Aktives maschinelles Lernen bei unvollständiger Information

Habilitationsschrift

zur Erlandung der Venia legendi für

das Fach Informatik

angenommen durch die Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg

von: Dr. rer. soc. oec. Georg Krempl
geb. am 10. März 1982 in Graz, Österreich

Gutachterin/Gutachter:
Prof. Dr. Myra Spiliopoulou
Prof. Dr. Eyke Hüllermeier
Prof. Dr. Bernhard Pfahringer

Magdeburg, den 7. September 2016

Zusammenfassung

Im Gegensatz zu den begrenzten menschlichen Steuerungs- und Überwachungskapazitäten nimmt das Volumen an automatisch generierten Daten kontinuierlich zu. Dies erfordert es, die dazu verfügbaren Kapazitäten effizient zur Verarbeitung und Kategorisierung dieser großen Datenmenge einzusetzen.
Diese Fragestellung wird innerhalb dieser Habilitationsschrift zum Themengebiet des aktiven und adaptiven maschinellen Lernens behandelt. Sie fasst zunächst die Literatur und Herausforderungen auf den Gebieten des aktiven Lernens und des Data Minings auf Datenströmen zusammen. Anschließend stellt sie eine neue Strategie zum probabilistischen aktiven Lernen vor. Diese entscheidungstheoretisch motivierte Strategie dient zur Auswahl des für einen aktiven, maschinellen Klassifikator informativsten Klassifikationsbeispiels unter einer Menge an potentiellen Klassifikationsbeispielen. Dazu bestimmt sie den aus dem Klassifikationsbeispiel zu erwartenden Klassifikationsgütezuwachs. Für diese Erwartungswertberechung werden nicht nur die Klasse des Klassifikationsbeispiels, sondern auch die tatsächliche A-posteriori-Wahrscheinlichkeit der Klassen an der Position des Klassifikationsbeispiels im Merkmalsraum als Zufallsvariablen modelliert. Über beide Zufallsvariablen wird der Erwartungswert berechnet, welcher anschließend mit der Dichte an der Position des Klassifikationsbeispiels im Merkmalsraum gewichtet wird. Anschließend wird das Klassifikationsbeispiel mit dem höchsten dichtegewichteten zu erwartendem Klassifikationsgütezuwachs ausgewählt.
Diese Berücksichtigung der tatsächlichen A-posteriori-Wahrscheinlichkeit in der Erwartungswertberechung stellt ein Novum auf dem Gebiet entscheidungstheoretischer aktiver Lernansätze dar, wo stattdessen bislang nur der wahrscheinlichste oder aber ein pessimistischer A-posteriori-Wahrscheinlichkeitswert verwendet wurden. Im Gegensatz zur informationstheoretischen Uncertainty Sampling-Strategie berücksichtigt die vorgeschlagene probabilistische aktive Lernstrategie die Anzahl der bereits vorhandenen ähnlichen Klassifikationsbeispiele. Dies ist von Vorteil, da somit der Explorationsgrad in der Nachbarschaft des Klassifikationsbeispiels in die Nutzenwertberechnung miteinbezogen wird. Für die Bestimmung dieser Nachbarschaft wird auf die im maschinellen Lernen gebräuchliche Annahme zurückgegriffen, dass die Nähe zweier Punkte im Merkmalsraum einen direkten Einfluss auf die Wahrscheinlichkeit ihrer gemeinsamen Klassenzugehörigkeit hat. Ausgehend von dieser probabilistischen aktiven Lernstrategie werden für die Berechnung des erwartenden Klassifikationsgütezuwachs im Rahmen der Arbeit ein flexibler, auf numerischer Integration basierender Ansatz, sowie mehrere auf geschlossenen Lösungen beruhende spezialisierte und schnelle Ansätze vorgestellt.

Die vorgeschlagene probabilistische aktive Lernstrategie bietet somit Auswahlansätze, die effizient und schnell berechenbar sind, direkt ein vorgegebenes Klassifikationsgütemaß optimieren, nicht auf eine bestimmte Klassifikatortechnologie beschränkt sind, nicht-myopisch und auch auf Mehrklassenprobleme sowie auf kostensensitive Klassifikationsprobleme anwendbar sind. Im Rahmen der Habilitationsschrift wird darüber hinaus gezeigt, wie diese Lernstrategie über das poolbasierte aktive Lernszenario hinaus angewendet werden kann. Dazu werden zwei Ansätze für aktives Lernen in Datenströmen sowie ein Ansatz für die aktive Auswahl von Klassen vorgestellt.

Die vorgeschlagenen Ansätze werden experimentell gegen mehrere dem aktuellen Stand der Forschung entnommene Vergleichsansätze evaluiert, darunter solche auf Basis von Uncertainty Sampling oder Expected Error Reduction. Die experimentelle Evaluation zeigt die Konkurrenzfähigkeit der vorgeschlagenen Ansätze für verschiedene Szenarien und aktive Lernaufgaben.

Abstract

The volume of automatically generated data is constantly increasing. However, human supervision and labelling capacities remain limited. Facing the task to mine large amounts of unlabelled data, an efficient allocation of annotation efforts is important. The research on active and adaptive machine learning in this habilitation thesis addresses this challenge. First, this thesis surveys the fields of active learning and data stream mining. Then, it contributes a novel probabilistic active learning strategy to these fields. This is a decision-theoretic strategy that computes the expected gain in classification performance from labelling additional instances. Given a candidate for labelling, as well as the number of already labelled similar instances, it models the possible label realisations from labelling this (or additional similar) instances as a random variable. In addition, it also models the true posterior in the candidate's neighbourhood as a random variable. Then, it calculates the expectation of the performance gain over both random variables. This expected performance gain is subsequently weighted by the density over labelled and unlabelled instances in the candidate's neighbourhood. Finally, the candidate with the highest density-weighted expected performance gain is selected for labelling. Considering also the true posterior as a random variable advances existing decision-theoretic strategies, which consider solely the most likely (or most pessimistic) posterior value. In contrast to the information-theoretic uncertainty sampling, the number of already labelled similar instances is considered as well. Thus, the proposed probabilistic active learning strategy accounts for the degree of exploration in a candidate's neighbourhood. For defining this neighbourhood, the approach follows the smoothness assumption. For the calculation of the expected performance gain, flexible approaches based on numerical integration as well as fast approaches based on closed-form solutions are derived. Thus, the strategy is computationally efficient, optimises directly a classification performance measure, is not limited to a particular classifier technology, is not myopic but allows to consider multiple label acquisitions at once, and it is applicable to multi-class and cost-sensitive classification tasks. Furthermore, this habilitation thesis shows that this strategy is not limited to the pool-based scenario: for evolving data streams, it proposes two approaches that use this strategy. In addition, it studies active machine learning also in its broader sense: for the task of active class-conditional example acquisition, it proposes an active class selection approach. The proposed algorithms are evaluated against state-of-the-art approaches, which are based on other active learning strategies, including uncertainty sampling and expected error reduction. The experimental evaluation shows the competitiveness of the proposed probabilistic active learning-based approaches in different scenarios and tasks.

Acknowledgements

I would like to take this opportunity to thank the many people, who have supported my work on this habilitation thesis in various ways.

First of all, I would like to thank my parents Barbara and Peter W. Krempl, and my brother Peter M. Krempl, who have supported me in all possible ways throughout my life.

I would like to thank Prof. Myra Spiliopoulou for providing me support and inspiration, as well as integrating me in her Knowledge Management and Discovery Group. From this group, I would like to thank in particular Daniel Kottke and Pawel Matuszyk for our inspiring and fruitful work, as well as Silke Reifgerste, Tommy Hielscher, and Uli Niemann, as well as all other colleagues from University of Magdeburg.

I would like to thank the many colleagues, distant and close, with whom I have discussed and collaborated. In particular, Katarzyna Bijak, Dariusz Brzeziński, Benedikt Budig, Antoine Cornuéjols, Matthias Deliano, Michael Denker, Thomas van Dijk, George Forman, João Gama, Sonja Grün, Michael Hanke, Pascal Held, Vera Hofer, Eyke Hüllermeier, Hans Kellerer, Roman Kern, Reinhard König, Rudolf Kruse, Mark Last, Vincent Lemaire, Tino Noack, Andreas Nürnberger, Frank Ohl, Mykola Pechenizkiy, Bernhard Pfahringer, Stefan Pollmann, Michele Sebag, Ammar Shaker, Sonja Sievi, Jerzy Stefanowski, Lyn Thomas, Klaus Tönnies, and Indrė Zliobaitė.

Furthermore, the many current and former students who have participated in discussions and research projects. In particular, Jannis Becke, Florian Bethe, Christian Beyer, David Bodnar, Fabian Göcke, Tuan Cuong Ha, Sebastian Hesse, Anita Hrubos, Andy Koch, Dominik Lang, Tuan Pham Minh, Stephan Möhring, Anton Niadzelka, Tino Reising, Tim Sabsch, Marianne Stecklina, Cornelius Styp, and Johannes Teschner.

To all those who remain unnamed, I offer my thanks and my apologies. Any errors remaining are, of course, mine.

Contents

Chapter 1

Introduction

1.1 Motivation and Outline

Recent years have seen a large increase in the creation and collection of digital data. Numbers are provided for example by the Digital Universe Study [33], which states a volume of 2.8ZB of data being created and processed in 2012, and projects this volume to increase 15 times by 2020. Another example is the number of webpages indexed by Google, which according to [26] increased from one million in 1998 to one billion in 2000, and further to one trillion in 2008. For organisations, for example, making use of this “big” data by data mining techniques is considered to provide competitive advantages [26]. However, the increasing volume of available data is contrasted by the limited available human supervision and labelling capacities. Therefore, the task of mining large amounts of mostly unlabelled data requires techniques that allocate the limited annotation efforts optimally.

This habilitation thesis studies such techniques from the fields of active and adaptive machine learning. Subsequently, it contributes a novel probabilistic active learning strategy to these fields. This active learning strategy is decision-theoretic and based on the calculation of the density-weighted, expected performance gain from labelling additional candidate instances. In contrast to existing expected error reduction strategies, the proposed approach models not only the possible label realisation as a random variable. Rather, it models also the true posterior in a candidate’s neighbourhood as a further random variable, and performs the expectation over both variables. This provides an advantage over existing expected error reduction strategies, which consider solely the most likely (or most pessimistic) posterior value. Furthermore, it uses statistics on the number of labelled instances in a candidate’s neighbourhood, thereby accounting for the degree of exploration therein. It provides a flexible approach based on numerical integration for calculating the expected performance gain, and derives closed-form solutions for fast approaches. Therefore, the contributed novel probabilistic active learning strategy is computationally efficient and is optimising directly a classification performance measure. The proposed strategy is not myopic, as it allows to consider multiple label acquisitions at once. In addition, it is also applicable to multi-class and cost-sensitive classification tasks. Furthermore, the use of this proposed strategy in combination with different classifier technologies is shown. Extending this initial pool-based approach, further approaches for evolving data streams as well as for the task of active class-conditional example acquisition are proposed. All these approaches are evaluated against other state-of-the-art approaches. These include active learning approaches based on uncertainty sampling and expected error reduction. The results of the experimental evaluations show that the proposed probabilistic active learning strategy is competitive in different scenarios and tasks.

1.1.1 Outline

The main part of this habilitation thesis consists of methodological chapters, each corresponding to a separate publication. Therefore, this first introductory chapter provides the frame for these later chapters. In its next Section 1.1, it gives an introduction to supervised machine learning, with particular focus on active learning and adaptive learning in evolving data streams. This is followed by two sections that review the literature and challenges in active learning, again with focus on evolving data streams. Therein, Section 1.3 starts with a meta review of surveys and position papers on this topic, before identifying and discussing of challenges for active learning in evolving data streams. Section 1.4 reviews the existing active learning strategies that are the most relevant to this work. It starts with a review of each strategy by presenting its main idea, before discussing its use in a non-stationary and stream-based scenario. The last Section 1.5 is dedicated to the proposed probabilistic active learning strategy. First, the strategy and its main idea are presented for the pool-based scenario. Then, its use in other active learning scenarios is discussed.

Each of the following Chapters 2–9 corresponds to a separate publication. Therein, chapter 2 is a position paper on challenges in data stream mining research. It includes a section on timing and availability of information, where active learning and handling incomplete information are discussed.

It is followed by Chapters 3–6 that address pool-based active learning. Therein, Chapter 3 introduces the probabilistic active learning strategy. Chapter 4 extends it to non-myopic and cost-sensitive classification and derives a fast closed-form solution. This is extended further in Chapter 5, where active learning for multi-class classification is addressed. The discussion of pool-based active learning closes with a comparative study in Chapter 6, which experimentally evaluates different combinations of classifier techniques and active learning strategies.

In Chapters 7–9, active learning beyond the pool-based scenario is addressed. This starts with Chapter 7 presenting an instance-wise approach for probabilistic active learning in evolving data streams. In Chapter 8, a clustering-based approach for probabilistic active learning in evolving data streams is introduced. In Chapter 9, an approach for the task of active class selection is presented.

Finally, documents specifying the bibliographical details of each publication, their contribution to computer science, and my personal contribution in each publication are given in the appendix.

1.2 Introduction

This section provides an introduction to active machine learning and data stream mining. It gives an introduction to supervised machine learning in Subsection 1.2.1, to active learning in Subsection 1.2.2, and to evolving data streams in Subsection 1.2.3.

1.2.1 Supervised Machine Learning

Machine learning is the field of study concerned with computer algorithms that improve automatically through experience [54, page XV]. The important aspect is learning as *improving through experience*, in contrast to being *explicitly programmed*. This was pointed out in a famous definition of machine learning as "the field of study that gives computers the ability to learn without being explicitly programmed", which is credited to Arthur Lee Samuel [68, see second footnote on page 3]. This learning from past experience, or exemplary data, is often motivated by situations where human expertise does either not exist or is not explicitly programmable, e.g. when humans are unable to explain their expertise [3, page XXV]. For example[1], it is difficult to derive an explicit program for the conversion of acoustic speech signals to words, despite the existence of human expertise. Furthermore, it is desirable (and often necessary) that a machine will adapt to new circumstances. For example, a speech converting machine might be required to adapt to a new user, by learning from experience in its interaction with the user. A formal definition of this learning, according to [54, page 2], is improving a *performance*, measured by a *performance measure* P, in a *task* T from *experience* E. For the speech converter above, an exemplary task is the classification of acoustic sensory input into ASCII-encoded textual output classes. Therein, experience might consist of pairs of input-output data that connect recorded acoustic signals to the corresponding word, e.g. provided by the user as feedback. The performance measure might be the error rate, that is, the percentage of incorrectly converted words.

Machine learning is categorised in supervised, unsupervised and reinforcement learning, based on the type of feedback (or supervision) provided to the learning system. The classification task above is an example of *supervised* machine learning, where a supervisor provides a learning system with labelled training data in the form of input-output tuples. More formally, supervised learning [66] is concerned with inferring a predictor $f : \mathcal{X} \to \mathcal{Y}$. Here, $x \in \mathcal{X}$ is the feature vector (or set of attributes or explanatory variables) of an instance, e.g. the acoustic sensory input sequence. Furthermore, $y \in \mathcal{Y}$ is its corresponding response variable (or target attribute), e.g. a class label corresponding to a word. Finally, $f \in \mathcal{F}$ is a prediction function from the hypothesis space, i.e. either a classifier, in the case of a categorical response variable, or a regression function, in the case of a numerical response variable. As this work addresses active learning for classification, solely classifier functions are considered further on. A classifier function is learned on a training data set $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^{l} \sim \Pr(x, y)$, e.g. on a sample of exemplary words with their corresponding sensory input sequences. Then, $\Pr(x, y)$ denotes

[1]This example is inspired by an example given in [3, page XXV].

the joint distribution of features x and class labels y. Marginalising $\Pr(x, y)$ over the features x yields the class prior distribution $\Pr(y) = \sum_x \Pr(x, y)$. Marginalising $\Pr(x, y)$ over the class labels y results in the feature distribution $\Pr(x) = \sum_y \Pr(x, y)$. Using Bayes' theorem [9], the posterior probability of an instance with given feature x being from class y is derived as $\Pr(y|x) = \frac{\Pr(x,y)}{\Pr(x)} = \frac{\Pr(x|y)\cdot\Pr(y)}{\Pr(x)}$, where $\Pr(x|y)$ is the class conditional feature distribution. The typical objective is generalisation, that is, the prediction of the class label of new instances. In the example above, this is the prediction of the word (class label) that corresponds to a new acoustic sensory input sequence. Some applications require probabilistic classification, that is, the computation of probabilistic estimates for $\Pr(y|x)$. However, for discriminative applications, it is sufficient to know the label of the most probable class y^*. This label might be obtained as $y^* = \arg\max_y (\Pr(x|y) \cdot \Pr(y))$, by exploiting the fact that for a given x the probability $\Pr(x)$ is a constant factor, thus $\Pr(y|x) \propto \Pr(x|y) \cdot \Pr(y)$. In any case, the learner's generalisation performance is typically evaluated on a so-called test set $\mathcal{T}$, which comprises solely hold-out data that was withheld from training.

The other two categories of machine learning, *unsupervised* and *reinforcement* learning, are not within the scope of this work. However, summarising them for matter of completeness, the former category corresponds to machine learning *without supervision*, where the objective is to find structure in the input that consists of unlabelled training examples. The latter category corresponds to machine learning without explicit training examples, but rather by *reinforcement* from receiving rewards on reaching goals in the interaction with a dynamic environment.

However, this categorisation into completely supervised or completely unsupervised learning is too restrictive for some applications. For example, in some applications the training data set is partially labelled and partially unlabelled. *Semi-supervised* learning, a hybrid between supervised and unsupervised learning, addresses machine learning on such data. Its objective corresponds either to learning a predictor, as in the supervised case, or to find structure in the input, as in the unsupervised case [83]. In the former, the additional unlabelled data provides additional information about structure in the input data, which might help to modify or re-prioritise classification hypotheses that are obtained from the labelled data alone [82, page 6]. In the latter, the labelled data provides additional information such as must-link constraints (all instances from the same class must be assigned to the same cluster) or cannot-link constraints (any two instances from different classes must not be assigned to the same cluster) [83, pages 892–893].

Typically, it is assumed that labelled and unlabelled data come from the same domain, i.e. from the same distribution (see e.g. the definition given in [83, page 892ff]) and the same task. However, some applications involve learning and applying models on different domains, where distributions or tasks differ. This corresponds to *transfer learning* [57], where knowledge is transferred between source and target domains that differ in their distributions or tasks. An example is the adaptation of a model[2] learned on labelled training data from a source domain to testing data from a different target domain. This

[2]Some authors denote this as domain adaptation. Following the notation in [5], we use transfer learning as the more generic term in this work.

is illustrated by the task of adapting a speech converter developed on training data from users in one region, such as northern Germany, for deployment in another region, such as Austria. Although the task of converting acoustic signals corresponding to spoken German into written words is the same, the source domain (northern Germany) and target domain (Austria) might differ in their distributions, e.g. due to idiosyncrasies in the pronunciation or vocabulary. A common categorisation within transfer learning is based on the availability of labelled information in each domain [57]: *Inductive transfer learning* denotes the case where labelled data is available for both domains, but the learning task is different. *Unsupervised transfer learning* denotes the case where neither in the source nor in the target domain labels are available. Finally, transfer learning under the constraint that labelled data is solely available from the source domain is known as *transductive transfer learning* [57] (also denoted unsupervised domain adaptation or unsupervised transductive transfer learning [5]). In the context of this work, it is important to emphasise that research in transfer learning is not limited to domains that differ in their task or their (geographical) population, as in the example above. Even when applied to the same task in the same population, a machine learning model might require adaptation due to changes in the distribution over time. In the speech converter above, a model learned and applied to users from the same region might require adaptation, as for example the users' vocabulary might evolve over time. Addressing such *temporal* difference between domains is important when working with evolving data streams, as discussed in the later Chapter 2.

A further categorisation is provided by the interaction between a machine learner and its supervisor, and the resulting control the learner has on its training samples [19]. In the *passive* learning paradigm, the learner is assumed to be a passive receiver of labelled training instances. That is, there is no interaction between learner and supervisor concerning the selection of training instances. Thus, it is the responsibility of the supervisor to provide a representative and informative sample of training instances. In contrast, the *active* learning paradigm assumes an interaction between learner and supervisor on the selection of training instances, such that the active learner influences or controls the sample selection. The focus of this work is on active machine learning, which is therefore discussed in further detail below.

1.2.2 Active Learning

Active learning deals with learning computer systems that *actively* develop and test new hypotheses, thereby improving by experience and training [72, pages 3–4]. It is often assumed that such active learning systems *interact* with their environment and influence or control the acquisition of new data [21]. Thus, such systems actively construct experiments for training, rather than passively processing given ones [19]. For example, the speech converter from the previous subsection might actively select few instances, for which the corresponding word label is uncertain. Subsequently, the learning system might interact with the user to obtain the true class label for those instances, in order to improve its future predictions.

In the case of classification, active learning corresponds to optimising the interaction between a classifier system and an oracle that provides supervision. Such an oracle is, for example, a human expert providing labelled data, constraints or categorisations. The objective in active learning is to optimise this interaction [71]. For example, by requesting the most insightful labels first. One motivation for active learning are applications where supervision is limited or costly, and thereby efforts should be focused on the data that improves the classification performance the most. This corresponds to optimising exploration and exploitation of the data space of a domain [10]: selecting instances from non-sampled areas (exploration) in order to limit the error therein, and selecting further instances in already sampled ambiguous areas (exploitation) to improve the classification model in critical areas.

Active machine learning has been approached from different angles under different names for over two decades now. It is inspired by optimal experiment design, reinforcement learning, and human learning. *Optimal experimental design*, also denoted as regression experimental design, is concerned with the optimal organisation of experiments [28]. That is, when and where to take measurements, such that the informativeness of the gathered data is maximised [18]. Ultimately, techniques from optimal experimental design are incorporated into a learning system that uses them directly to generate hypotheses and to automatically perform experiments to evaluate them. An exemplary implementation of such a learning robotic system for the domain of functional genomics is presented in [41]. The task of repeatedly selecting, gathering, and processing information autonomously is also important in *reinforcement learning*, another field of machine learning. In reinforcement learning, the learning problem is to determine which action yields the highest reward in which situation [75, chapter 1]. Therefore, the learning system is in a different trade-off between exploitation and exploration than above: on one hand, the system should exploit its knowledge by selecting the action that is currently known to maximise the reward. On the other hand, it should aim to find actions with even higher rewards. The latter requires to try new, unexplored actions. This helps in selecting better actions in the future, thereby potentially improving the system's performance in the long term. In this exploration, the selection of actions to gather informative data from the environment corresponds to active learning. This computational active learning has a correspondence in the educational concept of *human* active learning. This concept, researched in learning sciences, cognitive psychology, and educational psychology [53], defines active learning as engaging students in "such higher-order thinking tasks as analysis, synthesis, and evaluation." [11, page iii].

In active machine learning, different scenarios for the interaction between machine learner and supervising oracle have been studied. One of the first publications in this field is [4], who investigates the use of queries to learn an unknown concept. Therein, the learning system poses queries on membership, equivalence, subset, superset, disjointness, and exhaustiveness. This is a particular active learning scenario, later denoted as *query synthesis* [70]. In this scenario, the active learning system asks membership queries for synthetic instances. That is, the learner requests the label for synthetic instances that are generated upon its query, rather than relying on instances provided by an oracle. Another name for this query synthesis scenario [72] is *construc-*

tive active learning [19]. For this active learning problem, the use of optimal experimental design as a tool for guiding a neural network learner by using queries in the exploration of its domain is proposed in [21].

In the query synthesis scenario above, the freedom of an active learner to create queries that correspond to any point in the feature space might cause problems. For example, this might result in queries that are not meaningful or difficult to label by a (human) oracle. Thus, other active learning scenarios restrict the learner to queries on existing instances provided by the oracle. One scenario is that a pool $\mathcal{U}$ of such unlabelled candidate instances is provided for free, from which the active learner has to select instances for labelling requests. A common assumption in this so-called *pool-based active learning* [72, page 9] scenario is that the learner has access to all unlabelled instances at once, and that this pool is static. This allows the active learner to compare or rank candidates by their estimated usefulness. Another scenario restricting queries to existing instances is *stream-based active learning* [19], also called *sequential active learning* [72, p. 8–9] or *selective sampling* [20]. In this scenario, instances are provided by the oracle one after another, and the active learner has to decide immediately whether to request an instance's label or not. Thus, in contrast to the pool-based setting, the active learner might not request the labels of previous instances at later time points. This scenario is motivated by applications where either limited computational capacities prevent storage or processing of all instances at once, or where data is arriving sequentially. Thus, it is relevant for active learning in data streams, in particular to evolving data streams, as discussed further below.

Finally, research on active learning is not limited to the selection of instances for labelling. In its narrower sense active learning corresponds to this very problem. However, in its broader sense active learning is applicable to the targeted selection of any type of informative data. This more general topic has recently been denoted as *selective data acquisition* in [7]. Therein, the authors distinguish based on whether active selection is done during training or at prediction time, and based on the type of information that is requested. Combining this recent and more complete categorisation with the prototypical categorisation provided in [70] yields the following categorisation of the different problems in selective data acquisition.

Selective data acquisition at *training time* might concern feature values, labels, feature labellings, or whole examples. In *active feature (value) acquisition* [65, 7], the value of one or several features is acquired during training. This is motivated by situations where not all feature values are available for free, and not all features are equally important for all instances. Then, the aim is to actively select the most informative subset of features for a given instance at training time. *Active label acquisition* corresponds to active learning in its narrower sense, i.e. to the active selection of instances for labelling. As denoted in [7], the problems of active feature value and active label acquisition might occur together. Then, class labels might be interpreted as particular "features". *Active labelling of features* is defined in [7] as the active selection of a given feature value, for which a label is requested. They discuss an example from sentiment detection, where particular words (feature values) might have a positive or negative connotation (label), which might be provided upon request from an oracle. *Active class-conditional example acquisition* [7], denoted *active class selection* in [51] and [70, page 33], is an inverted active learning setting, where the learner actively selects a class, for which it queries instances (i.e. exemplary feature vectors) from the oracle.

Likewise, selective data acquisition at *prediction time* might be differentiated according to the same categorisation of types of information as at training time. However, the terminology used to describe these is not consistent in literature. The *active acquisition of feature values at prediction time* is denoted as *active classification* in [35], and the *active acquisition of labels at prediction time* for the purpose of training[3] is denoted as *active inference* in [7], meaning that labels are requested to improve the classifier during prediction. Active labelling of features at prediction time seems to not have been researched so far, and the active acquisition of class-conditional examples seems solely justified for the purpose of training.

Furthermore, the problem of *active learning from noisy acquisitions* [7][4] considers cases where oracles are not perfect providers of noiseless information. In these cases, the information provided by an oracle is subject to noise, but the active learner is allowed to formulate multiple queries of the same kind, either repeatedly from the same oracle, or from independent oracles.

1.2.3 Learning in Data Streams and Non-Stationary Environments

In the scenario of stream-based active learning [70, page 10] described above, instances arrive sequentially in a so-called *data stream*. Learning on such streaming data that is collected over time, as opposed to *batch learning* on all instances at once, has received attention in passive machine learning, for example in [24]. In [8], four characteristics are stated that define such a data stream model: (1) instances arrive *online*, (2) in an *ordering* that is not controlled by the learning system, (3) streams are *potentially unbounded* in size, (4) *processed instances are not easily accessible* as they are discarded or archived. These characteristics have a counterpart in the challenges caused by the ever increasing amount of data, a problem that has attracted attention under the term “big data” [47]. Three such challenges are listed in [47], namely volume, variety, and velocity. Therein, *volume* refers to the large volume of data, which increases faster than the speed of data processing tools. *Variety* refers to the many different types of data. Finally, *velocity* refers to the speed of continuously arriving data, which should be processed in real-time. Considering the characteristics of a data stream given in [8], volume and velocity correspond to challenges due to limitations in the available computational resources in time and memory.

In addition, recent literature (e.g. [26, 30]) considers further challenges. In the context of “big data”, variability and value are named as additional challenges in [26]. There, the former refers to changes, e.g. in the structure or interpretation of the data, while the latter refers to the business value

[3]Note that for the purpose of *prediction* of the same instance, the active acquisition of its label from the oracle corresponds to *classification with a reject option* or *selective classification* [25]. There, the classifier is allowed to vote for not classifying uncertain instances.

[4]In the categorisation proposed in [7], this appears as sub-category of active acquisition at training time. However, as it appears to be a potential problem in any type of active acquisition of information from oracles, it is presented as orthogonal issue in this work.

resulting from making use of this data. The counterpart of variability is the characteristic of *change* in data streams: patterns might evolve over time, data might not be considered independently and identically distributed but rather temporally as well as spatially situated [30]. Thus, in addition to limited computational resources, *evolving* data streams pose the further challenge of requiring adaptation to change. Changes might concern the target variable, the available feature information, or correspond to changes in the distribution of variables. In the foremost case, classes might be added to or removed from the target variable, or the task itself might change, requiring prediction of a different target variable. In the second case, the feature space might change, for example due to changes in the availability of features. In the last case, the target and feature space remain static, but the variables are affected by so-called concept or population drift. This problem has been subject to extensive research and has recently been surveyed for example in [32]. Thus, solely a short categorisation is provided here. Based on which component of the joint distribution $\Pr(x, y)$ of features x and class labels y is affected, one might distinguish drift of the posterior $\Pr(y|x)$, of the conditional feature $\Pr(x|y)$, of the feature $\Pr(x)$ and of the class prior $\Pr(y)$ distribution.

While the challenges of limited computational resources and adaptation to change are inherent to mining in evolving data streams, they might be also relevant outside streaming applications. For example, efficient algorithms are also important for batch learning in large pools of data with limited computational resources. Likewise, change might also effect models learned offline. This might then necessitate adaptation of these models. Thus, machine learning techniques addressing these challenges are of general interest, under passive as well as under active learning paradigms.

1.3 From Static to Adaptive Active Learning

Active learning in general has been reviewed in several recent works. However, no survey dedicated to adaptive active learning exists to the best of my knowledge. One aim of this section is to provide a broad overview on the general problem of active learning and its different aspects considered in literature. Another aim is to discuss in depth adaptive active learning in non-stationary environments such as evolving data streams, as depicted in Fig. 1.1. Thus, this section begins with a brief meta review on active learning surveys, in order to provide an overview and key references for its different aspects. Then, the specific challenges arising in adaptive active learning are elaborated. Finally, the different active learning strategies and their use in non-stationary environments are discussed in the next section.

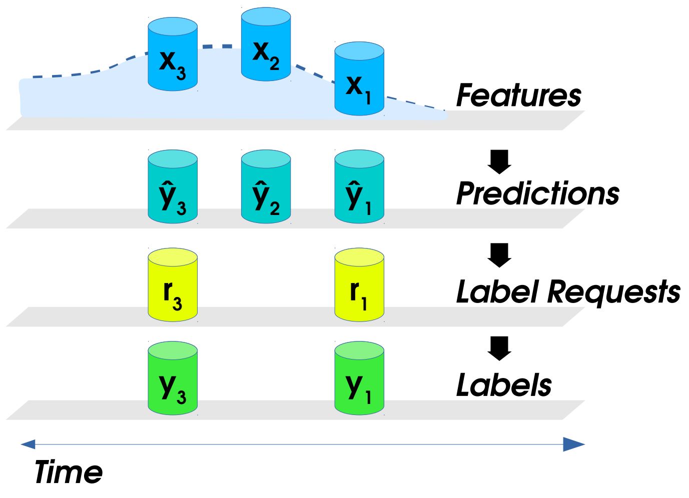


Figure 1.1: Active learning in evolving data streams comprises features x_i, predicted class labels $\hat{y}_i$, label requests r_i, and received true labels y_i. These four types of information might arrive at different points in time and constitute separate, although interdependent streams.

1.3.1 Related Surveys on Active Learning

Several recent publications have surveyed active learning in general. These include an introductory book [72], which provides an introduction to the different active learning scenarios and strategies and discusses practical considerations. A compact review is given in the Encyclopedia of Machine Learning [19], with focus on active learning scenarios and selected exemplary active learning techniques.

Several surveys with focus on different aspects exist [70, 7, 29, 6, 2]. The survey in [70] precedes the introductory book [72] by the same author. It is an introduction to active learning, its different scenarios and selection strategies, problem setting variants and related research areas. Furthermore, it provides a brief analysis on the question whether active learning works, and discusses practical considerations. It also includes a section on stream-based selective sampling [70, page 10]. However, the focus therein is on stationary environments. The book chapter [7] reviews active learning in its broader sense of selective data acquisition, thereby extending the problem setting variants from [70]. In [29], the literature on active learning is reviewed from an instance-selection perspective. Therein, active learning strategies are categorised based on whether or not they explicitly consider correlations between instances. Finally, emerging challenges and trends are briefly discussed, including active learning on streaming data platforms. In the book chapter [6], the use of active learning as a tool for addressing class imbalancedness is reviewed. Finally, the recently published book chapter [2, Chapter 22.6.4] includes a short section on streaming active learning with a discussion of a few particular adaptive active learning approaches.

In addition, different aspects of active learning have been addressed in recent PhD theses [50, 76, 51, 59, 61, 23]. The PhD thesis of [50] addresses active learning in cost-sensitive environments, with focus on resampling-based approaches. Active learning in the context of linguistic annotation is studied in [76], addressing problems such as stopping the annotation process and the reusability of samples between classifiers. In [51], the general problem of selective data acquisition is studied, with focus on active class-conditional example acquisition. New methodologies for active classification and active regression are discussed in [59]. The thesis of [61] studies active learning with focus on the trade-off between exploitation and exploration, as well as its industrial applications. Finally, [23] proposes a new family of decision trees for streaming data, which are based on confidence and also extended to active learning.

Some of these surveys discuss selected adaptive active learning approaches. However, none of these provides a structured overview on the challenges or existing approaches. Furthermore, the term stream-based selective sampling is frequently used for describing active learning in stationary streams (e.g. [70]). This makes it even more difficult, but also more important, to identify and survey adaptive active learning approaches for non-stationary environments. This is done in the following subsections and continued further in Section 1.4. Next, an overview on the different challenges for adaptive active learning is provided, and references to related existing works are given where available.

1.3.2 Efficient Use of Limited Computational Resources

As outlined in Section 1.2.3 above, the limitation in computational resources is a challenge that is inherent to data streams. It requires efficient processing of data. As in passive learning, the arriving data might be processed instance-wise, i.e. one instance after another, or in chunks, i.e. several instances at once. However, in active learning the protocol for the interaction with the oracle needs to be considered. If the protocol restricts queries to the most recently arrived instance, instance-wise processing is required, as assumed for example in [87, 88, 42]. In contrast, the protocol might allow querying the label of older, already processed instances as well. Then, instance-wise and chunk-based processing are possible, as assumed for example in [84, 85, 40, 44]. Furthermore, the interaction protocol might allow requesting the labels of a single instance or of batches of instances. The former limits active learning to myopic approaches, while the latter allows non-myopic active learning approaches as well. While the majority of adaptive active learning publications is limited to myopic approaches, few non-myopic approaches exist that ensure diversity among the queried instances in a batch [15, 14].

Another question concerning chunk-based processing is whether labels obtained in a chunk are allowed to affect the prediction on the remaining unlabelled instances in the very same chunk. Although the resulting effect might seem negligible, it directly affects the availability of most recent[5] labelled data. This suggests an effect increasing with chunk size and the extent of drift, although this has not been investigated so far.

1.3.3 Adaptation to Change

The inherent challenge of active learning in non-stationary environments is adapting the classifier and sampling strategy to changes. As outlined in section 1.2.3 above, there are various types of change that might occur. However, the focus within this work is on changes due to drift. This challenge has been identified in previous works, most notably in [87] and its extension in [88]. There, the authors highlight that concept drift might affect any part of the feature space, and provide two illustrative examples[6] for changes close and remote of the decision boundary. They show experimentally that the popular active learning strategy of uncertainty sampling, discussed in detail in Subsection 1.4.1, fails to identify remote changes. The reason is that uncertainty sampling is an exploitation strategy, which requests labels solely for instances located closely to the expected decision boundary. Thus, it fails to explore areas that are further away.

It is noteworthy that this problem related to balancing exploration and exploitation is not limited to adaptive active learning, but is also relevant in stationary environments. For example, [56] notes this issue of focusing label acquisitions to regions close to the expected decision boundary in many conventional active learning strategies. For static active learning, an approach that dynamically balances between exploitation and exploration is proposed. Similarly, [13] combines so-called exploration sampling in dense, unexplored regions with exploitation sampling in well-explored but uncertain regions, i.e. regions where the sampled labels are contradictory. The idea of both strategies is to initially favour exploration, while later focusing on exploitation, in order to ensure convergence to a good decision boundary. In stationary environments, the experimental results in the above works indicate that this initial exploration might be sufficient. However, as pointed out in [87], in non-stationary environments previous exploration is *not* sufficient, as drift might affect any part of the feature space and therefore requires continuous exploration. Otherwise, a change affecting the classification rule and its subsequent performance might go unnoticed.

This is illustrated in Fig. 1.2, which shows the distributions at three consecutive time points $T = 0, 1, 2$. The true underlying joint distribution for this binary classification problem (positive against negative class) is plotted left to the ordinate axis, while the joint distribution as predicted by the model is plotted to the right. The blue lines on the left indicate the Bayes' optimal

[5] Actually even some "future" data, from the point of view of instance-wise stream processing.

[6] See Fig. 1 in [88].

decision boundaries, while the black lines on the right show the learned decision boundaries. The shaded area on the right indicates the importance attributed to sampling instances in a region. Here, dark red indicates high importance, and white indicates no importance. The indicated importance corresponds to a strategy that starts with exploration and subsequently focuses on exploitation, as discussed above. Initially, the exploration at $T = 0$ ensures that the main characteristics of the underlying true distribution are captured by the model, even if not perfectly. Then, at $T = 1$ and $T = 2$, sampling is focused on the region close to the expected decision boundary. As a result, the (upper) predicted decision boundary converges towards the (upper) Bayes' optimal decision boundary. However, at $T = 1$ also a drift occurs that effects the label of the cluster located at the bottom of the plot. As this is in a remote region that is not further explored at consecutive time points, this change is never noticed.

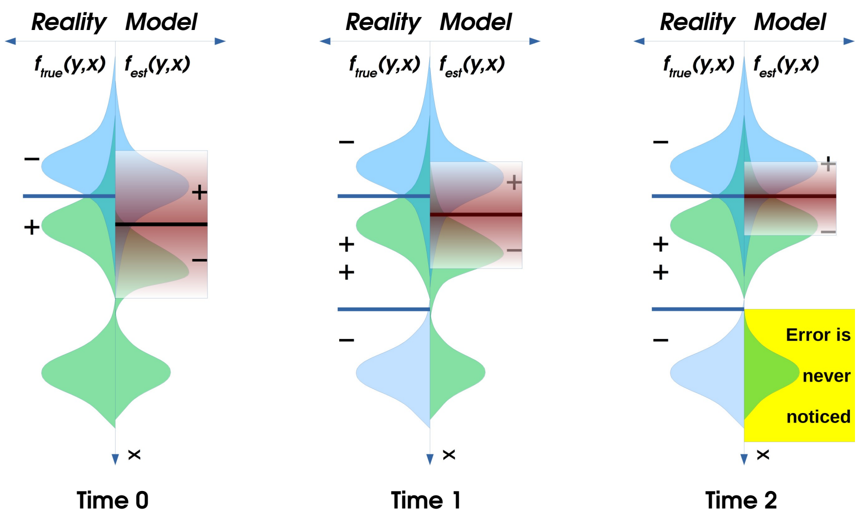
This challenge has been addressed in different ways in literature. In [87, 88], the authors propose to use random sampling in combination with an exploitative active learning strategy (uncertainty sampling in their case). Random sampling selects with equal probability any instance for labelling. This corresponds to exploration, although without consideration of the previous degree of exploration around an instance. In chunk-based active learning, a similar strategy is to ensure exploration in each newly obtained chunk. This is either done by randomly sampling a certain percentage of instances therein (e.g. in the approaches proposed in [84, 85, 64]), or by building (parts of) the prediction model from scratch on each chunk (e.g. in [39, 44]).

Another strategy is to monitor the distribution of incoming data by using change detection techniques. These techniques, reviewed for example in [69], trigger a signal when changes in the distribution are detected. Upon a detected change, the sampling strategy is adjusted, for example by triggering a new exploration phase. This is used for active learning of decision trees in [27, 38]. However, as pointed out in [86], some types of drift require labelled data to be detectable. In particular, drift that does not affect the unconditional feature distribution $\Pr(x)$ is not detectable from unlabelled data alone.

1.3.4 Budget Management

Active learning is often motivated by limited annotation resources [70, page 4ff], which define the *labelling budget* [87]. This requires to carefully allocate them to the most informative instances. How to distribute the use of these resources over time is a challenge that is closely related to the previous question on adaptation to change.

In stationary environments, the active learner might eventually focus on exploitation around the expected decision boundary, which thereby ideally converges towards the true decision boundary. Thus, the usefulness of additionally requested labels is likely to decrease over time. However, due to the reasons given above, this should not be assumed in non-stationary environments. Upon change, new label requests might provide more insights than previous ones before the change. A simplistic approach is to distribute the labelling budget either equally or decreasingly over time. However, in the previously described situation this approach might perform worse than a more sophisticated one that focuses its efforts on moments after changes. Thus,



(a) Time $T = 0$ (b) Time $T = 1$ (c) Time $T = 2$

Figure 1.2: Illustration of issues due to drift and insufficient exploration at later time points. The three plots show for three consecutive time points $T = 1, 2, 3$ the true (left of the ordinate axes) and estimated (right of the ordinate axes) joint distributions $\Pr(x, y)$, as well as the Bayes' optimal decision boundaries (horizontal blue lines) and the learned decision boundaries (horizontal black lines). Furthermore, the sampling preferences are shown in shades of red for an active learning strategy that favours exploration at early and exploitation at later time points (red indicating high preference). As seen in the rightmost plot, a change affecting a remote region that is not further explored at consecutive time points is never noticed.

managing the budget such that it is optimally allocated over both space and time is a relevant challenge for active learning in non-stationary environments. In chunk-based approaches, this corresponds to determining for each chunk the optimal number of label acquisitions therein, and subsequently selecting the most promising candidates in a pool-based approach. This is addressed in [85], where the use of a minimum-variance approach for estimating the number of required instances in a chunk is proposed. In instance-based approaches this is even more challenging: even if the optimal percentage of label requests per processed instance was known, the question remains, how to identify and select the most valuable instance online in a streaming sequence. First approaches for addressing this problem are proposed in [87, 88]. Here, the aim is to maintain a fixed sampling proportion over time, which is done by adjusting a selection threshold heuristically. More recently, in [42] adaptive filtering techniques were proposed for this purpose, and the problems of spatial and temporal sampling were discussed in more detail.

A further issue is the combination of budget management and change detection. While first approaches exist (see Subsection 1.3.3 above), these do not provide a generic budget management framework but rather are limited to a particular classifier technology.

1.3.5 Providing Bounds for Adaptive Active Learning

Related to budget management is the question of how well an active classifier currently performs, in particular in a streaming environment. In passive classification, labelled data might be used in a prequential evaluation [31]. However, in active learning labels are scarce and costly. Thus, some works have derived theoretical bounds for the number of mistakes and necessary label requests. The only such work addressing a somehow non-stationary environment is [80]. However, it proves bounds for the number of mistakes and necessary label requests in *static concepts* with covariate drift. A potential reason for the absence of such bounds for non-stationary concepts might be that strong assumptions on the type of drift would be required, limiting the applicability of results to very particular situations.

1.4 Existing Active Learning Strategies

This section reviews the most relevant among the existing active learning strategies. Most of these strategies were originally designed for the static, pool-based scenario. Some were later extended to the stream-based scenario. Therefore, each subsection first presents a strategy and its main idea in the pool-based scenario. This is then followed by a discussion on the use of this strategy in a non-stationary and stream-based scenario.

1.4.1 Uncertainty Sampling

Uncertainty sampling [48] is an information theoretic, heuristic active learning strategy. The main idea behind uncertainty sampling is to measure the current classifier's *uncertainty* in an instance's label, and to select the instance for labelling that has the highest uncertainty. Here, *uncertainty measures* serve as proxies for the impact an instance's label would have on the classification performance. In [48], the authors proposed to use a probabilistic classifier for estimating the posterior of a candidate's most likely class. As uncertainty measure they used the absolute difference between this posterior estimate and 0.5. This uncertainty measure is commonly denoted as *confidence* (see e.g. [72, page 14ff]), with low values indicating high uncertainty. The *entropy* over the classifier's posterior estimates is another uncertainty measure (see e.g. [72, page 14ff]), where high values indicate high uncertainty. A further uncertainty measure is the *margin* between a classifier's decision boundary and the candidate's position in a feature space [72, page 14ff]. Here, low values indicate high uncertainty.

Given a classifier that provides a posterior estimate or margin, the implementation of this approach is trivial. Furthermore, in the pool-based setting, its asymptotic time complexity is $O(|\mathcal{U}|)$, where $\mathcal{U}$ is the set of unlabelled candidate instances. In data streams it is constant. However, these advantages are contrasted by the known problem [81, 46] that these uncertainty measures do not consider the certainty of the posterior estimate itself. That is, they do not account for the number of instances, on which the posterior estimates are made or the decision boundaries are drawn. For example, this might lead to problems with unreliable uncertainty measures at early learning stages, when posterior estimates are based on few instances. At later learning stages these uncertainty measures tend to sample in regions with high Bayesian error, where the performance is not improvable, as we have shown in [46]. The former issue might be mitigated to some extend by using beta priors on the posterior estimates. The latter issue, however, remains. The results of some empirical evaluations (e.g. [16, 67, 45]), including our own, indicate that this approach performs often worse than random sampling baselines.

Nevertheless, its algorithmic simplicity and computational efficiency make it one of the most popular active learning baselines, as for example in the active learning competition [36].

Use of Uncertainty Sampling in Evolving Data Streams

Uncertainty sampling is easily applicable in evolving data streams. Its sole requirement is that the underlying classifier is suited for evolving data streams.

This means that the classifier should provide at any time either reliable posterior estimates for confidence- or entropy-based uncertainty measures, or reliable margins for margin-based uncertainty measures. As a result, this strategy is frequently used in evolving data streams [38, 49, 74, 87, 88, 40, 79, 78].

The chunk-based approach proposed in [38] actively constructs a decision tree. It uses a Naive Bayes classifier to perform confidence-based uncertainty sampling to select labelling candidates. Upon labelling, a decision tree classifier is trained. It is noteworthy that this corresponds to the label reusability scenario discussed in [77]. Translated to the terminology of [77], in this approach the Naive Bayes classifier acts as *selector* of instances for labelling, while the decision tree classifier acts as *consumer* of these labels. Furthermore, the approach in [38] is related to earlier work in [27] that also addresses active learning for decision trees in evolving data streams. However, the instance-wise approach in [27] uses a change detection on the distribution of unlabelled data, and upon a detected change it selects a fixed number of instances using random sampling.

In [49], a chunk-based approach is designed for support vector machines. This approach uses the margin as uncertainty measure, and a sliding window technique for adaptation. A support vector machine is also used in the more recent work [74] on sentiment analysis in evolving data streams, which includes a section on active learning. Therein, the authors experiment with margin-based uncertainty sampling and random sampling.

In [87] and its extended version [88], uncertainty sampling is used in instance-wise approaches. It is combined with different budget management and randomisation strategies. As explained in Subsections 1.3.3 and 1.3.4 above, randomisation serves to ensure sufficient exploration at any time, while adaptive thresholding is used to influence the consumed budget.

In [39], a chunk-based approach is proposed. For improving diversity among selected instances, it uses an amnesic clustering model. However, it is designed to be usable with any stream classifier technology. On each arriving chunk of instances, clustering is performed from scratch, Then, the most informative instances from each cluster are selected for labelling and fed to the classifier. For active learning, the approach distinguishes between a macro and a micro step. The *macro* step is used to rank clusters according to their homogeneity in terms of their predicted class distribution. This distribution is estimated by the current classifier, which was learned on the labelled instances from previous chunks. This corresponds to confidence-based uncertainty sampling. The later *micro* step determines the most useful instance within a given cluster. Thus, it ranks instances by combining geometrical information inside their cluster with the instance's maximum a posteriori classification probability. After selected instances are labelled, the clustering information is discarded.

The instance-wise approach proposed in [40] combines uncertainty sampling with density weighting. Density weighting is applied for filtering out candidates from low-density regions. On a remaining candidate from high-density regions, the candidate's margin-based uncertainty score is compared against a adaptive threshold with randomisation.

In [79], a wrapper classifier is evaluated that uses an uncertainty sampling approach. The used uncertainty sampling employs a (not further specified) support function value provided by the underlying classifier.

Finally, [78] is an application-oriented paper that uses uncertainty sampling. For random forest classifiers, the authors propose to use the posterior difference between the most and second-most probable labels as uncertainty measure. For evolving fuzzy classifiers, the authors propose a specialised uncertainty measure that considers at an instance's position *conflict*, i.e. how much opposing classes are overlapping, and *ignorance*, i.e. how close the nearest rule is. If rules are clusters, conflict measures the overlap of opposing clusters, while ignorance measures the distance to the nearest cluster. This is related to the idea of combined measures for *representativeness* and *diversity* discussed in [29].

1.4.2 Version Space Partitioning and Query by Committee

According to [19], the oldest practical active learning work is [63]. This work uses the strategy of *partitioning the version space.* This strategy is based on selecting those labelling candidates, for which the disagreement between hypotheses in the current version space is maximal [19]. A particular case of version space partitioning is the *query by committee* strategy for ensemble classifiers: Here, all ensemble members predict the class label of candidate instances. Subsequently, the candidate with the highest disagreement in its predicted class labels is selected for labelling [73]. In particular the query by committee strategy is frequently used in data streams, although it requires training and maintaining an ensemble of classifiers.

Use of Query by Committee in Evolving Data Streams

The approaches proposed in [84, 85, 64] use the query by committee strategy for active learning within an ensemble, without restriction on the type of base classifiers therein. In the chunk-based approach in [84] and its extension in [85], a new base classifier is learned on each arriving chunk. For initial training of this classifier, a fixed proportion of instances within this chunk is labelled randomly. Then, the disagreement within the ensemble is used to select further instances for labelling. In [84], in each chunk a fixed proportion of the remaining instances is labelled in this way. In [85], this proportion is automatically determined, and adaptive weighting of base classifiers is introduced. The instance-wise approach proposed in [64] also learns an ensemble. However, it combines this with change detection techniques to learn new base classifiers on demand. For change detection, it approximates the feature distribution by a mixture of spherical clusters. Each cluster corresponds to the training set used for learning one base classifier. Instances that are outside the mean-variance range of previous clusters are considered suspicious. Finally, neighbouring suspicious instances forming a spherical cluster are used to train a new base classifier. For classification of a new instance, a weighted voting scheme is used. Here, a base classifier's vote is weighted according to the distance between the centre of its corresponding cluster and the position of the new instance.

The idea to use clustering in combination with a query by committee strategy is taken further by approaches, which directly use the clustering model for classification. Several such approaches have been proposed for evolving data streams [52, 1, 55, 34]. The chunk-based approach in [52] also maintains a mixture of spherical clusters, denoted as pseudo points. This mixture is used as ensemble for the classification of new instances. For active learning, a combination of outlier selection and query by committee is used: instances outside all pseudo point ranges, or instances with high disagreement are selected for labelling and stored in a buffer. New clusters emerging in this buffer are then added to the ensemble. The semi-supervised stream classification approach proposed in [1] uses active learning solely to resolve ties due to votes from opposing classes in the ensemble. Thus, this approach uses active learning techniques for reducing the requested number of labels, but it provides no means for controlling its budget consumption. The active learning approach suggested in [55] differentiates between clusters with and without labelled instances. In the latter case, the algorithm seeks to obtain the label of the centre-most instance. In the former case, the algorithm checks for a skewed label distribution. That is, if all labels are located on one side of the cluster, an additional label at the opposite side of the cluster is requested. Subsequently, cluster with inhomogeneously distributed labels are split. The approach proposed in [34] addresses active learning for binary spam classification. It learns a separate clustering model for both classes. For new instances, each clustering model computes a similarity score. Similar to a likelihood, this score expresses the confidence that the instance belongs to the particular clustering model. When both clustering models indicate low but similar confidence, an instance is selected for labelling. Thus, the approach does not consider a predefined budget limit.

1.4.3 Loss Minimisation

In contrast to the information-theoretic uncertainty sampling, the decision-theoretic strategies summarised in [19, page 12] as *loss minimisation* aim at directly optimising a classifier's performance [72, page 37]. This is achieved by estimating the influence on the performance from acquiring a candidate's label.

The first strategy of this kind is *expected error reduction* [22, 62]. Here, the influence on the performance is measured by the expected reduction in the misclassification error. Expectation therein is over the different possible realisations of the candidate's label. For each possible realisation, the approach proposed in [62] simulates a classifier update and calculates the error reduction of this updated classifier compared to the current one. This Monte-Carlo-based approach is usable with any classifier and any user-specified classification performance measure, although computationally expensive. In [22], a closed-form solution is derived for mixtures of Gaussians. However, deriving such closed-form solutions is often not feasible. Thus, *variance reduction* has been proposed in [21, 22] as a variation of this strategy. Variance reduction aims at minimising the variance term, while ignoring the noise and bias terms. Thus, it allows closed-form solutions for some cases of regression problems, where such closed-form solutions are not available for expected error reduction [72, pages 40 and 64]. A drawback of loss-minimising strategies such as expected error reduction is the need for accurate posterior estimates in the expectation step. This is problematic when the posterior estimates from the current model are unreliable, as for example in early learning stages. This problem has been identified for example in [16]. Therein, different regularisation approaches have been explored to address this problem, including the use of Beta priors. A further drawback in expected error reduction is the use of en evaluation sample $\mathcal{V}$, on which the classifier's error after its simulated update is calculated. Different implementations for obtaining $\mathcal{V}$ are discussed in literature, using either the sample $\mathcal{L}$ of already labelled instances [62], or self-labelling the unlabelled candidate pool $\mathcal{U}$ [16], or a combination of the two. However, in any case the resulting evaluation sample $\mathcal{V}$ depends on the already acquired labels in $\mathcal{L}$. Thus, in early learning stages neither $\mathcal{L}$ nor $\mathcal{V}$ are reliable, as we have shown in [46]. In addition, this strategy's asymptotic time complexity is as high as $O(|\mathcal{V}| \cdot |\mathcal{U}|)$. Thus, according to [72, page 64], these strategies are not applicable in stream-based classification scenarios. This is underlined by the absence of papers reporting their use for classification in evolving data streams.

1.4.4 Sufficient Weights

The recently proposed strategy of computing so-called *sufficient weights* [12] is related to confidence-based uncertainty sampling, expected error reduction, and to probabilistic active learning discussed in Section 1.5. The sufficient weight is a proxy for the *certainty* of the classifier in a candidate's label. It is defined as the minimum weight this candidate would require, when added to the training set, in order to alter its own classification from one class to another. The authors propose to determine this sufficient weight by logarithmic search and simulation. Thus, the approach aims to consider the influence of an instance on the classifier given the current training set. This is related to the aims in expected error reduction and probabilistic active learning. However, this sufficient weight might be small for instances in regions of high Bayesian error, independently of the number of already acquired nearby labels. Thus, one might speculate that the sufficient weight strategy might request instances from very well-explored but noisy regions, in contrast to probabilistic active learning. However, this remains to be shown.

This approach is designed for evolving data streams and supposed to work with a given budget restriction. Similar to the budget management in [88], its selection threshold is increased in case an acquired instance did match the predicted label, and decreased otherwise.

1.5 Probabilistic Active Learning

The main contribution of this work is a novel probabilistic active learning strategy. Although originally presented for the static, pool-based scenario, this strategy has been designed for expandability to the non-stationary, stream-based scenario. Thus, the strategy aims to be computationally efficient, to be decision-theoretic in optimising directly a classification performance measure, and not to be limited to a particular classifier technology. In addition, it is not myopic but allows to consider multiple label acquisitions at once, and it is not limited to binary classification. Furthermore, this strategy is shown to be applicable in pool-based, stream-based and active class-conditional example acquisition scenarios.

The strategy and its main idea are presented in the next subsection for the pool-based scenario. This is then followed by subsections that discuss its use in other active learning scenarios.

1.5.1 The Probabilistic Active Learning Strategy

Probabilistic active learning is a decision-theoretic strategy that computes the expected gain in classification performance from labelling additional instances. Given a labelling candidate x, probabilistic active learning models the possible label realisations from labelling this (or m additional similar) instances as a multinomial random variable. Likewise, it models the true posterior in the labelling candidate's neighbourhood as another random variable. The expectation of the performance gain in the candidate's neighbourhood is then calculated over *both* random variables. Subsequently, this expected performance gain is weighted by the density of labelled and unlabelled instances in the candidate's neighbourhood. Thereby, the importance of the candidate's neighbourhood in the classification task is approximated. Finally, the candidate with the highest density-weighted expected performance gain is selected for labelling.

In contrast to probabilistic active learning, loss minimisation strategies consider the label realisation as only random variable in their expectation. Considering also the true posterior as a random variable advances the state-of-the-art decision-theoretic active learning literature, which considers solely the most likely (or most pessimistic) posterior value. In contrast to the information-theoretic uncertainty sampling, probabilistic active learning considers how well a candidate's neighbourhood has been explored. This is done by counting the number of already acquired labelled instances in the candidate's neighbourhood. As we have shown in [46], this provides theoretical as well as empirical advantages over other state-of-the-art active learning strategies.

The use of the concept of neighbourhoods in probabilistic active learning follows the *smoothness assumption* [17, page 7]. This assumes that neighbouring positions in the feature space have similar posteriors. In this work, two neighbourhood concepts are differentiated. The first corresponds to disjoint neighbourhoods and applies to categorical or pre-clustered data. Such data allows to count directly the number of labelled instances that are similar to the candidate w.r.t. their features (or assigned cluster). These label counts are then summarised by so-called *label statistics* $\mathit{ls} = (n, \hat{p})$. These are tuples consisting of the absolute number n of labels in a candidate's neighbourhood, and the share $\hat{p}$ of positives therein. The second concept corresponds to smooth, continuous neighbourhoods. It is applicable to numerical data, where the influence of instances increases with the similarity of their features. In analogy to counts in the disjoint case, frequency estimates are used in the smooth, continuous case. These might be obtained from probabilistic classifiers that are modified to return unnormalised estimates for the absolute frequencies. This is shown to work particularly well for generative probabilistic classifiers such as Naive Bayes. If such estimates are not available from the classifier, frequency estimation techniques might be used, e.g. those based on kernels.

For calculating the expected performance gain, a numerical integration approach is proposed for optimising any user-specified point classification performance measure [58], e.g. accuracy. In addition, a faster closed-form solution is derived for misclassification loss [37]. This is a performance measure for cost-sensitive classification, where the cost τ of a false positive classification potentially differs from that of a false negative one $(1-\tau)$. As a result, probabilistic active learning is also applicable in cost-sensitive classification.

To illustrate this approach, consider the case of binary classification with a given unlabelled instance x from a pool of labelling candidates $\mathcal{U}$, a set of labelled instances $\mathcal{L} = \{(x_i, y_i)\}$, a labelling budget m, and a costs τ for a false positive classification.

First, the label statistics are acquired for the candidate's neighbourhood. Using Gaussian kernel frequency estimation with bandwidth matrix Σ yields:

$$LC(x,\mathcal{L}) \approx KFE(x,\mathcal{L}) = \sum_{x_i \in \mathcal{L}} \exp\left(-\frac{1}{2} \cdot (x-x_i)'\Sigma^{-1}(x-x_i)\right) \tag{1.1}$$

Based on such frequency estimates, the total number of labels is derived as $n = LC(x,\mathcal{L})$. For the label statistics $ls = (n, \hat{p})$, the share of positives therein is $\hat{p} = LC(x,\mathcal{L}_+)/LC(x,\mathcal{L})$. Here, $\mathcal{L}_+$ is the subset of labelled positive instances in $\mathcal{L}$.

Since $\mathcal{L} \cup \mathcal{U}$ is static, it is recommended to precompute the densities in the neighbourhood d_x of all candidates $x \in \mathcal{U}$. For example, by calculating this density based on Eq. 1.1:

$$d_x = \frac{LC(x,\mathcal{L}\cup\mathcal{U})}{|\mathcal{L}\cup\mathcal{U}|} \tag{1.2}$$

Given the label statistics of a candidate, the expected gain G_{OPAL} in the candidate's neighbourhood is computed as:

$$G_{\text{OPAL}}(n,\hat{p},\tau,m) = \frac{n+1}{m} \cdot \binom{n}{n\cdot\hat{p}} \cdot \left(\mathrm{I}_{ML}(n,\hat{p},\tau,0,0) - \sum_{k=0}^{m} \mathrm{I}_{ML}(n,\hat{p},\tau,m,k)\right) \tag{1.3}$$

Here, $\mathrm{Bin}_{m,p}(k)$ is the generalised binomial coefficient for non-integer arguments using Legendre's gamma function $\Gamma(z)$[7]:

$$\binom{m}{k} = \frac{\Gamma(m+1)}{\Gamma(k+1)\cdot\Gamma(m-k+1)} \tag{1.4}$$

[7]See for example pages 206–208 in [60].

The function $\mathrm{I}_{ML}(n, \hat{p}, \tau, m, k)$ is proportional to the expected misclassification loss within the neighbourhood, given that among m additionally acquired labels k are positive:

$$\mathrm{I}_{ML}(n, \hat{p}, \tau, m, k) = \binom{m}{k} \cdot \begin{cases} (1-\tau) \cdot \frac{\Gamma(1-k+m+n-n\hat{p})\Gamma(2+k+n\hat{p})}{\Gamma(3+m+n)} & \frac{n\hat{p}+k}{n+m} < \tau \\ (\tau-\tau^2) \cdot \frac{\Gamma(1-k+m+n-n\hat{p})\Gamma(1+k+n\hat{p})}{\Gamma(2+m+n)} & \frac{n\hat{p}+k}{n+m} = \tau \\ \tau \cdot \frac{\Gamma(2-k+m+n-n\hat{p})\Gamma(1+k+n\hat{p})}{\Gamma(3+m+n)} & \frac{n\hat{p}+k}{n+m} > \tau \end{cases} \tag{1.5}$$

For $m > 1$, the value m_x^* maximising the G_{OPAL} (see Eq. 1.3) is determined by using, for example, a *logarithmic search* over $m' = 1, 2, \cdots, m$:

$$m_x^* \leftarrow \underset{m' \in 1,2,\cdots,m}{\arg\max}\ G_{\mathrm{OPAL}}((n_x, \hat{p}_x), \tau, m') \tag{1.6}$$

Finally, the candidate x^* with the highest density-weighted expected performance gain is requested:

$$x^* \leftarrow \underset{x \in \mathcal{U}}{\arg\max}\,(d_x \cdot G_{\mathrm{OPAL}}((n_x, \hat{p}_x), \tau, m_x^*)) \tag{1.7}$$

1.5.2 Probabilistic Active Learning in Evolving Data Streams

In this work, two different approaches for using the probabilistic active learning strategy in evolving data streams are shown.

The first, instance-wise approach is presented in [42]. In addition to extending the probabilistic active learning strategy for the use in evolving data streams, it complements the notion of usefulness within a topological space ("spatial usefulness") with the concept of "temporal usefulness". Furthermore, it addresses budget management by proposing the Balanced Incremental Quantile Filter (BIQF). This filtering technique is used for assessing the usefulness of instances in a sliding window and for ensuring that a predefined budget is not exceeded above a given tolerance.

The second approach, presented in [42], is based on the concept of disjoint neighbourhoods. These are obtained from clustering the data in a chunk-based approach. This requires adjusting the calculation of the label statistics and the density estimates. Furthermore, a framework for clustering-based active learning in evolving data streams is presented, which is partially inspired by [39].

1.5.3 Probabilistic Active Learning for Active Class-Conditional Example Acquisition

Probabilistic active learning is not limited to pool-based and stream-based active label acquisition. A further scenario addressed in this work is active class-conditional example acquisition, also known as active class selection. In [43], probabilistic active learning is extended to this scenario. The proposed algorithm uses pseudo instances to estimate in expectation the classifier's benefit from additional information. For yielding the final class selection score, this expected value is then weighted with the pseudo instance's density and its class conditional probability.

Bibliography

[1] Zahraa S. Abdallah, Mohamed Gaber, Bala Srinivasan, and Shonali Krishnaswamy. Streamar: Incremental and active learning with evolving sensory data for activity recognition. In *Proceedings of the 24th IEEE International Conference on Tools with Artificial Intelligence*, 2012.

[2] Charu C. Aggarwal, Xiangnan Kong, Quanquan Gu, Jiawei Han, and Philip S Yu. Active learning: A survey. In Charu C. Aggarwal, editor, *Data Classification: Algorithms and Applications*, page 571–605. CRC Press, 2014.

[3] Ethem Alpaydin. *Introduction to Machine Learning.* Adaptive Computation and Machine Learning. The MIT Press, 2004.

[4] Dana Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[5] Andrew Arnold, Ramesh Nallapati, and William W. Cohen. A comparative study of methods for transductive transfer learning. In *Proceedings of the Seventh IEEE International Conference on Data Mining Workshops*, ICDMW '07, pages 77–82, Washington, DC, USA, 2007. IEEE Computer Society.

[6] Josh Attenberg and Seyda Ertekin. Class imbalance and active learning. In Haibo He and Yunqian Ma, editors, *Imbalanced Learning: Foundations, Algorithms, and Applications*, pages 101–150. IEEE, 2013.

[7] Josh Attenberg, Prem Melville, Foster Provost, and Maytal Saar-Tsechansky. Selective data acquisition for machine learning. In Balaji Krishnapuram, Shipeng Yu, and R. Bharat Rao, editors, *Cost-Sensitive Machine Learning.* CRC Press, Boca Raton, FL, USA, 1st edition, 2011.

[8] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, pages 1–16, New York, NY, USA, 2002. ACM.

[9] Thomas Bayes. An essay towards solving a problem in the doctrine of chance. *Philosophical Transactions of the Royal Society of London*, 53(0), 1763. Communicated by Richard Price in a letter to John Canton.

[10] Alexis Bondu, Vincent Lemaire, and Marc Boullé. Exploration vs. exploitation in active learning : A Bayesian approach. In *International*

Joint Conference on Neural Networks, IJCNN 2010, Barcelona, Spain, 18-23 July, 2010, pages 1–7. IEEE, 2010.

[11] Charles C. Bonwell and James A. Eison. Active learning: Creating excitement in the classroom. *ASHE-ERIC Higher Education Report*, 1, 1991.

[12] Mohamed-Rafik Bouguelia, Yolande Belaïd, and Abdel Belaïd. An adaptive streaming active learning strategy based on instance weighting. *Pattern Recognition Letters*, 70:38–44, 2016.

[13] Nicolas Cebron and Michael R. Berthold. Active learning for object classification: from exploration to exploitation. *Data Mining and Knowledge Discovery*, 18:283–299, 2009.

[14] Shayok Chakraborty, Vineeth Balasubramanian, and Sethuraman Panchanathan. Adaptive batch mode active learning. *IEEE Transactions on Neural Networks and Learning Systems*, 26(8), 2014.

[15] Shayok Chakraborty, Vineeth Nallure Balasubramanian, and Sethuraman Panchanathan. Optimal batch selection for active learning in multi-label classification. In K. Selçuk Candan, Sethuraman Panchanathan, Balakrishnan Prabhakaran, Hari Sundaram, Wu-Chi Feng, and Nicu Sebe, editors, *Proceedings of the 19th International Conference on Multimedia 2011, Scottsdale, AZ, USA, November 28 - December 1, 2011*, pages 1413–1416. ACM, 2011.

[16] Olivier Chapelle. Active learning for parzen window classifier. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 49–56, 2005.

[17] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. *Semi-supervised Learning.* MIT Press, 2006.

[18] Yunfei Chu and Jürgen Hahn. Optimal experiment design. In Werner Dubitzky, Olaf Wolkenhauer, Kwang-Hyun Cho, and Hiroki Yokota, editors, *Encyclopedia of Systems Biology*, pages 1572–1573. Springer New York, New York, NY, 2013.

[19] David Cohn. Active learning. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, pages 10–14. Springer, 2010.

[20] David Cohn, L. Atlas, R. Ladner, M. El-Sharkawi, R. II Marks, M. Aggoune, and D. Park. Training connectionist networks with queries and selective sampling. In *Advances in Neural Information Processing Systems (NIPS).* Morgan Kaufmann, 1990.

[21] David A. Cohn. Neural network exploration using optimal experiment design. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems 6, [7th NIPS Conference, Denver, Colorado, USA, 1993]*, pages 679–686. Morgan Kaufmann, 1993.

[22] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[23] Rocco De Rosa. *Confidence-based and Nonparametric Classification: Applications to Classification, Online and Active Learning.* PhD thesis, Universita Degli Studi Di Milano, 2014.

[24] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD00)*, KDD '00, pages 71–80. ACM, 2000.

[25] Ran El-Yaniv and Yair Wiener. Active learning via perfect selective classification. *J. Mach. Learn. Res.*, 13:255–279, March 2012.

[26] Wei Fan and Albert Bifet. Mining big data: Current status, and forecast to the future. *SIGKDD Explor. Newsl.*, 14(2):1–5, April 2013.

[27] Wei Fan, Yi-an Huang, Haixun Wang, and Philip S. Yu. Active mining of data streams. In *Proceedings of the 4th SIAM International Conference on Data Mining, SDM 2004, USA*, pages 457—461, 2004.

[28] Valerii V. Fedorov. *Theory of Optimal Experiments Design.* Academic Press, 1972.

[29] Yifan Fu, Xingquan Zhu, and Bin Li. A survey on instance selection for active learning. *Knowledge and Information Systems*, 35(2):249–283, 2012.

[30] João Gama. *Knowledge Discovery from Data Streams.* Chapman and Hall, 2010.

[31] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90:317–346, 2013.

[32] João Gama, Indrė Zliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):1–44, 2014.

[33] John Gantz and David Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east, December 2012.

[34] Kleanthi Georgala, Aris Kosmopoulos, and George Paliouras. Spam filtering: An active learning approach using incremental clustering. In *Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14)*, WIMS '14, pages 23:1–23:12, New York, NY, USA, 2014. ACM.

[35] Russell Greiner, Adam J. Grove, and Dan Roth. Learning cost-sensitive active classifiers. *Artificial Intelligence*, 139(2):137–174, 2002.

[36] Isabelle Guyon, Gavin Cawley, Gideon Dror, Vincent Lemaire, and Alexander Statnikov, editors. *Active Learning Challenge*, volume 6 of *Challenges in Machine Learning.* Microtome Publishing, 2011.

[37] David J. Hand. Measuring classifier performance: a coherent alternative to the area under the roc curve. *Machine Learning*, 77(1):103–123, 2009.

[38] Shucheng Huang and Yisheng Dong. An active learning system for mining time-changing data streams. *Intelligent Data Analysis*, 11(4):401–419, 2007.

[39] Dino Ienco, Albert Bifet, Indrė Zliobaitė, and Bernhard Pfahringer. Clustering based active learning for evolving data streams. In Johannes Fürnkranz, Eyke Hüllermeier, and Tomoyuki Higuchi, editors, *Proceedings of the 16th Int. Conf. on Discovery Science (DS), Singapore*, volume 8140 of *Lecture Notes in Artificial Intelligence*, pages 79–93. Springer, 2013.

[40] Dino Ienco, Bernhard Pfahringer, and Indrė Zliobaitė. High density-focused uncertainty sampling for active learning over evolving stream data. In *Proceedings of the 3rd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, pages 133–148, 2014.

[41] Ross D. King, Kenneth E. Whelan, Ffion M. Jones, Philip G. K. Reiser, Christopher H. Bryant, Stephen H. Muggleton, Douglas B. Kell, and Stephen G. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427:247–252, 2004.

[42] Daniel Kottke, Georg Krempl, and Myra Spiliopoulou. Probabilistic active learning in data streams. In Elisa Fromont, Tijl De Bie, and Matthijs van Leeuwen, editors, *Advances in Intelligent Data Analysis XIV*, volume 9385 of *LNCS*, pages 145–157. Springer, 2015.

[43] Daniel Kottke, Georg Krempl, Marianne Stecklina, Cornelius Styp von Rekowski, Tim Sabsch, Tuan Pham Minh, Matthias Deliano, Myra Spiliopoulou, and Bernhard Sick. Probabilistic active learning for active class selection. In Kory Mathewson, Kaushik Subramanian, and Robert Loftin, editors, *Proc. of the NIPS Workshop on the Future of Interactive Learning Machines*, 2016.

[44] Georg Krempl, Tuan Cuong Ha, and Myra Spiliopoulou. Clustering-based optimised probabilistic active learning (COPAL). In Nathalie Japkowicz and Stan Matwin, editors, *Proc. of the 18th Int. Conf. on Discovery Science (DS 2015)*, volume 9356 of *LNCS*, pages 101–115. Springer, 2015.

[45] Georg Krempl, Daniel Kottke, and Vincent Lemaire. Optimised probabilistic active learning (OPAL) for fast, non-myopic, cost-sensitive active classification. *Machine Learning*, 100(2), 2015.

[46] Georg Krempl, Daniel Kottke, and Myra Spiliopoulou. Probabilistic active learning: Towards combining versatility, optimality and efficiency. In Saso Dzeroski, Pance Panov, Dragi Kocev, and Ljupco Todorovski, editors, *Proceedings of the 17th Int. Conf. on Discovery Science (DS), Bled*, volume 8777 of *Lecture Notes in Computer Science*, pages 168–179. Springer, 2014.

[47] Douglas Laney. 3-d data management: Controlling data volume, velocity and variety. Technical report, META Group Research Note, February 6 2001.

[48] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, pages 3–12, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[49] Patrick Lindstrom, Sarah Jane Delany, and Brian Mac Namee. Handling concept drift in a text data stream constrained by high labelling cost. In *Proceedings of the 23rd Int. Florida Artificial Intelligence Research Society Conference (FLAIRS 2010)*, pages 32–37, 2010.

[50] Alexander Yun-chung Liu. *Active learning in cost-sensitive environments.* PhD thesis, University of Texas, Electrical and Computer Engineering, 2009.

[51] Rachel Lomasky. *Active Acquisition of Informative Training Data.* PhD thesis, Tufts University, 2010.

[52] Mohammad M. Masud, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani Thuraisingham. Classification and novel class detection in data streams with active mining. In *Proceedings of the 14th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining - Volume Part II*, PAKDD 2010, pages 311–324, Berlin, Heidelberg, 2010. Springer-Verlag.

[53] Joel Michael. Where's the evidence that active learning works? *Advances in Physiology Education*, 30(4):159–167, 2006.

[54] Tom M. Mitchell. *Machine Learning.* McGraw Hill, 1st edition, 1997.

[55] Hai-Long Nguyen, Wee-Keong Ng, and Yew-Kwong Woon. Concurrent semi-supervised learning with active learning of data streams. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems VIII*, pages 113–136. Springer, 2013.

[56] Thomas Osugi, Deng Kim, and Stephen Scott. Balancing exploration and exploitation: A new algorithm for active machine learning. In *Data Mining, Fifth IEEE International Conference on*, pages 8–pp. IEEE, 2005.

[57] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[58] Charles Parker. An analysis of performance measures for binary classifiers. In *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM2011)*, pages 517 – 526. IEEE, 2011.

[59] Edoardo Pasolli. *Active learning methods for classification and regression problems.* PhD thesis, University of Trento, 2011.

[60] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in Fortran 77: The Art of Scientific Computing.* Cambridge University Press, 2 edition, 1992.

[61] Jens Röder. *Active Learning: New Approaches, and Industrial Applications.* PhD thesis, University of Heidelberg, 2013.

[62] Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proc. of the 18th Int. Conf. on Machine Learning, ICML 2001, Williamstown, MA, USA*, pages 441–448, San Francisco, CA, USA, 2001. Morgan Kaufmann.

[63] Ritchey A. Ruff and Thomas Dietterich. What good are experiments? In *Proc. of the sixth int. workshop on machine learning*, 1989.

[64] Joung Woo Ryu, Mehmed M Kantardzic, Myung-Won Kim, and A Ra Khil. An efficient method of building an ensemble of classifiers in streaming data. In *Big Data Analytics*, pages 122–133. Springer, 2012.

[65] Maytal Saar-Tsechansky, Prem Melville, and Foster Provost. Active feature-value acquisition. *Management Science*, 55(4):664–684, April 2009.

[66] Claude Sammut and Geoffrey I. Webb, editors. *Supervised Learning*, page 941. Springer, 2010.

[67] Andrew I. Schein and Lyle H. Ungar. Active learning for logistic regression: an evaluation. *Machine Learning*, 68(3):235–265, 2007.

[68] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015.

[69] Raquel Sebastião and João Gama. A study on change detection methods. In *Proceedings of the 4th Portuguese Conf. on Artificial Intelligence, Lisbon*, 2009.

[70] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin-Madison, Madison, Wisconsin, USA, 2009.

[71] Burr Settles. From theories to queries: Active learning in practice. In Isabelle Guyon, Gavin Cawley, Gideon Dror, Vincent Lemaire, and Alexander Statnikov, editors, *Active Learning and Experimental Design workshop*, volume 16 of *JMLR Workshop and Conference Proceedings*, pages 1–18. JMLR, 2010.

[72] Burr Settles. *Active Learning.* Number 18 in Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers, 2012.

[73] H. Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In Warmuth M.K. and Valiant L.G., editors, *Proc. of the fifth workshop on computational learning theory.* Morgan Kaufmann, 1992.

[74] Jasmina Smailović, Miha Grčar, Nada Lavrač, and Martin Žnidaršič. Stream-based active learning for sentiment analysis in the financial domain. *Information Sciences*, 285:181–203, 2014.

[75] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[76] Katrin Tomanek. *Resource-aware annotation through active learning.* PhD thesis, Technische Universität Dortmund, 2010.

[77] Katrin Tomanek and Katharina Morik. Inspecting sample reusability for active learning. In Isabelle Guyon, Gavin C. Cawley, Gideon Dror, Vincent Lemaire, and Alexander R. Statnikov, editors, *AISTATS workshop on Active Learning and Experimental Design*, volume 16 of *JMLR Proceedings*, pages 169–181. JMLR.org, 2011.

[78] Eva Weigl, Wolfgang Heidl, Edwin Lughofer, Thomas Radauer, and Christian Eitzinger. On improving performance of surface inspection systems by online active learning and flexible classifier updates. *Machine Vision and Applications*, 27(1):103–127, 2015.

[79] Michał Woźniak, Bogusław Cyganek, Andrzej Kasprzak, Paweł Ksieniewicz, and Krzysztof Walkowiak. Active learning classifier for streaming data. In Francisco Martínez-Álvarez, Alicia Troncoso, Héctor Quintián, and Emilio Corchado, editors, *Proc. of the 11th Int. Conf. on Hybrid Artificial Intelligent Systems*, pages 186–197. Springer International Publishing, 2016.

[80] Liu Yang. Active learning with a drifting distribution. *Neural Information Processing Systems*, 2011.

[81] Jingbo Zhu, Huizhen Wang, Benjamin K. Tsou, and Matthew Y. Ma. Active learning with sampling by uncertainty and density for data annotations. *IEEE Transactions on Audio, Speech & Language Processing*, 18(6):1323–1331, 2010.

[82] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, University of Wisconsin, 2008.

[83] Xiaojin Zhu. Semi-supervised learning. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, page 892ff. Springer, 2010.

[84] Xingquan Zhu, Peng Zhang, Xiaodong Lin, and Yong Shi. Active learning from data streams. In *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, ICDM '07, pages 757–762, Washington, DC, USA, 2007. IEEE Computer Society.

[85] Xingquan Zhu, Peng Zhang, Xiaodong Lin, and Yong Shi. Active learning from stream data using optimal weight classifier ensemble. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 40(6):1607 – 1621, 2010.

[86] Indrė Zliobaitė. Change with delayed labeling: When is it detectable? In *IEEE International Conference on Data Mining Workshops (ICDMW 2010)*, pages 843 – 850, 2010.

[87] Indrė Zliobaitė, Albert Bifet, Bernhard Pfahringer, and Geoffrey Holmes. Active learning with evolving streaming data. In *Proceedings of the 21st European Conference on Machine Learning and Principles and Practice*

of Knowledge Discovery in Databases (ECML PKDD'11), volume 6913 of *Lecture Notes in Computer Science*, pages 597–612. Springer, 2011.

[88] Indrė Zliobaitė, Albert Bifet, Bernhard Pfahringer, and Geoffrey Holmes. Active learning with drifting streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):27–39, 2013.

Chapter 2

Open Challenges for Data Stream Mining Research

The following chapter has been published as:

Open Challenges for Data Stream Mining Research
Georg Krempl, Indrė Zliobaitė, Dariusz Brzeziński, Eyke Hüllermeier, Mark Last, Vincent Lemaire, Tino Noack, Ammar Shaker, Sonja Sievi, Myra Spiliopoulou, and Jerzy Stefanowski.

ACM SIGKDD Explorations Newsletter, Special Issue on Big Data, volume 16(1) 2014, pp. 1-10, ACM.
DOI: `https://doi.org/10.1145/2674026.2674028`

Open Challenges for Data Stream Mining Research

Georg Krempl
University Magdeburg, Germany
georg.krempl@iti.cs.uni-magdeburg.de

Indre Žliobaite
Aalto University and HIIT, Finland
indre.zliobaite@aalto.fi

Dariusz Brzeziński
Poznan U. of Technology, Poland
dariusz.brzezinski@cs.put.poznan.pl

Eyke Hüllermeier
University of Paderborn, Germany
eyke@upb.de

Mark Last
Ben-Gurion U. of the Negev, Israel
mlast@bgu.ac.il

Vincent Lemaire
Orange Labs, France
vincent.lemaire@orange.com

Tino Noack
TU Cottbus, Germany
noacktin@tu-cottbus.de

Ammar Shaker
University of Paderborn, Germany
ammar.shaker@upb.de

Sonja Sievi
Astrium Space Transportation, Germany
sonja.sievi@astrium.eads.net

Myra Spiliopoulou
University Magdeburg, Germany
myra@iti.cs.uni-magdeburg.de

Jerzy Stefanowski
Poznan U. of Technology, Poland
jerzy.stefanowski@cs.put.poznan.pl

ABSTRACT

Every day, huge volumes of sensory, transactional, and web data are continuously generated as streams, which need to be analyzed online as they arrive. Streaming data can be considered as one of the main sources of what is called big data. While predictive modeling for data streams and big data have received a lot of attention over the last decade, many research approaches are typically designed for well-behaved controlled problem settings, overlooking important challenges imposed by real-world applications. This article presents a discussion on eight open challenges for data stream mining. Our goal is to identify gaps between current research and meaningful applications, highlight open problems, and define new application-relevant research directions for data stream mining. The identified challenges cover the full cycle of knowledge discovery and involve such problems as: protecting data privacy, dealing with legacy systems, handling incomplete and delayed information, analysis of complex data, and evaluation of stream mining algorithms. The resulting analysis is illustrated by practical applications and provides general suggestions concerning lines of future research in data stream mining.

1. INTRODUCTION

The volumes of automatically generated data are constantly increasing. According to the Digital Universe Study [18], over 2.8ZB of data were created and processed in 2012, with a projected increase of 15 times by 2020. This growth in the production of digital data results from our surrounding environment being equipped with more and more sensors. People carrying smart phones produce data, database transactions are being counted and stored, streams of data are extracted from virtual environments in the form of logs or user generated content. A significant part of such data is volatile, which means it needs to be analyzed in real time as it arrives. Data stream mining is a research field that studies methods and algorithms for extracting knowledge from volatile streaming data [14; 5; 1]. Although data streams, online learning, big data, and adaptation to concept drift have become important research topics during the last decade, truly autonomous, self-maintaining, adaptive data mining systems are rarely reported. This paper identifies real-world challenges for data stream research that are important but yet unsolved. Our objective is to present to the community a position paper that could inspire and guide future research in data streams. This article builds upon discussions at the International Workshop on Real-World Challenges for Data Stream Mining (RealStream)[1] in September 2013, in Prague, Czech Republic.

Several related position papers are available. Dietterich [10] presents a discussion focused on predictive modeling techniques, that are applicable to streaming and non-streaming data. Fan and Bifet [12] concentrate on challenges presented by large volumes of data. Zliobaite et al. [48] focus on concept drift and adaptation of systems during online operation. Gaber et al. [13] discuss ubiquitous data mining with attention to collaborative data stream mining. In this paper, we focus on research challenges for streaming data inspired and required by real-world applications. In contrast to existing position papers, we raise issues connected not only with large volumes of data and concept drift, but also such practical problems as privacy constraints, availability of information, and dealing with legacy systems.

The scope of this paper is not restricted to algorithmic challenges, it aims at covering the full cycle of knowledge discovery from data (CRISP [40]), from understanding the context of the task, to data preparation, modeling, evaluation, and deployment. We discuss eight challenges: making models simpler, protecting privacy and confidentiality, dealing with legacy systems, stream preprocessing, timing and availability of information, relational stream mining, analyzing event data, and evaluation of stream mining algorithms. Figure 1 illustrates the positioning of these challenges in the CRISP cycle. Some of these apply to traditional (non-streaming) data mining as well, but they are critical in streaming environments. Along with further discussion of these challenges, we present our position where the forthcoming focus of research and development efforts should be directed to address these challenges.

In the remainder of the article, section 2 gives a brief introduction to data stream mining, sections 3–7 discuss each identified challenge, and section 8 highlights action points for future research.

[1]http://sites.google.com/site/realstream2013

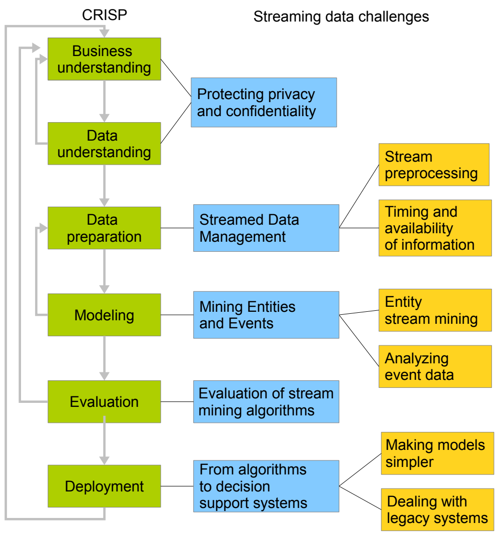

CRISP
Streaming data challenges
Business understanding
Data understanding
Data preparation
Modeling
Evaluation
Deployment
Protecting privacy and confidentiality
Streamed Data Management
Mining Entities and Events
Evaluation of stream mining algorithms
From algorithms to decision support systems
Stream preprocessing
Timing and availability of information
Entity stream mining
Analyzing event data
Making models simpler
Dealing with legacy systems

Figure 1: CRISP cycle with data stream research challenges.

2. DATA STREAM MINING

Mining big data streams faces three principal challenges: *volume*, *velocity*, and *volatility*. Volume and velocity require a high volume of data to be processed in limited time. Starting from the first arriving instance, the amount of available data constantly increases from zero to potentially infinity. This requires incremental approaches that incorporate information as it becomes available, and online processing if not all data can be kept [15]. Volatility, on the other hand, corresponds to a dynamic environment with ever-changing patterns. Here, old data is of limited use, even if it could be saved and processed again later. This is due to change, that can affect the induced data mining models in multiple ways: *change of the target variable*, *change in the available feature information*, and *drift*.

Changes of the target variable occur for example in credit scoring, when the definition of the classification target "default" versus "non-default" changes due to business or regulatory requirements. Changes in the available feature information arise when new features become available, e.g. due to a new sensor or instrument. Similarly, existing features might need to be excluded due to regulatory requirements, or a feature might change in its scale, if data from a more precise instrument becomes available. Finally, drift is a phenomenon that occurs when the distributions of features x and target variables y change in time. The challenge posed by drift has been subject to extensive research, thus we provide here solely a brief categorization and refer to recent surveys like [17].

In supervised learning, drift can affect the posterior $P(y|x)$, the conditional feature $P(x|y)$, the feature $P(x)$ and the class prior $P(y)$ distribution. The distinction based on which distribution is assumed to be affected, and which is assumed to be static, serves to assess the suitability of an approach for a particular task. It is worth noting, that the problem of changing distributions is also present in unsupervised learning from data streams.

A further categorization of drift can be made by:

- **smoothness of concept transition**: Transitions between concepts can be sudden or gradual. The former is sometimes also denoted in literature as shift or abrupt drift.
- **singular or recurring contexts**: In the former case, a model becomes obsolete once and for all when its context is replaced by a novel context. In the latter case, a model's context might reoccur at a later moment in time, for example due to a business cycle or seasonality, therefore, obsolete models might still regain value.
- **systematic or unsystematic**: In the former case, there are patterns in the way the distributions change that can be exploited to predict change and perform faster model adaptation. Examples are subpopulations that can be identified and show distinct, trackable evolutionary patterns. In the latter case, no such patterns exist and drift occurs seemingly at random. An example for the latter is fickle concept drift.
- **real or virtual**: While the former requires model adaptation, the latter corresponds to observing outliers or noise, which should not be incorporated into a model.

Stream mining approaches in general address the challenges posed by volume, velocity and volatility of data. However, in real-world applications these three challenges often coincide with other, to date insufficiently considered ones.

The next sections discuss eight identified challenges for data stream mining, providing illustrations with real world application examples, and formulating suggestions for forthcoming research.

3. PROTECTING PRIVACY AND CONFIDENTIALITY

Data streams present new challenges and opportunities with respect to protecting privacy and confidentiality in data mining. Privacy preserving data mining has been studied for over a decade (see. e.g. [3]). The main objective is to develop such data mining techniques that would not uncover information or patterns which compromise confidentiality and privacy obligations. Modeling can be done on original or anonymized data, but when the model is released, it should not contain information that may violate privacy or confidentiality. This is typically achieved by controlled distortion of sensitive data by modifying the values or adding noise.

Ensuring privacy and confidentiality is important for gaining trust of the users and the society in autonomous, stream data mining systems. While in offline data mining a human analyst working with the data can do a sanity check before releasing the model, in data stream mining privacy preservation needs to be done online. Several existing works relate to privacy preservation in publishing streaming data (e.g. [46]), but no systematic research in relation to broader data stream challenges exists.

We identify two main challenges for privacy preservation in mining data streams. The first challenge is incompleteness of information. Data arrives in portions and the model is updated online. Therefore, the model is never final and it is difficult to judge privacy preservation before seeing all the data. For example, suppose GPS traces of individuals are being collected for modeling traffic situation. Suppose person A at current time travels from the campus to the airport. The privacy of a person will be compromised, if there are no similar trips by other persons in the very near future. However, near future trips are unknown at the current time, when the model needs to be updated.

On the other hand, data stream mining algorithms may have some inherent privacy preservation properties due to the fact that they do not need to see all the modeling data at once, and can be incrementally updated with portions of data. Investigating privacy preservation properties of existing data stream algorithms makes another interesting direction for future research.

The second important challenge for privacy preservation is concept drift. As data may evolve over time, fixed privacy preservation rules may no longer hold. For example, suppose winter comes, snow falls, and much less people commute by bike. By knowing that a person comes to work by bike and having a set of GPS traces, it may not be possible to identify this person uniquely in summer, when there are many cyclists, but possible in winter. Hence, an important direction for future research is to develop adaptive privacy preservation mechanisms, that would diagnose such a situation and adapt themselves to preserve privacy in the new circumstances.

4. STREAMED DATA MANAGEMENT

Most of the data stream research concentrates on developing predictive models that address a simplified scenario, in which data is *already pre-processed*, *completely* and *immediately available* for *free*. However, successful business implementations depend strongly on the alignment of the used machine learning algorithms with both, the business objectives, and the available data. This section discusses often omitted challenges connected with streaming data.

4.1 Streamed Preprocessing

Data preprocessing is an important step in all real world data analysis applications, since data comes from complex environments, may be noisy, redundant, contain outliers and missing values. Many standard procedures for preprocessing offline data are available and well established, see e.g. [33]; however, the data stream setting introduces new challenges that have not received sufficient research attention yet.

While in traditional offline analysis data preprocessing is a once-off procedure, usually done by a human expert prior to modeling, in the streaming scenario manual processing is not feasible, as new data continuously arrives. Streaming data needs fully automated preprocessing methods, that can optimize the parameters and operate autonomously. Moreover, preprocessing models need to be able to update themselves automatically along with evolving data, in a similar way as predictive models for streaming data do. Furthermore, all updates of preprocessing procedures need to be synchronized with the subsequent predictive models, otherwise after an update in preprocessing the data representation may change and, as a result, the previously used predictive model may become useless.

Except for some studies, mainly focusing on feature construction over data streams, e.g. [49; 4], no systematic methodology for data stream preprocessing is currently available.

As an illustrative example for challenges related to data preprocessing, consider predicting traffic jams based on mobile sensing data. People using navigation services on mobile devices can opt to send anonymized data to the service provider. Service providers, such as Google, Yandex or Nokia, provide estimations and predictions of traffic jams based on this data. First, the data of each user is mapped to the road network, the speed of each user on each road segment of the trip is computed, data from multiple users is aggregated, and finally the current speed of the traffic is estimated.

There are a lot of data preprocessing challenges associated with this task. First, *noisiness* of GPS data might *vary* depending on location and load of the telecommunication network. There may be *outliers*, for instance, if somebody stopped in the middle of a segment to wait for a passenger, or a car broke. The number of pedestrians using mobile navigation may vary, and require *adaptive instance selection*. Moreover, road networks may change over time, leading to changes in average speeds, in the number of cars and even car types (e.g. heavy trucks might be banned, new optimal routes emerge). All these issues require automated preprocessing actions before feeding the newest data to the predictive models.

The problem of preprocessing for data streams is challenging due to the challenging nature of the data (continuously arriving and evolving). An analyst cannot know for sure, what kind of data to expect in the future, and cannot deterministically enumerate possible actions. Therefore, not only models, but also the procedure itself needs to be fully automated.

This research problem can be approached from several angles. One way is to look at existing predictive models for data streams, and try to integrate them with selected data preprocessing methods (e.g. feature selection, outlier definition and removal).

Another way is to systematically characterize the existing offline data preprocessing approaches, try to find a mapping between those approaches and problem settings in data streams, and extend preprocessing approaches for data streams in such a way as traditional predictive models have been extended for data stream settings.

In either case, developing individual methods and methodology for preprocessing of data streams would bridge an important gap in the practical applications of data stream mining.

4.2 Timing and Availability of Information

Most algorithms developed for evolving data streams make simplifying assumptions on the timing and availability of information. In particular, they assume that information is *complete*, *immediately available*, and *received passively and for free*. These assumptions often do not hold in real-world applications, e.g., patient monitoring, robot vision, or marketing [43]. This section is dedicated to the discussion of these assumptions and the challenges resulting from their absence. For some of these challenges, corresponding situations in offline, static data mining have already been addressed in literature. We will briefly point out where a mapping of such known solutions to the online, evolving stream setting is easily feasible, for example by applying windowing techniques. However, we will focus on problems for which no such simple mapping exists and which are therefore open challenges in stream mining.

4.2.1 Handling Incomplete Information

Completeness of information assumes that the true values of all variables, that is of features and of the target, are revealed eventually to the mining algorithm.

The problem of missing values, which corresponds to incompleteness of features, has been discussed extensively for the offline, static settings. A recent survey is given in [45]. However, only few works address data streams, and in particular *evolving* data streams. Thus several open challenges remain, some are pointed out in the review by [29]: how to address the problem that the frequency in which missing values occur is unpredictable, but largely affects the quality of imputations? How to (automatically) select the best imputation technique? How to proceed in the trade-off between speed and statistical accuracy?

Another problem is that of missing values of the target variable. It has been studied extensively in the static setting as semi-supervised learning (SSL, see [11]). A requirement for applying SSL techniques to streams is the availability of at least some labeled data from the most recent distribution. While first attempts to this problem have been made, e.g. the online manifold regularization approach in [19] and the ensembles-based approach suggested by [11], improvements in speed and the provision of performance guarantees remain open challenges. A special case of incomplete information is “censored data” in Event History Analysis (EHA), which is described in section 5.2. A related problem discussed below is active learning (AL, see [38]).

4.2.2 Dealing with Skewed Distributions

Class imbalance, where the class prior probability of the minority class is small compared to that of the majority class, is a frequent problem in real-world applications like fraud detection or credit scoring. This problem has been well studied in the offline setting (see e.g. [22] for a recent book on that subject), and has also been studied to some extent in the online, stream-based setting (see [23] for a recent survey). However, among the few existing stream-based approaches, most do not pay attention to drift of the minority class, and as [23] pointed out, a more rigorous evaluation of these algorithms on real-world data needs yet to be done.

4.2.3 Handling Delayed Information

Latency means information becomes available with significant delay. For example, in the case of so-called verification latency, the value of the preceding instance's target variable is not available before the subsequent instance has to be predicted. On *evolving* data streams, this is more than a mere problem of streaming data integration between feature and target streams, as due to concept drift patterns show temporal locality [2]. It means that feedback on the current prediction is not available to improve the subsequent predictions, but only eventually will become available for much later predictions. Thus, there is *no recent sample of labeled data at all* that would correspond to the most-recent unlabeled data, and semi-supervised learning approaches are not directly applicable.

A related problem in static, offline data mining is that addressed by unsupervised transductive transfer learning (or unsupervised domain adaptation): given labeled data from a source domain, a predictive model is sought for a related target domain in which no labeled data is available. In principle, ideas from transfer learning could be used to address latency in evolving data streams, for example by employing them in a chunk-based approach, as suggested in [43]. However, adapting them for use in evolving data streams has not been tried yet and constitutes a non-trivial, open task, as adaptation in streams must be fast and fully automated and thus cannot rely on iterated careful tuning by human experts.

Furthermore, consecutive chunks constitute several domains, thus the transitions between several subsequent chunks might provide exploitable patterns of systematic drift. This idea has been introduced in [27], and a few so-called *drift-mining* algorithms that identify and exploit such patterns have been proposed since then. However, the existing approaches cover only a very limited set of possible drift patterns and scenarios.

4.2.4 Active Selection from Costly Information

The challenge of intelligently selecting among costly pieces of information is the subject of active learning research. Active stream-based selective sampling [38] describes a scenario, in which instances arrive one-by-one. While the instances' feature vectors are provided for free, obtaining their true target values is costly, and the definitive decision whether or not to request this target value must be taken before proceeding to the next instance. This corresponds to a data stream, but *not necessarily* to an *evolving* one. As a result, only a small subset of stream-based selective sampling algorithms is suited for non-stationary environments. To make things worse, many contributions do not state explicitly whether they were designed for drift, neither do they provide experimental evaluations on such evolving data streams, thus leaving the reader the arduous task to assess their suitability for evolving streams. A first, recent attempt to provide an overview on the existing active learning strategies for evolving data streams is given in [43]. The challenges for active learning posed by evolving data streams are:

- **uncertainty regarding convergence**: in contrast to learning in static contexts, due to drift there is no guarantee that with additional labels the difference between model and reality narrows down. This leaves the formulation of suitable stop criteria a challenging open issue.
- **necessity of perpetual validation**: even if there has been convergence due to some temporary stability, the learned hypotheses can get invalidated at any time by subsequent drift. This can affect any part of the feature space and is not necessarily detectable from unlabeled data. Thus, without perpetual validation the mining algorithm might lock itself to a wrong hypothesis without ever noticing.
- **temporal budget allocation**: the necessity of perpetual validation raises the question of optimally allocating the labeling budget over time.
- **performance bounds**: in the case of drifting posteriors, no theoretical work exists that provides bounds for errors and label requests. However, deriving such bounds will also require assuming some type of systematic drift.

The task of active feature acquisition, where one has to actively select among costly features, constitutes another open challenge on evolving data streams: in contrast to the static, offline setting, the value of a feature is likely to change with its drifting distribution.

5. MINING ENTITIES AND EVENTS

Conventional stream mining algorithms learn over a single stream of arriving entities. In subsection 5.1, we introduce the paradigm of *entity stream mining*, where the entities constituting the stream are linked to instances (structured pieces of information) from further streams. Model learning in this paradigm involves the incorporation of the streaming information into the stream of entities; learning tasks include cluster evolution, migration of entities from one state to another, classifier adaptation as entities re-appear with another label than before.

Then, in subsection 5.2, we investigate the special case where entities are associated with the occurrence of events. Model learning then implies identifying the moment of occurrence of an event on an entity. This scenario might be seen as a special case of entity stream mining, since an event can be seen as a degenerate instance consisting of a single value (the event's occurrence).

5.1 Entity Stream Mining

Let T be a stream of entities, e.g. customers of a company or patients of a hospital. We observe entities over time, e.g. on a company's website or at a hospital admission vicinity: an entity appears and re-appears at discrete time points, new entities show up. At a time point t, an entity $e \in T$ is linked with different pieces of information - the purchases and ratings performed by a customer, the anamnesis, the medical tests and the diagnosis recorded for the patient. Each of these information pieces $i_j(t)$ is a structured record or an unstructured text from a stream T_j, linked to e via the foreign key relation. Thus, the entities in T are in 1-to-1 or 1-to-n relation with entities from further streams $T_1, \ldots, T_m$ (stream of purchases, stream of ratings, stream of complaints etc). The schema describing the streams $T, T_1, \ldots, T_m$ can be perceived as a conventional relational schema, except that it describes streams instead of static sets.

In this relational setting, the *entity stream mining* task corresponds to learning a model ζ_T over T, thereby incorporating information from the adjoint streams $T_1, \ldots, T_m$ that "feed" the entities in T.

Albeit the members of each stream are entities, we use the term ”entity” only for stream T – the target of learning, while we denote the entities in the other streams as ”instances”. In the unsupervised setting, entity stream clustering encompasses learning and adapting clusters over T, taking account the other streams that arrive at different speeds. In the supervised setting, entity stream classification involves learning and adapting a classifier, notwithstanding the fact that an entity’s label may change from one time point to the next, as new instances referencing it arrive.

5.1.1 Challenges of Aggregation

The first challenge of entity stream mining task concerns information summarization: how to aggregate into each entity e at each time point t the information available on it from the other streams? What information should be stored for each entity? How to deal with differences in the speeds of the individual streams? How to learn over the streams efficiently? Answering these questions in a seamless way would allow us to deploy conventional stream mining methods for entity stream mining after aggregation.

The information referencing a relational entity cannot be held perpetually for learning, hence aggregation of the arriving streams is necessary. Information aggregation over time-stamped data is traditionally practiced in document stream mining, where the objective is to derive and adapt content summaries on learned topics. Content summarization on entities, which are referenced in the document stream, is studied by Kotov et al., who maintain for each entity the number of times it is mentioned in the news [26].

In such studies, summarization is a task by itself. Aggregation of information for subsequent learning is a bit more challenging, because summarization implies information loss - notably information about the evolution of an entity. Hassani and Seidl monitor health parameters of patients, modeling the stream of recordings on a patient as a sequence of events [21]: the learning task is then to predict forthcoming values. Aggregation with selective forgetting of past information is proposed in [25; 42] in the classification context: the former method [25] slides a window over the stream, while the latter [42] forgets entities that have not appeared for a while, and summarizes the information in frequent itemsets, which are then used as new features for learning.

5.1.2 Challenges of Learning

Even if information aggregation over the streams $T_1, \ldots, T_m$ is performed intelligently, entity stream mining still calls for more than conventional stream mining methods. The reason is that entities of stream T re-appear in the stream and evolve. In particular, in the unsupervised setting, an entity may be linked to conceptually different instances at each time point, e.g. reflecting a customer’s change in preferences. In the supervised setting, an entity may change its label; for example, a customer’s affinity to risk may change in response to market changes or to changes in family status. This corresponds to *entity drift*, i.e. a new type of drift beyond the conventional concept drift pertaining to model ζ_T. Hence, how should entity drift be traced, and how should the interplay between entity drift and model drift be captured?

In the unsupervised setting, Oliveira and Gama learn and monitor clusters as *states* of evolution [32], while [41] extend that work to learn Markov chains that mark the entities’ evolution. As pointed out in [32], these states are not necessarily predefined – they must be subject of learning. In [43], we report on further solutions to the entity evolution problem and to the problem of learning with forgetting over multiple streams and over the entities referenced by them.

Conventional concept drift also occurs when learning a model over entities, thus the challenges pertinent to stream mining also apply here. One of these challenges, and one much discussed in the context of big data, is *volatility*. In relational stream mining, volatility refers to the entity itself, not only to the stream of instances that reference the entities. Finally, an entity is ultimately *big data* by itself, since it is described by multiple streams. Hence, next to the problem of dealing with new forms of learning and new aspects of drift, the subject of efficient learning and adaption in the Big Data context becomes paramount.

5.2 Analyzing Event Data

Events are an example for data that occurs often yet is rarely analyzed in the stream setting. In static environments, events are usually studied through *event history analysis* (EHA), a statistical method for modeling and analyzing the temporal distribution of events related to specific objects in the course of their lifetime [9]. More specifically, EHA is interested in the duration before the occurrence of an event or, in the *recurrent* case (where the same event can occur repeatedly), the duration between two events. The notion of an *event* is completely generic and may indicate, for example, the failure of an electrical device. The method is perhaps even better known as *survival analysis*, a term that originates from applications in medicine, in which an event is the death of a patient and *survival time* is the time period between the beginning of the study and the occurrence of this event. EHA can also be considered as a special case of entity stream mining described in section 5.1, because the basic statistical entities in EHA are monitored objects (or subjects), typically described in terms of feature vectors $\boldsymbol{x} \in \mathbb{R}^n$, together with their survival time s. Then, the goal is to model the dependence of s on $\boldsymbol{x}$. A corresponding model provides hints at possible cause-effect relationships (e.g., what properties tend to increase a patient’s survival time) and, moreover, can be used for predictive purposes (e.g., what is the expected survival time of a patient).

Although one might be tempted to approach this modeling task as a standard regression problem with input (regressor) $\boldsymbol{x}$ and output (response) s, it is important to notice that the survival time s is normally not observed for all objects. Indeed, the problem of *censoring* plays an important role in EHA and occurs in different facets. In particular, it may happen that some of the objects survived till the end of the study at time t_{end} (also called the *cut-off point*). They are censored or, more specifically, *right censored*, since t_{event} has not been observed for them; instead, it is only known that $t_{event} > t_{end}$. In *snapshot monitoring* [28], the data stream may be sampled multiple times, resulting in a new cut-off point for each snapshot. Unlike standard regression analysis, EHA is specifically tailored for analyzing event data of that kind. It is built upon the *hazard function* as a basic mathematical tool.

5.2.1 Survival function and hazard rate

Suppose the time of occurrence of the next event (since the start or the last event) for an object $\boldsymbol{x}$ is modeled as a real-valued random variable T with probability density function $f(\cdot \,|\, \boldsymbol{x})$. The hazard function or hazard rate $h(\cdot \,|\, \boldsymbol{x})$ models the *propensity* of the occurrence of an event, that is, the marginal probability of an event to occur at time t, given that no event has occurred so far:

$$h(t \,|\, \boldsymbol{x}) = \frac{f(t \,|\, \boldsymbol{x})}{S(t \,|\, \boldsymbol{x})} = \frac{f(t \,|\, \boldsymbol{x})}{1 - F(t \,|\, \boldsymbol{x})} \enspace ,$$

where $S(\cdot \,|\, \boldsymbol{x})$ is the survival function and $F(\cdot \,|\, \boldsymbol{x})$ the cumulative distribution of $f(\cdot \,|\, \boldsymbol{x})$. Thus,

$$F(t \,|\, \boldsymbol{x}) \;=\; \mathbf{P}(T \leq t) = \int_0^t f(u \,|\, \boldsymbol{x}) \, du$$

is the probability of an event to occur before time t. Correspondingly, $S(t\,|\,\boldsymbol{x}) = 1 - F(t\,|\,\boldsymbol{x})$ is the probability that the event did not occur until time t (the survival probability). It can hence be used to model the probability of the right-censoring of the time for an event to occur.

A simple example is the Cox proportional hazard model [9], in which the hazard rate is constant over time; thus, it does depend on the feature vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ but not on time t. More specifically, the hazard rate is modeled as a log-linear function of the features x_i:

$$h(t\,|\,\boldsymbol{x}) = \lambda(\boldsymbol{x}) = \exp\left(\boldsymbol{x}^\top \beta\right)$$

The model is proportional in the sense that increasing x_i by one unit increases the hazard rate $\lambda(\boldsymbol{x})$ by a factor of $\alpha_i = \exp(\beta_i)$. For this model, one easily derives the survival function $S(t\,|\,\boldsymbol{x}) = 1 - \exp(-\lambda(\boldsymbol{x}) \cdot t)$ and an expected survival time of $1/\lambda(\boldsymbol{x})$.

5.2.2 EHA on data streams

Although the temporal nature of event data naturally fits the data stream model and, moreover, event data is naturally produced by many data sources, EHA has been considered in the data stream scenario only very recently. In [39], the authors propose a method for analyzing earthquake and Twitter data, namely an extension of the above Cox model based on a sliding window approach. The authors of [28] modify standard classification algorithms, such as decision trees, so that they can be trained on a snapshot stream of both censored and non-censored data.

Like in the case of clustering [35], where one distinguishes between clustering observations and clustering data sources, two different settings can be envisioned for EHA on data streams:

1. In the first setting, events are generated by multiple data sources (representing monitored objects), and the features pertain to these sources; thus, each data source is characterized by a feature vector $\boldsymbol{x}$ and produces a stream of (recurrent) events. For example, data sources could be users in a computer network, and an event occurs whenever a user sends an email.

2. In the second setting, events are produced by a single data source, but now the events themselves are characterized by features. For example, events might be emails sent by an email server, and each email is represented by a certain set of properties.

Statistical event models on data streams can be used in much the same way as in the case of static data. For example, they can serve predictive purposes, i.e., to answer questions such as "How much time will elapse before the next email arrives?" or "What is the probability to receive more than 100 emails within the next hour?". What is specifically interesting, however, and indeed distinguishes the data stream setting from the static case, is the fact that the model may change over time. This is a subtle aspect, because the hazard model $h(t\,|\,\boldsymbol{x})$ itself may already be time-dependent; here, however, t is not the absolute time but the duration time, i.e., the time elapsed since the last event. A change of the model is comparable to concept drift in classification, and means that the way in which the hazard rate depends on time t and on the features x_i changes over time. For example, consider the event "increase of a stock rate" and suppose that $\beta_i = \log(2)$ for the binary feature $x_i =$ `energy sector` in the above Cox model (which, as already mentioned, does not depend on t). Thus, this feature doubles the hazard rate and hence halves the expected duration between two events. Needless to say, however, this influence may change over time, depending on how well the energy sector is doing.

Dealing with model changes of that kind is clearly an important challenge for event analysis on data streams. Although the problem is to some extent addressed by the works mentioned above, there is certainly scope for further improvement, and for using these approaches to derive predictive models from censored data. Besides, there are many other directions for future work. For example, since the *detection* of events is a main prerequisite for analyzing them, the combination of EHA with methods for *event detection* [36] is an important challenge. Indeed, this problem is often far from trivial, and in many cases, events (such as frauds, for example) can only be detected with a certain time delay; dealing with delayed events is therefore another important topic, which was also discussed in section 4.2.

6. EVALUATION OF DATA STREAM ALGORITHMS

All of the aforementioned challenges are milestones on the road to better algorithms for real-world data stream mining systems. To verify if these challenges are met, practitioners need tools capable of evaluating newly proposed solutions. Although in the field of static classification such tools exist, they are insufficient in data stream environments due to such problems as: concept drift, limited processing time, verification latency, multiple stream structures, evolving class skew, censored data, and changing misclassification costs. In fact, the myriad of additional complexities posed by data streams makes algorithm evaluation a highly multi-criterial task, in which optimal trade-offs may change over time.

Recent developments in applied machine learning [6] emphasize the importance of understanding the data one is working with and using evaluation metrics which reflect its difficulties. As mentioned before, data streams set new requirements compared to traditional data mining and researchers are beginning to acknowledge the shortcomings of existing evaluation metrics. For example, Gama et al. [16] proposed a way of calculating classification accuracy using only the most recent stream examples, therefore allowing for time-oriented evaluation and aiding concept drift detection. Methods which test the classifier's robustness to drifts and noise on a practical, experimental level are also starting to arise [34; 47]. However, all these evaluation techniques focus on single criteria such as prediction accuracy or robustness to drifts, even though data streams make evaluation a constant trade-off between several criteria [7]. Moreover, in data stream environments there is a need for more advanced tools for visualizing changes in algorithm predictions with time.

The problem of creating complex evaluation methods for stream mining algorithms lies mainly in the size and evolving nature of data streams. It is much more difficult to estimate and visualize, for example, prediction accuracy if evaluation must be done online, using limited resources, and the classification task changes with time. In fact, the algorithm's ability to adapt is another aspect which needs to be evaluated, although information needed to perform such evaluation is not always available. Concept drifts are known in advance mainly when using synthetic or benchmark data, while in more practical scenarios occurrences and types of concepts are not directly known and only the label of each arriving instance is known. Moreover, in many cases the task is more complicated, as labeling information is not instantly available. Other difficulties in evaluation include processing complex relational streams and coping with class imbalance when class distributions evolve with time. Finally, not only do we need measures for evaluating single aspects of stream mining algorithms, but also ways of combining several of these aspects into global evaluation models, which would take into account expert knowledge and user preferences.

Clearly, evaluation of data stream algorithms is a fertile ground for novel theoretical and algorithmic solutions. In terms of prediction measures, data stream mining still requires evaluation tools that would be immune to class imbalance and robust to noise. In our opinion, solutions to this problem should involve not only metrics based on relative performance to baseline (chance) classifiers, but also graphical measures similar to PR-curves or cost curves. Furthermore, there is a need for integrating information about concept drifts in the evaluation process. As mentioned earlier, possible ways of considering concept drifts will depend on the information that is available. If true concepts are known, algorithms could be evaluated based on: how often they detect drift, how early they detect it, how they react to it, and how quickly they recover from it. Moreover, in this scenario, evaluation of an algorithm should be dependent on whether it takes place during drift or during times of concept stability. A possible way of tackling this problem would be the proposal of graphical methods, similar to ROC analysis, which would work online and visualize concept drift measures alongside prediction measures. Additionally, these graphical measures could take into account the state of the stream, for example, its speed, number of missing values, or class distribution. Similar methods could be proposed for scenarios where concepts are not known in advance, however, in these cases measures should be based on drift detectors or label-independent stream statistics. Above all, due to the number of aspects which need to be measured, we believe that the evaluation of data stream algorithms requires a multi-criterial view. This could be done by using inspirations from multiple criteria decision analysis, where trade-offs between criteria are achieved using user-feedback. In particular, a user could showcase his/her criteria preferences (for example, between memory consumption, accuracy, reactivity, self-tuning, and adaptability) by deciding between alternative algorithms for a given data stream. It is worth noticing that such a multi-criterial view on evaluation is difficult to encapsulate in a single number, as it is usually done in traditional offline learning. This might suggest that researchers in this area should turn towards semi-qualitative and semi-quantitative evaluation, for which systematic methodologies should be developed.

Finally, a separate research direction involves rethinking the way we test data stream mining algorithms. The traditional train, cross-validate, test workflow in classification is not applicable for sequential data, which makes, for instance, parameter tuning much more difficult. Similarly, ground truth verification in unsupervised learning is practically impossible in data stream environments. With these problems in mind, it is worth stating that there is still a shortage of real and synthetic benchmark datasets. Such a situation might be a result of non-uniform standards for testing algorithms on streaming data. As community, we should decide on such matters as: What characteristics should benchmark datasets have? Should they have prediction tasks attached? Should we move towards online evaluation tools rather than datasets? These questions should be answered in order to solve evaluation issues in controlled environments before we create measures for real-world scenarios.

7. FROM ALGORITHMS TO DECISION SUPPORT SYSTEMS

While a lot of algorithmic methods for data streams are already available, their deployment in real applications with real streaming data presents a new dimension of challenges. This section points out two such challenges: making models simpler and dealing with legacy systems.

7.1 Making models simpler, more reactive, and more specialized

In this subsection, we discuss aspects like the simplicity of a model, its proper combination of offline and online components, and its customization to the requirements of the application domain. As an application example, consider the French Orange Portal[2], which registers millions of visits daily. Most of these visitors are only known through anonymous cookie IDs. For all of these visitors, the portal has the ambition to provide specific and relevant contents as well as printing ads for targeted audiences. Using information about visits on the portal the questions are: what part of the portal does each cookie visit, and when and which contents did it consult, what advertisement was sent, when (if) was it clicked. All this information generates hundreds of gigabytes of data each week. A user profiling system needs to have a back end part to preprocess the information required at the input of a front end part, which will compute appetency to advertising (for example) using stream mining techniques (in this case a supervised classifier). Since the ads to print change regularly, based on marketing campaigns, the extensive parameter tuning is infeasible as one has to react quickly to change. Currently, these tasks are either solved using bandit methods from game theory [8], which impairs adaptation to drift, or done offline in big data systems, resulting in slow reactivity.

7.1.1 Minimizing parameter dependence

Adaptive predictive systems are intrinsically parametrized. In most of the cases, setting these parameters, or tuning them is a difficult task, which in turn negatively affects the usability of these systems. Therefore, it is strongly desired for the system to have as few user adjustable parameters as possible. Unfortunately, the state of the art does not produce methods with trustworthy or easily adjustable parameters. Moreover, many predictive modeling methods use a lot of parameters, rendering them particularly impractical for data stream applications, where models are allowed to evolve over time, and input parameters often need to evolve as well.

The process of predictive modeling encompasses fitting of parameters on a training dataset and subsequently selecting the best model, either by heuristics or principled methods. Recently, model selection methods have been proposed that do not require internal cross-validation, but rather use the Bayesian machinery to design regularizers with data dependent priors [20]. However, they are not yet applicable in data streams, as their computational time complexity is too high and they require all examples to be kept in memory.

7.1.2 Combining offline and online models

Online and offline learning are mostly considered as mutually exclusive, but it is their combination that might enhance the value of data the most. Online learning, which processes instances one-by-one and builds models incrementally, has the virtue of being fast, both in the processing of data and in the adaptation of models. Offline (or batch) learning has the advantage of allowing the use of more sophisticated mining techniques, which might be more time-consuming or require a human expert. While the first allows the processing of "*fast* data" that requires real-time processing and adaptivity, the second allows processing of "*big* data" that requires longer processing time and larger abstraction.

Their combination can take place in many steps of the mining process, such as the data preparation and the preprocessing steps. For example, offline learning on big data could extract fundamental and sustainable trends from data using batch processing and massive parallelism. Online learning could then take real-time decisions

[2]`www.orange.fr`

from online events to optimize an immediate pay-off. In the online advertisement application mentioned above, the user-click prediction is done within a context, defined for example by the currently viewed page and the profile of the cookie. The decision which banner to display is done online, but the context can be preprocessed offline. By deriving meta-information such as "the profile is a young male, the page is from the sport cluster", the offline component can ease the online decision task.

7.1.3 Solving the right problem

Domain knowledge may help to solve many issues raised in this paper, by systematically exploiting particularities of application domains. However, this is seldom considered, as typical data stream methods are created to deal with a large variety of domains. For instance, in some domains the learning algorithm receives only partial feedback upon its prediction, i.e. a single bit of right-or-wrong, rather than the true label. In the user-click prediction example, if a user does not click on a banner, we do not know which one would have been correct, but solely that the displayed one was wrong. This is related to the issues on timing and availability of information discussed in section 4.2.

However, building predictive models that systematically incorporate domain knowledge or domain specific information requires to choose the right optimization criteria. As mentioned in section 6, the data stream setting requires optimizing multiple criteria simultaneously, as optimizing only predictive performance is not sufficient. We need to develop learning algorithms, which minimize an objective function including intrinsically and simultaneously: memory consumption, predictive performance, reactivity, self monitoring and tuning, and (explainable) auto-adaptivity. Data streams research is lacking methodologies for forming and optimizing such criteria.

Therefore, models should be simple so that they do not depend on a set of carefully tuned parameters. Additionally, they should combine offline and online techniques to address challenges of big and fast data, and they should solve the right problem, which might consist in solving a multi-criteria optimization task. Finally, they have to be able to learn from a small amount of data and with low variance [37], to react quickly to drift.

7.2 Dealing with Legacy Systems

In many application environments, such as financial services or health care systems, business critical applications are in operation for decades. Since these applications produce massive amounts of data, it becomes very promising to process these amounts of data by real-time stream mining approaches. However, it is often impossible to change existing infrastructures in order to introduce fully fledged stream mining systems. Rather than changing existing infrastructures, approaches are required that integrate stream mining techniques into legacy systems. In general, problems concerning legacy systems are domain-specific and encompass both technical and procedural issues. In this section, we analyze challenges posed by a specific real-world application with legacy issues — the ISS Columbus spacecraft module.

7.2.1 ISS Columbus

Spacecrafts are very complex systems, exposed to very different physical environments (e.g. space), and associated to ground stations. These systems are under constant and remote monitoring by means of telemetry and commands. The ISS Columbus module has been in operation for more than 5 years. For some time, it is pointed out that the monitoring process is not as efficient as previously expected [30]. However, we assume that data stream mining can make a decisive contribution to enhance and facilitate the required monitoring tasks. Recently, we are planning to use the ISS Columbus module as a technology demonstrator for integrating data stream processing and mining into the existing monitoring processes [31]. Figure 2 exemplifies the failure management system (FMS) of the ISS Columbus module. While it is impossible to simply redesign the FMS from scratch, we can outline the following challenges.

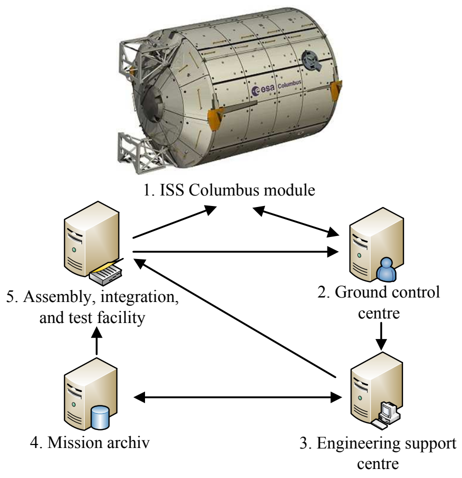

1. ISS Columbus module
2. Ground control centre
3. Engineering support centre
4. Mission archiv
5. Assembly, integration, and test facility

Figure 2: ISS Columbus FMS

7.2.2 Complexity

Even though spacecraft monitoring is very challenging by itself, it becomes increasingly difficult and complex due to the integration of data stream mining into such legacy systems. However, it was assumed to enhance and facilitate current monitoring processes. Thus, appropriate mechanism are required to integrate data stream mining into the current processes to decrease complexity.

7.2.3 Interlocking

As depicted in Figure 2, the ISS Columbus module is connected to ground instances. Real-time monitoring must be applied aboard where computational resources are restricted (e.g. processor speed and memory or power consumption). Near real-time monitoring or long-term analysis must be applied on-ground where the downlink suffers from latencies because of a long transmission distance, is subject to bandwidth limitations, and continuously interrupted due to loss of signal. Consequently, new data stream mining mechanisms are necessary which ensure a smooth interlocking functionality of aboard and ground instances.

7.2.4 Reliability and Balance

The reliability of spacecrafts is indispensable for astronauts' health and mission success. Accordingly, spacecrafts pass very long and expensive planning and testing phases. Hence, potential data stream mining algorithms must ensure reliability and the integration of such algorithms into legacy systems must not cause critical side effects. Furthermore, data stream mining is an automatic process which neglects interactions with human experts, while spacecraft monitoring is a semi-automatic process and human experts (e.g.

the flight control team) are responsible for decisions and consequent actions. This problem poses the following question: How to integrate data stream mining into legacy systems when automation needs to be increased but the human expert needs to be maintained in the loop? Abstract discussions on this topic are provided by expert systems [44] and the MAPE-K reference model [24]. Expert systems aim to combine human expertise with artificial expertise and the MAPE-K reference model aims to provide an autonomic control loop. A balance must be struck which considers both aforementioned aspects appropriately.

Overall, the Columbus study has shown that extending legacy systems with real time data stream mining technologies is feasible and it is an important area for further stream-mining research.

8. CONCLUDING REMARKS

In this paper, we discussed research challenges for data streams, originating from real-world applications. We analyzed issues concerning privacy, availability of information, relational and event streams, preprocessing, model complexity, evaluation, and legacy systems. The discussed issues were illustrated by practical applications including GPS systems, Twitter analysis, earthquake predictions, customer profiling, and spacecraft monitoring. The study of real-world problems highlighted shortcomings of existing methodologies and showcased previously unaddressed research issues.

Consequently, we call the data stream mining community to consider the following action points for data stream research:

- developing methods for ensuring privacy with incomplete information as data arrives, while taking into account the evolving nature of data;
- considering the availability of information by developing models that handle incomplete, delayed and/or costly feedback;
- taking advantage of relations between streaming entities;
- developing event detection methods and predictive models for censored data;
- developing a systematic methodology for streamed preprocessing;
- creating simpler models through multi-objective optimization criteria, which consider not only accuracy, but also computational resources, diagnostics, reactivity, interpretability;
- establishing a multi-criteria view towards evaluation, dealing with absence of the ground truth about how data changes;
- developing online monitoring systems, ensuring reliability of any updates, and balancing the distribution of resources.

As our study shows, there are challenges in every step of the CRISP data mining process. To date, modeling over data streams has been viewed and approached as an extension of traditional methods. However, our discussion and application examples show that in many cases it would be beneficial to step aside from building upon existing offline approaches, and start blank considering what is required in the stream setting.

Acknowledgments

We would like to thank the participants of the RealStream2013 workshop at ECMLPKDD2013 in Prague, and in particular Bernhard Pfahringer and George Forman, for suggestions and discussions on the challenges in stream mining. Part of this work was funded by the German Research Foundation, projects SP 572/11-1 (IMPRINT) and HU 1284/5-1, the Academy of Finland grant 11865: (ALGODAN), and the Polish National Science Center grants DEC-2011/03/N/ST6/00360 and DEC-2013/11/B/ST6/00963.

9. REFERENCES

[1] C. Aggarwal, editor. *Data Streams: Models and Algorithms*. Springer, 2007.

[2] C. Aggarwal and D. Turaga. Mining data streams: Systems and algorithms. In *Machine Learning and Knowledge Discovery for Engineering Systems Health Management*, pages 4–32. Chapman and Hall, 2012.

[3] R. Agrawal and R. Srikant. Privacy-preserving data mining. *SIGMOD Rec.*, 29(2):439–450, 2000.

[4] C. Anagnostopoulos, N. Adams, and D. Hand. Deciding what to observe next: Adaptive variable selection for regression in multivariate data streams. In *Proc. of the 2008 ACM Symp. on Applied Computing*, SAC, pages 961–965, 2008.

[5] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proc. of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS, pages 1–16, 2002.

[6] C. Brodley, U. Rebbapragada, K. Small, and B. Wallace. Challenges and opportunities in applied machine learning. *AI Magazine*, 33(1):11–24, 2012.

[7] D. Brzezinski and J. Stefanowski. Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Trans. on Neural Networks and Learning Systems.*, 25:81–94, 2014.

[8] D. Chakrabarti, R. Kumar, F. Radlinski, and E. Upfal. Mortal multi-armed bandits. In *Proc. of the 22nd Conf. on Neural Information Processing Systems*, NIPS, pages 273–280, 2008.

[9] D. Cox and D. Oakes. *Analysis of Survival Data*. Chapman & Hall, London, 1984.

[10] T. Dietterich. Machine-learning research. *AI Magazine*, 18(4):97–136, 1997.

[11] G. Ditzler and R. Polikar. Semi-supervised learning in non-stationary environments. In *Proc. of the 2011 Int. Joint Conf. on Neural Networks*, IJCNN, pages 2741 – 2748, 2011.

[12] W. Fan and A. Bifet. Mining big data: current status, and forecast to the future. *SIGKDD Explorations*, 14(2):1–5, 2012.

[13] M. Gaber, J. Gama, S. Krishnaswamy, J. Gomes, and F. Stahl. Data stream mining in ubiquitous environments: state-of-the-art and current directions. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(2):116 – 138, 2014.

[14] M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining data streams: A review. *SIGMOD Rec.*, 34(2):18–26, 2005.

[15] J. Gama. *Knowledge Discovery from Data Streams*. Chapman & Hall/CRC, 2010.

[16] J. Gama, R. Sebastiao, and P. Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90(3):317–346, 2013.

[17] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept-drift adaptation. *ACM Computing Surveys*, 46(4), 2014.

[18] J. Gantz and D. Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east, December 2012.

[19] A. Goldberg, M. Li, and X. Zhu. Online manifold regularization: A new learning setting and empirical study. In *Proc. of the European Conf. on Machine Learning and Principles of Knowledge Discovery in Databases*, ECMLPKDD, pages 393–407, 2008.

[20] I. Guyon, A. Saffari, G. Dror, and G. Cawley. Model selection: Beyond the bayesian/frequentist divide. *Journal of Machine Learning Research*, 11:61–87, 2010.

[21] M. Hassani and T. Seidl. Towards a mobile health context prediction: Sequential pattern mining in multiple streams. In *Proc. of , IEEE Int. Conf. on Mobile Data Management*, MDM, pages 55–57, 2011.

[22] H. He and Y. Ma, editors. *Imbalanced Learning: Foundations, Algorithms, and Applications*. IEEE, 2013.

[23] T. Hoens, R. Polikar, and N. Chawla. Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, 1(1):89–101, 2012.

[24] IBM. An architectural blueprint for autonomic computing. Technical report, IBM, 2003.

[25] E. Ikonomovska, K. Driessens, S. Dzeroski, and J. Gama. Adaptive windowing for online learning from multiple inter-related data streams. In *Proc. of the 11th IEEE Int. Conf. on Data Mining Workshops*, ICDMW, pages 697–704, 2011.

[26] A. Kotov, C. Zhai, and R. Sproat. Mining named entities with temporally correlated bursts from multilingual web news streams. In *Proc. of the 4th ACM Int. Conf. on Web Search and Data Mining*, WSDM, pages 237–246, 2011.

[27] G. Krempl. The algorithm APT to classify in concurrence of latency and drift. In *Proc. of the 10th Int. Conf. on Advances in Intelligent Data Analysis*, IDA, pages 222–233, 2011.

[28] M. Last and H. Halpert. Survival analysis meets data stream mining. In *Proc. of the 1st Worksh. on Real-World Challenges for Data Stream Mining*, RealStream, pages 26–29, 2013.

[29] F. Nelwamondo and T. Marwala. Key issues on computational intelligence techniques for missing data imputation - a review. In *Proc. of World Multi Conf. on Systemics, Cybernetics and Informatics*, volume 4, pages 35–40, 2008.

[30] E. Noack, W. Belau, R. Wohlgemuth, R. Müller, S. Palumberi, P. Parodi, and F. Burzagli. Efficiency of the columbus failure management system. In *Proc. of the AIAA 40th Int. Conf. on Environmental Systems*, 2010.

[31] E. Noack, A. Luedtke, I. Schmitt, T. Noack, E. Schaumlöffel, E. Hauke, J. Stamminger, and E. Frisk. The columbus module as a technology demonstrator for innovative failure management. In *German Air and Space Travel Congress*, 2012.

[32] M. Oliveira and J. Gama. A framework to monitor clusters evolution applied to economy and finance problems. *Intelligent Data Analysis*, 16(1):93–111, 2012.

[33] D. Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann Publishers Inc., 1999.

[34] T. Raeder and N. Chawla. Model monitor (m^2): Evaluating, comparing, and monitoring models. *Journal of Machine Learning Research*, 10:1387–1390, 2009.

[35] P. Rodrigues and J. Gama. Distributed clustering of ubiquitous data streams. *WIREs Data Mining and Knowledge Discovery*, pages 38–54, 2013.

[36] T. Sakaki, M. Okazaki, and Y. Matsuo. Tweet analysis for real-time event detection and earthquake reporting system development. *IEEE Trans. on Knowledge and Data Engineering*, 25(4):919–931, 2013.

[37] C. Salperwyck and V. Lemaire. Learning with few examples: An empirical study on leading classifiers. In *Proc. of the 2011 Int. Joint Conf. on Neural Networks*, IJCNN, pages 1010–1019, 2011.

[38] B. Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers, 2012.

[39] A. Shaker and E. Hüllermeier. Survival analysis on data streams: Analyzing temporal events in dynamically changing environments. *Int. Journal of Applied Mathematics and Computer Science*, 24(1):199–212, 2014.

[40] C. Shearer. The CRISP-DM model: the new blueprint for data mining. *J Data Warehousing*, 2000.

[41] Z. Siddiqui, M. Oliveira, J. Gama, and M. Spiliopoulou. Where are we going? predicting the evolution of individuals. In *Proc. of the 11th Int. Conf. on Advances in Intelligent Data Analysis*, IDA, pages 357–368, 2012.

[42] Z. Siddiqui and M. Spiliopoulou. Classification rule mining for a stream of perennial objects. In *Proc. of the 5th Int. Conf. on Rule-based Reasoning, Programming, and Applications*, RuleML, pages 281–296, 2011.

[43] M. Spiliopoulou and G. Krempl. Tutorial ”mining multiple threads of streaming data”. In *Proc. of the Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, PAKDD, 2013.

[44] D. Waterman. *A Guide to Expert Systems*. Addison-Wesley, 1986.

[45] W. Young, G. Weckman, and W. Holland. A survey of methodologies for the treatment of missing values within datasets: limitations and benefits. *Theoretical Issues in Ergonomics Science*, 12, January 2011.

[46] B. Zhou, Y. Han, J. Pei, B. Jiang, Y. Tao, and Y. Jia. Continuous privacy preserving publishing of data streams. In *Proc. of the 12th Int. Conf. on Extending Database Technology*, EDBT, pages 648–659, 2009.

[47] I. Zliobaite. Controlled permutations for testing adaptive learning models. *Knowledge and Information Systems*, In Press, 2014.

[48] I. Zliobaite, A. Bifet, M. Gaber, B. Gabrys, J. Gama, L. Minku, and K. Musial. Next challenges for adaptive learning systems. *SIGKDD Explorations*, 14(1):48–55, 2012.

[49] I. Zliobaite and B. Gabrys. Adaptive preprocessing for streaming data. *IEEE Trans. on Knowledge and Data Engineering*, 26(2):309–321, 2014.

Chapter 3

Probabilistic Active Learning: Towards Combining Versatility, Optimality and Efficiency

The following chapter has been published as:

Probabilistic Active Learning:
Towards Combining Versatility, Optimality & Efficiency.
Georg Krempl, Daniel Kottke, Myra Spiliopoulou.

S. Džeroski, P. Panov, D. Kocev, and L. Todorovski (eds.).
Proceedings of the 17th International Conference on Discovery Science (DS), October 8–10, 2014, Bled, Slovenia. Lecture Notes in Computer Science, Springer. ISSN: 0302-9743.
DOI: `10.1007/978-3-319-11812-3_15`

The original publication is available at Springer International Publishing
`http://link.springer.com/chapter/10.1007/978-3-319-11812-3_15`

Probabilistic Active Learning: Towards Combining Versatility, Optimality and Efficiency

Georg Krempl, Daniel Kottke, and Myra Spiliopoulou

Knowledge Management and Discovery Lab, University Magdeburg, Germany
{georg.krempl,daniel.kottke,myra}@iti.cs.uni-magdeburg.de,
kmd.cs.ovgu.de

Abstract. Mining data with minimal annotation costs requires efficient active approaches, that ideally select the optimal candidate for labelling under a user-specified classification performance measure. Common generic approaches, that are usable with any classifier and any performance measure, are either slow like error reduction, or heuristics like uncertainty sampling. In contrast, our Probabilistic Active Learning (PAL) approach offers versatility, direct optimisation of a performance measure and computational efficiency. Given a labelling candidate from a pool, PAL models both the candidate's label and the true posterior in its neighbourhood as random variables. By computing the expectation of the gain in classification performance over both random variables, PAL then selects the candidate that in expectation will improve the classification performance the most. Extending our recent poster, we discuss the properties of PAL and perform a thorough experimental evaluation on several synthetic and real-world data sets of different sizes. Results show comparable or better classification performance than error reduction and uncertainty sampling, yet PAL has the same asymptotic time complexity as uncertainty sampling and is faster than error reduction.

1 Introduction

Recently, the application of machine learning to large data pools and fast data streams has gained attention. This application often requires classification of data where features are cheap but labels are costly [8]. Examples are applications where features are obtained from an automated process but labels require human annotation efforts. Active learning (AL) [15, p. 4] addresses such applications, where the machine learning system can actively select instances for labelling, rather than passively processing a given set of labelled instances. Its tasks are to decide a) for which instance to request a label, and b) whether to continue labelling at all, given some labels have already been acquired.

The ideal active learning strategy should select those instances first that, once incorporated into the training data, will result in the highest gain in terms of a classification performance measure. Furthermore, it provides a quantification of this performance gain, needed for a sound answer to the stop-criterion related second question. It therefore considers the already acquired amount of training data. Finally, it is fast, requiring solely linear asymptotic computational time per instance with respect to the pool size, in order to enable its application in large data pools and fast data streams. Active learning strategies that are usable in conjunction with any classifier technology provide some of the above qualities. However, as discussed further in Section 2, they do not offer a *combination of all these qualities* in one single approach.

We address this challenge by a novel, probabilistic active learning (PAL) technique for classification that combines the above qualities and constitutes an alternative to other generic strategies like error reduction or uncertainty sampling. It is not limited to a particular classifier technology, and usable with any point [12] performance performance measure. Given a pool of candidates, it computes for each candidate the expected gain in classification performance from obtaining its label. This expectation models the candidate's label *and* the true posterior at its location as a random variables, and uses likelihood weights according to the already obtained labels in the candidate's neighbourhood. Subsequently, it selects the optimal candidate under this expected overall performance gain for labelling. This active selection from a pool requires asymptotic computational time that is solely linear in the size of the pool, as fast uncertainty sampling approaches do. While deriving stop-criteria is not within the scope of this paper, but our quantification of a label's expected impact provides a fundamental first step.

This paper is a full-version of our recent poster [10], extending it by a more detailed discussion of related work, an additional discussion of PAL's properties, and additional experiments. It is structured as follows: In the next section, we provide the necessary background and discuss related approaches. In section 3, we present our probabilistic active learning approach. In section 4, we report on our evaluation results, where we compare *PAL* to the strategy considered to be optimal for minimising classification error (error reduction), and to a popular fast heuristic strategy (uncertainty sampling). [1]

2 Background and Related Work

This paper addresses *pool-based* active learning (AL) for binary classifiers, as described in [15, p. 9] and [4]. In this scenario, an active classifier has access to a pool of unlabelled instances $\mathcal{U} = \{(x, .)\}$. From this pool of labelling candidates it repeatedly selects an instance $(x^*, .)$ for labelling. Upon receiving its label y^*, the instance (x^*, y^*) is moved to a pool of labelled instances $\mathcal{L} = \{(x, y)\}$, the classifier is retrained, and the process is repeated. There exist various approaches for this scenario, recent surveys are provided in [15], [6], [4] and [14]. We will focus on popular families of approaches that are usable with any classification technique, and discuss the ones most related to our approach: error reduction, uncertainty sampling and query-by-transduction.

Expected error reduction (ER) is a decision-theoretic approach. It considers the improvement in classification performance by selecting the candidate, that has the minimal expected classification error if incorporated into the training pool. The seminal work of [5], which coined the term "statistically optimal active learning", derived closed-form solutions for optimal data selection for two specific learning methods. In contrast, the approach suggested in [13] is generic, both with respect to arbitrary performance measures and classifiers: using a Monte Carlo sampling approach, it estimates the performance on a labelled validation sample $\mathcal{V}$, rather than integrating over the full feature distribution $\mathcal{P}r(x)$. It uses the posterior estimate $\hat{p} = \hat{\mathcal{P}r}(y|x)$ provided by the current classifier as proxy for the true posterior $\mathcal{P}r(y|x)$ that is required for the expectation over

[1] For additional resources please consult `http://kmd.cs.ovgu.de/res/pal/`.

the label realisations y. However, as discussed in [2], this proxy is not reliable if solely few labels are available, requiring regularisation approaches such as using Beta priors. Furthermore, the labelled (or self-labelled) validation sample $\mathcal{V}$ must be representative of the data. Not only is this difficult, in particular at the beginning with few available labels and a still unreliable classifier, but it also makes error reduction prohibitively slow [14] for using it in applications that require fast processing of big amounts of data, as even for incremental classifiers its asymptotic time complexity is $O(|\mathcal{V}| \cdot |\mathcal{U}|)$.

In comparison, a faster method [15, p.64] is uncertainty sampling (US), introduced in [11]. It uses simple uncertainty measures, like sample margin, confidence, or entropy as proxies for a candidate's value, and selects the candidate with maximal uncertainty. However, these proxies do not consider the number of similar instances on which posterior estimates are made. This is problematic, as Figure 2 (next page) illustrates on four exemplary active learning situations. These situations could, for example, occur simultaneously in different regions of a feature space such that the next label must be actively requested in either of them[2] The first (in Roman numeral) and second situation differ in the number of obtained labels (6 vs. 1), but lead to the same posterior estimate $\hat{\mathcal{P}r}(+|x) = 1$, as all obtained labels are positive. Uncertainty sampling is indifferent between them, as both entropy and confidence are zero. This indicates equal and absolute certainty, which is not justified as in II the single positive label can simply be due to chance, even if the true posterior of the positive class is actually smaller than 0.5 and the classifier is wrong. In contrast, in I a high true positive posterior is indeed very likely, and additional labels have less impact on the classifier. Similarly, in IV the classifier's prediction is quite reliable, but uncertainty according to measures like entropy or confidence is maximal. This leads to sampling in regions of high Bayesian error rate, even if the classifier can not be further improved there.

Some of the many existing *classifier-specific* AL approaches offer high processing speeds for particular applications. However, they require classifier selection to be made with respect to the available active learning strategy, as sample reusability between different types of classifiers for selector and consumer strategies is an open question [16]. Finally, even recently proposed classifier-specific approaches are mostly either information-theoretic (i.e. agnostic to the decision task at hand) or use the most likely or most pessimistic posterior under the current model, thus ignoring the reliability associated with this estimate, as for example [7]. A very recent information-theoretic approach that considers the reliability of a predictive model is Query-By-Transduction (QbT) [9]. QbT is based on conformal prediction and selects the instances with respect to the p-values obtained using transduction. This quantification of the reliability using p-values is related to ours, although we use the likelihood weights of the posterior estimates and follow a decision-theoretic Bayes-optimal active learning approach that directly optimises a classification performance measure.

[2] For simplicity, this illustration assumes conditional independence of the posterior from the feature given the region, i.e. $\mathcal{P}r(y|x,z) = \mathcal{P}r(y|z)$, where y is the class, x the feature vector, and z the region. Thus no further differentiation can be made within a region. We also assume equal numbers of instances in all regions, making accuracy everywhere equally important.

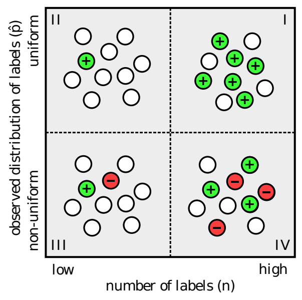


Fig. 1. Different AL situations, where entropy- or confidence-based uncertainty measures differentiate only on a class' *relative* (vert.) but not on all classes' *total* (horiz.) number of labels

3 Probabilistic Active Learning

Following the common smoothness assumption [3], we consider that an instance x influences the classification the most in its neighbourhood. Thus, the impact of an additional label primarily depends on the already obtained labels in its neighbourhood. We summarise these by their *absolute* number n, and the share of positives $\hat{p}$ therein, yielding the *label statistics* $\mathit{ls} = (n, \hat{p})$. Here, n is obtained by counting the similar labelled instances for pre-clustered or categorical data (as for the partitions in Figure 2), or approximated by frequency estimates such as kernel frequency estimates for smooth, continuous data. Thus, in x's neighbourhood, n expresses the absolute quantity of labelled information, whereas the density d_x of unlabelled instances quantifies the importance of this neighbourhood, i.e. the share of future classifications that will take place therein compared to other regions of the feature space.

Given a labelling candidate $(x, .)$ from a pool of unlabelled instances $\mathcal{U}$ for a user-specified point classification performance measure [12] like accuracy, we want to compute the expected overall gain in classification performance if requesting its label. This requires knowledge of its label statistics ls, but also of its label y and the true posterior p of the positive class within its neighbourhood. As the latter values of y and p are not directly accessible, we use a probabilistic approach and model Y and P as random variables. This allows us to compute the *expected value* of the gain in performance over all different true posteriors and label realisations, which we denote as *probabilistic gain*[3] (pgain). Finally, we weight it by the neighbourhood's density d_x (over labelled and unlabelled data) to consider the importance of the neighbourhood on the whole data set, quantifying the overall expected performance change. Comparing the overall expected performance change of all candidates, we select the optimal candidate for labelling.

We now first provide the modelling and derive the necessary equations, present the framework of *Probabilistic Active Learning* (*PAL*) with its pseudo-code, and close with discussing its properties.

[3] We do this to differentiate it from the expected gain as in expected error reduction methods like [2], where expectation is solely over label outcomes, but not over the true posterior.

3.1 Probabilistic Gain Calculation

Given a candidate $(x, .)$, the label statistics ls summarise the obtained labels in its neighbourhood. We model the true posterior P of the positive class $(y = 1)$ in this neighbourhood as a Beta-distributed random variable, whose realisation p is itself the parameter of the Bernoulli distribution controlling the label realisation $y \in \{0, 1\}$ of any instance within the neighbourhood. Consequently, the number of positives $n \cdot \hat{p}$ among the n already obtained labels in the neighbourhood is the realisation of a Binomial-distributed random variable:

$$P \sim \mathrm{Beta}_{n\cdot\hat{p}+1,n\cdot(1-\hat{p})+1} \tag{1}$$

$$Y \sim \mathrm{Bernoulli}_p = \mathrm{Ber}_p \tag{2}$$

$$(n \cdot \hat{p}) \sim \mathrm{Binomial}_{n,p} \tag{3}$$

The true posterior's Beta distribution above results from its normalised likelihood given the already observed labels, that is

$$\omega_{\mathit{ls}}(p) = \frac{L(p|\mathit{ls})g(p)}{\int_0^1 L(\psi|\mathit{ls})g(\psi)\mathrm{d}\psi} = (1+n)\cdot L(p|\mathit{ls}) \tag{4}$$

$$= \frac{\Gamma(n+2)\cdot p^{n\cdot\hat{p}}\cdot(1-p)^{n\cdot(1-\hat{p})}}{\Gamma(n\cdot\hat{p}+1)\cdot\Gamma(n\cdot(1-\hat{p})+1)} = \mathrm{Beta}_{\alpha,\beta}(p) \tag{5}$$

where the parameters $\alpha = n \cdot \hat{p} + 1$ and $\beta = n \cdot (1 - \hat{p}) + 1$ of the Beta-distribution's pdf $\mathrm{Beta}_{\alpha,\beta}(p)$ are obtained by following a Bayesian approach under a uniform prior for P such that $g()$ is a constant function, and by using the probability mass function according to Eq. 3 for the likelihood $L(p|\mathit{ls})$, and $(1 + n) \cdot \Gamma(n + 1) = \Gamma(n + 2)$.

We take the expectation on the performance gain over these two random variables, yielding the candidate's probabilistic gain (pgain), that defines the expected change of the performance measure for its neighbourhood:

$$\mathrm{pgain}(\mathit{ls}) = \mathtt{E}_p\left[\mathtt{E}_y\left[\mathrm{gain}_p(\mathit{ls}, y)\right]\right] \tag{6}$$

$$= \int_0^1 \mathrm{Beta}_{\alpha,\beta}(p)\cdot\sum_{y\in\{0,1\}}\mathrm{Ber}_p(y)\cdot\mathrm{gain}_p(\mathit{ls}, y)\,\mathrm{d}p \tag{7}$$

Here, $\mathrm{gain}_p(\mathit{ls}, y)$ is the candidate's $(x, .)$ performance gain given its label realisation y and the neighbourhood's true posterior p:

$$\mathrm{gain}_p(\mathit{ls}, y) = \mathrm{perf}_p\left(\frac{n\hat{p}+y}{n+1}\right) - \mathrm{perf}_p(\hat{p}) \tag{8}$$

The definition of Eq. 7 and 8 allow the use of any point performance measure (see e.g. [12]) for perf. An example is accuracy (acc), defined as

$$\mathrm{perf}_p(\hat{p}) = 1 - \mathrm{err}_p(\hat{p}) = 1 - \begin{cases} p & \hat{p} < 0.5 \\ 1-p & \mathit{otherwise} \end{cases} \tag{9}$$

where $\mathrm{err}_p(\hat{p})$ is the error rate under Bayes' optimal classification, given a true posterior p and observed posterior $\hat{p}$ of the positive class.

Plugging this in Eq. 7 yields the probabilistic accuracy gain

$$\mathrm{pgain}_{\mathrm{acc}}(\mathit{ls}) = \\ = \int_0^1 \mathrm{Beta}_{\alpha,\beta}(p) \sum_{y\in\{0,1\}} \mathrm{Ber}_p(y) \left(\mathrm{err}_p(\hat{p}) - \mathrm{err}_p\left(\frac{n\hat{p}+y}{n+1}\right)\right) \mathrm{d}p$$

which we compute by trapezoidal numerical integration over p.

Finally, we weight each candidate's probabilistic gain with the density d_x over *labelled* and *unlabelled* instances in its neighbourhood, and select the candidate with the highest density-weighted probabilistic gain for labelling:

$$x^* = \arg\max_{x\in\mathcal{U}} \left(d_x \cdot \mathrm{pgain}_{\mathrm{acc}}(\mathit{ls}_x)\right) \tag{10}$$

3.2 PAL Algorithm

The pseudo-code for the resulting probabilistic, pool-based active learning algorithm is given in Figure 2. Iterating over the candidate pool $\mathcal{U}$ (Lines 2-6), for each labelling candidate x one computes its label statistics $\mathit{ls}_x = (n_x, \hat{p}_x)$, its density weight d_x, and using numerical integration its probabilistic gain, which is weighted by its density weight to obtain g_x. Finally, the candidate with the highest g_x is selected (Line 7).

```
1: function PoolBasedPAL($\mathcal{U}$,$\mathcal{L}$)
2:     for $x \in \mathcal{U}$ do
3:         $(n_x, \hat{p}_x) \leftarrow labelstatistics(x, \mathcal{L})$
4:         $d_x \leftarrow densityweight(x, \mathcal{L} \cup \mathcal{U})$
5:         $g_x \leftarrow pgain((n_x, \hat{p}_x)) \cdot d_x$
6:     end for
7:     return arg $\max_{x\in\mathcal{U}}(g_x)$
8: end function
```

Fig. 2. The PAL Algorithm

3.3 PAL's Properties

Statistical Optimality in Disjoint Neighbourhoods. For a disjoint neighbourhood concept, like in pre-clustered or categorical data, where instances are partitioned such that instances having an influence on each others' classification belong to the same subset, the density-weighted probabilistic gain of a candidate corresponds precisely to the expected change in overall performance from acquiring the candidate's label. Thus selecting the candidate with highest probabilistic gain is statistically optimal.

For smooth, continuous neighbourhoods, the density-weighted probabilistic gain is the expected change at the candidate's location, serving as an approximation of the overall performance gain. We use this latter concept in our evaluation, as it applies to more data sets and is better comparable to the baseline active learning algorithms.

Computational Efficiency. In this subsection, we discuss the asymptotic (with respect to data set size) computational time complexity of PAL and related algorithms for active learning of binary, incremental classifiers. For selecting a candidate from a pool $\mathcal{U}$ of labelling candidates, the PAL algorithm above needs to iterate over all candidates in the pool (Lines 2 – 6). Each iteration consists of 1) querying labelstatstics, 2) querying density weights, and 3) computing the probabilistic gain. The first step requires absolute frequency estimates of labels in the candidate's neighbourhood, similar to the relative frequency estimates needed by entropy or confidence uncertainty measures. These are obtained in constant time by probabilistic classifiers. The second step requires density estimates over all instances, that is over labelled $\mathcal{L}$ and unlabelled $\mathcal{U}$ ones. Precomputing these density estimates once for all later calls of PAL leads to constant query time, as in the pool-based setting the union $\mathcal{L} \cup \mathcal{U}$ is constant. The third step consists of a numeric integration over the true posterior p and a summing over possible label realisations y. Both factors do not depend on the data set size. We used fifty numeric integration steps in all our experiments to get highly precise estimates for expected classification accuracy gain, resulting in a constant factor of $O(50 \cdot 2)$ per probabilistic gain computation. Overall, the iteration over the pool is done in $O(|\mathcal{U}|)$ time.

Selecting the candidate with highest density-weighted probabilistic gain in Line 7 is done in constant time, by using a sweep line approach and storing the maximal value and its corresponding candidate in the previous for-loop.

Overall, PAL requires $O(|\mathcal{U}|)$ time for selecting a candidate from the pool. Uncertainty sampling, using probabilistic classifiers and entropy or confidence uncertainty measures, requires asymptotically the same time, but due the simplicity of its computation with a smaller constant factor involved. In contrast, error reduction as discussed in [15], requires $O(|\mathcal{U}| \cdot |\mathcal{V}|)$ time, where $|\mathcal{V}| \approx |\mathcal{U}|$, as $\mathcal{V}$ needs to be a representative sample of the data.

Characteristics of the Probabilistic Gain. For a better understanding of the probabilistic gain function, Figure 3.3 shows the computed probabilistic gain (in terms of accuracy) for different label statistics, i.e. combinations of different numbers of already obtained labels n and observed posteriors $\hat{Pr}(+|x)$. The following main characteristics of the curve underline its reasonable behaviour:

Monotonicity with variable $\mathbf{n}$: With increasing n and a fixed $\hat{\mathcal{P}r}(+|x)$ the probabilistic gain decreases, because it is more likely that the posterior already is correct.

Symmetry with respect to $\hat{\mathcal{P}r}(+|\mathbf{x}) = \mathbf{0.5}$: Evaluating accuracy, pos. and neg. labels count the same, i.e. the probabilistic gain is equal for $\hat{\mathcal{P}r}(+|x)$ and $\hat{\mathcal{P}r}(-|x)$.

Zero for irrelevant candidates: If one label would not change the decision in its neighbourhood, the accuracy remains the same. Thus, gain and probabilistic gain are 0.

This figure is inspired by an illustration of Settles, where different uncertainty measures are plotted as functions of the posterior of a class (see figure 2.4 in [15, p. 15]). Comparing the least confident curve (plot (a) in [15]), it behaves nearly similarly as our probabilistic gain for $n = 1$, but does not change with n.

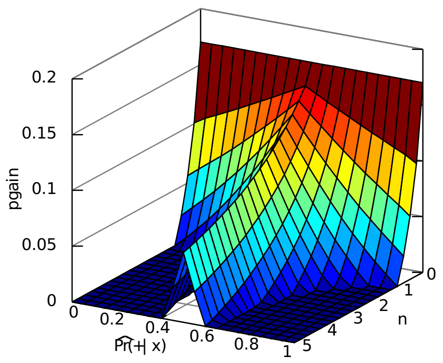


Fig. 3. Illustration of the probabilistic gain (pgain) as a function of $\hat{Pr}(+|x)$, which is the observed posterior of the positive class, and of n, which is the number of already obtained labels

4 Experimental Evaluation

From its theoretical characteristics, we expect PAL to be comparable to error-reduction in terms of classification performance, yet faster, and we expect PAL to be better than uncertainty sampling. This section will now verify these characteristics empirically. After outlining the experimental setup, we will discuss the results in the second subsection.

4.1 Evaluation Settings

We compare our new base method PAL with expected error-reduction (in the extended variant proposed by Chapelle in [2], denoted Chap), with uncertainty sampling (using confidence [15] as uncertainty measure, den. Uncer), and with random sampling (den. Rand). While error-reduction is considered as one of the best available AL-methods [15, p. 64], uncertainty sampling is fast and very popular for large or streaming data.

We used Gaussian kernels for frequency estimation, and a Parzen window classifier as in [2] for ensuring comparability with [2]. So, the estimated label frequencies $labelFreq_c$, $c \in \{+,-\}$ at an instance x for the the positive $\mathcal{L}_+$ and the negative class $\mathcal{L}_-$ are calculated by an unnormalised Gaussian function. These frequencies build the label statistics $n = labelFreq_+ + labelFreq_-$ and $\hat{p} = labelFreq_+/n$.

$$labelFreq_c(x) = \sum_{x' \in \mathcal{L}_c} \exp\left(-\frac{\|x'-x\|^2}{2\sigma^2}\right)$$

Our framework starts *without* initial labels, and finishes after 40 label requests. The classifiers, implemented in Octave/MATLAB and run separately on a cluster, use the same pre-tuned, data set-specific bandwidth, and are re-evaluated in each of the 40 steps on the same, dedicated (labelled) test sample. This ensures that only the difference in the active learning strategy is influencing the performance. For better performance assessment, we generated 100 random training and test subsets for each data set, and averaged the results. Evaluation is done on 2 synthetic (based on [2]) and 6 real-world data sets (from [1]). The main characteristics (number of instances, number of attributes), such as training and test set size and the σ of the Parzen window, are summarised in Table 4. The synthetic data sets consist of $4x4$ clusters, arranged in a checker-board formation. While the clusters are low-density-separated in `Che`, they are adjoined in `Che2`. The real-world data sets are Mammographic mass (`Mam`), Vertebral (`Ver`), Haberman's survival (`Hab`), Blood transfusion (`Blo`), Seeds (`See`) and Abalone (`Aba`). All attributes are scaled to a $[0; 1]$-range. We evaluate the performance over the first 40 active label acquisitions and provide the results as learning curves for the optimised performance measure accuracy for all data sets and algorithms.

4.2 Evaluation Results

In accordance to [2] and [15], we provide learning curves in the subfigures of Figure 6. These curves depict the progress in the active classifier's accuracy as 40 training instances are selected one after another for training. This allows to evaluate the performance based on several criteria, and is more informative than tables of the performance at arbitrarily selected learning stages.

(1) When does a curve become flat, i.e. when does the learner converge? On subfigure g) for data set Seeds, the curves become flat already after reading 10 labels, while the curves for data set Checkboard 2 (b) do not converge. Convergence indicates that additional labels do not provide additional use to the classifier, ideally a classifier converges fast and to a high level of performance. This is seen on subfigures a and c, where PAL in contrast to Random Sampling quickly converges to a high performance level.

(2) At what accuracy does a learner stop improving? Clearly, a learner that achieves a 99% accuracy after reading 10 labels is better than one that needs 40 labels to reach the same accuracy value, and also better than one that converges at 75%. Hence, PAL outperforms all other algorithms except on Blood (f), Seeds (g) Abalone (h). The moment of convergence gives also indication on the appropriateness of the data set for

Dataset	Inst	Attr	$\mathcal{P}r(+)$	|Train|	|Test|	σ
`See`	210	7	33 %	160	50	0.1
`Che`	308	2	44 %	200	108	0.08
`Che2`	392	2	49 %	250	142	0.08
`Hab`	306	3	73 %	256	50	0.1
`Ver`	310	6	32 %	260	50	0.1
`Aba`	4177	8	50 %	400	1177	0.06
`Blo`	748	4	24 %	600	148	0.1
`Mam`	830	11	51 %	630	200	0.1

Fig. 4. Dataset characteristics and parameters (number of instances, number of attributes, proportion of positive instances, training set size, test set size, bandwidth for Parzen window classifier)

Dataset	PAL	Chap	Uncer	Rand
`See`	0.50	0.93	0.03	0.01
`Che`	0.61	1.16	0.03	0.01
`Che2`	0.92	1.54	0.03	0.02
`Hab`	0.89	1.72	0.03	0.02
`Ver`	0.91	1.84	0.04	0.02
`Aba`	1.51	3.82	0.07	0.04
`Blo`	2.34	6.14	0.1	0.05
`Mam`	2.56	8.48	0.25	0.12

Fig. 5. Average execution time (in seconds), ordering of rows is in ascending training dataset size

active learning. If we contrast subfigures b and g, we must assert that data set Seeds is not truly interesting in terms of active learning: after reading the labels of 5 or at most 10 instances, all learners converge to an accuracy very close to 1. Thus, comparative performance of the active learners on Seeds is not truly informative; this data set is not very appropriate for experiments on active learning (except as a counterexample). The curves on the Blood Transfusion data set (cf. subfigure f) also indicate that active learning is not truly beneficial on this data set.

(3) Does a learner recover from previous errors? If a curve becomes flat early, then the learner might be trapped in low accuracy values. This is the case for the algorithm Chapelle on data set Mammographic Mass (c). In contrast, PAL recovers on this data set, as well as on data sets Checkboard, Vertebral and Habermans Survival (a, d, e). Random Sampling never recovers from earlier choices: its performance curves are either flat or go upwards, indicating that an early poor choice cannot be amended. Uncertainty Sampling recovers in some data sets, while Chapelle and PAL always manages

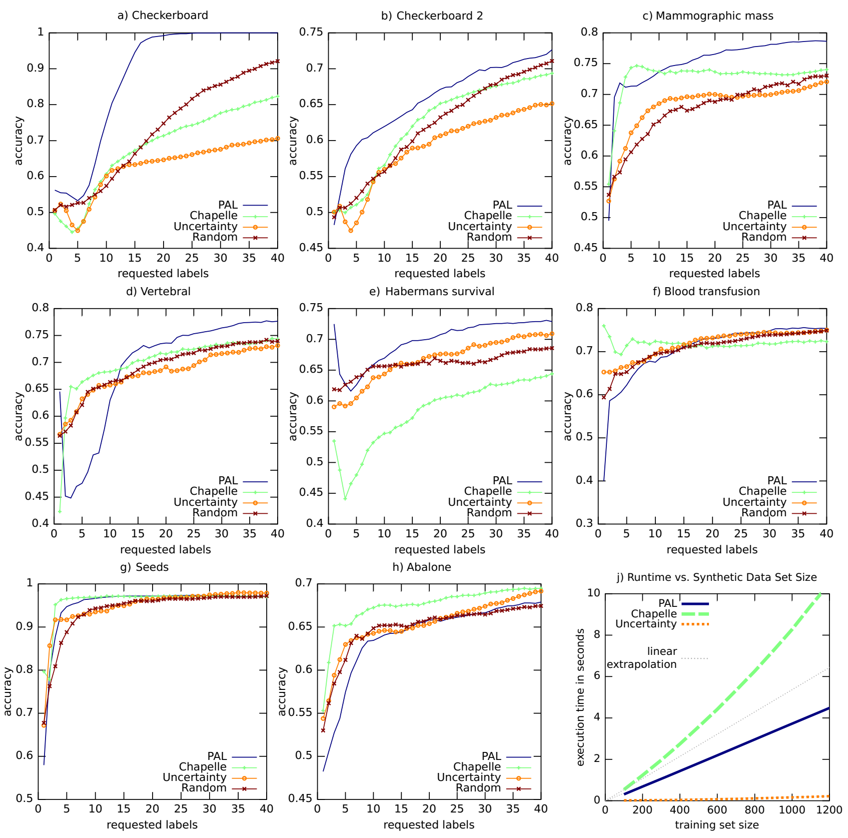


Fig. 6. a-h: accuracy curves for the algorithms on each dataset; early convergence to very high values is best; improvement after a performance drop is better than a flat curve on low accuracy values; **j**: runtime of PAL on a synthetic data set of varying size (100–1200 candidate instances)

to recover if they err in their early choices of label. Summarising the results on accuracy progress, PAL exhibits high performance in all data sets, manages to recover from poor choices and makes best use of available labels, as long as needed (i.e. longer for Checkboard 2 than for Seeds). PAL reaches the best accuracy values on 5 of the data sets, achieves comparable accuracy to the other learners on two data sets (Seeds and Blood Transfusion). PAL is only outperformed once on the Abalone data set.

(4) Execution time The execution time of PAL is shown in Table 5 and plot j of figure 6. Table 5 indicates the execution times of all active learning algorithms on each dataset. We see that PAL achieves better accuracy curves with lower (up to $1/2.5$ times) execution time than the error-reduction algorithm of Chapelle. Nevertheless, the execution time is still significantly higher than that of uncertainty sampling, but like the former its time increases solely linearly with the training set size, i.e. the number of labelling candidates. This is also shown in plot j) of Figure 6, where the execution times on various training set sizes of the same synthetic dataset are plotted. Overall, the uniformly low execution time of uncertainty sampling is accompanied by a stronger variance among the accuracy curves (cf. Figure 6): while PAL has very high performance on all data sets, escapes from earlier errors and exploits well all labels (whenever reasonable, see counterexample on Subfigure 6g), the accuracy curves of Uncertainty Sampling and Random Sampling vary in dependence on the data set. Thus, PAL exhibits stable performance at lower execution time than the expensive error-reduction mechanism, while the simpler algorithms are affected stronger by the idiosyncrasies of the data sets.

5 Conclusion

In this paper, we introduced the probabilistic active learning approach (PAL). It uses probabilistic estimates (label statistics) calculated within the neighbourhood of a labelling candidate. In contrast to Monte-Carlo-based error reduction approach proposed in [13], it models both the true posterior and the candidate's label as random variables. Given a user-specified performance measure, PAL computes the probabilistic gain, that is the expected performance gain over *both* random variables by numeric integration. It subsequently selects the candidate with highest density-weighted probabilistic gain. Like uncertainty sampling [11], PAL requires asymptotically linear time with respect to the pool size, in contrast to quadratic time required by error reduction in [13].

Thus PAL combines two previously incompatible qualities: being fast, and computing and optimising directly a point-performance measure. Given such a user-specified performance measure and the label statistics as input, no additional parameters are required. Our experimental evaluation shows that PAL yields comparable or better classification performance than error-reduction, uncertainty-sampling or random active learning strategies, while requiring less computational time than error-reduction.

Future work will comprise deriving *specific* closed-form solutions for some point-performance measures such as misclassification loss, as this promises further improvements in speed. Further research is also needed to address *non-myopic* scenarios, where optimising the resulting performance gain from acquiring several labels is required. Finally, as PAL is fast and requires only label statistics but no samples to be kept, its application in *data streams* seems a promising direction for future research.

Acknowledgements. We thank Vincent Lemaire from Orange Labs, France, for the insightful discussions.

References

1. Asuncion, A., Newman, D.J.: UCI ML repository (2013)
2. Chapelle, O.: Active learning for parzen window classifier. In: Proc. 10th Int. Workshop on AI and Statistics, pp. 49–56 (2005)
3. Chapelle, O., Schölkopf, B., Zien, A. (eds.): Semi-Supervised Learning. MIT Press (2006)
4. Cohn, D.: Active learning. In: Sammut, C., Webb, G.I. (eds.) Encyclopedia of ML, pp. 10–14. Springer (2010)
5. Cohn, D.A., Ghahramani, Z., Jordan, M.I.: Active learning with statistical models. J. of AI Research 4, 129–145 (1996)
6. Fu, Y., Zhu, X., Li, B.: A survey on instance selection for active learning. Knowledge and Inf. Syss. 35(2), 249–283 (2012)
7. Garnett, R., Krishnamurthy, Y., Xiong, X., Schneider, J.G., Mann, R.: Bayesian optimal active search and surveying. In: Proc. of the 29th ICML (2012)
8. Gopalkrishnan, V., Steier, D., Lewis, H., Guszcza, J.: Big data, big business: Bridging the gap. In: Workshop on Big Data, Streams and Heterogeneous Source Mining, pp. 7–11 (2012)
9. Ho, S.S., Wechsler, H.: Query by transduction. IEEE Trans. on Pattern A. & Mach. Int. 30(9), 1557–1571 (2008)
10. Krempl, G., Kottke, D., Spiliopoulou, M.: Probabilistic active learning: A short proposition. In: Proc. 21st Europ. Conf. on AI (ECAI 2014). IOS Press (2014)
11. Lewis, D.D., Gale, W.A.: A sequential algorithm for training text classifiers. In: Proc. of the 17th ACM SIGIR, pp. 3–12 (1994)
12. Parker, C.: An analysis of performance measures for binary classifiers. In: Proc. of the 11th ICDM, pp. 517–526. IEEE (2011)
13. Roy, N., McCallum, A.: Toward optimal active learning through sampling estimation of error reduction. In: Proc. of the 18th ICML, pp. 441–448 (2001)
14. Settles, B.: Active Learning literature survey. CS Tech. Rep. 1648, U. Wisconsin (2009)
15. Settles, B.: Active Learning. in Synth. Lect. AI and ML. Morgan Claypool, vol. 18 (2012)
16. Tomanek, K., Morik, K.: Inspecting sample reusability for active learning. In: Guyon, I., et al. (eds.) AISTATS workshop on Act. Learning and Exp. Design., vol. 16, pp. 169–181 (2011)

Chapter 4

Optimised Probabilistic Active Learning for Fast, Non-Myopic, Cost-Sensitive Active Classification

The following chapter has been published as:

Optimised Probabilistic Active Learning (OPAL)
For Fast, Non-Myopic, Cost-Sensitive Active Classification
Georg Krempl, Daniel Kottke, Vincent Lemaire.

C. Bielza, J. Gama, A. Jorge, I. Zliobaite (eds.). Machine Learning, Special Issue of the Journal Track of ECML/PKDD 2015, vol. 100(2), pp. 449–476, 2015, Springer. ISSN: 1573-0565.
DOI: `10.1007/s10994-015-5504-1`

The original publication is available at Springer International Publishing
`http://link.springer.com/article/10.1007/s10994-015-5504-1`

Optimised probabilistic active learning (OPAL)

For fast, non-myopic, cost-sensitive active classification

Georg Krempl[1] · **Daniel Kottke**[1] · **Vincent Lemaire**[2]

Received: 11 August 2014 / Accepted: 8 May 2015 / Published online: 20 June 2015

Abstract In contrast to ever increasing volumes of automatically generated data, human annotation capacities remain limited. Thus, fast active learning approaches that allow the efficient allocation of annotation efforts gain in importance. Furthermore, cost-sensitive applications such as fraud detection pose the additional challenge of differing misclassification costs between classes. Unfortunately, the few existing cost-sensitive active learning approaches rely on time-consuming steps, such as performing self-labelling or tedious evaluations over samples. We propose a fast, non-myopic, and cost-sensitive probabilistic active learning approach for binary classification. Our approach computes the expected reduction in misclassification loss in a labelling candidate's neighbourhood. We derive and use a closed-form solution for this expectation, which considers the possible values of the true posterior of the positive class at the candidate's position, its possible label realisations, and the given labelling budget. The resulting myopic algorithm runs in the same linear asymptotic time as uncertainty sampling, while its non-myopic counterpart requires an additional factor of $O(m \cdot \log m)$ in the budget size. The experimental evaluation on several synthetic and real-world data sets shows competitive or better classification performance and runtime, compared to several uncertainty sampling- and error-reduction-based active learning strategies, both in cost-sensitive and cost-insensitive settings.

Keywords Active learning · Non-myopic · Cost-sensitive · Unequal misclassification costs · Misclassification loss · Imbalanced data · Uncertainty sampling · Error reduction

Editors: João Gama, Indrė Žliobaitė, Alípio M. Jorge, and Concha Bielza.

✉ Georg Krempl
georg.krempl@ovgu.de; georg.krempl@iti.cs.uni-magdeburg.de

Daniel Kottke
daniel.kottke@iti.cs.uni-magdeburg.de

Vincent Lemaire
vincent.lemaire@orange.com

[1] KMD Lab, University Magdeburg, Magdeburg, Germany

[2] Orange Labs, Lannion, France

1 Introduction

The volume of automatically generated data increases constantly (Gantz and Reinsel 2012) but human annotation capacities remain limited. Learning from large pools or fast streams of unlabelled data, yet scarce and expensive labelled data, requires the development of fast and efficient active learning algorithms (Gopalkrishnan et al. 2012; Krempl et al. 2014). Such algorithms actively construct a training set, rather than passively processing a given one. Their objective is to select among the unlabelled instances (candidates) the ones for labelling that are deemed to maximise the classification performance the most (Settles 2012), thus focusing annotation efforts to the most valuable candidates. While *fast* active learning itself poses a challenge, an even bigger one is *cost-sensitivity*[1] (Liu et al. 2009), where misclassification costs differ between classes, as for example in the diagnosis of rare but dangerous ailments in patients (Attenberg and Ertekin 2013), where classifying a sick patient as healthy incurs more severe consequences than classifying a healthy one as sick.

We address these challenges by presenting a novel, fast probabilistic active learning approach, which is suitable for binary classification, both under equal and non-equal misclassification costs. This probabilistic (Krempl et al. 2014a, b) active learning approach computes the expected performance gain, thereby considering both a candidate's *label realisation* and the *true posterior* of the positive class in the candidate's neighbourhood. The latter directly incorporates the likelihoods of different possible posteriors under the already labelled data. This advances most state-of-the-art decision-theoretic active learning literature (e.g. Freytag et al. 2013; Garnett et al. 2012), which considers solely the most likely (or most pessimistic) posterior.

We make three important contributions beyond (Krempl et al. 2014a, b) and other works: First, we optimise label selection for the minimisation of misclassification loss, a *cost-sensitive* performance measure (Hand 2009). Second, we derive a *fast, closed-form solution* for calculating the probabilistic gain of an instance. We show that this yields a myopic active learning approach with the same *linear asymptotic computational time* as uncertainty sampling, which is one of the fastest available approaches (Settles 2012, p. 64). Third, we propose a *non-myopic* extension of our *optimised probabilistic active learning (OPAL)* approach, where the myopic, isolated view on the value of each label is exchanged for considering the possible remaining number of similar label acquisitions under a given budget. We show that this provides an advantage in particular in cost-sensitive settings, while it requires solely additional time of a factor $O(m \cdot \log m)$ of the budget size. In practical applications, neither this budget size, which for example results from limited human annotation capacities, nor the misclassification costs, which for example are determined by economic consequences as in the German Credit dataset (Elkan 2001), do constitute tunable parameters. Our approach is simple to implement, and neither requires maintaining an evaluation set, nor self-labelling. Experimental evaluation shows its competitiveness in classification performance and speed, compared to other cost-sensitive and cost-insensitive active learning approaches.

The rest of this paper is organised as follows: first, Sect. 2 provides the necessary background and discusses the related work. Our OPAL-approach is presented in Sect. 3. The results of its experimental evaluation are reported in Sect. 4, comparing it to several active learning strategies, including error-reduction and uncertainty sampling approaches specialised for cost-sensitive settings.

[1] Some authors use *cost-sensitive* for differing label acquisition costs between candidates (Liu et al. 2009).

2 Background and related work

Our work addresses cost-sensitive active learning for binary classification. Given the existing overviews on the various existing cost-**in**sensitive approaches in recent surveys such as Settles (2012), Fu et al. (2012), Cohn (2010) and Settles (2009), we focus on the most related approaches and start with cost-insensitive approaches before moving to cost-sensitive ones.

In expected *error reduction* (ER) (Roy and McCallum 2001; Cohn et al. 1996), the expected error upon incorporating a candidate into the training set is calculated. This is done by simulating for each of its possible label realisations the classifier update, and calculating the resulting error on an evaluation set (e.g. using the set of already labelled instances). In contrast to earlier work (Cohn et al. 1996), the error reduction approach in Roy and McCallum (2001) is usable with any classifier, and directly optimises a user-specified classification performance measure. Nevertheless, ER requires reliable estimates for the true posteriors (Chapelle 2005), which are difficult to obtain in early learning stages, where solely few labels are available. Therefore, several regularisation approaches, such as Beta priors, have been explored (Chapelle 2005). This family of approaches is known (see e.g. Settles 2012) to yield good results, but it is slow, requiring an asymptotic runtime of $O(|\mathcal{V}| \cdot |\mathcal{U}|)$, where $\mathcal{V}$ is the evaluation set and $\mathcal{U}$ the pool of candidates.

A fast active learning approach is *uncertainty sampling* (US) (Lewis and Gale 1994), which has an asymptotic time complexity of $O(|\mathcal{U}|)$ and is usable on fast data streams (Zliobaitė et al. 2013). It employs so-called *uncertainty measures* as proxies for a candidate's impact on the classification performance, and the candidate with the highest uncertainty is selected for labelling. In the seminal work of Lewis and Gale (1994), a probabilistic classifier is used on a candidate to compute the posterior of its most likely class. The absolute difference between this posterior estimate and 0.5 is used as uncertainty measure (lower values denoting higher uncertainty). In addition to this confidence-based uncertainty measure, other measures are common as well (Settles 2012), like entropy or the margin between a candidate and the decision boundary. Similar to the issue of the true posterior above, a known drawback (Zhu et al. 2010) of US is that these proxies do not consider the number of similar instances on which the posterior estimates are made or the decision boundaries are drawn. The reported results of empirical evaluations are somewhat inconclusive, with some authors [e.g. Chapelle (2005) or Schein and Ungar (2007)] reporting for US on some data sets even worse performance than random sampling.

Our recently (Krempl et al. 2014a, b) proposed *probabilistic active learning* (PAL) approach combines the qualities of uncertainty sampling and error reduction, namely being fast and optimising directly a performance measure. Following a smoothness assumption (Chapelle et al. 2006), our approach uses probabilistic estimates for summarising the labelled information in a candidate's neighbourhood and evaluating the impact of acquiring a label therein. This impact is expressed by the expected performance gain (the so-called probabilistic gain), measured in terms of an user-defined point classification performance measure (Parker 2011) like accuracy. Expectation is not only done over the possible realisations of a candidate's label as in error reduction, but also over the true posterior in the candidate's neighbourhood. PAL then selects the candidate that in expectation improves the classification performance the most within its neighbourhood. PAL runs in the same asymptotic time $O(|\mathcal{U}|)$ as uncertainty sampling and showed good results in cost-insensitive classification experiments (Krempl et al. 2014b), yet its suitability for cost-sensitive applications is an open question.

Cost-sensitive learning (Liu et al. 2009) is a particular challenging task for active learning algorithms, where misclassification costs are not equal among different classes. This occurs for example in fraud detection (Elkan 2001), where positives are rare, but misclassifying them (i.e. producing a false negative) is more costly than misclassifying a negative instance as positive. The objective is then to minimise the misclassification loss (Hand 2009), i.e. the cost-weighted sum of false positives and false negatives. A related, yet different problem is that of skewed or imbalanced class prior distributions (see He and Ma (2013) for an overview), where one class is far less frequent than the other. This latter problem is addressed in passive classification (where labels are known) by resampling (Chawla et al. 2002; Attenberg and Ertekin 2013), i.e. oversampling the minority or undersampling the majority class. In Attenberg and Ertekin (2013), active-learning-based approaches for resampling are reviewed. However, while resampling strategies are useful for creating a balanced training sample, they do not directly address the former problem of cost-sensitive classification itself. Furthermore, the reported empirical results in Elkan (2001), Liu (2009) suggest that their suitability for cost-sensitive classification is highly classifier dependent. Thus, we focus on approaches that directly address cost-sensitive classification.

In *passive* classification, unequal misclassification costs are addressed by using classification rules that minimise the conditional risk (Domingos 1999). A corresponding *active* learning strategy is to use cost-sensitive measures for label selection. This is done in the cost-sensitive variant of ER (Margineantu 2005), where misclassification loss is used as error measure, and varying label acquisition costs between instances are considered. Nevertheless, it inherits the slow runtime of ER and its issues associated with the ignorance of the true posterior. For query-by-committee approaches, which use the disagreement between an ensemble of classifiers as a proxy for a candidate's value, Tomanek and Hahn (2009) proposes a class-weighted, vote entropy-based measure as disagreement metric. However, this approach is specific for natural language processing, where active selection is between given conglomerates of labels.

Uncertainty measures can be made cost-sensitive by weighting posterior estimates with class-specific misclassification costs (Liu et al. 2009). However, an active learning component might induce a sampling bias, such that with additional labels the posterior estimates deviate further from the true posterior (Liu et al. 2009). This poses a problem especially in cost-sensitive classification tasks, where reliable posterior estimates are required to determine the misclassification-loss optimal decision boundary. Liu et al. (2009) addresses this by proposing a so-called cost-sensitive uncertainty sampling approach that performs self-labelling of all remaining unlabelled instances after each label request. This aims at de-biasing the training sample for a cost-sensitive classifier, but also increases the asymptotic time complexity to $O(|\mathcal{U}|^2)$. To the best of our knowledge, no direct empirical comparison between the approaches of Margineantu (2005) and Liu et al. (2009) has been published yet. Furthermore, they share another shortcoming in addition to requiring time-consuming steps: they are myopic, as they evaluate the impact of the next label acquisition without considering the remaining labelling budget. Nevertheless, as already stated in early works on active learning (Roy and McCallum 2001), the optimal query may very well depend on this remaining budget, which defines how many additional label requests will follow. Thus, extending active learning approaches to become non-myopic (also called far-sighted) is considered relevant (Zhao et al. 2012; Vijayanarasimhan et al. 2010). Vijayanarasimhan et al. (2010) proposed a far-sighted cost-sensitive active learning method for support vector machines that chooses a set of instances out of the pool of unlabelled candidates incorporating the individual labelling costs into the SVM's objective function. Zhao et al. (2012) select a set of instances greedily based on expected entropy reduction. They furthermore suggest to be near-optimal and define a stopping criterion for the active learning process.

3 Optimised probabilistic active learning (OPAL)

We address fast active learning for binary classification in a cost-sensitive environment, where the costs τ of a false positive classification potentially differ from that of a false negative one $(1-\tau)$. The objective of our approach is to select from the pool of unlabelled candidates the one that reduces the misclassification loss (Hand 2009) the most, once it is labelled and incorporated into the training set.

In the next Sect. 3.1, we provide the detailed modelling and derivation of our probabilistic performance gain estimate (G_{OPAL}), the pseudo-code[2] and a numeric example. This is followed by a discussion of OPAL's properties (Sect. 3.2), in particular under varying misclassification cost ratios and budgets. For convenience, we summarise in Table 1 the notation that is subsequently used.

3.1 Modelling and derivation of G_{OPAL}

Our approach follows a *smoothness assumption* [see ch. 1.2.1, p. 7 in Chapelle et al. (2006)], such that neighbouring positions in the feature space are assumed to have similar posteriors. Given a labelling candidate $(x, \cdot)$ from a pool of unlabelled instances $\mathcal{U}$, and a set $\mathcal{L}$ of already labelled instances (x, y), our approach needs to assess how well its neighbourhood has been explored, i.e. to count the number of already labelled instances that are similar to the candidate, in order to further assess the value of additional labels therein. Estimates on the *posterior probabilities* $\Pr(y|x)$ are *not sufficient*, as their normalisation cancels out the *absolute number* of labels, keeping solely information on the proportion of each class. Therefore, we resort to the unnormalised values. That is, we use the *absolute frequencies* for the number of labels of each class in the candidate's neighbourhood.

We differentiate between two neighbourhood concepts: The first, disjoint one applies to categorical or pre-clustered data. Such data allows to count the number $LC(x, \mathcal{L})$ of labelled instances that are similar to the candidate w.r.t. their features (or assigned cluster). These label counts for x are summarised by its *label statistics* $ls = (n, \hat{p})$, a tuple consisting of the absolute number n of labels in a candidate's neighbourhood, and the share $\hat{p}$ of positives therein. The second concept of smooth, continuous neighbourhoods corresponds for example to numerical data, where the influence of instances increases with the similarity of their features. In analogy to counts in the first case, we use frequency estimates in this second case. Using probabilistic classifiers that are modified to return unnormalised estimates for the absolute frequencies is one option. We recommend to use generative probabilistic classifiers like Naive Bayes rather than discriminiative ones like logistic regression. The information on the labelled data kept by the former by modelling $\Pr(X, Y)$ and $\Pr(X)$ allows to compute the label statistics directly. Furthermore, generative classifiers converge with fewer labels, as shown in Ng and Jordan (2001), which is important in active learning contexts. If these classifiers are not available, we propose to use Gaussian kernel frequency estimation (here, Σ is the bandwidth matrix):

$$LC(x, \mathcal{L}) \approx KFE(x, \mathcal{L}) = \sum_{x_i \in \mathcal{L}} \exp\left(-\frac{1}{2} \cdot (x - x_i)' \Sigma^{-1} (x - x_i)\right) \tag{1}$$

[2] Implementations are available on our companion website: http://kmd.cs.ovgu.de/res/opal.

Table 1 Used symbols and notation

Symbol	Description	Reference
Input data		
x	Feature vector of an instance	p. 5
$y \in \{0, 1\}$	Class label of an instance (0 = neg., 1 = pos.)	p. 5
$\mathcal{U} = \{(x, \cdot)\}$	Pool of unlabelled instances	p. 5
$\mathcal{L} = \{(x, y)\}$	Pool of labelled instances	p. 5
Variables imposed by the application domain		
$\tau \in [0, 1]$	Cost of each false positive classification	p. 5, p. 9, Eq. 16
$m \geq 0$	Budget for the candidate's neighbourhood	p. 7
Variables within the neighbourhood of a candidate $(x, \cdot)$		
$d_x \geq 0$	Density weight (w.r.t. all instances in $\mathcal{U} \cup \mathcal{L}$)	p. 6, Eq. 2
g_x	Density weighted optimised probabilistic gain	p. 11
$ls = (n, \hat{p})$	Label statistics with:	p. 5
n	Total number of already obtained labels	
$\hat{p}$	Share of positives therein (a posterior estimate)	
$k \in \{0, \ldots, m\}$	Number of positives among future label realisations	p. 7
$p \in [0, 1]$	True posterior probability of the positive class	p. 7
Functions		
$L(p\|ls)$	Likelihood of a possible true posterior	p. 7, Eq. 7
$\omega_p(S_{\text{label}}) \in [0, 1]$	Normalised likelihood of a possible true posterior	p. 8, Eqs. 10–12
$\Gamma(z)$	Legendre's gamma function, see pp. 206–208 in Press et al. (1992)	p. 7
$\mathrm{I}_{ML}(n, \hat{p}, \tau, m, k)$	Integral, proportional to the expected performance	p. 11, Eq. 32
$G_{\text{OPAL}}(n, \hat{p}, \tau, m)$ $\in [-1, 1]$	Optimised probabilistic gain, i.e. a candidate's Exp. average misclassification loss reduction	p. 11, Eq. 35

Based on this, we derive the total number of labels $n = LC(x, \mathcal{L})$ and the share of positives therein $\hat{p} = LC(x, \mathcal{L}_+)/LC(x, \mathcal{L})$, where $\mathcal{L}_+$ is the subset of labelled positive instances. The tuple $ls = (n, \hat{p})$ constitutes the label statistics of x's neighbourhood. Using Eq. 1, we also derive the density in the candidate's neighbourhood

$$d_x = \frac{LC(x, \mathcal{L} \cup \mathcal{U})}{|\mathcal{L} \cup \mathcal{U}|} \tag{2}$$

This serves as a weight for the importance of the classification performance within this neighbourhood, as compared to other regions in feature space. Therefore, we later weight the average misclassification loss reduction by this density-weight. It is useful to precompute d_x for all candidates in the pool, as $\mathcal{L} \cup \mathcal{U}$ is static in the pool-based active learning scenario.

3.1.1 A non-myopic, cost-sensitive probabilistic gain

Following our recently (Krempl et al. 2014a, b) proposed probabilistic active learning approach, we use a candidate's label statistics ls to compute its probabilistic gain, which corresponds to the expected gain in classification performance from acquiring the candidate's label. This is done by first modelling both, the candidate's label realisation y and the true posterior p of the positive class in its neighbourhood, as random variables, and computing the expectation over both variables simultaneously, using the normalised likelihood given the label statistics. The resulting *probabilistic gain* is weighted by the feature density d_x at its position, and the candidate with highest density-weighted probabilistic gain is selected.

However, in contrast to (Krempl et al. 2014a, b), our *optimised probabilistic active learning* (OPAL) offers three advantages for fast, cost-sensitive applications: first, it quantifies a candidate's probabilistic gain (its label's value) in terms of *misclassification loss reduction*, which is a cost-sensitive measure. Second, it uses a closed-form solution for computing the probabilistic gain, making it faster. Third, it is *non-myopic*, considering the effect of more than one label acquisition at once. Thus, given a budget that allows to acquire m labels at once within the neighbourhood, we compute the expectation over a set $y_1, y_2, \ldots, y_m$ of label realisations, rather than on a single label realisation y. However, the ordering of labels is irrelevant in this additional training set. By counting the number of positive realisations k in the possible sets, $m + 1$ different cases ($k = 0, 1, \ldots, m$) are distinguishable, and the number of positive realisations is a binomial-distributed random variable $K \sim \text{Bin}_{m,p}$. Thus, we perform the expectation over its realisation k and over the true posterior p, rather than over y and p as in Krempl et al. (2014b).

The true posterior in this neighbourhood is unknown, but for its possible values, the likelihoods are calculable by using the frequency estimates from the label statistics. For this, we model the unknown true posterior in this neighbourhood by a Beta-distributed random variable P. Its realisation p is itself the parameter of the Bernoulli distribution that controls the label realisation $y \in \{0, 1\}$ of any instance within the neighbourhood. Consequently, for any set of m label realisations in the neighbourhood, the number k of positives therein is the realisation of a Binomial-distributed random variable K:

$$P \sim \text{Beta}_{n\cdot\hat{p}+1,n\cdot(1-\hat{p})+1} \tag{3}$$

$$Y \sim \text{Bernoulli}_p = \text{Ber}_p \tag{4}$$

$$K \sim \text{Binomial}_{m,p} = \text{Bin}_{m,p} \tag{5}$$

We will denote the probability (density) functions (pdf's) of the above distributions by $\text{Beta}_{\alpha,\beta}()$, $\text{Ber}_p()$, and $\text{Bin}_{m,p}()$, respectively. For computing the binomial coefficient in $\text{Bin}_{m,p}(k) = \binom{m}{k} \cdot p^k \cdot (1-p)^{m-k}$, as well as in the subsequent equations below, we use the generalised binomial coefficient for non-integer arguments, and the gamma function $\Gamma(z)$ as defined by Legendre[3]:

$$\binom{m}{k} = \frac{\Gamma(m+1)}{\Gamma(k+1)\cdot\Gamma(m-k+1)} \tag{6}$$

The true posterior's Beta distribution above is the result of its normalised likelihood, given the already observed labels, as we will show below. According to Eq. 5, the likelihood of a true posterior p given the data summarised in ls corresponds to the probability mass function $\text{Bin}_{n,p}(n\hat{p})$:

[3] See for example pages 206–208 in Press et al. (1992).

$$L(p|\mathit{ls}) = L(p|(n,\hat{p})) = \mathrm{Bin}_{n,p}(n\hat{p}) \tag{7}$$

$$= \frac{\Gamma(n+1)\cdot p^{n\cdot\hat{p}}\cdot(1-p)^{n\cdot(1-\hat{p})}}{\Gamma(n\cdot\hat{p}+1)\cdot\Gamma(n\cdot(1-\hat{p})+1)} \tag{8}$$

Following a Bayesian approach, we consider a prior $g(p)$ for P, and obtain the normalised likelihood

$$\omega_{\mathit{ls}}(p) = \frac{L(p|\mathit{ls})g(p)}{\int_0^1 L(\psi|\mathit{ls})g(\psi)\mathrm{d}\psi} \tag{9}$$

The choice of a suitable prior $g(p)$ depends on our a priori information about the class prior distribution $\Pr(Y=+)$ in the data. Without any a priori information, we chose a uniform prior for P, i.e. $g(p) \sim U(0,1)$. As a result, $g(p)$ is a constant function, and the integral in the denominator sums up to $(1+n)^{-1}$, yielding $(1+n)$ as normalising constant:

$$\omega_{\mathit{ls}}(p) = (1+n)\cdot L(p|\mathit{ls}) \tag{10}$$

Expanding this using Eq. 7, and setting $(1+n)\cdot\Gamma(n+1) = \Gamma(n+2)$, we obtain precisely the probability function of the Beta-distribution:

$$\omega_{\mathit{ls}}(p) = \frac{\Gamma(n+2)\cdot p^{n\cdot\hat{p}}\cdot(1-p)^{n\cdot(1-\hat{p})}}{\Gamma(n\cdot\hat{p}+1)\cdot\Gamma(n\cdot(1-\hat{p})+1)} \tag{11}$$

$$= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\cdot\Gamma(\beta)}\cdot p^{\alpha-1}\cdot(1-p)^{\beta-1} = \mathrm{Beta}_{\alpha,\beta}(p) \tag{12}$$

Here, we rewrite in the last step the arguments of the Γ-functions and obtain $\alpha = n\cdot\hat{p}+1$ and $\beta = n\cdot(1-\hat{p})+1$. These parameters for the Beta-distribution have a correspondence in the positive and negative labels in the candidate's neighbourhood: with each positive label therein, α increases by one, while with each negative, β increases by one. Thus, the normalised likelihood expressed by the Beta-distribution is a uniform distribution if no labels are available. However, if labels are available, its peak around $\hat{p}$ becomes more and more distinct with an increase in the number of available labels.

In contrast to Krempl et al. (2014b), we do the expectation in OPAL over k and p, rather than over y and p, yielding the candidate's probabilistic gain (G_{OPAL}), defining the expected change of the performance measure for its neighbourhood in average per additional label:

$$G_{\mathrm{OPAL}}(\mathit{ls},\tau,m) = \frac{1}{m}\cdot\mathrm{E}_p\Big[\mathrm{E}_k\left[\mathrm{gain}_p(\mathit{ls},k,m)\right]\Big] \tag{13}$$

$$= \frac{1}{m}\cdot\int_0^1 \mathrm{Beta}_{\alpha,\beta}(p)\cdot\sum_{0\le k\le m}\mathrm{Bin}_{m,p}(k)\cdot\mathrm{gain}_p(\mathit{ls},k,m)\,\mathrm{d}p \tag{14}$$

Here, $\mathrm{gain}_p(\mathit{ls},k,m)$ is the performance gain within the neighbourhood with label statistics ls and true posterior p, given that k among m additional label realisations are positive. Using a point performance measure $\mathrm{perf}_p(\hat{p})$ that calculates the classification performance under a posterior estimate $\hat{p}$ and a true posterior p, the performance gain is written as difference between future and current performance:

$$\mathrm{gain}_p(\mathit{ls},k,m) = \mathrm{perf}_p\left(\frac{n\hat{p}+k}{n+m}\right) - \mathrm{perf}_p(\hat{p}) \tag{15}$$

We address active learning for cost-sensitive binary classification tasks, where $\tau \in [0;1]$ indicates the cost for each false positive instance, and $1-\tau$ is the corresponding cost for each false negative one, assuming zero costs for correct classifications. A point performance measure (Parker 2011) for this setting is misclassification loss (Hand 2009), which is the product of the misclassification cost matrix and the confusion matrix. Within a neighbourhood with true posterior p and a classification rule classifying a share of q instances therein as positive,[4] the misclassification loss is

$$MLoss(p,q) = p \cdot (1-q) \cdot cost_{FN} + (1-p) \cdot q \cdot cost_{FP} = \tag{16}$$

$$p \cdot (1-q) \cdot (1-\tau) + (1-p) \cdot q \cdot \tau = q \cdot (\tau - p) + p \cdot (1-\tau) \tag{17}$$

Thus, given p and τ, the misclassification loss is a linear function of $q \in [0;1]$. It has a positive slope for $p < \tau$, and a negative for $p > \tau$. Due to the positive slope, it is optimal to set $q = 0$ in the former case (and $q = 1$ in the latter), in order to minimise the loss function. Thus, the cost-optimal decision is:

$$q^* = \begin{cases} 0 & p < \tau \\ 1-\tau & p = \tau \\ 1 & p > \tau \end{cases} \tag{18}$$

One could argue that in the case $\tau = p$ the choice of q^* is an arbitrary one, as the first factor $q \cdot (\tau - p)$ is zero, meaning equal loss of $p^2 = \tau^2$ for all choices of q^*. However, in order to obtain a *consistent* classification rule, one should specify the assignment $q^* = 1 - \tau$ at ties, rather than simply replacing one strict inequality condition in Eq. 18 with a non-strict one. This is illustrated when studying the classification under extreme values for τ. For example, if $\tau = 0$, false positives do not cost anything, while false negatives are very expensive. A cost-optimal classification rule should thus classify every instance as positive, i.e. q^* should be one for all possible p. While cases of $p \in]0;1]$ are covered by the third clause in Eq. 18, the second clause must return $q^* = 1$ for cases of $p = 0$. Vice versa, if $\tau = 1$ this second clause must return $q^* = 0$. Thus, it should be set to $1 - \tau$ (or $1 - p$, equivalently). As the true posterior p is not directly observable, the counted observed share $\hat{p}$ of positives is used as proxy instead. Thus, ties might occur frequently enough to consider this as relevant.

Given p and τ, using negated misclassification loss (Eq. 16) under cost-optimal classification (Eq. 18), we derive a performance measure suited for Eq. 15:

$$perf_{p,\tau}(\hat{p}) = -ML_{p,\tau}(\hat{p}) = -\begin{cases} p \cdot (1-\tau) & \hat{p} < \tau \\ \tau \cdot (1-\tau) & \hat{p} = \tau \\ \tau \cdot (1-p) & \hat{p} > \tau \end{cases} \tag{19}$$

Intentionally, we do not include the density of the neighbourhood here, to measure the effect of this misclassification loss reduction for the whole data set, because we intend to separate data set specific information from this value. Of course, $ML_{p,\tau}(\hat{p})$ should have been multiplied with the neighbourhood's density d_x, but this factor can be delivered to the very left side of the whole formula.

Plugging this into Eq. 13 yields the probabilistic misclassification loss reduction

$$G_{\text{OPAL}}(ls,\tau,m) = \frac{1}{m} \cdot \int_0^1 \text{Beta}_{\alpha,\beta}(p) \sum_{k=0}^{m} \text{Bin}_{m,p}(k) \left(ML_{p,\tau}(\hat{p}) - ML_{p,\tau}\left(\frac{n\hat{p}+k}{n+m}\right)\right) \mathrm{d}p \tag{20}$$

[4] Classification within a neighbourhood is assumed to be indifferently of the precise location within the neighbourhood, i.e. we assume conditional independence of the posterior given the neighbourhood of an instance.

3.1.2 Derivation of the closed-form solution

For deriving a closed-form solution, we split Eq. 20 it into a term for to the expected current performance E_{cur} and another for the expected future performance E_{fut}:

$$G_{\mathrm{OPAL}}(\ell s, \tau, m) = \frac{1}{m} \cdot \left(E_{cur} - E_{fut}\right) \tag{21}$$

The first term E_{cur}, where $m = k = 0$ and $\mathrm{Bin}_{0,p}(0) = 1$, is simplified to:

$$E_{cur} = \int_0^1 \mathrm{Beta}_{\alpha,\beta}(p) \cdot ML_{p,\tau}(\hat{p})\,\mathrm{d}p \tag{22}$$

Expanding the Beta-distributed probability $\mathrm{Beta}_{\alpha,\beta}(p)$ by Eq. 12 and the misclassification loss by the case-by-case formula in Eq. 18, and integrating out yields:

$$E_{cur} = \int_0^1 \frac{\Gamma(n+2) \cdot p^{n\cdot\hat{p}} \cdot (1-p)^{n\cdot(1-\hat{p})}}{\Gamma(n\cdot\hat{p}+1)\cdot\Gamma(n\cdot(1-\hat{p})+1)} \cdot \begin{cases} p\cdot(1-\tau)\,\mathrm{d}p & \hat{p} < \tau \\ \tau\cdot(1-\tau)\,\mathrm{d}p & \hat{p} = \tau \\ \tau\cdot(1-p)\,\mathrm{d}p & \hat{p} > \tau \end{cases} \tag{23}$$

$$= (n+1)\cdot\binom{n}{n\cdot\hat{p}}\cdot\begin{cases} (1-\tau) & \cdot\ \frac{\Gamma(1+n-n\hat{p})\Gamma(2+n\hat{p})}{\Gamma(3+n)} & \hat{p} < \tau \\ (\tau-\tau^2) & \cdot\ \frac{\Gamma(1+n-n\hat{p})\Gamma(1+n\hat{p})}{\Gamma(2+n)} & \hat{p} = \tau \\ \tau & \cdot\ \frac{\Gamma(2+n-n\hat{p})\Gamma(1+n\hat{p})}{\Gamma(3+n)} & \hat{p} > \tau \end{cases} \tag{24}$$

The second term, E_{fut}, in Eq. 21 is:

$$E_{fut} = \int_0^1 \mathrm{Beta}_{\alpha,\beta}(p) \sum_{k=0}^{m} \mathrm{Bin}_{m,p}(k)\cdot ML_{p,\tau}\left(\frac{n\hat{p}+k}{n+m}\right)\mathrm{d}p \tag{25}$$

$$= \sum_{k=0}^{m} \cdot \int_0^1 \mathrm{Beta}_{\alpha,\beta}(p)\cdot\mathrm{Bin}_{m,p}(k)\cdot ML_{p,\tau}\left(\frac{n\hat{p}+k}{n+m}\right)\mathrm{d}p \tag{26}$$

As above, we expand the terms therein, which now include the Binomial-distributed probability $\mathrm{Bin}_{m,p}(k) = \binom{m}{k}\cdot p^k\cdot(1-p)^{m-k}$:

$$E_{fut} = \sum_{k=0}^{m}\int_0^1 \frac{\Gamma(n+2) \cdot p^{n\cdot\hat{p}} \cdot (1-p)^{n\cdot(1-\hat{p})}}{\Gamma(n\cdot\hat{p}+1)\cdot\Gamma(n\cdot(1-\hat{p})+1)}\cdot \tag{27}$$

$$\cdot\binom{m}{k}\cdot p^k\cdot(1-p)^{m-k}\cdot ML_{p,\tau}\left(\frac{n\hat{p}+k}{n+m}\right)\mathrm{d}p \tag{28}$$

$$= (n+1)\cdot\binom{n}{n\cdot\hat{p}}\cdot\sum_{k=0}^{m}\cdot\mathrm{I}_{ML}(n,\hat{p},\tau,m,k) \tag{29}$$

where I_{ML} is a function of n, $\hat{p}$, τ, m, and k, containing the integral and proportional to the expected performance, which is integrated out as follows:

$$\mathrm{I}_{ML}(n,\hat{p},\tau,m,k) = \int_0^1 \binom{m}{k} \cdot p^{n\cdot\hat{p}+k} \cdot (1-p)^{n+m-n\cdot\hat{p}-k} \cdot ML_{p,\tau}\left(\frac{n\hat{p}+k}{n+m}\right) \mathrm{d}p \quad (30)$$

$$= \binom{m}{k} \cdot \int_0^1 p^{n\cdot\hat{p}+k} \cdot (1-p)^{n+m-n\cdot\hat{p}-k} \cdot \begin{cases} p\cdot(1-\tau)dp & \frac{n\hat{p}+k}{n+m} < \tau \\ (\tau-\tau^2)dp & \frac{n\hat{p}+k}{n+m} = \tau \\ \tau\cdot(1-p)dp & \frac{n\hat{p}+k}{n+m} > \tau \end{cases} \quad (31)$$

$$= \binom{m}{k} \cdot \begin{cases} (1-\tau) & \cdot \frac{\Gamma(1-k+m+n-n\hat{p})\Gamma(2+k+n\hat{p})}{\Gamma(3+m+n)} & \frac{n\hat{p}+k}{n+m} < \tau \\ (\tau-\tau^2) & \cdot \frac{\Gamma(1-k+m+n-n\hat{p})\Gamma(1+k+n\hat{p})}{\Gamma(2+m+n)} & \frac{n\hat{p}+k}{n+m} = \tau \\ \tau & \cdot \frac{\Gamma(2-k+m+n-n\hat{p})\Gamma(1+k+n\hat{p})}{\Gamma(3+m+n)} & \frac{n\hat{p}+k}{n+m} > \tau \end{cases} \quad (32)$$

Using this in Eqs. 24 and 29, we obtain

$$E_{cur} = (n+1)\cdot\binom{n}{n\cdot\hat{p}}\cdot \mathrm{I}_{ML}(n,\hat{p},\tau,0,0) \quad (33)$$

$$E_{fut} = (n+1)\cdot\binom{n}{n\cdot\hat{p}}\cdot \sum_{k=0}^{m} \cdot \mathrm{I}_{ML}(n,\hat{p},\tau,m,k) \quad (34)$$

and Eq. 35 to compute the G_{OPAL} in the candidate's neighbourhood:

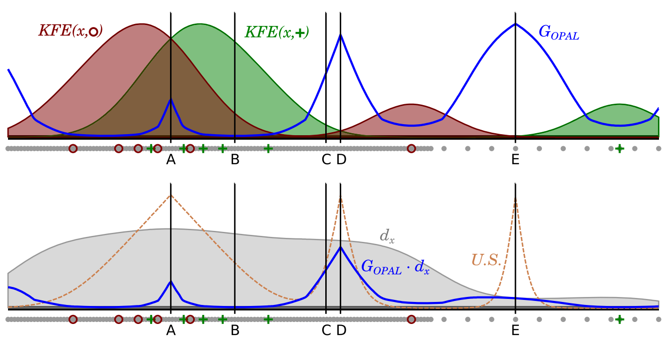
$$G_{\mathrm{OPAL}}(n,\hat{p},\tau,m) = \frac{(n+1)}{m}\cdot\binom{n}{n\cdot\hat{p}}\cdot\left(\mathrm{I}_{ML}(n,\hat{p},\tau,0,0) - \sum_{k=0}^{m}\mathrm{I}_{ML}(n,\hat{p},\tau,m,k)\right) \quad (35)$$

3.1.3 Pseudocode and numeric examples

The pseudocode for OPAL in pool-based active learning is given in Fig. 1. Lines 2–7 iterate over each labelling candidate $(x,\cdot)$ in the pool $\mathcal{U}$. First (line 3), a candidate's label statistics $\mathfrak{ls} = (n_x, p_x)$ are computed according to Eq. 1. Second (line 4), the density weight d_x is estimated, i.e. the proportion of all labelled and unlabelled instances within the candidate's neighbourhood divided by those in all neighbourhoods, see Eq. 2. In line 5, the optimal m_x^* that maximises the G_{OPAL} (see Eq. 35) is found, using a *logarithmic search* over $m' = 1, 2, \ldots, m$. Density-weighting this maximal G_{OPAL} (line 6) yields g_x. , and the candidate maximising the density-weighted probabilistic gain is returned (line 8).

```
1: function PoolBasedOPAL($\mathcal{U}$,$\mathcal{L}$,$\tau$,$m$)
2:     for $x \in \mathcal{U}$ do
3:         $(n_x, \hat{p}_x) \leftarrow labelstatistics(x, \mathcal{L})$
4:         $d_x \leftarrow densityweight(x, \mathcal{L} \cup \mathcal{U})$
5:         $m_x^* \leftarrow \arg\max_{m' \in 1,2,\cdots,m} G_{\mathrm{OPAL}}((n_x, \hat{p}_x), \tau, m')$
6:         $g_x \leftarrow G_{\mathrm{OPAL}}((n_x, \hat{p}_x), \tau, m_x^*) \cdot d_x$
7:     end for
8:     return $\arg\max_{x \in \mathcal{U}}(g_x)$
9: end function
```

Fig. 1 The OPAL algorithm



	$\hat{p}$	n	d_x
A	.5	6.0	.16
B	.8	3.6	.15
C	.8	0.5	.14
D	.5	0.4	.13
E	.5	0.1	.02

	G_{OPAL}	
	$\cdot 1$	$\cdot d_x$
A	.0278	.00435
B	.0011	.00016
C	.0459	.00617
D	.0737	.00982
E	.0817	.00164

Fig. 2 Visualisation of G_{OPAL}-values for $\tau = 0.5$ on a one-dimensional dataset with labelled (red resp. *green dots*) and unlabelled (*grey dots*) data points. The upper plot shows the kernel frequency estimates (KFE) for each class and the corresponding G_{OPAL}-value (*blue curve*). The lower plot shows the density (*grey area*) and the density-weighted G_{OPAL}-values (*blue curve*). Additionally, the negative confidence values from the Uncertainty Sampling approach are plotted for comparison. For exemplary data points (A–E) the corresponding label statistics and the unweighted ($\cdot 1$) and density weighted ($\cdot d_x$) G_{OPAL}-values are given in the tables

The G_{OPAL}, visualised in Fig. 2, corresponds to the expected *average* reduction in misclassification loss in each subsequent classification[5] in the candidate's neighbourhood. For equal misclassification costs, the G_{OPAL} is proportional to the expected average gain in accuracy and is highest for candidates close to the decision boundary (where $\hat{p} \approx \tau$), like the points D and E (compared to B and C) in Fig. 2. For a very small number n of already obtained similar labels, G_{OPAL} approximates random sampling as $n \to 0$, corresponding to the barely available information. Nevertheless, as n increases (compared to the remaining budget m), the equations above are dominated by the observed posterior $\hat{p}$. Thus, the difference between expected future $\mathrm{I}_{ML}(n, \hat{p}, \tau, m, k)$ and current performance $\mathrm{I}_{ML}(n, \hat{p}, \tau, 0, 0)$ converges towards zero, making candidates in well-explored regions (e.g. A) less valuable than those in unexplored ones (e.g. D, E). In the lower subplot in Fig. 2, the points (D, E) show the importance of the density-weighting: Point E is in a less explored but also sparser area than D, thus E has a higher G_{OPAL} (0.0817 vs. 0.0737) but a 6.5-times lower density weight, as improving the performance in its region will effect 6.5 times fewer future classifications. Thus, the density-weighted probabilistic gain of E (0.00164) is lower than that of D (0.00982). In contrast, US neither incorporates the amount of available information (e.g. A vs. D), nor the importance of neighbourhoods (e.g. D vs. E). Note that for unequal misclassification costs, the G_{OPAL} is not symmetric around τ, but rather favours sampling instances from the regions where potentially a more costly error is made. That is, if false positive costs are relatively low compared to false negative ones (e.g. $\tau = 0.1$), misclassification of positives (as false negatives) is expensive compared to the misclassification of negatives. Accordingly, the probabilistic gain is higher in regions where currently instances are classified as negative, as the possible error therein is more expensive. Therefore, our cost-sensitive approach will favour sampling in these regions. A further discussion of the properties of G_{OPAL} is provided in Sect. 3.2.2, where Fig. 3 on page 14 illustrates the shape of this function.

3.2 Properties of G_{OPAL}

We now briefly discuss the asymptotic (with respect to data set size) computational time complexity of OPAL in Sect. 3.2.1, comparing it to related algorithms for active learning of

[5] Assuming cost-optimal classification (Domingos 1999), see Eq. 18 in Sect. 3.1.

binary, incremental classifiers, before illustrating the effect of the myopic extension on the probabilistic gain in Sect. 3.2.2.

3.2.1 Computational complexity

For the non-myopic selection of a candidate from a pool $\mathcal{U}$ of labelling candidates, OPAL iterates first over all candidates in the pool (lines 2–7). Each iteration consists of (1) querying label statistics, (2) querying density weights, (3) determining the locally optimal budget, and (4) computing the density-weighted probabilistic gain. The first step requires absolute frequency estimates of labels in the candidate's neighbourhood, similar to the relative frequency estimates needed by entropy or confidence uncertainty measures. These are obtained in constant time by probabilistic classifiers. The second step requires density estimates over all instances, that is over labelled $\mathcal{L}$ and unlabelled $\mathcal{U}$ ones. Precomputing these density estimates once for all later calls of OPAL leads to constant query time, as in the pool-based setting the union $\mathcal{L} \cup \mathcal{U}$ is constant. The third step requires a logarithmic search over all possible $m' \in 1, 2, \ldots, m$, where for each m' the G_{OPAL} is calculated. The latter is done in $O(m)$ time, due to the closed-form solution obtained for Eq. 35. Thus, the third step requires $O(m \log(m))$ time. The fourth step computes the density-weighted probabilistic gain g_x, requiring constant time. For the subsequent selection of the best candidate in line 8, the maximum of g_x as well as the index of the corresponding candidate are kept. Thus, the iteration over the pool is determining the overall asymptotic time complexity of $O(|\mathcal{U}| \cdot m \log(m))$ for the *non-myopic* OPAL, where m is the remaining labelling budget that is in general much smaller than $|\mathcal{U}|$. OPAL's *myopic* counterpart requires asymptotically linear time $O(|\mathcal{U}|)$, as $m = 1$.

In comparison, uncertainty sampling also requires asymptotically linear time $O(|\mathcal{U}|)$, whereas error reduction as discussed in Settles (2012) requires $O(|\mathcal{U}| \cdot |\mathcal{V}|)$ time, where $|\mathcal{V}| \approx |\mathcal{U}|$, as $\mathcal{V}$ needs to be a representative sample of the data.

3.2.2 Effect of the non-myopic extension

Besides of being faster and cost-sensitive, OPAL extends PAL by adding the ability of acting non-myopic. The first two properties are the result from using a closed-form solution and misclassification loss as performance measure, and have already been discussed. Thus, we will now focus on the third one, which allows OPAL to consider a budget m when computing the probabilistic gain of a candidate.

We illustrate the usefulness and effect of this extension on the probabilistic gain function in Fig. 3. In this figure, the first two columns of plots show the probabilistic gain function (in terms of average misclassification loss reduction) for different label statistics, i.e. combinations of different numbers of already obtained labels n and observed posteriors $\hat{p} = \hat{Pr}(+|x)$. The first column shows the myopic probabilistic gain ($m = 1$), the second the non-myopic one ($m = m^*$), where $m \in \{1, 2, \ldots, 21\}$ is chosen such that the probabilistic gain is maximal. The third column corresponds to the difference (in a logarithmic scale) between the two probabilistic gains. The three rows correspond to the different misclassification costs $\tau = 0.1$, 0.25, and 0.5. The plots for $\tau = 0.75$ (and $\tau = 0.9$) are not shown, as they are reflection symmetric to those of $\tau = 0.25$ (and $\tau = 0.1$, respectively).

Given equal misclassification costs ($\tau = 0.5$, third row) and a candidate in a neighbourhood, where already two labels (all positive) have been acquired ($n = 2$, $\hat{p} = 1$). In this neighbourhood, a single additional label can not alter the classification decision, thus the probabilistic gain in a myopic setting is zero. This is seen in the left-bottom plot, where the

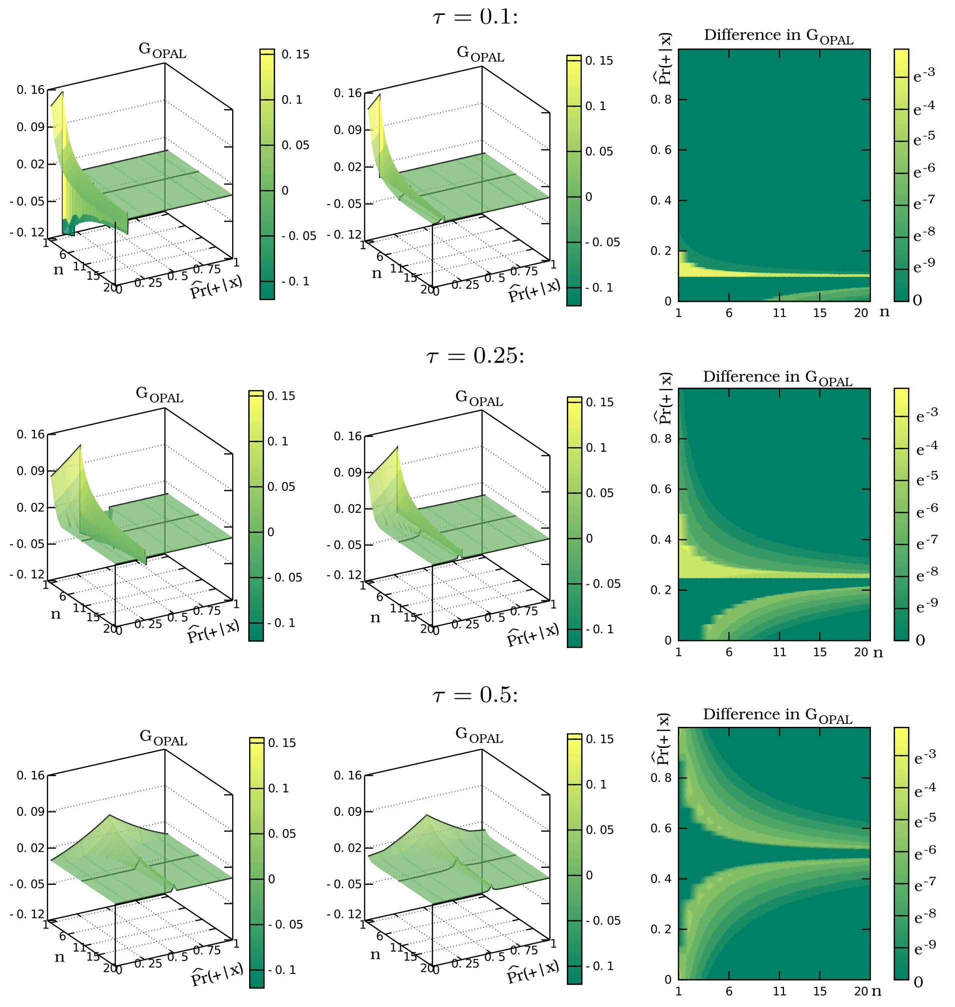


Fig. 3 Plots of the G_{OPAL} as function of observed posterior $\hat{p} = \hat{Pr}(+|x)$ and number of labels n for different cost-rations $\tau = 0.1, 0.25, 0.5$ (*rows*). The left column shows the myopic G_{OPAL}, the centre column shows the non-myopic G_{OPAL}, and the right column shows the difference between the two (in logarithmic scale)

probabilistic gain is zero for $n = 2$, $\hat{p} = 1$ (the corner in the uttermost back). However, if more than one label can be acquired in this neighbourhood, these labels might change the classification. Thus, under a non-myopic setting and equal misclassification costs, the probabilistic gain should be positive. Indeed, the probabilistic gain is $G_{\text{OPAL}} = 0.0274$ for the optimal acquisition of three labels ($m^* = 3$). Correspondingly, the flanks in the centred plots are flatter than in the left ones.

For unequal misclassification costs (e.g. in first row with $\tau = 0.1$, meaning false positives are cheap compared to false negatives), the expected gain from a single additional label might even be negative. This corresponds to situations, where just sufficiently positive labels were acquired within a neighbourhood to classify instances therein as positive (i.e. $\hat{p} = \hat{Pr}(+|x) > \tau$). In the myopic setting, the realisation of the single additional label is binary. If it is positive, it does not alter the classification. If it is negative, it inverts the classification. The latter results in a wrong classification if the true posterior p is actually greater τ, which is very likely, given that the share of positives $\hat{p}$ among the already seen labels was greater τ. Therefore, in the left-upper plot, the myopic probabilistic gain is negative for $n = 1$ and $\hat{p} \in [0.1, 0.2]$, with a negative peak at $G_{\text{OPAL}} = -0.12$ for $\hat{p} = 0.199$.

In contrast to the myopic setting with its binary label realisation, the non-myopic setting uses a rational number: the share of positives among the realisation of additional labels. Thus, the effect of this special case decreases with increasing m, as shown in the upper-centre plot. Nevertheless, for extreme misclassification cost inequalities (e.g. $\tau = 0.1$ or $\tau = 0.9$), the probabilistic gain remains non-positive for all neighbourhoods with $\hat{p} > \tau$, which are therefore not selected for label requests. In contrast, for moderate misclassification cost inequalities (e.g. $\tau = 0.25$ or $\tau = 0.75$), it becomes positive for some neighbourhoods therein, namely those having an observed posterior $\hat{p}$ close to τ. As a consequence, the effect of the non-myopic extension might be more important for situations with moderate misclassification cost inequalities, than for those with either equal misclassification costs or extreme unequal ones. However, this requires empirical evaluation, which we provide in Sect. 4.3.

Concerning the probabilistic gain function's mode, our numerical experiments indicate it to be an unimodal function of m for a given combination of n, $\hat{p}$ and τ.

4 Experimental evaluation

We expect our new *cost-sensitive* method OPAL to perform *at least equally well* in terms of resulting classification performance as other (cost-sensitive) active learning approaches, while being *faster* than other cost-sensitive approaches. Furthermore, we expect OPAL to be better than PAL towards the end of the learning process, through its *non-myopic* extension. Therefore, we designed a framework that ensures a fair evaluation of our contributions.

In the first subsection, we describe our evaluation setting, the active learning approaches used in the comparison, the data sets and our framework. In the second subsection, we present and discuss the results of the experimental evaluation. There, we first assess the usefulness of our cost-sensitive extension. Then, we show that OPAL is in most cases superior, both to its myopic counterpart PAL and to other active learning approaches, while having the same asymptotic time complexity as uncertainty sampling.

4.1 Evaluation settings

4.1.1 Active learning algorithms

For experimental evaluation, we use the fast version of *OPAL* described in Sect. 3, which applies a logarithmic search for determining the optimal budget. In pretests, there was no significant difference in classification performance between this approach and another variant of OPAL doing exhaustive search. In addition, we use a cost-sensitive, myopic *PAL* (Krempl et al. 2014b) with the presented speed optimisation (here denoted as csPAL). This is the equivalent to OPAL with a fixed budget of $m = 1$.

Furthermore, we use a cost-sensitive variant of *Uncertainty Sampling* (Liu et al. 2009) (denoted as U.S.) and *Certainty Sampling* (Ferdowsi et al. 2011) (denoted as C.S.), which both optimise confidence[6] (posterior difference to 0.5). Here, the posterior probabilities are calculated from a cost-weighted frequency estimation. Liu et al. (2009) proposed to use a

[6] We tested confidence- and entropy-based uncertainty measures in pretests and used the one with the best performance over all data sets for the final evaluation.

self-training approach for uncertainty sampling, such that posterior estimates are optimised for confidence calculation. We denote this extension as U.S. st.

For error reduction, we use the cost-sensitive algorithm proposed by Margineantu (2005) (denoted as Marg) and the non-cost-sensitive version by Chapelle (2005) (denoted as Chap). As a non-myopic representative, we use the method by Zhao et al. (2012) (denoted as Zhao). As the latter originally needs initial labels, we use a beta-correction for the classifier predictions of 0.001 (like for Chap). This simulates that in each evaluation neighbourhood an equal number of positives and negatives has been seen. In our framework, we always use 40 labels to be acquired by the active learners. Therefore, we disabled the automatic stopping criterion (otherwise learning often stopped far too early). Furthermore, we use random selection (denoted as Rand) as a baseline.

4.1.2 Data sets

In our experiments, we used 3 synthetic and 5 real data sets (from Asuncion and Newman (2013)). Each attribute was scaled to a [0; 1]-range, because we use Gaussian Kernel Frequency estimates with a fixed and pre-tuned bandwidth $sigma$ (see Sect. 1). The main characteristics (number of instances, number of attributes), such as training and test set size and the bandwidth σ of the Gaussian Kernel, are given in Table 2.

Two of the synthetic data sets are based on the generator used in Chapelle (2005). They consist of $4x4$ clusters, arranged in a checker-board formation (on a 2 dimensional feature space). While the clusters are low-density-separated in `Che` (as in Chapelle (2005)), they are adjoined in `Che2`. The third synthetic data set (`Sim`) consists of two normal distributed, overlapping clusters in a two dimensional space. We used this very simple example as a proof of concept for active learning methods.

The real-world data sets are Seeds (`See`), Vertebral (`Ver`), Mammographic mass (`Mam`), Yeast (`YeaU`) and Abalone (`Aba`), see Asuncion and Newman (2013). As show in Table 2, balanced as well as unbalanced class distributions occur. Categorical features (as in `Mam`) have been dichotomised into multiple binary features. In `Mam`, instances with missing values have been removed. For the multi-class dataset `See`, we classified Kama and Canadian vs. Rose; for `Ver`, we used normal vs. abnormal; for `Aba`, we used trees with rings <10 vs.

Table 2 Data set characteristics and parameters (number of instances, number of attributes (real-valued, categorical), proportion of positive instances, training set size, test set size, bandwidth for Parzen window classifier) in ascending training set size order

Data set	Instances	Attributes		$\mathcal{P}r(+)$	|Train|	|Test|	σ
		Real	Cat.				
`See`	210	7	–	33 %	160	50	0.2
`Che`	308	2	–	44 %	200	108	0.08
`Che2`	392	2	–	49 %	250	142	0.08
`Ver`	310	6	–	32 %	260	50	0.08
`Mam`	830	2	2	51 %	630	200	0.7
`Sim`	1200	2	–	50 %	800	400	0.08
`YeaU`	1484	8	–	90 %	1000	484	0.1
`Aba`	4177	8	–	50 %	3500	677	0.25

rings $\geq$10; for `YeaU`, we used MIT vs. the rest. For better comparison to Chapelle (2005) and Krempl et al. (2014b), we used a Parzen window classifier, which is a generative probabilistic classifier as discussed in Sect. 3.1. However, for some cost-ratios this classifier is not able to discriminate in the data, thus it is classifying all instances into the more expensive class. This is detectable in the bandwidth tuning curves (see Fig. 1), when the best σ-value (the one with lowest misclassification loss) is the maximal, uttermost left one. We reported the results on those data-set/cost-ratio-combinations for completeness, but emphasise that the classification performance on such ill-posed learning problems is not meaningful.

4.1.3 Framework

Our framework decouples classification and active learning. All runs behave exactly the same except for the active sampling component, which decides the instance whose label should be acquired next. Thus, we use exactly the same frequency estimates (see Eq. 1) and the same classification algorithm (a Parzen window classifier (Chapelle 2005)) with the identical parameters for any active learning strategy during the calculation. Furthermore, we decoupled the classification and evaluation process to ensure, that every active learning method just differs in the set of labelled instances. To get more significant results, we used a cross-validation with random sub-samplings in 100 runs. The training and test set sizes are listed in Table 2.

Here, active learning starts *without* initial labels on the unlabelled training sample, and finishes after 40 label acquisitions (steps). We implemented the framework in Octave/MATLAB, which was parallelised to run on a cluster. Every run uses the same pre-tuned, data set-specific bandwidth, and each of the 40 steps is evaluated on the same, dedicated (labelled) test sample with the same cost-sensitive Parzen window classifier, recording misclassification loss and speed.

The presented learning curves show the arithmetic mean of the misclassification loss over all 100 runs for a given combination of data set, algorithm and cost-ratio.

4.2 Relevance of the cost-sensitivity

From OPAL's theoretical characteristics, we expect OPAL to choose the best instances, with respect to a given cost-ratio τ. To evaluate the relevance of this cost-sensitivity empirically, we run OPAL for its G_{OPAL} calculation with 5 different τ values ($\tau \in \{.1, .25, .5, .75, .9\}$), and evaluate its misclassification loss regarding the true cost-ratio τ^*. If the cost-sensitivity is meaningful and relevant, the curve with $\tau = \tau^*$ should have the best performance compared to all other τ-values.

Figure 4 shows a selection of data sets (columns) and the evaluation cost-ratios τ^* (rows). Each plot shows the learning curves with the classification performance in terms of misclassification loss on its y-axis and the steps of its learning process in the number of already requested labels on the x-axis. The 5 different variants of OPAL with the varying τ are printed in different colours while the correct one is plotted in bold. The results of the other data sets are given in the appendix (see Sect. 1; Fig. 8).

The first interesting fact is that the correct usage of $\tau = \tau^*$ leads to a converging curve, while some wrong ones lead to diverging curves. Thus, ignoring the application-specific cost-ratio results in a low classification performance on these data sets. Furthermore, the curves for neighbouring τ-values behave similarly, especially the curves for $\tau = 0.25$ and $\tau = 0.1$, respectively $\tau = 0.75$ and $\tau = 0.9$.

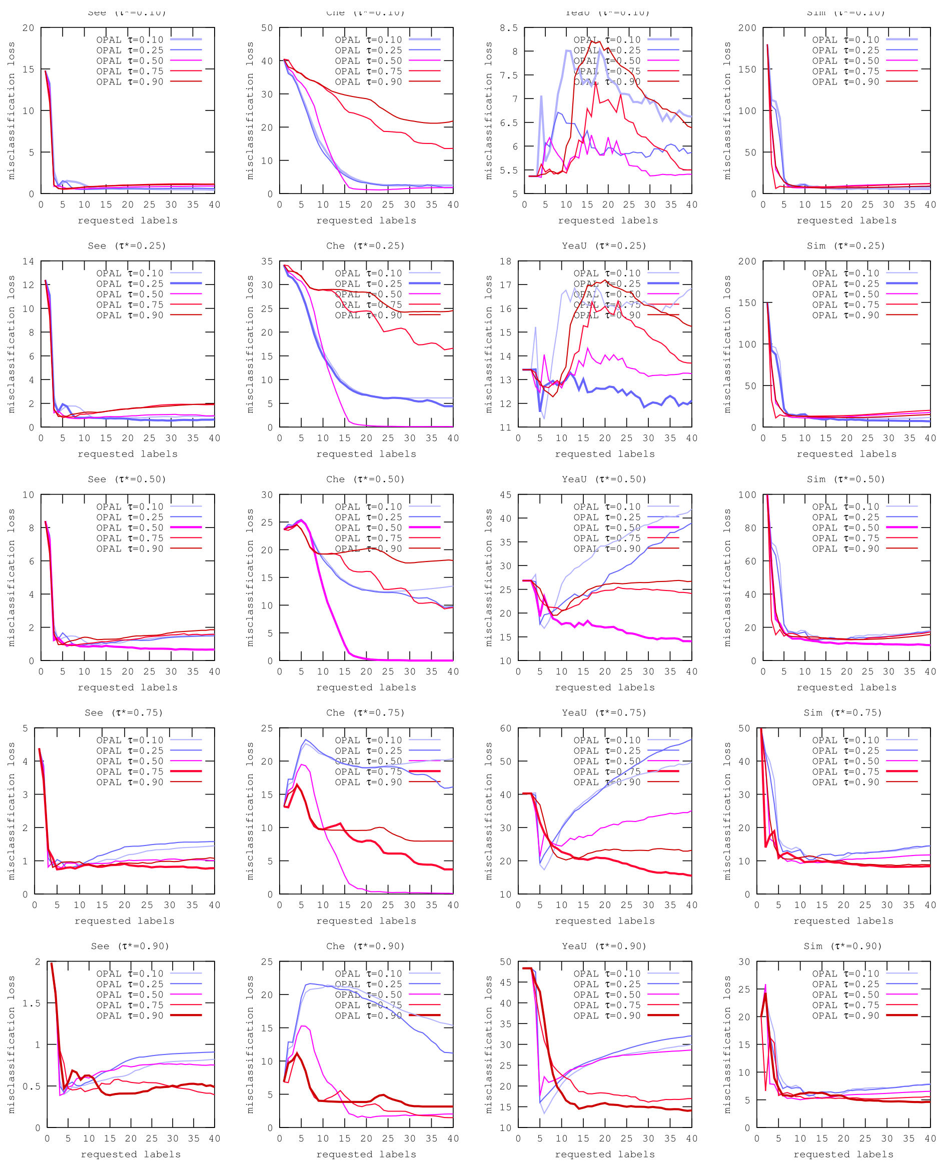


Fig. 4 Misclassification loss curves for OPAL on a selection of data sets with different cost-ratios (τ) for active learning and true cost-ratios (τ^*) for evaluation; curves for correct cost-ratios ($\tau = \tau^*$) are plotted in bold and should be superior; early convergence to very low values is best

Comparing the level and velocity of the misclassification curves, the bold ones are mostly superior. The single exception occurs on Che, where OPAL selects optimal labels for $\tau = 0.5$ (that is one instance of each cluster), so it performs very well for the other evaluation cost-ratios too. This is due to the very special characteristics (well-separated clusters) of this data set. When the separation between clusters is reduced, as in Che2, the effect vanishes. The other exceptions, like in YeaU for $\tau^* = .1$, occur all on ill-posed learning problems (see Fig. 1), where the results are not meaningful.

Summarising the results, the use of the cost-sensitive extension is beneficial, although there are some exceptional cases, where $\tau \neq \tau^*$ achieves better performance due to special structure in the data. It is noteworthy that in a real-world application, τ is not a tunable parameter but rather imposed by the application domain. The results show that ignoring this application-specific cost-ratio will in most cases result in a non-optimal classification performance.

4.3 Comparison between OPAL and other active learning strategies

This subsection assess whether (1) OPAL's non-myopic extension is beneficial for a given labelling budget, (2) OPAL's performance is superior or at least equal to other active learning approaches, and (3) its time complexity increases solely linearly with training set size (like uncertainty sampling).

For comparison, we use learning curves measuring the performance in terms of misclassification loss as before. The plots for all data sets and algorithms are given in Figs. 5 and 6. The best active learning method is the one, that has a fast-converging, low final misclassification loss level. Furthermore, we give a numerical value (see Table 3) for comparing two algorithms directly over all 8 data sets. It is computed as the portion of OPAL being better than the compared algorithm on all 100 runs of all data sets (thus on 800 pairs) for a given cost-ratio (τ^*) and labelling step. We also report the results of one-sided Wilcoxon signed-rank tests with significance level 0.001 on these pairs, by indicating a significantly better performance of OPAL by *, and a significantly worse performance by †.

(1) OPAL's non-myopic extension is beneficial Here, we compare the non-myopic OPAL and the myopic csPAL. Both algorithms just differ in their available budget size. While OPAL chooses the best G_{OPAL} for a given m-vector, csPAL just considers the very next possible label ($m = 1$).

As already discussed in Sect. 3.2.2, the learning curves for $\tau^* = 0.5$ of both algorithms are quite similar. Furthermore, the tables state that OPAL is at most in 4 % better than csPAL. This value might confuse, but one must be reminded, that this is just the portion of OPAL being *better*. Because there is no significance of being worse, we can derive that OPAL and csPAL behave quite similar (have the same values). However, if the number of already obtained labels is very high, the myopic G_{OPAL} used in csPAL will get zero, resulting in a random-sampling-like behaviour. In contrast, if m is sufficiently large, meaning that still several more labels will be acquired, the non-myopic variant in OPAL is advantageous, as it will still perform a differentiated selection.

For $\tau^* \neq 0.5$, the non-myopic variant is slightly advantageous in terms of final misclassification loss (e.g. `Mam` for $\tau^* = 0.75$, `YeaU` for $\tau^* = 0.9$ or `See`) or at least performs equally well. Interestingly, while for extreme cost-ratios ($\tau = 0.1$ or $\tau = 0.9$) the non-myopic variant is still advantageous over its myopic counterpart (in 44 or 48 % of the cases, see Table 3), the difference is not as big as it is for moderately unequal cost-ratios ($\tau = 0.25$ or $\tau = 0.75$). However, this is in accordance to the theoretical observations made in Sect. 3.2.2, where a strongest effect for moderately unequal misclassification cost ratios was predicted.

Furthermore, we observe that csPAL converges faster (see e.g. Table 3 for 10 label acquisitions). However, this again matches with the theoretical discussion in Sect. 3.2.2, because we set the budget initially to $m = 40$ (and not to 10), thus OPAL has optimised its learning path for 40 label acquisitions. Evaluating its performance after less steps is therefore slightly

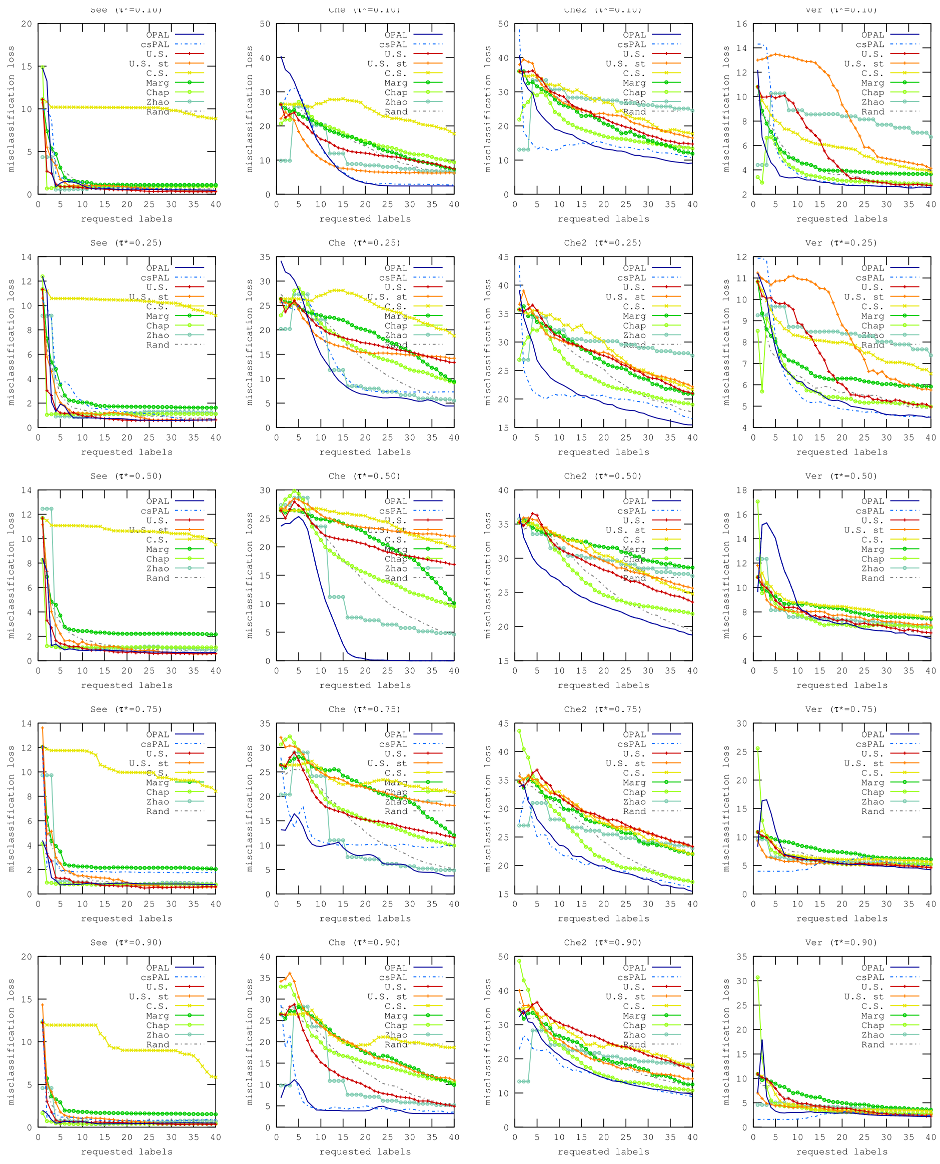


Fig. 5 Misclassification loss curves for presented algorithms on all data sets with different evaluation cost-ratios (τ^*); early convergence to very low values is best

malicious. However, the results indicate that (1) setting the budget correctly to the remaining one is beneficial for final classification performance, and (2) a faster learning is achievable by setting m to low values, if one is willing to forfeit long-term performance.

(2) OPAL's performance is superior Measuring the overall performance of active learning methods over different data sets and cost-ratios is complex, due to weighting and measuring the characteristics of learning curves, which ideally converge fast to a low final misclassification loss level. Therefore, Table 3 provides a summary of the total percentage of wins of OPAL

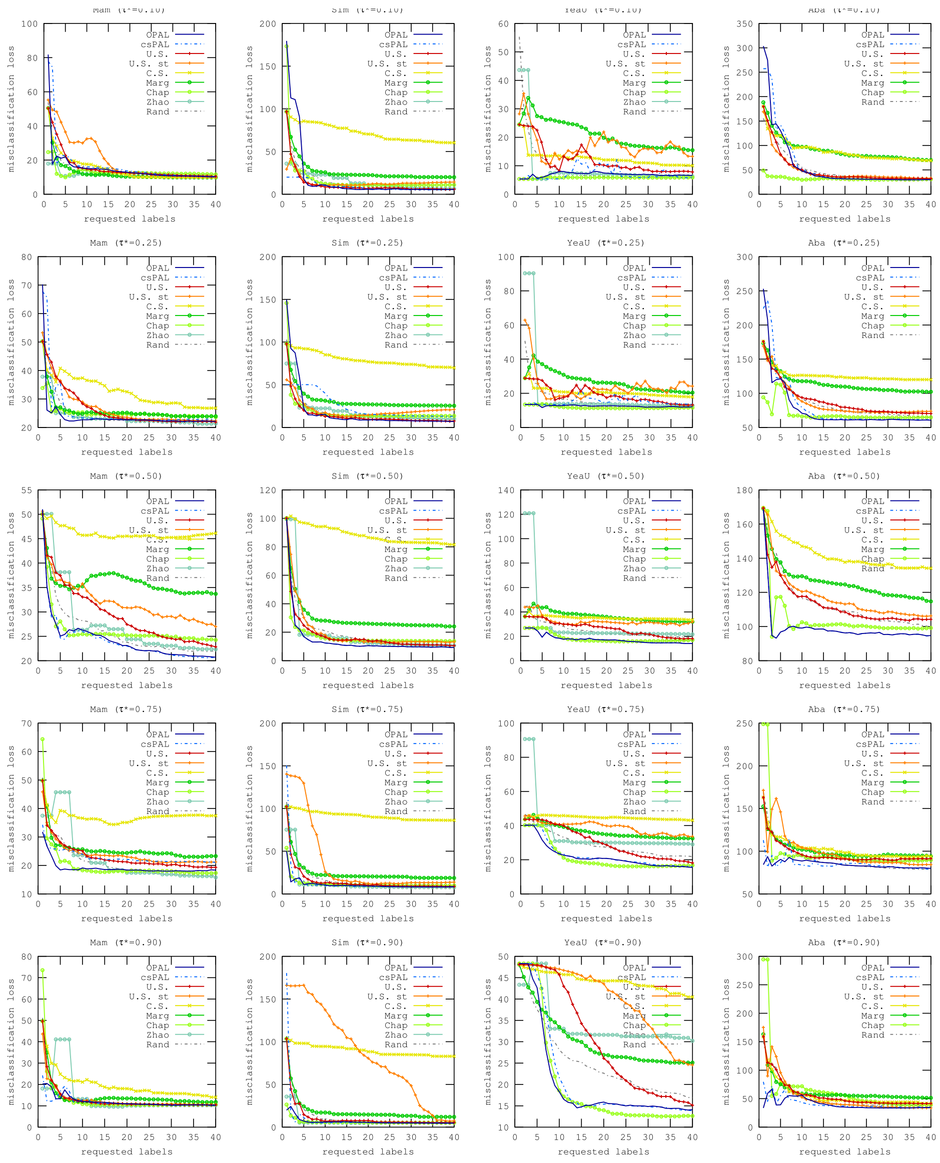


Fig. 6 Continuation of Fig. 5 on additional data sets

against each other approach. The learning curves show that OPAL is better than U.S. (with and without self-training) after 6 label requests (when learning becomes meaningful) in most cases. Explanations according to Settles (2012, pp. 19–20) are that US (a) ignores the extend of exploration in a neighbourhood, (b) relies on a hypothesis biased by its sampling, and (c) is fairly myopic. We can not confirm a superiority of C.S. (Ferdowsi et al. 2011) compared to any other (even random) active learning approach in our experiments. The cost-sensitive error reduction method Marg performs worse than expected. It is outperformed by its cost-insensitive counterpart Chap, maybe due to solely using labelled instances for evaluation, as opposed to the self-labelling used by Chap. Chap and the non-myopic, cost-insensitive error

Table 3 Percentages of runs over all data sets, where OPAL performs better than its competitor. Significantly better performance is denoted by *, significantly worse performance by †

	OPAL vs.							
	csPAL	U.S.	U.S. st	C.S.	Marg[1]	Chap[1]	Zhao[1]	Rand
10 Labels acquired								
$\tau^* = 0.10$	38% †	51 %	52 %*	62% *	58% *	47 %	64% *	54% *
$\tau^* = 0.25$	60% *	66% *	68% *	82% *	76% *	61% *	73% *	66% *
$\tau^* = 0.50$	1 %	70% *	74% *	89% *	80% *	62% *	69% *	72% *
$\tau^* = 0.75$	46 %	62% *	65% *	81% *	78% *	58% *	60% *	67% *
$\tau^* = 0.90$	41% †	63% *	64% *	70% *	71% *	58% *	60% *	62% *
20 Labels acquired								
$\tau^* = 0.10$	47 %	62% *	70% *	72% *	66% *	56% *	72% *	62% *
$\tau^* = 0.25$	51% *	63% *	75% *	88% *	81% *	62% *	70% *	65% *
$\tau^* = 0.50$	1 %	64% *	72% *	92% *	87% *	63% *	69% *	68% *
$\tau^* = 0.75$	53% *	60% *	67% *	86% *	80% *	50% *	48% *	58% *
$\tau^* = 0.90$	42 %	61% *	66% *	77% *	75% *	53% *	57% *	62% *
40 Labels acquired								
$\tau^* = 0.10$	43 %	55% *	71% *	75% *	69% *	62% *	69% *	57% *
$\tau^* = 0.25$	56% *	59% *	73% *	89% *	79% *	65% *	69% *	58% *
$\tau^* = 0.50$	4 %	61% *	72% *	93% *	89% *	74% *	76% *	62% *
$\tau^* = 0.75$	57% *	64% *	71% *	90% *	81% *	59% *	56% *	54% *
$\tau^* = 0.90$	46 %	55% *	63% *	82% *	77% *	57% *	64% *	56% *

The used significance level in the one-sided Wilcoxon signed-rank test was for both 0.001. Algorithms are marked with [1] if not every data set could be used in the evaluation due to their long execution time

reduction method Zhao sometimes achieve competitive results (esp. on `YeaU`), but only at very high computational costs, which prevented them to be completed on `Aba`. Using the numbers of Table 3, Rand is surprisingly the best competitor. All in all, we can argue that OPAL outperforms all other tested algorithms with high significance (see Table 3) and has a good trade-off between fast convergence and low final misclassification loss value. Although such an evaluation is not in the scope of this paper, the results on `YeaU` indicate that OPAL is also suitable for unbalanced data sets.

(3) OPAL's runtime in comparison with training set size To experimentally verify OPAL's time complexity (cmp. Sect. 3.2.1), we measured the time in seconds per run used by each active learning process and summed it over all 40 label acquisitions in Table 4 (all differences w.r.t. OPAL are significant at level 0.001). Obviously, Rand is fastest, followed by U.S., C.S., csPAL and OPAL, which slow down constantly with increasing training set size. Our myopic approach csPAL is just slightly slower than U.S., due to its more complex value calculation. OPAL takes about 7 times longer than csPAL, because it computes the G_{OPAL} more often when searching for the optimal budget over $m = 1, 2, \ldots, 40$. In contrast, the execution times of Marg, Chap, and Zhao explode on bigger data sets, taking for a single cost ratio

Table 4 Average execution time (in s), rows ordered in ascending data set size

Data	OPAL	csPAL	U.S.	U.S. st	C.S.	Marg	Chap	Zhao	Rand
See	1.867	0.254	0.206	0.468	0.162	43.535	51.87	254.8	0.015
Che	1.905	0.249	0.201	0.452	0.183	54.897	56.60	319.9	0.016
Che2	1.968	0.261	0.199	0.510	0.198	66.282	69.68	440.7	0.015
Ver	1.987	0.269	0.202	0.653	0.207	71.126	78.66	451.7	0.015
Mam	2.580	0.353	0.268	3.913	0.277	192.86	280.1	1577	0.016
Sim	2.827	0.335	0.239	2.422	0.202	242.98	302.6	1641	0.016
YeaU	2.993	0.379	0.272	9.318	0.260	285.51	499.9	3050	0.017
Aba	7.000	1.001	0.703	136.1	0.706	NaN	NaN	NaN	0.023

All differences w.r.t. OPAL are significant (level 0.001, one-sided Wilcoxon signed-rank test)

more than 8 (Marg), 13 (Chap), or 84 (Zhao) hours. Their calculations on Aba were aborted after one week.

5 Conclusion

In this paper, we addressed the problem of fast, non-myopic active learning for binary classification in cost-sensitive applications. In such applications, unlabelled data is abundant but annotation capacities are limited and require an efficient allocation between labelling candidates. Furthermore, the costs of misclassifications differ between classes, and ultimately the optimal candidate given a remaining labelling budget should be chosen.

We proposed a novel approach, OPAL, that optimises probabilistic active learning for such situations. Given the misclassification cost ratio and remaining budget, which are predetermined by the application, our approach follows a smoothness assumption and computes the expected misclassification loss reduction within a candidate's neighbourhood. For this expectation over the true posterior in the neighbourhood and over the subsequent label realisations therein, we derived a fast, closed-form solution. This allows to select the candidate that reduces the expected misclassification loss in its neighbourhood the most. We have shown that for a myopic setting, our approach runs in asymptotically linear time in the size of the candidate pool. For the non-myopic setting, we have shown that an additional factor that is solely $O(m \cdot \log m)$ in the budget size is required. Furthermore, we have illustrated the effect of the non-myopic extension, indicating its usefulness for unequal misclassification costs. This is confirmed in experimental evaluations on several synthetic and real-world data sets, where our approach has shown comparable or better classification performance than several uncertainty sampling- or error-reduction-based active learning strategies, both in cost-sensitive and cost-insensitive settings.

Our fast approach requires no tunable parameters, yet it is simple to implement, and it neither requires an evaluation sample, nor self-labelling. Thus, its natural extension to data streams has not missed our attention. However, this remains to be done in further research.

Acknowledgments We would like to thank our colleagues from the University Magdeburg and the KMD Lab, in particular Myra Spiliopoulou, Pawel Matuszyk and Christian Braune. Furthermore, we thank the anonymous reviewers for their helpful comments and suggestions.

Appendix

Bandwidth (σ) tuning

The plots in Fig. 7 below show the misclassification loss (y-axis) on the test set, given full label information on the training sets of size 40, for different σ-bandwidths of the Parzen window classifier. The maximal bandwidth (leftmost values of the x-axis) corresponds to a non-discriminating classifier, which simply classifies any instance into the class with higher misclassification cost. Thus, ideally each cost-ratio/data-set-combination exhibits a unique minimum that is smaller than this rightmost value. Combinations with monotonically decreasing misclassification loss curves indicate ill-posed learning problems.

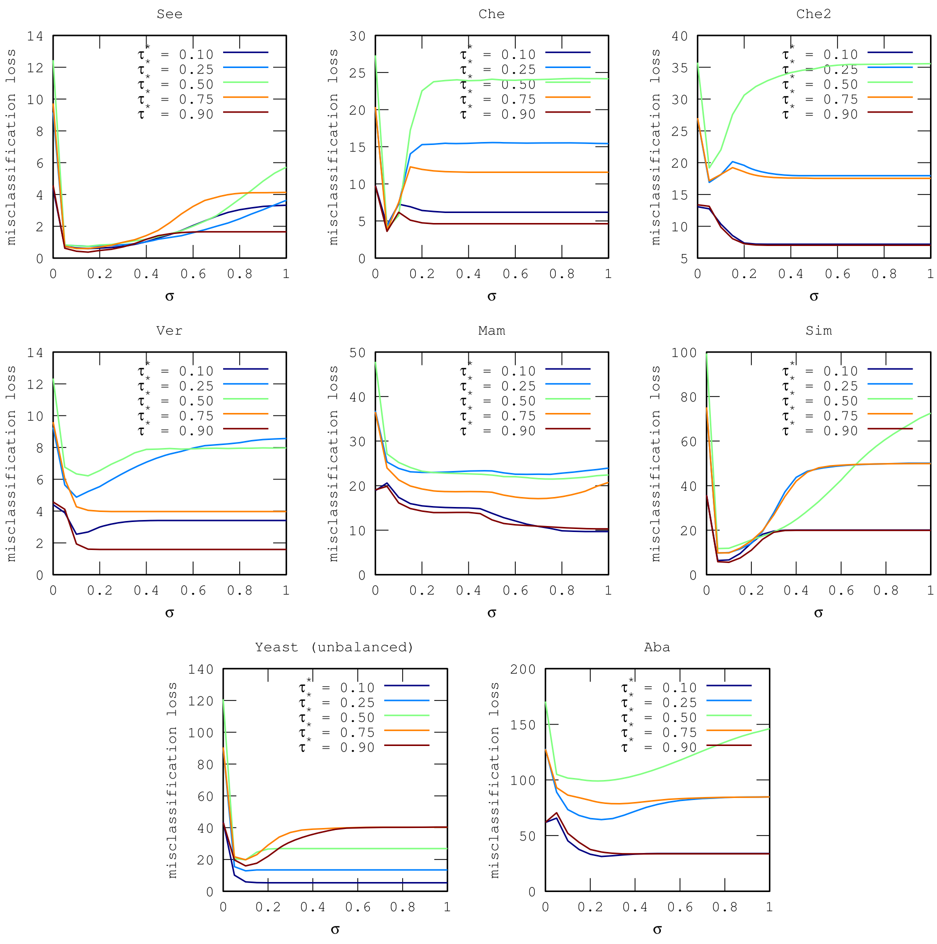


Fig. 7 Misclassification loss for different bandwidth (σ) values of a Parzen window classifier

Cost-sensitivity (cont.)

In Fig. 8, we continue the results from Fig. 4 on the relevance of the cost-sensitiveness, for details see Sect. 4.2.

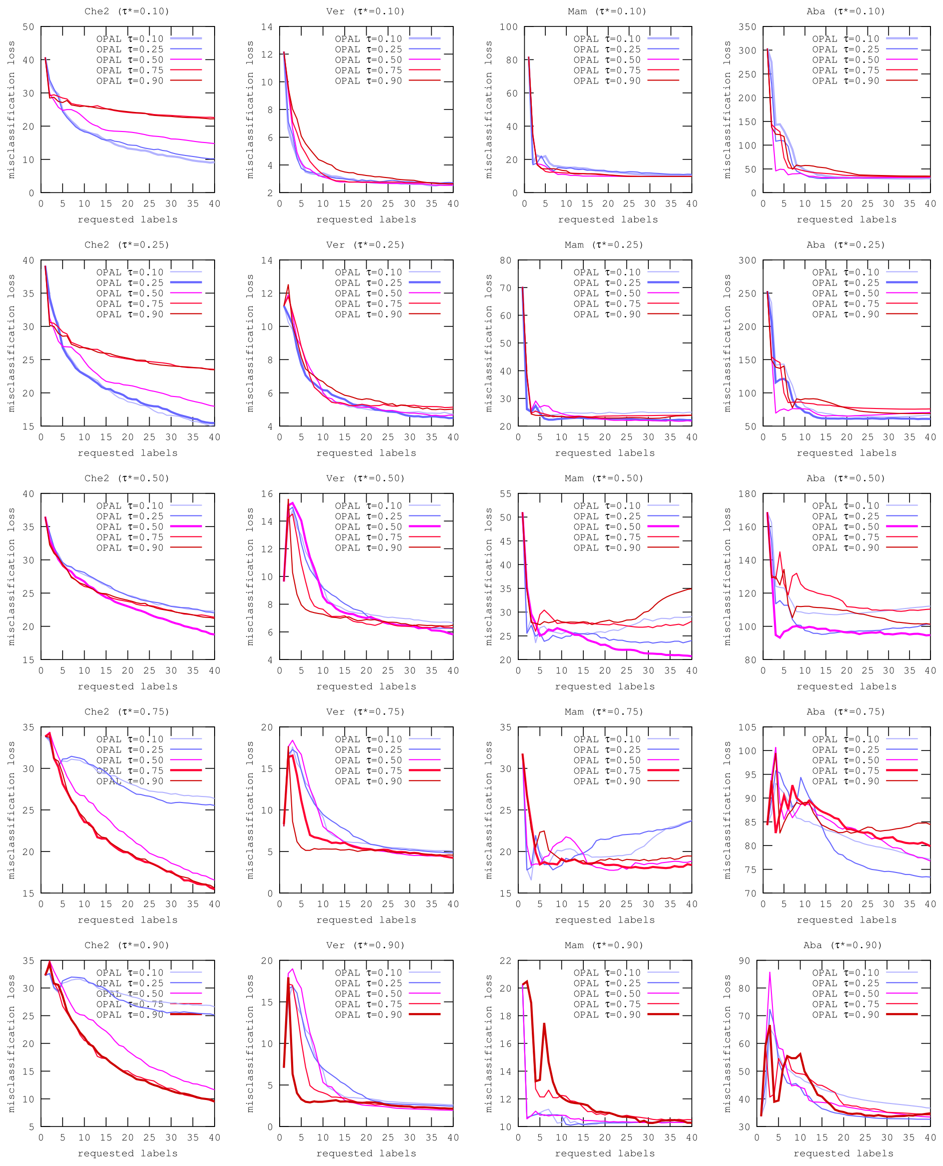


Fig. 8 Misclassification loss curves for OPAL on a selection of data sets with different cost-ratios (τ) for active learning and true cost-ratios (τ^*) for evaluation; curves for correct cost-ratios ($\tau = \tau^*$) are plotted in bold and should be superior; early convergence to very high low values is best. Continuation of Fig. 4 on additional data sets

Detailed derivation of Eq. 24 from Eq. 22

Starting with Eq. 22, we apply Eq. 12, the misclassification loss def. from Eq. 16 and expand the latter by the cost-optimal classification rule from Eq. 18:

$$\begin{aligned}
E_{cur} &= \int_0^1 \mathrm{Beta}_{\alpha,\beta}(p) \cdot ML_{p,\tau}(\hat{p})\,\mathrm{d}p && (36)\\
&= \int_0^1 \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1} \cdot q(\tau-p) + p(1-\tau)\,\mathrm{d}p && (37)\\
&= \int_0^1 \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1} \cdot \begin{cases} 0(\tau-p)+p(1-\tau) & \hat{p}<\tau \\ (1-\tau)(\tau-p)+p(1-\tau) & \hat{p}=\tau \\ 1(\tau-p)+p(1-\tau) & \hat{p}>\tau \end{cases} \mathrm{d}p && (38)\\
&= \int_0^1 \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1} \cdot \begin{cases} p(1-\tau) & \hat{p}<\tau \\ (1-\tau)\tau & \hat{p}=\tau \\ \tau(1-p) & \hat{p}>\tau \end{cases} \mathrm{d}p && (39)
\end{aligned}$$

Following Eq. 12, we set $\alpha = n\hat{p}+1$ and $\beta = n(1-\hat{p})+1$, and obtain Eq. 24.

Detailed derivation of Eq. 32 from Eq. 31

Using the definition of the Beta Integral

$$\int_0^1 x^a(1-x)^b\,\mathrm{d}x = \mathrm{Beta}(a+1,b+1) = \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+b+2)} \tag{40}$$

and setting

$$a = \begin{cases} n\hat{p}+k+1 & \frac{n\hat{p}+k}{n+m} < \tau \\ n\hat{p}+k & \frac{n\hat{p}+k}{n+m} \geq \tau \end{cases} \tag{41}$$

$$b = \begin{cases} n+m-n\hat{p}-k & \frac{n\hat{p}+k}{n+m} \leq \tau \\ n+m-n\hat{p}-k+1 & \frac{n\hat{p}+k}{n+m} > \tau \end{cases} \tag{42}$$

we can express the first factors in the integral in Eq. 31 and thus derive Eq. 32:

$$\mathrm{I}_{ML}(n,\hat{p},\tau,m,k) = \binom{m}{k} \cdot \int_0^1 p^{n\cdot\hat{p}+k} \cdot (1-p)^{n+m-n\cdot\hat{p}-k} \cdot \begin{cases} p\cdot(1-\tau)dp & \frac{n\hat{p}+k}{n+m} < \tau \\ (\tau-\tau^2)dp & \frac{n\hat{p}+k}{n+m} = \tau \\ \tau\cdot(1-p)dp & \frac{n\hat{p}+k}{n+m} > \tau \end{cases} \tag{43}$$

$$= \binom{m}{k} \cdot \begin{cases} (1-\tau) & \cdot \frac{\Gamma(1-k+m+n-n\hat{p})\Gamma(2+k+n\hat{p})}{\Gamma(3+m+n)} & \frac{n\hat{p}+k}{n+m} < \tau \\ (\tau-\tau^2) & \cdot \frac{\Gamma(1-k+m+n-n\hat{p})\Gamma(1+k+n\hat{p})}{\Gamma(2+m+n)} & \frac{n\hat{p}+k}{n+m} = \tau \\ \tau & \cdot \frac{\Gamma(2-k+m+n-n\hat{p})\Gamma(1+k+n\hat{p})}{\Gamma(3+m+n)} & \frac{n\hat{p}+k}{n+m} > \tau \end{cases} \tag{44}$$

References

Asuncion, A., & Newman, D. J. (2013). *UCI machine learning repository*. http://archive.ics.uci.edu/ml/

Attenberg, J., & Ertekin, S. (2013). Imbalanced learning: Foundations, algorithms, and applications, chap. *Class Imbalance and Active Learning*, pp. 101–150. IEEE.

Chapelle, O. (2005). Active learning for parzen window classifier. In *Proceedings of the tenth international workshop on artificial intelligence and statistics*, pp. 49–56.

Chapelle, O., Schölkopf, B., & Zien, A. (Eds.). (2006). *Semi-supervised learning*. Cambridge: MIT Press.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal Artificial Intelligence Research (JAIR)*, *16*, 321–357.

Cohn, D. (2010). Active learning. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of machine learning* (pp. 10–14). Berlin: Springer.

Cohn, D. A., Ghahramani, Z., & Jordan, M. I. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*, *4*, 129–145.

Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. In U. M. Fayyad, S. Chaudhuri, & D. Madigan (Eds.) *Proceedings of the fifth ACM SIGKDD international conference on knowledge discovery and data mining*, KDD, San Diego, CA, USA, August 15–18, 1999, pp. 155–164. ACM.

Elkan, C. (2001). The foundations of cost-sensitive learning. In B. Nebel (Ed.) *Proceedings of the seventeenth international joint conference on artificial intelligence, IJCAI 2001*. Seattle, Washington, USA, August 4–10, 2001, pp. 973–978. Morgan Kaufmann.

Ferdowsi, Z., Ghani, R., & Kumar, M. (2011). An online strategy for safe active learning. In *ICML workshop on combining learning strategies to reduce label cost*.

Freytag, A., Rodner, E., Bodesheim, P., & Denzler, J. (2013). Labeling examples that matter: Relevance-based active learning with gaussian processes. In *German conference on computer vision (GCPR)*, pp. 282–291.

Fu, Y., Zhu, X., & Li, B. (2012). A survey on instance selection for active learning. *Knowledge and Information Systems*, *35*(2), 249–283.

Gantz, J., & Reinsel, D. (2012). *The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east*. http://estonia.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf

Garnett, R., Krishnamurthy, Y., Xiong, X., Schneider, J. G., & Mann, R. (2012). Bayesian optimal active search and surveying. In *Proceedings of the 29th international conference on machine learning (ICML 2012)*. icml.cc/Omnipress.

Gopalkrishnan, V., Steier, D., Lewis, H., & Guszcza, J. (2012). Big data, big business: Bridging the gap. In *Proceedings of the 1st international workshop on big data* (pp. 7–11). Streams and heterogeneous source mining: Algorithms, systems, programming models and applications, BigMine'12 New York, NY: ACM.

Hand, D. J. (2009). Measuring classifier performance: A coherent alternative to the area under the roc curve. *Machine Learning*, *77*(1), 103–123.

He, H., & Ma, Y. (Eds.) (2013). *Imbalanced learning: Foundations, algorithms, and applications*. IEEE.

Krempl, G., Kottke, D., & Spiliopoulou, M. (2014a). Probabilistic active learning: A short proposition. In *Proceedings of the 21st European conference on artificial intelligence (ECAI2014)*, August 18–22, 2014. Prague: IOS Press.

Krempl, G., Kottke, D., & Spiliopoulou, M. (2014b). Probabilistic active learning: Towards combining versatility, optimality and efficiency. In *Proceedings of the 17th international conference on discovery science (DS), Bled*, Lecture Notes in Computer Science. Springer.

Krempl, G., Zliobaitė, I., Brzeziński, D., Hüllermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M., & Stefanowski, J. (2014). Open challenges for data stream mining research. *SIGKDD Explorations*. Special Issue on Big Data (to appear).

Lewis, D. D., & Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 3–12). SIGIR'94 New York, NY: Springer.

Liu, A., Jun, G., & Ghosh, J. (2009). A self-training approach to cost sensitive uncertainty sampling. *Machine Learning*, *76*, 257–270.

Liu, A., Jun, G., & Ghosh, J. (2009). Spatially cost-sensitive active learning. In *Proceedings of the SIAM international conference on data mining*, SDM 2009, April 30–May 2, 2009, Sparks, Nevada, USA, pp. 814–825. SIAM

Liu, A. Y. C. (2009). *Active learning in cost-sensitive environments*. Ph.D. thesis, University of Texas, Electrical and Computer Engineering

Margineantu, D. D. (2005). Active cost-sensitive learning. In *Proceedings of the 19th international joint conference on Artificial intelligence, IJCAI'05*, pp. 1622–1623. Morgan Kaufmann Publishers Inc.

Ng, A. Y., & Jordan, M. I. (2001). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in Neural Information Processing Systems*, *14*, 841–848.

Parker, C. (2011). An analysis of performance measures for binary classifiers. In *Proceedings of the 11th IEEE international conference on data mining (ICDM2011)*, pp. 517–526. IEEE.

Press, W. H., Flannery, B. P., Teukolsky, S. A., & Vetterling, W. T. (1992). *Numerical recipes in Fortran 77: The art of scientific computing* (2nd ed.). Cambridge: Cambridge University Press.

Roy, N., & McCallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the 18th international conference on machine learning, ICML 2001*, Williamstown, MA, USA, ICML'01, pp. 441–448. Morgan Kaufmann Publishers Inc.

Schein, A. I., & Ungar, L. H. (2007). Active learning for logistic regression: An evaluation. *Machine Learning*, *68*(3), 235–265.

Settles, B. (2009). *Active learning literature survey*. Computer Sciences Technical Report 1648, University of Wisconsin-Madison, Madison, Wisconsin, USA. http://pages.cs.wisc.edu/bsettles/pub/settles.activelearning.pdf

Settles, B. (2012). *Active learning. No. 18 in synthesis lectures on artificial intelligence and machine learning*. San Rafael: Morgan and Claypool Publishers.

Tomanek, K., & Hahn, U. (2009). Reducing class imbalance during active learning for named entity annotation. In Y. Gil, & N. Fridman Noy (Eds.) *Proceedings of the 5th international conference on knowledge capture (K-CAP 2009)*, September 1–4, 2009, Redondo Beach, California, USA, pp. 105–112. ACM.

Vijayanarasimhan, S., Jain, P., & Grauman, K. (2010). Far-sighted active learning on a budget for image and video recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition, CVPR*, 13–18 June 2010, San Francisco, CA, pp. 3035–3042. IEEE.

Zhao, Y., Yang, G., Xu, X., & Ji, Q. (2012). A near-optimal non-myopic active learning method. In *Proceedings of the 21st international conference on pattern recognition, ICPR 2012*, Tsukuba, Japan, November 11–15, 2012, pp. 1715–1718. IEEE.

Zhu, J., Wang, H., Tsou, B. K., & Ma, M. Y. (2010). Active learning with sampling by uncertainty and density for data annotations. *IEEE Transactions on Audio, Speech and Language Processing*, *18*(6), 1323–1331.

Zliobaitė, I., Bifet, A., Pfahringer, B., & Holmes, G. (2013). Active learning with drifting streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(1), 27–39.

Chapter 5

Multi-Class Probabilistic Active Learning

A paper based on this chapter has been published as:
Multi-Class Probabilistic Active Learning
Daniel Kottke, Georg Krempl, Dominik Lang, Johannes Teschner, Myra Spiliopoulou. Gal A. Kaminka, Maria Fox, Paolo Bouquet, Eyke Hüllermeier, Virginia Dignum, Frank Dignum, Frank van Harmelen (eds.). ECAI 2016 – 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands. Frontiers in Artificial Intelligence and Applications, vol. 285, pp. 586 – 594, 2016, IOS Press. ISBN 978-1-61499-671-2.
DOI: `10.3233/978-1-61499-672-9-586`
The final version of the resulting paper is available at IOS Press
`http://ebooks.iospress.nl/publication/44803`

Multi-Class Probabilistic Active Learning

Daniel Kottke[1] and **Georg Krempl**[1] and
Dominik Lang[2] and **Johannes Teschner**[2] and **Myra Spiliopoulou**[3]

Abstract. This work addresses active learning for multi-class classification. Active learning algorithms optimize classifier training by subsequently selecting those instances for labeling by an expert, which improve the classifier's performance the most. In this work, we identify different influence factors that positively affect active learning. These factors are (1) an instance's impact, (2) its posterior, and (3) the reliability of this posterior. We contribute a new decision-theoretic approach, called multi-class probabilistic active learning (McPAL). Building on a probabilistic active learning framework, our approach is non-myopic, fast, and optimizes a performance measure (like accuracy) directly. Considering all influence factors, McPAL determines the expected gain in performance to compare the usefulness of instances. For this purpose, it calculates the density weighted expectation over the true posterior and over all possible labeling combinations in a closed-form solution. Thus, in contrast to other multi-class algorithms, it considers the posterior's reliability which improved the performance. In our experimental evaluation, we show the reasonability of the selected influence factors and the superiority of McPAL in comparison to various other multi-class active learning algorithms on six datasets.

1 INTRODUCTION

In supervised classification, prediction models are learned from labeled training data. In some applications, unlabeled data is available or easy to collect but the labeling (annotation) of this data is expensive, time-consuming or exhausting. For such applications, active learning methods provide solutions that optimize the labeling process by selecting the most useful unlabeled instances to be passed to an oracle for labeling. Thereby, active learning aims to achieve high performance with as few labeled instances as possible [22].

A particular and little researched challenge [25] in active learning is its generalization to multi-class settings, with multinomial rather than binary labels. The few works that have addressed this task so far mostly use either uncertainty sampling for active learning of support vector machines, thereby concentrating on instances close to the supposed decision boundary [6, 11, 27], optionally extended by information about density or diversity [4, 13], or they use expected error reduction by simulating an acquisition on the whole dataset to determine the expected performance [12]. Both approaches have known limitations [7, 14]: the former fast, information-theoretic heuristic often fails in exploring the dataspace, the latter decision-theoretic approach requires high computational resources.

In contrast, we contribute a multi-class active learning approach that combines the advantages of the approaches mentioned above, i.e. optimizing expected performance directly while being nearly as fast as uncertainty sampling. Following the recently proposed probabilistic active learning framework [16], the key idea is to compute the expectation over the true posterior by incorporating the number of nearby labels as a proxy for the posterior's reliability and to weight this score by the density as a proxy for the impact of the new label on the whole dataset. We compare our approach with the most relevant state-of-the art methods from the literature and present experiments on six datasets.

In addition, we expose the three influence factors that are used in our method: the posterior, the reliability of that posterior, and the impact of a labeling candidate. We explain their role in active learning and evaluate their effect experimentally. To the best of our knowledge, we are the first to use the number of nearby labels in multi-class active learning, which we show to have a strong impact on the learner's performance. Furthermore, by adding another decision-theoretic argument to propositions in the comparative study of [13], we contribute to the important research question on how to combine the posteriors of many classes into one comparable score.

The next section summarizes the related work by introducing the basic approaches of multi-class active learning. The main section presents our new approach including an analysis of its characteristics, and is followed by our experimental evaluation. The paper is concluding with a summarizing discussion.

2 RELATED WORK

Active learning aims to optimize the annotation of unlabeled instances (candidates), by selecting the ones that improve a given classifier's performance the most [22]. As active learning in general is far more researched than multi-class active learning, we concentrate on the most relevant work before summarizing multi-class approaches.

Most active learning techniques define a usefulness score for each label candidate. A simple but common information-theoretic heuristic is to use the instances with highest uncertainty [17]. This uncertainty sampling method chooses instances near the classifier's current decision boundary, i.e. instances with a posterior probability near the decision threshold (for binary cases 0.5). Related approaches like using the posteriors' entropy have been addressed in [22]. In contrast, the decision-theoretic expected error reduction approach estimates a candidate's usefulness by simulating its label's realizations and measuring the resulting model's performance on a representative set of evaluation instances [20]. This computationally expensive calculation of the expected performance over all possible labels and the instances of the representative set builds the usefulness score [3].

Krempl et al. [15] argue that using posterior estimates directly in

[1] Knowledge Management and Discovery Lab, Otto von Guericke University, Magdeburg, Germany, email: {daniel.kottke, georg.krempl}@ovgu.de
[2] Faculty of Computer Science, Otto von Guericke University, Magdeburg, Germany, email: {dominik.lang, johannes.teschner}@st.ovgu.de
[3] Knowledge Management and Discovery Lab, Otto von Guericke University, Magdeburg, Germany, email: myra@iti.cs.uni-magdeburg.de

the expectation step leads to inaccuracies. They observed that these posterior estimates are highly unreliable especially having only few labeled instances. Probabilistic active learning [16] therefore tries to overcome these difficulties by introducing label statistics that include the posterior of the positive class (they only consider binary classification tasks) and the number of nearby labels as a proxy for reliability. The usefulness score is calculated with the expectation over the true posterior as well as over the possibly appearing labels. Other approaches aim to reduce the classification variance by using an ensemble of classifiers and request instances where the ensemble's disagreement is high [23].

For active learning in multiple classes, the main challenge is the mapping of posterior values into a comparable score to select the most useful labeling candidate. Körner and Wrobel [13] analyzed different heuristics that have been also used by other papers: (1) usual confidence-based uncertainty sampling chooses the instance with the lowest posterior for the best decision, which is comparable to selecting the instances near the decision boundary (see also [5, 10, 26, 27]), (2) entropy-based sampling chooses the instance with highest posterior entropy (see also [28]), (3) Best-vs-Second-Best (BvsSB) sampling (also called margin-based) uses the difference between the posterior of the best and the second best class (see also [5, 11]), and (4) sampling using a specific disagreement that combines margin-based disagreement with the maximal probability[4].

Expected error reduction-based methods have also been considered for multi-class active learning. Joshi et al. [12] proposed an algorithm called *Value of Information (VoI)* that estimates the expected misclassification costs plus the expected labeling costs. They compare the performance of the current classifier and each hypothetical classifier which is generated for each labeling candidate and each class on an evaluation set. As these algorithms take long for execution, they propose three approximations for speedup. For music annotation applications, Chen et al. [4] developed a method that finds a set of instances to be labeled based on a volume criterion (similar to SVM volume reduction [24]), a density score to prefer dense regions and a diversity score to ensure that instances from one labeling set are diverse. More recently, Guo and Wang [6] developed a step-wise method consisting of an initial selection of instances to be labeled (via random, clustering or discrepancy), followed by an active learning step. This is based on the characteristics of One-versus-Rest (OvR) Support Vector Machines (SVMs) where a labeling candidate can belong to one class with support from zero, one or more than one OvR SVMs. To choose the next instance for labeling, they define a rejection, a compatibility and an uncertainty score, and some rules how they are considered. Wang et al. [25] propose an ambiguity-based multi-class approach that uses possibilistic memberships from One-vs-Rest SVMs. These memberships are between 0 and 1 but do not necessarily sum up to one like posteriors. Their ambiguity measure is based on fuzzy logic operations and has a parameter γ which has to be optimized and is not known in advance. A more theoretical work on cost-sensitive multi-class active learning is given by [1]. He analyzed the regret and label complexity for data with labels that are generated with a generalized linear model.

Some approaches consider settings with different costs for misclassifying an instance of a specific class [5, 12]. Additionally, [12] also includes annotation cost, i.e. the cost the expert induces while labeling an instance. The acquisition of instances can be done in a subsequent manner or in form of instance batches. Most approaches choose to acquire instances one-by-one, except for [4, 28]. Besides support vector machines (often used with a probabilistic version), [13] used an ensemble of trees, [10] proposed a probabilistic version of the k-nearest-neighbor (pKNN) classifier, [5] tested their algorithms on a random forest, and [28] used random walks over a markov chain.

[4] Note, that the selection of instance based on confidence and BvsSB would be exactly the same in a two-class problem but is different for multiple classes (see [22]).

3 OUR METHOD

In this section, we propose probabilistic active learning for multiple classes. This approach directly optimizes a performance measure like accuracy, is non-myopic, and works easily with any generative classifier [15]. In the first subsection, we present the active learning framework and explain our influence factors. Next, we propose our **M**ulti-**c**lass **P**robabilistic **A**ctive **L**earning (McPAL) approach, followed by the derivation of a closed-form solution. Finally, we conclude our results and compare its behavior to existing approaches in an analytical way.

3.1 AL framework and influence factors

In an active, multi-class classification tasks with C different classes, each instance has a feature vector $\vec{x}$ and a label $y \in \{1, \ldots, C\}$, which is unknown at the beginning. As shown in Fig. 1, the labeled set $\mathcal{L}$ is subsequently filled by the active learner, who selects the most useful instance $\vec{x}_{\text{opt}}$ from the candidate pool $\mathcal{U}$ and requests its label from the oracle. This is repeated until the budget b is consumed. In our setting, the active component's decision is based on outputs (posteriors and distribution of labeled instances) of a generative probabilistic classifier [18], which is updated according to changes in the labeled set.

```
function al_framework(U){
  L = {}
  cl = init_classifier()
  for(b=1; b<=60; b++){
    x* = active_learning(U, cl)
    y = ask_oracle(x*)
    U = remove(U, {x*})
    L = append(L, {x*, y})
    cl = train_classifier(L)
  }
}
```

Figure 1. Pseudocode of the active learning framework

Throughout our research on active learning, we identified different influence factors that affect active learning in a positive manner. The labeling candidate's *class posterior* $\hat{P}(y \mid \vec{x})$ is the most commonly used one, as it indicates the probability of an instance $\vec{x}$ to be classified as y. For simplicity, we denote $\vec{\hat{p}}$ as the vector of estimated posterior probabilities, i.e. $\hat{p}_i = \hat{P}(y = i \mid x), 1 \leq i \leq C$. If the posteriors for all classes are similar, this indicates a high uncertainty of the classifier at the instance's location $\vec{x}$. Here, we have to distinguish between the aleatoric uncertainty that is caused by high Bayesian error, and the epistemic uncertainty, which is caused by a lack of information [21]. We are not able to reduce the aleatoric uncertainty, but we can acquire more labels to reduce the epistemic uncertainty in the currently considered neighborhood.

Measuring the number of nearby labels n as a proxy for the *reliability of the class posterior* enables the separation of the aleatoric and the epistemic uncertainty. The higher this number is, the more likely it is for the observed posterior $\hat{p}$ to be close to the unknown true posterior.

The third influence factor is the *impact on the whole dataset*. Weighting the usefulness score by the instances' density as a proxy for its impact prefers instances in dense regions over those in sparse ones. We assume that it is more beneficial to focus on regions with high density as more future classification decision benefit from the information increment there.

One of the most important questions in multi-class active learning is how to combine the different posteriors to one comparable score. In binary situations, this function $\hat{p} \mapsto \mathbb{R}$ is only one-dimensional as $\hat{p}_2 = 1 - \hat{p}_1$ and can be easily visualized. Three-class problems typically are visualized with ternary plots (see also [13, 22]). In Fig. 2, we show a ternary heatmap plot where the darkness indicates the usefulness. Every position in that barycentric coordinate system stands for one specific posterior probability. The figure shows the usefulness values for confidence-based sampling (Conf), and for the Best-vs-Second-Best (BvsSB) approach. The entropy-based score has a more circular shape (not shown here) [22].

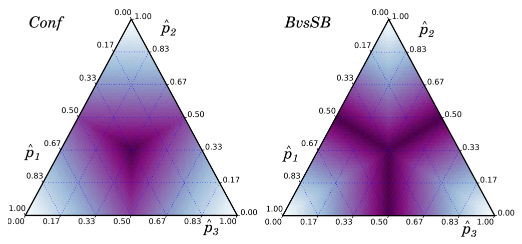


Figure 2. Ternary heatmap plot of the usefulness of confidence-based (Conf) and Best-vs-Second-Best (BvsSB) sampling. Dark color indicates high usefulness of a posterior in that barycentric coordinate system.

After we proposed our method that combines all three influence factors in a decision-theoretic way in the next section, we visualize the behavior of McPAL (without the density) also with ternary plots, and evaluate our theory of influence factors experimentally comparing their effects on active learning performance in Sec. 4.2. Our mathematical symbols are summarized in Tab. 1.[5]

C	- Number of classes
$Y = \{1, \ldots, C\}$	- Vector of all possible labels
$\mathcal{L}$	- Set of labeled instances (x, y)
$\mathcal{U}$	- Set of unlabeled instances $(x, .)$
$\vec{p} = (p_1, \ldots, p_C)$	- Vector of true posteriors
$\vec{k} = (k_1, \ldots, k_C)$	- Vector of frequency estimates
$n = \sum k_i$	- Number of observed labels (reliability)
$\vec{\hat{p}} = \vec{k}/n$	- Vector of observed posteriors
$\vec{d} = (d_1, \ldots, d_C)$	- Decision vector (see Eq. 8)
$m \in \mathbb{N}$	- Number of hypothetically considered labels
$\vec{l} = (l_1, \ldots, l_C) \in \mathbb{N}^C$	- Hypothetical labeling ($\sum l_i = m$)

Table 1. Overview of used mathematical symbols.

3.2 Multi-class probabilistic active learning

In probabilistic active learning for two classes, it is assumed that the appearance of a label of class y is a Bernoulli experiment [16]. A label of class i in the neighborhood of an instance $\vec{x}$ appears with a probability of $P(y = i \mid \vec{x}) =: p_i$ building the vector of true posteriors $\vec{p}$. For multiple classes, we naturally generalize the 2-class Binomial distribution to a Multinomial one. The probability of observing a specific labeling situation $\vec{k}$ given the true posterior $\vec{p}$ is then calculated according to Eq. 1. Each entry k_i in the vector $\vec{k}$ represents the number of instances with label $i, 1 \leq i \leq C$ in the neighborhood of $\vec{x}$. This vector also indicates the number of observed labels $n = \sum k_i$, which is used as the reliability proxy ($\vec{k} = n \cdot \vec{\hat{p}}$). In Eq. 1, we use the generalized multinomial coefficient for non-integer arguments containing the Γ function by Legendre [19].

$$P(\vec{k} \mid \vec{p}) = \text{Multinomial}_{\vec{p}}(\vec{k}) = \binom{\sum k_i}{k_1, \ldots, k_C} \cdot \prod \left(p_i^{k_i}\right) \quad (1)$$

$$= \frac{\Gamma\left(\left(\sum k_i\right) + 1\right)}{\prod\left(\Gamma\left(k_i + 1\right)\right)} \cdot \prod \left(p_i^{k_i}\right) \quad (2)$$

In the active learning setting, we do not know the true posteriors $\vec{p}$, but we are able to estimate the number of observations $\vec{k}$. To determine a probability distribution for the true posterior, we take the normalized likelihood function [15] as given in Eq. 3-5.

$$L(\vec{p} \mid \vec{k}) = P(\vec{k} \mid \vec{p}) \quad (3)$$

$$P(\vec{p} \mid \vec{k}) = \frac{L(\vec{p} \mid \vec{k})}{\int_{\vec{p}'} L(\vec{p}' \mid \vec{k})\, \mathrm{d}\vec{p}'} = \frac{\Gamma\left(\sum (k_i + 1)\right)}{\Gamma\left(\left(\sum k_i\right) + 1\right)} \cdot L(\vec{p} \mid \vec{k}) \quad (4)$$

$$= \frac{\Gamma\left(\sum (k_i + 1)\right)}{\prod\left(\Gamma\left(k_i + 1\right)\right)} \cdot \prod \left(p_i^{k_i}\right) \quad (5)$$

The density function $P(\vec{p} \mid \vec{k})$ has its maximum for $\vec{p} = \vec{\hat{p}}$ and the variance decreases by increasing $n = \sum k_i$.

Given a performance measure like accuracy, a Bayesian optimal decision [15] selects the most probable class $\hat{y}$ (based on its observed frequency $k_{\hat{y}}$) according to Eq. 6. The true posterior $p_{\hat{y}}$ of this selected class corresponds to the resulting accuracy, as expressed by the performance function in Eq. 7.

$$\hat{y} = \underset{y \in \{1, \ldots, C\}}{\arg\max}\, (k_y) \quad (6)$$

$$\text{perf}\left(\vec{k} \mid \vec{p}\right) = p_{\hat{y}}, \quad (7)$$

$$= \prod p_i^{d_i} \qquad d_i = \begin{cases} 1 & \text{if } i = \hat{y} \\ 0 & \text{if } i \neq \hat{y} \end{cases} \quad (8)$$

Given such a performance function, we calculate the expected performance for the neighborhood around $\vec{x}$ with observations $\vec{k}$:

$$\text{expPerf}\left(\vec{k}\right) = \mathop{\mathbb{E}}_{\vec{p}}\left[\text{perf}\left(\vec{k} \mid \vec{p}\right)\right] \quad (9)$$

$$= \int_{\vec{p}} P(\vec{p} \mid \vec{k}) \cdot \text{perf}\left(\vec{k} \mid \vec{p}\right) \mathrm{d}\vec{p} \quad (10)$$

The goal of our approach is (1) to estimate the gain of performance resulting from an upcoming label based on the unlabeled $\mathcal{U}$ and labeled data $\mathcal{L}$, and (2) to choose the candidate with maximal gain (see Eq. 11). Having chosen a generative, probabilistic classifier like the Parzen window classifier [3] or the probabilistic k-nearest-neighbor [10], we are able to count the number of labeled occurrences per class (see Eq. 12). Finally, we define our active learning score as the density weighted performance gain given in Eq. 13.

$$\vec{x}^* = \underset{\vec{x} \in \mathcal{U}}{\arg\max}(\text{alScore}\left(\vec{x} \mid \mathcal{L}, \mathcal{U}\right)) \quad (11)$$

$$\vec{k} = \text{cl}\left(\vec{x} \mid \mathcal{L}\right) \quad (12)$$

$$\text{alScore}\left(\vec{x} \mid \mathcal{L}, \mathcal{U}\right) = P(\vec{x} \mid \mathcal{L} \cup \mathcal{U}) \cdot \text{perfGain}\left(\text{cl}\left(\vec{x} \mid \mathcal{L}\right)\right) \quad (13)$$

[5] All unspecified iterators start at $i = 1$ and end at C.

We determine the performance gain in Eq. 14 by the difference of the new expected expected performance and the current expected performance. The current expected performance is simply calculated as in Eq. 9, the new one is explicitly called expected expected performance (see Eq. 15), as there are multiple possibilities of an labeling to be considered. Therefore, we additionally calculate the expectation value over these possible labelings $\vec{l} = (l_1, \ldots, l_C) \in \mathbb{N}^C$. Given a number of hypothetical labels that are allowed to be acquired $m \in \mathbb{N}$, $\sum l_i = m$ in one step, the labeling vector represents the change of observations that would be added to the $\vec{k}$ vector if this labeling would be obtained. Hence, the classifier output after receiving a labeling $\vec{l}$ changes to $\vec{k} + \vec{l}$. Note that this calculation is exact for $m = 1$, but only an approximation for $m > 1$, as it is unlikely to have another instance $\vec{x}'$ at exactly the same location as our instance $\vec{x}$ (similarity of $\vec{x}$ and $\vec{x}'$ should be 1). However, as we only select one instance for labeling at each step, we divide the gain by m to have the average gain per label acquisition.

$$\text{perfGain}\left(\vec{k}\right) = \max_{m \leq M} \left(\frac{1}{m} \left(\text{expExpPerf}\left(\vec{k}, m\right) - \text{expPerf}\left(\vec{k}\right) \right) \right) \tag{14}$$

$$\text{expExpPerf}\left(\vec{k}, m\right) = \mathop{\mathbb{E}}_{\vec{p}} \left[\mathop{\mathbb{E}}_{\vec{l}} \left[\text{perf}\left(\vec{k} + \vec{l} \mid \vec{p}\right) \right] \right] \tag{15}$$

The labeling $\vec{l}$ is multinomial distributed given the true posterior:

$$P(\vec{l} \mid \vec{p}) = \text{Multinomial}_{\vec{p}}(\vec{l}) = \frac{\Gamma\left(\left(\sum l_i\right) + 1\right)}{\prod \left(\Gamma\left(l_i + 1\right)\right)} \cdot \prod \left(p_i^{l_i} \right) \tag{16}$$

With help of these equations it is possible to determine the next best instance for labeling as given in Eq. 13 numerically. To achieve a good numerical performance would be computationally expensive and highly dependent on the number of classes C as well as the step width for integrating the true posterior $\vec{p}$.

Hence, we propose a closed-form solution for this approach in the following section that reduces the computational cost seriously.

3.3 Fast closed-form solution

To simplify the integration, it is sufficient to optimize the expected expected performance, as the expected performance is a special case of the former (see Eq. 17ff.).

$$\text{expPerf}\left(\vec{k}\right) = \text{expExpPerf}\left(\vec{k}, 0\right) \tag{17}$$

$$\text{expExpPerf}\left(\vec{k}, m\right) = \mathop{\mathbb{E}}_{\vec{p}} \left[\mathop{\mathbb{E}}_{\vec{l}} \left[\text{perf}\left(\vec{k} + \vec{l} \mid \vec{p}\right) \right] \right] \tag{18}$$

$$= \int_{\vec{p}} P(\vec{p} \mid \vec{k}) \cdot \sum_{\vec{l}} P(\vec{l} \mid \vec{p}) \cdot \text{perf}\left(\vec{k} + \vec{l} \mid \vec{p}\right) \mathrm{d}\vec{p} \tag{19}$$

$$= \sum_{\vec{l}} \int_{\vec{p}} P(\vec{p} \mid \vec{k}) \cdot P(\vec{l} \mid \vec{p}) \cdot \text{perf}\left(\vec{k} + \vec{l} \mid \vec{p}\right) \mathrm{d}\vec{p} \tag{20}$$

$$= \sum_{\vec{l}} \int_{\vec{p}} \frac{\Gamma\left(\sum (k_i + 1)\right)}{\prod \left(\Gamma\left(k_i + 1\right)\right)} \cdot \prod \left(p_i^{k_i} \right) \cdot \frac{\Gamma\left(\left(\sum l_i\right) + 1\right)}{\prod \left(\Gamma\left(l_i + 1\right)\right)} \cdot \prod \left(p_i^{l_i} \right) \cdot \text{perf}\left(\vec{k} + \vec{l} \mid \vec{p}\right) \mathrm{d}\vec{p} \tag{21}$$

$$= \sum_{\vec{l}} \frac{\Gamma\left(\sum (k_i + 1)\right)}{\prod \left(\Gamma\left(k_i + 1\right)\right)} \cdot \frac{\Gamma\left(\left(\sum l_i\right) + 1\right)}{\prod \left(\Gamma\left(l_i + 1\right)\right)} \cdot \int_{\vec{p}} \prod \left(p_i^{k_i + l_i} \right) \cdot \text{perf}\left(\vec{k} + \vec{l} \mid \vec{p}\right) \mathrm{d}\vec{p} \tag{22}$$

After separating the normalization factors from the integral, we simplify the integral by inserting the performance from Eq. 8 and by calculating the definite integral as above in Eq. 4.

$$\int_{\vec{p}} \prod \left(p_i^{k_i + l_i} \right) \cdot \text{perf}\left(\vec{k} + \vec{l} \mid \vec{p}\right) \mathrm{d}\vec{p} \tag{23}$$

$$= \int_{\vec{p}} \prod \left(p_i^{k_i + l_i} \right) \cdot \prod p_i^{d_i} \, \mathrm{d}\vec{p} \tag{24}$$

$$= \int_{\vec{p}} \prod \left(p_i^{k_i + l_i + d_i} \right) \mathrm{d}\vec{p} = \frac{\prod \Gamma\left(k_i + l_i + d_i + 1\right)}{\Gamma\left(\sum (k_i + l_i + d_i + 1)\right)} \tag{25}$$

Reinserting the integral into Eq. 22 and sorting the terms yields the following equations.

$$\text{expExpPerf}\left(\vec{k}, m\right) = \sum_{\vec{l}} \frac{\Gamma\left(\sum (k_i + 1)\right)}{\prod \left(\Gamma\left(k_i + 1\right)\right)} \cdot \frac{\Gamma\left(\left(\sum l_i\right) + 1\right)}{\prod \left(\Gamma\left(l_i + 1\right)\right)} \cdot \frac{\prod \Gamma\left(k_i + l_i + d_i + 1\right)}{\Gamma\left(\sum (k_i + l_i + d_i + 1)\right)} \tag{26}$$

$$= \sum_{\vec{l}} \frac{\Gamma\left(\sum (k_i + 1)\right)}{\Gamma\left(\sum (k_i + l_i + d_i + 1)\right)} \cdot \frac{\prod \Gamma\left(k_i + l_i + d_i + 1\right)}{\prod \left(\Gamma\left(k_i + 1\right)\right)} \cdot \frac{\Gamma\left(\left(\sum l_i\right) + 1\right)}{\prod \left(\Gamma\left(l_i + 1\right)\right)} \tag{27}$$

The first and second factor are simplified as follows.

$$\frac{\Gamma\left(\sum (k_i + 1)\right)}{\Gamma\left(\sum (k_i + l_i + d_i + 1)\right)} \tag{28}$$

$$= \frac{\Gamma\left(\sum (k_i + 1)\right)}{\Gamma\left(\sum (k_i + 1) + \left(\sum l_i\right) + \left(\sum d_i\right)\right)} \tag{29}$$

$$= \left(\prod_{j = \sum (k_i + 1)}^{\left(\sum (k_i + l_i + d_i + 1)\right) - 1} \frac{1}{j} \right) \frac{\Gamma\left(\sum (k_i + 1)\right)}{\Gamma\left(\sum (k_i + 1)\right)} \tag{30}$$

$$= \prod_{j = \sum (k_i + 1)}^{\left(\sum (k_i + l_i + d_i + 1)\right) - 1} \frac{1}{j} \tag{31}$$

$$\frac{\prod \Gamma\left(k_i + l_i + d_i + 1\right)}{\prod \left(\Gamma\left(k_i + 1\right)\right)} = \prod \frac{\Gamma\left(k_i + l_i + d_i + 1\right)}{\Gamma\left(k_i + 1\right)} \tag{32}$$

$$= \prod \frac{\left(\prod_{j = k_i + 1}^{k_i + l_i + d_i} j \right) \Gamma\left(k_i + 1\right)}{\Gamma\left(k_i + 1\right)} = \prod \left(\prod_{j = k_i + 1}^{k_i + l_i + d_i} j \right) \tag{33}$$

Using Eq. 27, 31 and 33, we get the fast version of the expected expected performance.

$$\text{expExpPerf}\left(\vec{k}, m\right) = \sum_{\vec{l}} \left(\prod_{j = \sum (k_i + 1)}^{\left(\sum (k_i + l_i + d_i + 1)\right) - 1} \frac{1}{j} \right) \cdot \prod \left(\prod_{j = k_i + 1}^{k_i + l_i + d_i} j \right) \cdot \frac{\Gamma\left(\left(\sum l_i\right) + 1\right)}{\prod \left(\Gamma\left(l_i + 1\right)\right)} \tag{34}$$

Now, the final McPAL usefulness score from Eq. 13 is calculated using Eq. 14 and Eq. 34.

As an example, we calculate the expected expected performance for $m = 0$ which is equivalent to the expected performance. As mentioned before, $\hat{y} = \arg\max_{y \in \{1,\dots,C\}}(k_y)$.

$$\text{expExpPerf}\left(\vec{k}, 0\right) = \sum_{\vec{l}} \left(\prod_{j=\sum(k_i+1)}^{(\sum(k_i+1)+(\sum l_i)+(\sum d_i))-1} \frac{1}{j} \right)$$

$$\cdot \prod \left(\prod_{j=k_i+1}^{k_i+l_i+d_i} j \right) \cdot \frac{\Gamma\left(\left(\sum l_i\right)+1\right)}{\prod\left(\Gamma\left(l_i+1\right)\right)} \quad (35)$$

$$= \left(\prod_{j=\sum(k_i+1)}^{(\sum(k_i+1)+0+1-1} \frac{1}{j} \right) \cdot k_{\hat{y}} \cdot 1 \cdot 1 = \frac{k_{\hat{y}}}{\sum(k_i+1)} \quad (36)$$

3.4 Characteristics of McPAL

As briefly discussed in Sec. 3.1, there are different ways to combine the posterior estimates $\vec{\hat{p}}$ from the classifier to determine a usefulness score. The examples in Fig. 2 show different shapes that lead to different behavior, which is evaluated in Sec. 4.

Fig. 3 shows the ternary heatmap plots for the performance gain function of the McPAL algorithm, i.e. the active learning score without the density weight. In contrast to all other multi-class active learning approaches, McPAL does not only consider the observed probability $\vec{\hat{p}}$ but also includes the reliability $n = \sum k_i$, which is summarized in the frequency vector $\vec{k} = n \cdot \vec{\hat{p}}$. This extends the ternary plot by an additional degree of freedom. Therefore, we provide two exemplary figures, one showing the behavior for $n = 1$, and one for $n = 2$.

The left plot of Fig. 3 shows a similar but not identical shape as the confidence based (Conf in Fig. 2). While contour lines for confidence-based sampling are linear, these of McPAL are slightly concave. The highest gain is in the center, which represents regions of absolute uncertainty as the posteriors are equal. The lowest gains are in the corners of the triangle. An increase of reliability n decreases the gain (see right plot), as the epistemic uncertainty (caused by lack of information) decreases. This means that there are situations where instances with a non-equal posterior vector are preferred over those with equal posteriors if there is more evidence that the equal posteriors are more likely to be correct.

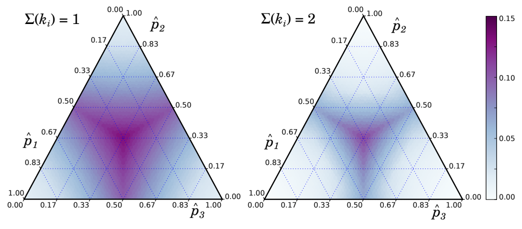


Figure 3. Ternary plot for performance gain for situations with $n = \sum k_i = 1$ (left) and $n = 2$ (right).

The maximal number of hypothetically considered labels M is not a tuning parameter. It should be set according to the application. For accuracy optimization, the optimal m value is the lowest number of labels that is able to switch the decision in the neighborhood of $\vec{x}$. If lots of instances should be labeled, the M value should be increased. In the upper plots the value was set to $M = 2$ as the highest observable n was 2. Hence, two labels are able to change the decision.

From a decision-theoretic view, it is more reasonable to prefer confidence based active learning over entropy or best-vs-second-best, but the reliability makes a huge difference in the performance as the next section will show.

4 EVALUATION

The goals of our evaluation are twofold: on the one hand, we show the reasonability of our previously defined impact factors, and on the other hand we compare our multi-class probabilistic active learning approach with state-of-the-art methods. All experiments are conducted based on the setup explained in the following subsection.

4.1 Experimental setup

The proposed method and several other active learning strategies are applied to six datasets, sampling single instances subsequently until the available budget of 60 label acquisitions has been depleted. This is done on multiple, seed-based splits of the datasets into independent training and test subsets (training 67%, test 33% of the data) where the number of different training-test-splits for the smaller datasets (ecoli, glass, iris, wine) is 100 and for the large datasets (vehicle, yeast) is set to 50 due to execution time. All experiments are reported by its mean and standard deviation across all splits.

The most used visualization of evaluation results are learning curves, which plot the performance in comparison to the number of acquired labels. Our learning curves in Fig. 4 and 5 show the classification error of each active learner on the y-axis, the standard deviation of the error across all splits indicated as an error bar, and the number of instances sampled for the labeled set on the x-axis. In addition to these plots, the results are given in Tab. 4, showing the error and standard deviation of the different active learning methods for all used datasets. The tables show the learner's performance at three different steps, i.e. after 20, 40 and 60 labels have been acquired. Since 60 is the maximum number of sampled instances in the experiments, these steps show the performance in the beginning, intermediate and end phase of the learning process. All results are reported separately for each classifier and dataset. We computed our experiments on a computer cluster running the Neurodebian [8] system.

Besides the proposed method of this paper, six other active learning strategies are used. The McPAL method is executed with $M = 2$, as higher M just increased the execution time but did not change the performance. As a standard baseline, we use a randomly sampling method (`Rand`). *Confidence-based sampling* (`Conf`) selects the instance with the lowest maximal posterior ($x^* = \arg\min_{x \in \mathcal{U}} \max_{y \in Y} \hat{p}_y$) [10]. The next approach uses the shannon entropy to model the uncertainty of an instance (`Entr`) [12]. *Best-vs-Second-Best* (`BvsSB`) samples this instance of the unlabeled set that minimizes the difference of the posterior probabilities of the most probable and the second most probable class [11, 12, 13]. *Maximum-Expected-Cost* (`MaxECost`) determines the value of an instance based on the expected cost associated with the misclassification of that instance. Consequently, the learner samples the instance tied to this score [5]. The last strategy belongs to the expected error reduction based methods. The original *Value of Information* (`VoI`) criterion as suggested by Joshi et al. [12] selects the instance $\vec{x}$ that minimizes a risk measure defined by them. It has to be mentioned that the computational effort of this algorithm forced us to exclude it from the experiments on the vehicle and yeast datasets, since they possess a large number of instances and/or classes, leading to infeasible execution times.

Active learning algorithms require robust classifiers for robust usefulness estimation. Therefore, we choose generative classifiers [18], namely the Parzen window classifier (PWC) [3], and a probabilistic variant of the k-nearest-neighbor classifier (pKNN with $k = 9$) proposed by Jain and Kapoor [10]. These classifiers can be used with any arbitrary similarity function. As the optimization of the overall performance level is not the scope of this paper, we choose to simply standardize each attribute (z-standardization) and use an univariate Gaussian kernel with fixed standard deviation of $\sigma = 0.7$ for all datasets and active learning algorithms. This ensures fair comparability that is independent of a classifier bias.

Table 2. Datasets with the number of instances, the number of attributes and the class frequencies.

Dataset	#Inst.	#Attr.	#Instances per class
Ecoli	336	8	143, 77, 52, 35, 20, 5, 2, 2
Glass	214	10	70, 76, 17, 13, 9, 29
Iris	150	4	50, 50, 50
Vehicle	946	18	212, 217, 218, 199
Wine	178	13	59, 71, 48
Yeast	1484	8	463, 429, 244, 163, 51, 44, 35, 30, 20

We evaluate our algorithm on six multi-class datasets from the UCI repository [2]. The distribution of classes and the number of instances and attributes are summarized in Tab. 2. The *ecoli* dataset was originally used for predicting protein localization sites in eukaryotic cells. The attributes describe properties of proteins. *Glass* was originally generated for classification of types of glass left at a crime scene. The attributes describe chemical ingredients to predict for example whether the glass is from a car window or a window of a building. The *iris* dataset classifies the type of an iris plant, the features describe physical measures of the plant. *Vehicle* contains features of car models for predicting the manufacturer. The attributes of the *wine* dataset describe the chemical ingredients of a wine instance. The class values are derived from three different cultivars. The *yeast* dataset is also used for predicting the localization site of protein in bacteria. The first column, which held the sequence name, was removed.

The complete results together with an implementation are available at our companion website[6].

4.2 Impact of influence factors

In Sec. 3.1, we introduced three different influence factors that are considered in McPAL. Fig. 4 shows learning curves on selected datasets and classifiers of McPAL variants with different input parameters using the previously described experimental setup. In that way, we aim to measure the importance of the different influence factors *posterior*, *reliability*, and *impact*. Besides using the original `McPAL` algorithm, we exclude information about the reliability by normalizing the $\vec{k}$ vector to $\sum \vec{k} = n = 1$ (denoted `w/o reliability`). Analogously, we proceed with the posterior by replacing the kernel frequency estimate with a uniform one $k_i = n/C, 1 \leq i \leq C$ (denoted `w/o posterior`), and with the density by setting it to a constant (denoted `w/o impact`).

Our selection in Fig. 4 shows that the combination of all influence factors works best. In some cases, the variant without impact is better than the McPAL method. We explain this behavior with the fact that the density, which is used as a proxy for the influence of a label on the complete dataset, gets inaccurate. Especially when there are many labels added to the dataset, this estimate gets worse as the influence also depends on the explicit label situation on the dataset. Nevertheless, the density improved the overall performance although leaving it out is less critical than leaving out one of the other factors.

Especially the results on yeast with the PWC are interesting. Here, leaving out the reliability or the posterior leads to no performance improvement, but unifying these approaches (McPAL) achieves the lowest error.

4.3 Competitiveness of our method

Fig. 5 shows the learning curves of the experiment results with the pKNN classifier, Tab. 4 shows the results using the PWC. As shown in Tab. 4 the McPAL algorithm outperforms its competitors consistently on 4 of the 6 datasets (best performance highlighted in bold text), for the first 20 sampled instances even on 5 out of 6. Using the PWC, our method is only the second best by a close margin after 40 and 60 samples on the vehicle data. After 20 samples random sampling performed best. On the wine dataset, our method scores best at 20 sampled instances but falls behind `Entr` later. As wine data is easy to learn, it is important to mention that the performance almost converged at 30 labels. In general the `BvsSB` and `Entr` algorithms seem to be the most consistent competitors to McPAL in the experiments, the former being the best scoring on the vehicle dataset after 40 samples and the latter outperforming McPAL on the wine dataset after 40 samples.

A good active learning algorithm is characterized by a fast convergence to a good final performance. As can be seen in Fig. 5, our proposed method manages to reduce the classification error quicker than its competitors, in some cases even starting out with a lower error (e.g. ecoli, glass, yeast). Over all datasets, McPAL reduces the error quicker than the other algorithms in the early steps. On top of that, the McPAL algorithm shows a lower standard deviation across all trials compared its competitors (indicated by the error bars in the plots and the brackets in Tab. 4), making it not only the best performing but also the most stable method in the experiments.

For another perspective on the results, the performance of the algorithms in comparison to randomly sampling instances (`Rand`, grey dotted line) should be considered. In case of both the vehicle and yeast dataset McPAL's competitors surpass random instance sampling only pretty late into the learning process in terms of classification error. Even on the iris dataset `Conf`, `BvsSB` and `VoI` struggle to perform better than random selection.

Table 3. Mean execution time for each algorithm for choosing one instance for labeling on the specified dataset (sorted by dataset size)

Dataset	McPAL	BvsSB	MaxEC.	Conf	Entr	VoI	Rand
Iris	0.363	0.085	0.083	0.097	0.092	15.94	0.001
Wine	0.584	0.145	0.148	0.153	0.147	36.22	0.001
Glass	1.794	0.200	0.205	0.204	0.204	136.1	0.001
Ecoli	4.590	0.306	0.317	0.313	0.308	518.5	0.001
Vehicle	2.128	0.389	0.394	0.385	0.386	NA	0.001
Yeast	28.06	1.175	1.207	1.171	1.186	NA	0.001

[6] `http://kmd.cs.ovgu.de/res/mcpal/`

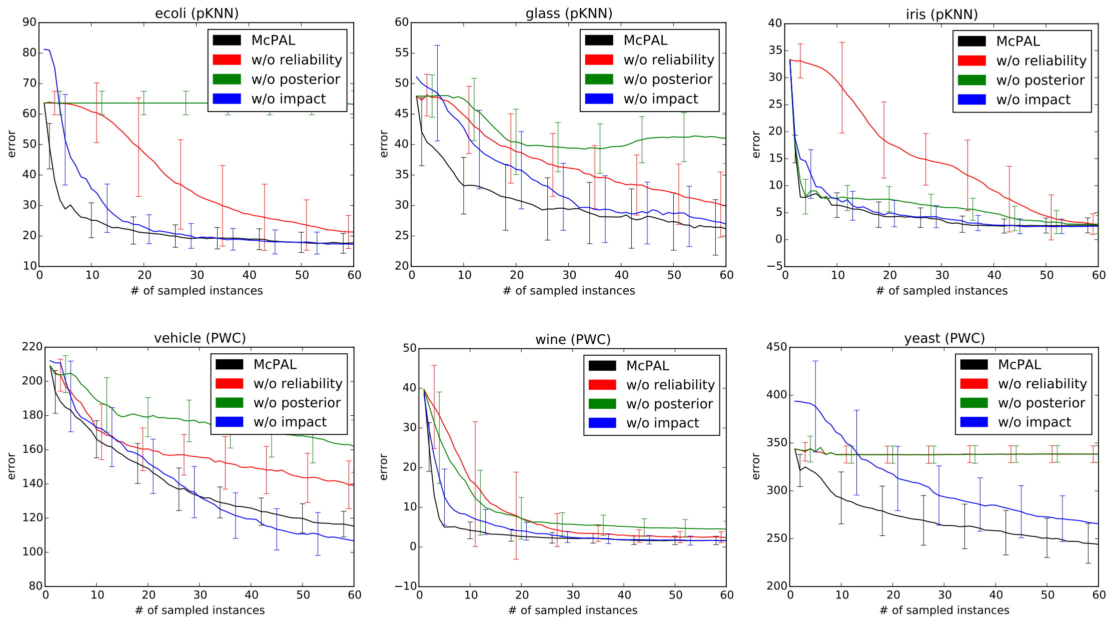


Figure 4. Learning curves of different variants of the McPAL algorithm on all six datasets. The upper plots show results from the pKNN classifier, the lower ones with the PWC.

In Tab. 3, we summarized the mean execution time of all algorithms on every dataset. Our proposed method does require more time to sample an instance than its competitors with exception of the VoI algorithm, which takes much longer than any other algorithm used in the experiments. Due to the higher complexity of the McPAL method in comparison to more simple methods like uncertainty-based ones, a longer execution time is to be expected. Considering the performance and stability of McPAL mentioned before, the increased time requirement is still a good trade off. In contrast to the fast methods, McPAL has an additional factor which is the sum over each labeling that is dependent on the M value.

5 CONCLUSION

This paper addresses active learning for multiple classes. This challenging topic opens up different aspects like the combination of the posterior vector into one comparable score. In this paper, we proposed a new multi-class probabilistic active learning method (McPAL) that addresses this problem in a decision-theoretic way. To this end, we developed a generalized probabilistic model that combines all of our mentioned influence factors impact, posterior, and the reliability of the posterior. Our approach directly optimizes a performance measure like accuracy, is non-myopic and fast. We showed how the influence factors depend on each other in our probabilistic framework and evaluated their behavior in multiple experiments. Especially the combination of the posterior and its reliability makes a huge difference. Our experimental comparison with the most relevant multi-class active learning approaches shows that McPAL is superior in most cases or at least comparable. We suggest that our approach can still be optimized by replacing the proxies of our influence factors by even more appropriate ones, which will be part of our future research. The complete results together with an implementation are available at our companion website[7].

ACKNOWLEDGEMENTS

We would like to thank the Pychoinformatics Labs at OVG-University Magdeburg for letting us use their cluster, in particular Michael Hanke and Alex Waite, as well as our colleague Pawel Matuszyk for all discussions. The ternary plots are generated with python-ternary [9].

[7] http://kmd.cs.ovgu.de/res/mcpal/

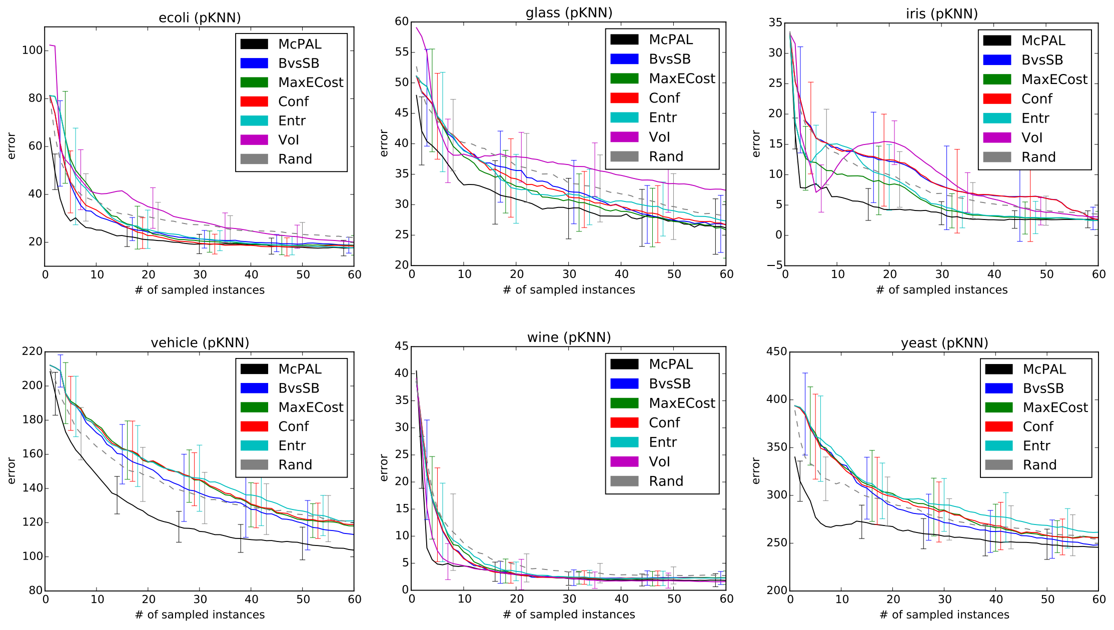


Figure 5. Learning curves of McPAL and its competitors on all six datasets using the pKNN classifier.

Table 4. Mean error and standard deviation of the all algorithms on our six datasets using the Parzen window classifier. We report the results after 20, 40, and 60 acquired labels. The best method is printed in bold numbers.

20 samples	ecoli	glass	iris	vehicle	wine	yeast
McPAL	**22.70** (± *4.45*)	**30.17** (± *4.22*)	**3.94** (± *1.97*)	149.14 (± *11.94*)	**2.66** (± *1.43*)	**275.24** (± *26.35*)
BvsSB	24.75 (± *4.84*)	35.95 (± *5.57*)	12.63 (± *7.06*)	148.68 (± *18.25*)	2.80 (± *1.67*)	289.90 (± *23.13*)
MaxECost	25.42 (± *6.63*)	33.33 (± *5.09*)	8.23 (± *6.24*)	155.98 (± *17.71*)	2.95 (± *1.88*)	294.20 (± *32.95*)
Conf	24.64 (± *7.07*)	33.93 (± *5.02*)	12.48 (± *7.32*)	156.52 (± *17.19*)	2.90 (± *1.79*)	292.92 (± *34.42*)
Entr	26.94 (± *8.01*)	33.04 (± *5.50*)	14.61 (± *3.17*)	153.44 (± *18.82*)	3.41 (± *1.76*)	298.60 (± *32.63*)
VoI	40.14 (± *9.59*)	38.20 (± *3.98*)	16.55 (± *2.67*)	NA	2.89 (± *2.68*)	NA
Rand	32.52 (± *7.89*)	36.69 (± *5.00*)	9.91 (± *4.47*)	**145.38** (± *13.27*)	4.35 (± *3.04*)	300.12 (± *23.56*)
40 samples	ecoli	glass	iris	vehicle	wine	yeast
McPAL	**19.15** (± *4.06*)	**29.14** (± *4.22*)	**2.85** (± *1.58*)	125.88 (± *8.99*)	1.78 (± *1.06*)	**258.36** (± *24.40*)
BvsSB	21.02 (± *4.42*)	32.28 (± *4.36*)	11.78 (± *7.78*)	**122.90** (± *14.43*)	1.92 (± *1.26*)	273.52 (± *22.95*)
MaxECost	20.80 (± *4.10*)	29.70 (± *4.46*)	7.70 (± *6.44*)	131.82 (± *14.44*)	1.90 (± *1.16*)	274.54 (± *30.65*)
Conf	19.60 (± *4.30*)	29.79 (± *4.87*)	11.69 (± *7.79*)	133.56 (± *14.90*)	1.94 (± *1.19*)	276.36 (± *32.40*)
Entr	23.55 (± *4.80*)	30.64 (± *4.61*)	13.88 (± *3.49*)	139.02 (± *18.57*)	**1.77** (± *1.14*)	284.38 (± *28.05*)
VoI	41.46 (± *7.22*)	38.06 (± *3.78*)	16.74 (± *2.58*)	NA	1.92 (± *1.89*)	NA
Rand	29.80 (± *6.57*)	34.57 (± *5.18*)	8.28 (± *4.03*)	129.88 (± *13.31*)	2.65 (± *1.61*)	281.84 (± *25.48*)
60 samples	ecoli	glass	iris	vehicle	wine	yeast
McPAL	**18.41** (± *3.69*)	**27.08** (± *3.95*)	**5.81** (± *2.54*)	115.26 (± *7.60*)	1.63 (± *1.06*)	**244.12** (± *20.71*)
BvsSB	19.69 (± *4.44*)	29.71 (± *4.22*)	12.71 (± *7.64*)	**113.42** (± *9.95*)	1.76 (± *1.13*)	259.68 (± *22.66*)
MaxECost	20.29 (± *4.55*)	27.99 (± *4.25*)	8.12 (± *5.62*)	120.06 (± *12.42*)	1.66 (± *1.03*)	257.60 (± *26.75*)
Conf	19.91 (± *4.29*)	28.46 (± *4.59*)	12.40 (± *7.59*)	122.34 (± *13.39*)	1.62 (± *1.12*)	259.98 (± *25.76*)
Entr	22.54 (± *4.55*)	31.65 (± *4.91*)	11.94 (± *4.07*)	126.06 (± *14.60*)	**1.53** (± *1.00*)	272.44 (± *24.93*)
VoI	34.20 (± *5.78*)	37.22 (± *4.72*)	15.06 (± *3.49*)	NA	1.54 (± *1.22*)	NA
Rand	28.32 (± *5.65*)	33.55 (± *5.17*)	6.92 (± *2.76*)	123.28 (± *13.26*)	2.30 (± *1.43*)	276.42 (± *26.98*)

REFERENCES

[1] Alekh Agarwal, 'Selective sampling algorithms for cost-sensitive multiclass prediction', in *Proceedings of the 30th International Conference on Machine Learning*, pp. 1220–1228, (2013).

[2] Arthur Asuncion and David J. Newman. UCI machine learning repository, 2015.

[3] Olivier Chapelle, 'Active learning for parzen window classifier', in *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pp. 49–56, (2005).

[4] Gang Chen, Tian-jiang Wang, Li-yu Gong, and Perfecto Herrera, 'Multi-class support vector machine active learning for music annotation', *International Journal of Innovative Computing, Information and Control*, **6**(3), 921–930, (2010).

[5] Po-Lung Chen and Hsuan-Tien Lin, 'Active learning for multiclass cost-sensitive classification using probabilistic models', in *Technologies and Applications of Artificial Intelligence (TAAI), 2013 Conference on*, pp. 13–18, (Dec 2013).

[6] Husheng Guo and Wenjian Wang, 'An active learning-based svm multiclass classification model', *Pattern Recognition*, **48**(5), 1577–1597, (2015).

[7] Isabelle Guyon, Gavin Cawley, Gideon Dror, Vincent Lemaire, and Alexander Statnikov. Active learning challenge: Challenges in machine learning, volumn 6, 2012.

[8] Yaroslav O Halchenko and Michael Hanke, 'Open is not enough. let's take the next step: an integrated, community-driven computing platform for neuroscience', *Frontiers in neuroinformatics*, **6**, (2012).

[9] Marc Harper, Bryan Weinstein, Cory Simon, chebee7i, Nick Swanson-Hysell, The Gitter Badger, Maximiliano Greco, and Guido Zuidhof. python-ternary: Ternary plots in python, December 2015.

[10] Paril Jain and Ajay Kapoor, 'Active learning for large multi-class problems', in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 762–769. IEEE, (2009).

[11] Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos, 'Multi-class active learning for image classification', in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 2372–2379, (June 2009).

[12] Ajay J Joshi, Fatih Porikli, and Nikolaos P Papanikolopoulos, 'Scalable active learning for multiclass image classification', *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **34**(11), 2259–2273, (2012).

[13] Christine Körner and Stefan Wrobel, 'Multi-class ensemble-based active learning', in *Machine Learning: ECML 2006*, 687–694, Springer, (2006).

[14] Daniel Kottke, Georg Krempl, and Myra Spiliopoulou, 'Probabilistic active learning in data streams', in *Advances in Intelligent Data Analysis XIV - 14th Int. Symposium, IDA 2015, St. Etienne, France*, eds., Tijl De Bie and Elisa Fromont, volume 9385 of *Lecture Notes in Computer Science*, pp. 145–157. Springer, (2015).

[15] Georg Krempl, Daniel Kottke, and Vincent Lemaire, 'Optimised probabilistic active learning (OPAL) for fast, non-myopic, cost-sensitive active classification', *Machine Learning (Special Issue of ECML PKDD 2015)*, **100**(2), (2015).

[16] Georg Krempl, Daniel Kottke, and Myra Spiliopoulou, 'Probabilistic active learning: A short proposition', in *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI2014), August 18 – 22, 2014, Prague, Czech Republic*, eds., Torsten Schaub, Gerhard Friedrich, and Barry O'Sullivan, volume 263 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, (2014).

[17] David D. Lewis and William A. Gale, 'A sequential algorithm for training text classifiers', in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, pp. 3–12, New York, NY, USA, (1994). Springer-Verlag New York, Inc.

[18] Andrew Y. Ng and Michael I. Jordan, 'On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes', in *Advances in Neural Information Processing Systems 14, NIPS*, (2001).

[19] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, *Numerical Recipes in Fortran 77: The Art of Scientific Computing*, Cambridge University Press, 2 edn., 1992.

[20] Nicholas Roy and Andrew McCallum, 'Toward optimal active learning through sampling estimation of error reduction', in *Proc. of the 18th Int. Conf. on Machine Learning, ICML 2001, Williamstown, MA, USA*, pp. 441–448, San Francisco, CA, USA, (2001). Morgan Kaufmann.

[21] Robin Senge, Stefan Bösner, Krzysztof Dembczyński, Jörg Haasenritter, Oliver Hirsch, Norbert Donner-Banzhoff, and Eyke Hüllermeier, 'Reliable classification: Learning classifiers that distinguish aleatoric and epistemic uncertainty', *Information Sciences*, **255**, 16–29, (January 2014).

[22] Burr Settles, *Active Learning*, number 18 in Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan and Claypool Publishers, 2012.

[23] Katrin Tomanek and Udo Hahn, 'Reducing class imbalance during active learning for named entity annotation', in *Proceedings of the 5th International Conference on Knowledge Capture (K-CAP 2009), September 1-4, 2009, Redondo Beach, California, USA*, eds., Yolanda Gil and Natasha Fridman Noy, pp. 105–112. ACM, (2009).

[24] Simon Tong and Edward Chang, 'Support vector machine active learning for image retrieval', in *Proceedings of the ninth ACM international conference on Multimedia*, pp. 107–118. ACM, (2001).

[25] R. Wang, C. Y. Chow, and S. Kwong, 'Ambiguity-based multiclass active learning', *IEEE Transactions on Fuzzy Systems*, **24**(1), 242–248, (Feb 2016).

[26] Rong Yan and Alexander Hauptmann, 'Multi-class active learning for video semantic feature extraction', in *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on*, volume 1, pp. 69–72. IEEE, (2004).

[27] Rong Yan, Jie Yang, and Alexander Hauptmann, 'Automatically labeling video data using multi-class active learning', in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 516–523. IEEE, (2003).

[28] Yi Yang, Zhigang Ma, Feiping Nie, Xiaojun Chang, and Alexander G Hauptmann, 'Multi-class active learning by uncertainty sampling with diversity maximization', *International Journal of Computer Vision*, **113**(2), 113–127, (2014).

Chapter 6

How to Select Information That Matters: A Comparative Study on Active Learning Strategies for Classification

The following chapter has been published as:

How to Select Information That Matters: A Comparative Study on Active Learning Strategies for Classification
Christian Beyer, Georg Krempl, and Vincent Lemaire.

Stefanie Lindstaedt, Tobias Ley, and Harald Sack. Proceedings of the 15th Int. Conf. on Knowledge Technologies and Data-driven Business, i-KNOW 2015, ACM, 2015. ISBN: 978-1-4503-3721-2.
DOI: 10.1145/2809563.2809594

The original publication is available at ACM
https://doi.org/10.1145/2809563.2809594

How to Select Information That Matters

A Comparative Study on Active Learning Strategies for Classification

Christian Beyer
University Magdeburg
Universitätsplatz 2, Building 29
39016 Magdeburg, Germany
christian.beyer@st.ovgu.de

Georg Krempl
Knowledge Management & Discovery, Univ. Magdeburg
Universitätsplatz 2, Building 29
39016 Magdeburg, Germany
georg.krempl@ovgu.de

Vincent Lemaire
Orange Labs
2 avenue Pierre Marzin
22300 Lannion, France
vincent.lemaire@orange.com

ABSTRACT

Facing ever increasing volumes of data but limited human annotation capabilities, active learning strategies for selecting the most informative labels gain in importance. However, the choice of an appropriate active learning strategy itself is a complex task that requires to consider different criteria such as the informativeness of the selected labels, the versatility with respect to classification algorithms, or the processing speed. This raises the question, which combinations of active learning strategies and classification algorithms are the most promising to apply. A general answer to this question, without application-specific, label-intensive experiments on each dataset, is highly desirable, as active learning is applied in situations with limited labelled data. Therefore, this paper studies several combinations of different active learning strategies and classification algorithms and evaluates them in a series of comparative experiments.

CCS Concepts

•**Theory of computation** → **Active learning;**

Keywords

Active Learning; Selective Sampling; Uncertainty Sampling; Probabilistic Active Learning

1. INTRODUCTION

While the volumes of data are constantly increasing [9], human annotation and supervision capacities remain limited. This raises the need for approaches that help in the efficient allocation of these capacities [15]. Active machine learning [22] provides such approaches for determining and selecting the most valuable information. In classification tasks, this corresponds to selecting the instance from a set of candidates, whose label is expected to improve a classifier's performance the most [23]. Given the large number of approaches that have been proposed in literature, the choice of the most appropriate active learning strategy constitutes itself a complex task: multiple criteria such as the informativeness of the selected labels, the versatility of the approach with respect to classification algorithms, or the processing speed of the approach need to be considered.

Active learning is applied in situations with very limited initial labelled data. Thus, knowing the overall most promising combinations of active learning strategies and classification algorithms without performing application-specific, label-intensive experiments on each novel dataset is highly desirable. This paper addresses this question by providing results of an experimental performance comparison of several combinations of popular classification algorithms and active learning strategies. In Section 2, related surveys are reviewed before discussing selected active learning strategies. These strategies are then experimentally evaluated in Section 3, before concluding in Section 4.

i-KNOW '15, October 21 - 23, 2015, Graz, Austria
© 2015 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-3721-2/15/10…$15.00
DOI: http://dx.doi.org/10.1145/2809563.2809594

2. ACTIVE LEARNING APPROACHES

This paper addresses the *pool-based* [23, 6] active learning scenario for *binary* classifiers, where an active classifier has access to a pool of unlabelled instances $\mathcal{U} = \{(x, .)\}$. Repeatedly, *the best instance* $(x^*, .) \in \mathcal{U}$ is selected, its label y^* is requested from an oracle, and it is moved from $\mathcal{U}$ to the set of labelled instanced $\mathcal{L} = \{(x, y)\}$ to retrain the classifier. In particular, this paper focuses on a sequential labelling scenario, in contrast to batch-based active learning where multiple instances are labelled in one iteration [10]. Various existing approaches for this scenario are surveyed in [22, 6, 23, 8]. The technical report [22], the machine learning encyclopedia entry [6] on active learning, and more recently the textbook [23] provide an introduction to active learning, as well as a good overview on various families of active learning approaches. While comparing theoretical aspects of the different approaches, they do not include an empirical evaluation. Recently, [8] surveys different approaches based on uncertainty sampling and instance correlation and provide a categorisation of different approaches. However, the performance analysis in that review is limited to runtime evaluations, thus leaving the question on the classification performance of different approaches open. An experimental classification performance evaluation and comparison of some approaches was done in the active learning challenge, published in [11]. It is remarked therein that a key to success in active learning is handling the trade-off between *exploration* and *exploitation*: the former samples in regions with yet little collected information, the latter investigates regions where the current model suspects the decision boundary. According to [11, page iv], the overall winners use combinations of random and uncertainty sampling to tackle this trade-off.

This comparative study's focus are *fast* approaches that are *usable with any classification technique*. Building on the results above, we compare random sampling, uncertainty sampling, and a combination of both that tackles exploration-exploration. In addition to these *popular* approaches, we include the *very recently* proposed probabilistic active learning approach, which implicitly balances exploration-exploration. We now briefly review these approaches, before continuing with the experimental evaluation in the next chapter.

2.1 Random Sampling

A simple and fast baseline is *random sampling*, where instances are selected at random with equal probability. Despite the simplicity of this *purely explorative* strategy, it has been shown to be difficult to be beaten consistently [2] and is one of the most popular active learning baselines [11].

2.2 Uncertainty Sampling

A very popular active learning strategy is *uncertainty sampling* [17], which is frequently used as baseline (e.g. in the active learning competition [11]). This is a purely exploitative strategy that relies on the current model to compute so-called *uncertainty measures.* These serve as proxies for a candidate's impact on the classification performance, and the candidate with the highest uncertainty is selected for labelling. In the seminal work of [17], a probabilistic classifier is used on a candidate to compute the posterior of its most likely class. The absolute difference between this posterior estimate and 0.5 is used as uncertainty measure (lower values denoting higher uncertainty). The formula for picking x^*_{LC} is the following according to [22]:

$$x^*_{LC} = \underset{x}{argmax}\ (1 - P_\theta(\hat{y} \mid x)) \tag{1}$$

x^*_{LC} is the instance from the pool of unlabelled data D_u which our model θ is least confident in while $\hat{y}$ is the class for which the model calculated the highest posterior estimate so $\hat{y} = \underset{y}{argmax}\ P_\theta(y \mid x)$. In addition to this confidence-based uncertainty measure, other common measures [23] are entropy or the margin between a candidate and the decision boundary. However, [22] notes that for *binary* classification problems classifiers the measures margin, confidence and entropy result in the same ranking and querying of instances.

This strategy is easy to implement and computationally efficient, having an asymptotic time complexity of $O(|\mathcal{U}|)$. Thus, it is also usable in time critical applications, or in big data scenarios with large numbers of unlabelled instances, or on fast data streams [27]. Nevertheless, a known disadvantage [25] of uncertainty sampling is that these proxies do not consider the number of similar instances on which the posterior estimates are made or the decision boundaries are drawn. The reported results of empirical evaluations are somewhat inconclusive, with some authors (e.g. [4, 20, 13]) reporting even worse performance on some data sets than random sampling. Its major problems are that it can get stuck in regions with high Bayesian error, especially when the data is not linearly separable. Additionally, as this strategy queries instances that are close to the current decision boundary, it is prone to missing subconcepts if the initial decision boundary is unfavourable for the data. Furthermore, it can also tend to query outliers which are not representative for the underlying distribution. Figure 1 illustrates some of the problems while the work of [26] and [25] discuss the issue of querying outliers. Following next is a short description of a mixed strategy that combines random and uncertainty sampling.

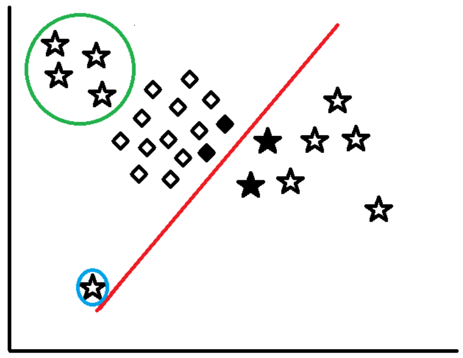


Figure 1: This figure shows a configuration during the active learning process on a two-class problem. The red line is the current decision boundary and the coloured stars and squares are the labelled instances. The stars on the top left are a subconcept which will probably be missed by uncertainty sampling because those instances are far away from the decision boundary which means the classifier is very confident in their prediction. The star with the blue circle on the other hand is an outlier that is very close to the current decision boundary and therefore highly likely to be queried for labelling.

2.3 Semi-Random Sampling

The combination of uncertainty and random sampling to combine exploitation and exploration has been suggested for example in [16, 11, 27]. Most recently, [27] uses a mix of random and uncertainty sampling on streams to tackle the problem of missing exploration with uncertainty measures. This is especially useful in stream-based active learning where concepts and thus the optimal decision boundary might change over time. The authors speculate that in a static scenario it is likely that uncertainty sampling beats the mixed strategies, as the decision boundary does not change over time. We investigate this hypothesis by studying the performance of a mixed strategy for pool-based active learning, which switches between uncertainty and random sampling. This strategy alternately applies random sampling and uncertainty sampling, beginning with the initial instance being selected randomly from the unlabelled pool D_u. This strategy has the same asymptotic time complexity as uncertainty sampling, but is faster by a constant factor due to using random selection half of the time.

2.4 Probabilistic Active Learning

Probabilistic active learning is a novel approach [14] that directly optimises a performance measure like accuracy, using statistically sound methods to guide the degree of exploitation and exploration. In this aspect it is comparable to error reduction approaches (proposed in [19]), while still having linear complexity like the fast uncertainty methods. For binary classification with Parzen Window classifiers, it was already shown that probabilistic active learning achieves comparable or superior performance than error reduction.

Probabilistic active learning builds on the smoothness assumption commonly used in semi-supervised learning [5], which suggests that the influence of an instance on the classification process is the highest in its neighbourhood:

> *Semi-supervised smoothness assumption*: If two points x_1, x_2 in a high-density region are close, then so should be the corresponding outputs y_1, y_2.

Therefore, the method proposed in [14] considers within the neighbourhood of an instance the number of labelled instances n and the share of positive labels therein $\hat{p} = \frac{n_+}{n}$. These two values are the necessary label statistics $ls = (n, \hat{p})$, which should be provided by the classifier being used. As the real posterior p of that neighbourhood and the label realisation y of the instance under consideration are unknown, they are modelled as hidden variables. The so-called probabilistic gain is calculated as the expectation over all possible realisations of p and y of the gain in classification performance. This gain is then weighted with the density in an instance's neighbourhood, considering the union of the labelled and unlabelled pool $D_u \cup D_l$, in order to prefer dense regions and avoid outliers. This probabilistic gain calculation models the true posterior p within the neighbourhood as being Beta-distributed, and the label realisation y as being Bernoulli-distributed with p as an input. Thus, the number of positive instances in the neighbourhood $n_+ = n \cdot \hat{p}$ is binomially distributed.

For accuracy or misclassification loss, a closed-form solution for computing the probabilistic gain is given in [13], which is called optimised probabilistic active learning (OPAL). The gain G can be written as:

$$G_{\text{OPAL}}(n, \hat{p}, \tau, m) = \frac{(n+1)}{m} \cdot \binom{n}{n \cdot \hat{p}} \cdot \tag{2}$$

$$\left(\mathrm{I}_{ML}(n, \hat{p}, \tau, 0, 0) - \sum_{k=0}^{m} \mathrm{I}_{ML}(n, \hat{p}, \tau, m, k) \right) \tag{3}$$

Here, τ is the cost of a false positive (normalised such that the costs of a false positive and a false negative add up to one), m denotes how many labels can be purchased in a given neighbourhood, and $I_{ML}(n, \hat{p}, \tau, m, k)$ is a function that is proportional to the expected misclassification loss in case k positive labels were among the m purchased ones:

$$\mathrm{I}_{ML}(n, \hat{p}, \tau, m, k) = \binom{m}{k} \cdot \tag{4}$$

$$\begin{cases} (1-\tau) \cdot \frac{\Gamma(1-k+m+n-n\hat{p})\Gamma(2+k+n\hat{p})}{\Gamma(3+m+n)} & \frac{n\hat{p}+k}{n+m} < \tau \\ (\tau - \tau^2) \cdot \frac{\Gamma(1-k+m+n-n\hat{p})\Gamma(1+k+n\hat{p})}{\Gamma(2+m+n)} & \frac{n\hat{p}+k}{n+m} = \tau \\ \tau \cdot \frac{\Gamma(2-k+m+n-n\hat{p})\Gamma(1+k+n\hat{p})}{\Gamma(3+m+n)} & \frac{n\hat{p}+k}{n+m} > \tau \end{cases} \tag{5}$$

Here, $\Gamma(z)$ is Legendre's gamma function (see e.g. [18, p. 206]).

For computing the probabilistic gain, the label statistics of an instance's neighbourhood are required, which consist of total number of labels (n) and the share of positives therein ($\hat{p}$). These statistics need to be estimated. In [13, 14], it is argued that using estimates provided by a probabilistic classifier might be favourable to using kernel frequency estimates as substitutes. For investigating this experimentally, different ways of computing the label statistics for different classifiers need to be specified.

When using kernel frequency estimates as substitutes, [14] propose the following formula that employs Gaussian kernels with a bandwidth of σ:

$$LC(x, \mathcal{L}) \approx \sum_{x_i \in \mathcal{L}} \exp\left(- \frac{\|x - x_i\|^2}{2\sigma^2} \right) \tag{6}$$

The total number of labels is then $n = LC(x, \mathcal{L})$, where $\mathcal{L}$ is the set of all labelled instances, and the the share of positives is $\hat{p} = LC(x, \mathcal{L}_+)/LC(x, \mathcal{L})$, where $\mathcal{L}_+$ is the subset of labelled positive instances.

For *Parzen-Window Classifiers* [4], which use kernel density estimates for computing an instance's posterior probabilities, the kernel frequency estimates above for $\hat{p}$ are identical to the classifier's posterior estimates. However, for *Naive Bayes Classifiers* these frequency estimates differ from the posterior estimates, due to the conditional independence assumed when computing the latter. Therefore, the classifier's estimates should be used directly for $\hat{p}$. For *k-Nearest Neighbour Classifiers*, these posterior estimates are obtained by the number of positives among an instance's k nearest neighbours. In analogy, for *Tree-Based Classifiers* such as Hoeffding Trees [7], the probabilistic estimates are obtained from the summary statistics in an instance's leaf, i.e. by simply dividing the number of positives by the total number of labels processed in that leaf.

In the classification algorithms discussed above, a label influences solely a particular region in the feature space. However, for some classifiers this does not hold. For example, in *Logistic Regression Classifiers* an instance might alter the decision on instances that are far away. Thus, even though Logistic Regression Classifiers provide probabilistic estimates that might be used for $\hat{p}$, they might be not suited for probabilistic active learning.

3. EXPERIMENTAL COMPARISON

Motivated by the relationship between the active learning strategies described in Section 2, the following three hypotheses guide the experimental evaluation:

1. Probabilistic active learning outperforms random, semi-random and uncertainty sampling.
2. The performance of probabilistic active learning drops if the label statistics are calculated independently of the classifier being used.
3. Semi-random sampling does not outperform random and uncertainty sampling at the same time.

The first hypothesis is motivated by the capability of probabilistic active learning to balance exploration and exploitation by computing the expected improvement in classification performance in an instance's neighbourhood, rather than using a heuristic approach. However, this relies on good estimates of the labelled information in an instance's neighbourhood, which are provided by the label statistics. These estimates depend on the classifier, thus computing them independently from the classifier is expected to deteriorate the performance, motivating the second hypothesis. The third hypothesis is motivated by the speculation in [27] that mixed strategies might be inferior in a pool-based setting with static concepts (as in our setting). According to this hypothesis, the performance of semi-random sampling should be between that of random and uncertainty sampling.

For testing these hypotheses, we follow the standard active learning assumptions, discussed and motivated in [22]:

1. All labels cost the same.
2. The labels that are bought are always correct.
3. The classifier learns incrementally on the actively selected labels, without any other change.

3.1 Experimental Setup

Active learning works on the trade-off between minimising the number of labels and maximising classification performance. For a single experiment, this trade-off is commonly visualised using learning curves, which depict the classifier's performance at different amounts of labelled instances. However, for a multitude of combinations of active learning approaches and datasets (as in this comparative study), a multitude of curves need to be compared. For matters of space and readability, different approaches for aggregating this information were used in literature. One proposed solution is to compare the area under the learning curve [11] but this method loses information about dominance at the different stages and might be misleading when learning curves intersect. Therefore, we use the approach suggested in [13] of pairwise comparisons at specific points in the learning process, in order to see which strategy dominates or is dominated by another strategy at which point in the learning process. Furthermore, in order to improve reliability of the results, we use n-fold-cross-validation to divide the datasets into different partitions of test and training sets. The experimental setup used for the comparison of active learning strategies is summarised by the following workflow:

1. Employ the selected strategies with the selected classifiers and datasets.
2. Compare the accuracy of two competing strategies after a specific number of instances were labelled and create two performance vectors for that point of comparison by collecting the achieved performance from all the folds of the 10-fold-cross-validation and do that for 10 random seeds. This gives us two vectors of the length 100 as there are 10 folds for each of the 10 seeds equalling a 100-fold-cross-validation.
3. Test if the performance vector of one strategy is significantly better or worse using a two-sided Wilcoxon test with a significance level of 0.05.
4. Repeat steps 2 and 3 for all classifiers on the individual datasets and also over all datasets at the same time which gives us a summary of how the strategies perform for a specific classifier over all datasets. The chosen comparison points are the performances obtained after labelling 20 and 40 instances. Accuracy is selected as performance measure.
5. Check if the results of step 4 are in line with the hypotheses or contradict them.

Table 1: Specifications of the data sets that were used for the experiments

Data Set	Instances	Attributes	Pr(+)
Seeds	210	7	33%
Vertebral	310	6	32%
Haberman	306	3	73%
Checkerboard1	308	2	44%
Checkerboard2	392	2	49%

3.1.1 Datasets

For the experiments the following real-world datasets from the UCI machine learning repository [1] are used: haberman, seeds, vertebral. Additionally, two synthetic datasets are included, namely checkerboard1 and checkerboard2 [4, 13]. The datasets are preprocessed such that there are no missing or invalid values and normalised such that all attribute values are between zero and one. The specifications of the data sets can be seen in Table 1. All the datasets are randomised and divided into ten folds, which are then used in the cross-validation of all active learning strategies. Since the datasets are small and the learning process converges quickly, the budget is set to 40 instances.

3.1.2 Algorithms

The compared active learning approaches are random sampling (uniform selection probability), semi-random and uncertainty sampling (both using confidence as uncertainty measure), and probabilistic active learning (using accuracy with $\tau = 0.5$ as performance measure) which were introduced in Sections 2.1 till 2.4.

All active learning strategies are evaluated on the same set of (incremental) classifiers. Those classifiers, implemented in MOA [3] and WEKA [12], are Hoeffding trees, Naive Bayes, logistic regression, k-nearest-neighbour and a Parzen-Window classifier which was implemented by the authors and is described in [4]. All algorithms were run on a desktop computer (Intel i5-760 with 2.8GHz and 8GB RAM).

The label statistics are once calculated by using the probabilistic classifier's posterior estimate for the values of the share of positives ($\hat{p}$) in a neighbourhood. Furthermore, to evaluate the effect of calculating the label statistics independently of the classifier, estimates based on kernel frequency estimates (as in [13]) over the labels are used.

3.2 Results

Based on the three hypotheses stated above, we now summarise our findings in the next subsections. Tables 3 and 2 provide the complete results of the experimental evaluation.

Table 3 shows the performance comparison after 20 and 40 labels over all datasets for different pairs of active learning strategies. The numbers are the percentages of wins of the strategy in the row versus the strategies in the columns, excluding ties. Thus, symmetric values sum up to one. Significantly better results of a two-sided Wilcoxon test with a significance level of 0.05 are denoted with a '*', significantly worse ones with a '-'. The active learning strategies are denoted with Pal (probabilistic active learning), Conf (confidence-based uncertainty sampling), Ran (random sampling), and Semi (semi-random sampling). For the columns on the left, the posterior estimates $\hat{p}$ come from the probabilistic classifier, while for the columns on the right they are calculated independently of the classifier by using kernel frequency estimates. In both cases, the number of labels n is calculated by kernel frequency estimates.

Table 2 summarises for different classifiers the effect on Pal's performance of using independently calculated posterior estimates against estimates takes from the probabilistic classifier. That is, the values correspond to the number of wins (excluding ties) of Pal with independently calculated posterior estimates (by using kernel frequency estimates) against Pal with estimates taken directly from the probabilistic classifier. A '*' shows that the performance is significantly better and a '-' shows that it is significantly worse using a two-sided Wilcoxon test with a significance level of 0.05. One can see that in the majority of cases calculating both parameters independently leads to a significantly worse classifier performance.

3.2.1 Probabilistic Active Learning Is Superior

In order to assess this statement, Table 3 provides the results for different classifiers. For a Parzen Window classifier (top-most cells), probabilistic active learning outperforms the other strategies significantly over all datasets, both after 20 and 40 acquired labels. This classifier's posterior estimates are kernel frequency estimates, thus there is no difference between its left and right subtables.

For Hoeffding Trees, this does only hold when posterior estimates by the classifier are used (64.26%, 64.92%, 62.7% at 20 labels, and 63.19%, 69%, 66.55% at 40 labels against confidence-based uncertainty sampling, random sampling, and semi-random sampling, respectively). When using independently calculated posterior estimates for probabilistic active learning, its performance is neither significantly better nor significantly worse than that of other approaches.

For Naive Bayes with posterior estimates by the classifier, Pal is again always significantly better. For Naive Bayes with kernel frequency estimates for the posterior, Pal is significantly better than random while not significantly worse than any other strategy.

For k-Nearest Neighbour and Logistic Regression, probabilistic active learning is not better: with k-NN it is significantly worse than uncertainty sampling or semi-random-sampling, but not significantly worse than random sampling. With logistic regression, results are inconclusive, but probabilistic active learning performs in some constellations significantly worse than uncertainty or random sampling. The reason for the weak performance of probabilistic active learning in combination with Logistic Regression is that here the smoothness assumption is violated, as an instance might influence the decision boundary at locations that are far away from its coordinates. The problem with k-Nearest Neighbour is a different one: here, the number of labels that are considered by the classifier is constantly set to three. Thus, the value n used in the label statistics is misleading the active learner. Overall, hypothesis one is confirmed for Parzen Window, Hoeffding Tree, and Naive Bayes, but not for k-Nearest Neighbour and Logistic Regression Classifiers.

3.2.2 Independently Calculated Label Statistics Reduce the Performance

The results discussed above already indicate an important relationship between the label statistics and the performance of the probabilistic active learning approach. To assess this relationship further, and to test the second hypothesis that classifier-independent calculation of these label statistics reduces the performance, Table 2 shows the results of a pairwise comparison between probabilistic active learning with and without independently computed posterior estimates. It depicts the percentage of cases where the performance with independent estimates was greater than the performance with estimates coming from the classifier. For example, the Naive Bayes classifier with independent estimates outperformed its counterpart with dependent estimates in 47.62% of the cases after 10 labels were bought and only in 25,76% of the cases after 40 labels were bought which is significantly worse being indicated by the '-' sign. A '*' ! would indicate that it performed better in most of the cases and that the result can be deemed significant.

Interestingly, the results depend on the learning stage: after processing the first ten labels (comparison point $CP = 10$), there is not yet a difference in performance between the two ways of calculating the label statistic's $\hat{p}$ (except for 3-Nearest Neighbour). In the later learning stages ($CP = 20, 30, 40$), this changes, and using independently estimated values for $\hat{p}$ significantly reduces performance for Hoeffding-Trees, Naive Bayes, and Logistic Regression. For 3-Nearest Neighbour, the results are different, but on this particular type of classifier the probabilistic active learning approach is not recommendable anyway.

One should note that this evaluation was limited to the effect of independent posterior estimates for $\hat{p}$, while always independently calculated estimates for the number of labels n were used. The situation of using for both values (for n and $\hat{p}$) kernel frequency estimates corresponds to using two classifiers, namely a Parzen-Window classifier for instance selection, and the chosen classifier for prediction. This is the typical scenario of label reusability as introduced by [24]. Summarising, the second hypothesis is confirmed for Hoeffding Trees, Naive Bayes, and Logistic Regression Classifiers.

Table 2: Effect of Independent Label Statistics Calculation

Labels	**H-Tree**	**Naive B.**	**Log. Reg.**	**3-NN**
CP=10	54.33%	47.62%	49.82%	34.92%-
CP=20	39.3%-	38.78%-	37.67%-	54.32%
CP=30	32.17%-	28.35%-	30.93%-	48.67%
CP=40	30.4%-	25.76%-	37.04%-	65.82%*

3.2.3 Semi-Random Sampling is not better than both Random and Uncertainty Sampling

The results in Table 3 confirm hypothesis three that that semi-random-sampling is with none of the classifiers consistently better (or worse) than both, random sampling and uncertainty sampling. That is, it is never at the same time dominating (or dominated by) both strategies. This supports the suggestions by [27] that a mixed strategy is inferior in a static setting because either uncertainty sampling will perform well or random will perform well and semi-random will end up in the middle of the two. However, this does not mean that a semi-random strategy is inferior to random or uncertainty sampling in every setting. For some configurations, semi-random sampling is slightly better than both, but in those cases the difference is never significant. Thus, in a real-world application where hold-out performance tests are difficult, semi-random sampling might help to avoid the worst-case performance. Nevertheless, for most classifier types probabilistic active learning seems to be the better choice, as it outperforms in general all three other methods when the label statistics are provided by the used classifier.

4. CONCLUSION

In this paper, the performance of popular active learning strategies in combination with different classification algorithms has been studied. These combinations were experimentally evaluated using 100-fold cross validation over several different real-world and synthetic datasets. The results confirm the finding of previous studies that neither pure exploration nor pure exploitation strategies perform consistently well, making the handling of the trade-off between *exploration* and *exploitation* a key challenge. In addition, the results show that the recently proposed probabilistic active learning approach significantly outperforms uncertainty-sampling-based strategies when used with Bayes, Naive Bayes or Decision-Tree Classifiers, but works not well on k-Nearest Neighbour or Logistic Regression Classifiers. Furthermore, it is shown that using a probabilistic classifier's estimates for the label statistics is in most cases better than using estimates that were calculated independently of the classifier. Finally, the results confirm the recently stated conjecture [27] that a hybrid between random and uncertainty sampling does not outperform both strategies at the same time in a pool-based setting.

While several combinations of active learning and classification approaches have been evaluated in this paper, this comparative study is by no means complete. Future work will focus on evaluating further combinations, as well as investigating further ways of computing better label statistics for some classification algorithms. Furthermore, comparisons for other active learning settings like user-based visually-supported active learning [21] would be insightful.

5. ACKNOWLEDGMENTS

We would like to thank Daniel Kottke and Myra Spiliopoulou.

6. REFERENCES

[1] A. Asuncion and D. J. Newman. UCI machine learning repository, 2015.

[2] J. Attenberg, P. Melville, F. Provost, and M. Saar-Tsechansky. *Selective Data Acquisition for Machine Learning*, chapter 5. CRC Press, Inc., 2011.

[3] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. Moa: Massive online analysis. *The Journal of Machine Learning Research*, 11:1601–1604, 2010.

[4] O. Chapelle. Active learning for parzen window classifier. In *Proc. of the 10th Int. Workshop on AI and Statistics*, pages 49–56, 2005.

[5] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-supervised Learning.* MIT Press, 2006.

[6] D. Cohn. Active learning. In C. Sammut and G. I. Webb, editors, *Encyclopedia of Machine Learning*, pages 10–14. Springer, 2010.

[7] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proc. of the 6th ACM SIGKDD int. conf. on Knowledge discovery and data mining (KDD00)*, pages 71–80. ACM, 2000.

[8] Y. Fu, X. Zhu, and B. Li. A survey on instance selection for active learning. *Knowledge and Information Systems*, 35(2):249–283, 2012.

[9] J. Gantz and D. Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east, December 2012.

[10] Y. Guo and D. Schuurmans. Discriminative batch mode active learning. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems (NIPS2007)*, pages 593–600, 2007.

[11] I. Guyon, G. Cawley, G. Dror, V. Lemaire, and A. Statnikov, editors. *Active Learning Challenge*, volume 6 of *Challenges in Machine Learning.* Microtome Publishing, 2011.

[12] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[13] G. Krempl, D. Kottke, and V. Lemaire. Optimised probabilistic active learning (OPAL) for fast, non-myopic, cost-sensitive active classification. *Machine Learning*, 2015.

[14] G. Krempl, D. Kottke, and M. Spiliopoulou. Probabilistic active learning: Towards combining versatility, optimality and efficiency. In S. Dzeroski, P. Panov, D. Kocev, and L. Todorovski, editors, *Proc. of the 17th Int. Conf. on Discovery Science (DS)*, volume 8777 of *LNCS*, pages 168–179. Springer, 2014.

[15] G. Krempl, I. Zliobaitė, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, and J. Stefanowski. Open challenges for data stream mining research. *SIGKDD Explorations*, 16(1):1–10, 2014.

[16] L. Lan, H. Shi, Z. Wang, and S. Vucetic. Active learning based on parzen window. *Journal of Machine Learning Research*, 16:99–112, 2011.

[17] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proc. of the 17th annual int. ACM SIGIR conf. on Research and development in information retrieval*, SIGIR '94, pages 3–12, 1994.

[18] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in Fortran 77: The Art of Scientific Computing.* Cambridge University Press, 2 edition, 1992.

[19] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proc. of the 18th Int. Conf. on Machine Learning, ICML 2001*, pages 441–448, 2001.

[20] A. I. Schein and L. H. Ungar. Active learning for logistic regression: an evaluation. *Machine Learning*, 68(3):235–265, 2007.

[21] C. Seifert and M. Granitzer. User-based active learning. In *Proc. of 10th Int. Conf. on Data Mining Workshops (ICDMW2010)*, pages 418–425, 2010.

[22] B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin-Madison, Madison, Wisconsin, USA, 2009.

[23] B. Settles. *Active Learning.* Number 18 in Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers, 2012.

[24] K. Tomanek and K. Morik. Inspecting sample reusability for active learning. In I. Guyon, G. C. Cawley, G. Dror, V. Lemaire, and A. R. Statnikov, editors, *AISTATS workshop on Active Learning and*

Experimental Design, volume 16, pages 169–181. JMLR.org, 2011.

[25] J. Zhu, H. Wang, B. K. Tsou, and M. Y. Ma. Active learning with sampling by uncertainty and density for data annotations. *IEEE Trans. on Audio, Speech & Language Processing*, 18(6):1323–1331, 2010.

[26] J. Zhu, H. Wang, T. Yao, and B. K. Tsou. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In D. Scott and H. Uszkoreit, editors, *22nd Int. Conf. on Computational Linguistics (COLING 2008)*, pages 1137–1144, 2008.

[27] I. Zliobaitė, A. Bifet, B. Pfahringer, and G. Holmes. Active learning with drifting streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99), 2013.

Table 3: Part A: Pairwise performance comparison after 20 and 40 labels over all datasets. It shows how often the classifier using the strategy from the row outperformed the same classifier using the strategies in the columns. For example, a Parzen-Window classifier using probabilistic active learning outperformed confidence-based uncertainty sampling significantly in 69.48%, random sampling in 73,22% and semi-random sampling in 71.98% of the cases (*continues on the next page*).

Parzen-Window Classifier

Posterior from Classifier, 20 labels					**Posterior from KFE, 20 labels**				
StrategyName	pal	conf	ran	semi	StrategyName	pal	conf	ran	semi
pal	0%	69.48%*	73.22%*	71.98%*	pal	0%	69.48%*	73.22%*	71.98%*
conf	30.52%-	0%	47.62%-	43.75%-	conf	30.52%-	0%	47.62%-	43.75%-
ran	26.78%-	52.38%*	0%	47.82%	ran	26.78%-	52.38%*	0%	47.82%
semi	28.02%-	56.25%*	52.18%	0%	semi	28.02%-	56.25%*	52.18%	0%
Posterior from Classifier, 40 labels					**Posterior from KFE, 40 labels**				
StrategyName	pal	conf	ran	semi	StrategyName	pal	conf	ran	semi
pal	0%	69.09%*	68.19%*	66.32%*	pal	0%	69.09%*	68.19%*	66.32%*
conf	30.91%-	0%	38.3%-	37.13%-	conf	30.91%-	0%	38.3%-	37.13%-
ran	31.81%-	61.7%*	0%	51.93%	ran	31.81%-	61.7%*	0%	51.93%
semi	33.68%-	62.87%*	48.07%	0%	semi	33.68%-	62.87%*	48.07%	0%

Hoeffding-Tree Classifier

Posterior from Classifier, 20 labels					**Posterior from KFE, 20 labels**				
StrategyName	pal	conf	ran	semi	StrategyName	pal	conf	ran	semi
pal	0%	64.26%*	64.92%*	62.7%*	pal	0%	51.66%	53.82%	49.1%
conf	35.74%-	0%	51.46%	48.92%	conf	48.34%	0%	51.68%	45.17%
ran	35.08%-	48.54%	0%	49.86%	ran	46.18%	48.32%	0%	44.48%
semi	37.3%-	51.08%	50.14%	0%	semi	50.9%	54.83%	55.52%	0%
Posterior from Classifier, 40 labels					**Posterior from KFE, 40 labels**				
StrategyName	pal	conf	ran	semi	StrategyName	pal	conf	ran	semi
pal	0%	63.19%*	69%*	66.55%*	pal	0%	53.06%	56.71%	50.16%
conf	36.81%-	0%	52.88%	51.24%	conf	46.94%	0%	52.12%	48.48%
ran	31%-	47.12%	0%	45.48%	ran	43.29%	47.88%	0%	45.42%-
semi	33.45%-	48.76%	54.52%	0%	semi	49.84%	51.52%	54.58%*	0%

Table 3: Part B: Pairwise performance comparison for Naive Bayes, K-Nearest Neighbour, and Logistic Regression Classifiers (*continuation from the previous page*).

Naive Bayes Classifier

Posterior from Classifier, 20 labels					**Posterior from KFE, 20 labels**				
StrategyName	pal	conf	ran	semi	StrategyName	pal	conf	ran	semi
pal	0%	58.97%*	64.66%*	60.7%*	pal	0%	51.25%	56.1%*	55.4%*
conf	41.03%-	0%	54.57%*	51.35%	conf	48.75%	0%	52.32%	55.41%*
ran	35.34%-	45.43%-	0%	47.26%	ran	43.9%-	47.68%	0%	51.26%
semi	39.3%-	48.65%	52.74%	0%	semi	44.6%-	44.59%-	48.74%	0%
Posterior from Classifier, 40 labels					**Posterior from KFE, 40 labels**				
StrategyName	pal	conf	ran	semi	StrategyName	pal	conf	ran	semi
pal	0%	63.17%*	74.5%*	67.24%*	pal	0%	54.55%	56.9%*	47.93%
conf	36.83%-	0%	60.92%*	53.65%	conf	45.45%	0%	56.51%	48.99%
ran	25.5%-	39.08%-	0%	41.18%-	ran	43.1%-	43.49%	0%	42.94%-
semi	32.76%-	46.35%	58.82%*	0%	semi	52.07%	51.01%	57.06%*	0%

K-Nearest Neighbour (K=3) Classifier

Posterior from Classifier, 20 labels					**Posterior from KFE, 20 labels**				
StrategyName	pal	conf	ran	semi	StrategyName	pal	conf	ran	semi
pal	0%	24.18%-	59.4%	30.34%-	pal	0%	27.53%-	50%	25%-
conf	75.82%*	0%	76.02%*	65.32%*	conf	72.47%*	0%	69.52%*	59.46%
ran	40.6%	23.98%-	0%	32.74%-	ran	50%	30.48%-	0%	30.81%-
semi	69.66%*	34.68%-	67.26%*	0%	semi	75%*	40.54%	69.19%*	0%
Posterior from Classifier, 40 labels					**Posterior from KFE, 40 labels**				
StrategyName	pal	conf	ran	semi	StrategyName	pal	conf	ran	semi
pal	0%	25%-	68.59%*	43.27%-	pal	0%	18.64%-	46.25%	27.55%-
conf	75%*	0%	80.57%*	80.75%*	conf	81.36%*	0%	78.24%*	72.89%*
ran	31.41%-	19.43%-	0%	35.05%-	ran	53.75%	21.76%-	0%	30.5%-
semi	56.73%*	19.25%-	64.95%*	0%	semi	72.45%*	27.11%-	69.5%*	0%

Logistic Regression Classifier

Posterior from Classifier, 20 labels					**Posterior from KFE, 20 labels**				
StrategyName	pal	conf	ran	semi	StrategyName	pal	conf	ran	semi
pal	0%	40.57%	37.5%-	45.31%	pal	0%	51.46%	46.05%	49.27%
conf	59.43%	0%	50.54%	55.75%*	conf	48.54%	0%	44.09%	47.43%
ran	62.5%*	49.46%	0%	56.52%*	ran	53.95%	55.91%	0%	51.63%
semi	54.69%	44.25%-	43.48%-	0%	semi	50.73%	52.57%	48.37%	0%
Posterior from Classifier, 40 labels					**Posterior from KFE, 40 labels**				
StrategyName	pal	conf	ran	semi	StrategyName	pal	conf	ran	semi
pal	0%	41.21%-	42.01%	43.53%	pal	0%	60.13%*	55.84%	58.13%*
conf	58.79%*	0%	52.66%	54.78%	conf	39.87%-	0%	44.03%	49.68%
ran	57.99%	47.34%	0%	48.78%	ran	44.16%	55.97%	0%	53.59%
semi	56.47%	45.22%	51.22%	0%	semi	41.88%-	50.32%	46.41%	0%

Chapter 7

Probabilistic Active Learning in Datastreams

The following chapter has been published as:

Probabilistic Active Learning in Datastreams
Daniel Kottke, Georg Krempl, Myra Spiliopoulou.

E. Fromont, T. De Bie, M. van Leeuwen. Advances in Intelligent Data Analysis XIV, Proc. of the 14th Int. Symposium (IDA 2015), Lecture Notes in Computer Science, vol. 9385, pp.145–157, Springer, 2015. ISSN: 978-3-319-24465-5.
DOI: `10.1007/978-3-319-24465-5_13`

The original publication is available at Springer International Publishing
`http://link.springer.com/chapter/10.1007/978-3-319-24465-5_13`

Probabilistic Active Learning in Datastreams

Daniel Kottke( ), Georg Krempl, and Myra Spiliopoulou

Knowledge Management and Discovery Lab, Otto-von-Guericke-University,
Universitätsplatz 2, 39106 Magdeburg, Germany
{daniel.kottke,georg.krempl,myra}@iti.cs.uni-magdeburg.de
http://kmd.cs.ovgu.de

Abstract. In recent years, stream-based active learning has become an intensively investigated research topic. In this work, we propose a new algorithm for stream-based active learning that decides immediately whether to acquire a label (selective sampling). To this purpose, we extend our pool-based Probabilistic Active Learning framework into a framework for streams. In particular, we complement the notion of usefulness within a topological space ("spatial usefulness") with the concept of "temporal usefulness". To actively select the instances, for which labels must be acquired, we introduce the Balanced Incremental Quantile Filter (BIQF), an algorithm that assesses the usefulness of instances in a sliding window, ensuring that the predefined budget restrictions will be met within a given tolerance window. We compare our approach to other active learning approaches for streams and show the competitiveness of our method.

1 Introduction

Facing continuously raising amounts of data but limited human supervis
capacities, active learning approaches that help in the efficient allocation
these capacities gain in relevance. The task in active learning is to decide
which instances to acquire labels from an oracle. An important active learn
scenario is stream-based active learning (also called selective sampling), wh
data arrives one-by-one on a stream and the algorithm has to decide imme
ately if the label is acquired [21]. Hence, there is no pool where instances
compared against each other by estimating their usefulness by their positior
feature space (spatial usefulness). Instead, the question becomes not only wh
but also when to query, i.e. the spatial aspect is complemented by a tempo
one [14]. Except for [26], the role of the temporal component was just fairly c
sidered in the algorithms as just simple thresholds have been tuned and appli
As it is not possible to tune a parameter without labeled data, we propos
method that ensures that a predefined budget will be definitely met withi
desired tolerance window. This also means that labeling resources like expe
or money remain constant (within the tolerance window) over time. Applicat
scenarios for those methods can be found in opinion mining of social comm
streams or annotation of sensor data like weather data or camera surveillar

Here, very fast classification systems are required because models might shift very fast (e.g. in twitter or stock exchange data). On the one hand, human experts only have limited (and constant) resources, and on the other hand, collecting a batch means to postpone the model updates.

We propose an active learning framework that explicitly distinguishes between the spatial and temporal component. This allows to study different combinations, and to separate their effects on the classification performance. Furthermore, we contribute an algorithm that chooses the most useful instances over time: the Balanced Incremental Quantile Filter (BIQF). BIQF uses a sliding window over the stream of spatial usefulness values as a representative of the most recent values to estimate an acquisition threshold. An adjustment of this threshold ensures that the aberration of the number of label acquisitions stays within a given tolerance window. We evaluate the performance of our new selective sampling algorithm that combines probabilistic active learning as the spatial and BIQF as the temporal component on multiple datastreams.

We start with a summary of the related work in Sect. 2. We propose our new stream active learning algorithm in Sect. 3 and present our new temporal active learning component BIQF in Sect. 3.2. After a detailed evaluation on several data sets in Sect. 4, we conclude this paper in Sect. 5.

2 Related Work

An active learning system aims to select the most promising instances for labeling, in order to build the best training basis for a given classifier [15]. Thus, in the beginning, no labeled information is available, but the target value (label) can be actively acquired from an oracle. This dynamic learning process develops the performance of the classifier directly over time [20].

In the pool-based setting, active learning has been researched for a long time. One of the simplest and most commonly used approaches, called *uncertainty sampling* [15], aims to request those instances that the classifier is most uncertain about, e.g. by measuring the confidence based on posterior estimates [20]. However, it is fairly easy to construct examples, where uncertainty sampling is not working [21, p.20], due to not doing any exploration [26]. This could also lead to even worse performance than a randomly sampling strategy [22]. Another approach is *Expected Error Reduction* (EER) [18], which aims to directly optimize a performance measure. It simulates each realization of a label for each unlabeled instance and trains a new classifier. On this classifier, it estimates the expected error on a validation set. In [3] it is observed that inaccuracies of the posterior estimates (esp. at the beginning) lead to problems for this algorithm, and the addition of a beta-prior is proposed. Other approaches, like Query by Committee (QbC) [7] minimize the variance between multiple classifiers. More recently, we proposed *Probabilistic Active Learning (PAL)* in [13] which includes the expectation value over the true posterior for a given instance to approximate the influence of an acquired label by the expected effect in its neighborhood. It measures the amount of already acquired labels in a neighborhood and balances

exploration and exploitation while directly optimizing a performance measu
Summarizing, EER and PAL optimize a performance measure, which ensure
good trade-off in exploration and exploitation [13,21]. While EER has high co
plexity, PAL and uncertainty sampling require only constant time per insta
[13] which enables their applicability in streams.

For active learning in datastreams, we have to separate those methods t
instantly decide whether to acquire a label or not, from those, that collect chu
or batches and apply pool-based methods. Chunk-based approaches use classi
ensembles [24,25] to determine the usefulness of instances or uncertainty-ba
measures [10,16,24]. A batch incremental stream active learning algorithm t
first clusters the chunk and ranks the instances based on an homogeneity a
certainty criterion was proposed in [11]. Most recently, we proposed a clusteri
based approach in [12] using the probabilistic description to select the cluster
choose instances from. The instantly deciding methods mostly are uncertain
based: the entropy uncertainty sampling with beta prior is used in [5], an ense
ble of radial clusters that evolve over time is proposed in [19], adaptively weigh
uncertainty and density scores are suggested in [4]. Zliobaite et al. [26] obser
that uncertainty sampling is not sufficient to react to drift and combined it w
random sampling. Except for [26], the latter group does not directly consi
budget restrictions as they use arbitrary tunable parameters or other impl
descriptions. In [26], an adaptive threshold method is proposed that ensures t
the budget is not exceeded. However, this threshold has issues as it is often do
inated by the flag that ensures that the budget is not exceeded. This leads
not finding the very best instances but to excluding only the very worse.

In Sect. 3.2, we propose an algorithm based on an incremental quantile
ter that handles the budget issue. In literature, quantile filters are prima
researched to address space limitations. A good review of existing methods a
their complexity is given in [23]. Quantiles have also been researched un
the condition of sliding windows [1], but with estimations for different ty
of windows and optimizations for approximations to save time and space. Su
approximations are not necessary in our setting with relatively short slid
windows.

3 Probabilistic Active Learning in Streams

We propose a probabilistic active learning framework for streams, building up
our original static framework PAL [13]. A core idea of the original, static PAL
to select instances for labeling by its probabilistic gain. Therefore, it consid
the observed posterior probabilities $\hat{p}$ (as determined by a classifier) but rat
model and exploit the *true* posterior probability p, which we express as a Be
distributed random variable, as we explain later on. The new stream algorit
uses this probabilistic gain as a measure for the instance's "spatial usefulnes
To identify what the spatial usefulness is currently worth in a temporal man
("temporal usefulness"), we propose the Balanced Incremental Quantile Fil
(BIQF). In the last subsection, we summarize all components and show
pseudocode.

3.1 Summary of the Probabilistic Gain Calculation

The probabilistic gain is a measure to determine the spatial usefulness of a labeling candidate x_i for active learning proposed in [13]. We use the term spatial usefulness to describe the usefulness for the instance's location in the feature space (characterized by its feature vector). Using the probabilistic gain, we extend the stream of instances (multi-dimensional feature vectors) by a stream of spatial usefulness values (single values). The core idea is to model the true posterior probability p as a Beta-distributed random variable, instead of using the observed posterior (determined by the classifier) as an estimate for the true posterior. This probability distribution uses the observed posterior probability ($\hat{p}$) and the among of neighbored labeled data (n) as parameters. For $n = 0$, the true posterior distribution is similar to an uniform distribution. The higher the n value, the higher the peak at the observed posterior $\hat{p}$. The final probabilistic gain calculates the expectation value over this true posterior probability p (assumed to be Beta-distributed) and each possible label realization y (assumed to be Bernoulli-distributed) [13].

$$\text{pgain}((n,\hat{p})) = \mathtt{E}_p\left[\mathtt{E}_y\left[\text{gain}_p((n,\hat{p}),y)\right]\right] \tag{1}$$

$$= \int_0^1 \text{Beta}_{n\hat{p}+1,n(1-\hat{p})+1}(p) \cdot \sum_{y\in\{0,1\}} \text{Ber}_p(y) \cdot \text{gain}_p((n,\hat{p}),y)\,\mathrm{d}p \tag{2}$$

The values for n and $\hat{p}$ can be determined by any generative classifier [17]. The gain in accuracy is directly derived from the true posterior (p), given the classification decision made from the observed posterior [13].

$$\text{gain}_p((n,\hat{p}),y) = \text{acc}_p(\hat{p}_{new}) - \text{acc}_p(\hat{p}) = \text{acc}_p\left(\frac{n\hat{p}+y}{n+1}\right) - \text{acc}_p(\hat{p}) \tag{3}$$

$$\text{acc}_p(\hat{p}) = \begin{cases} 1-p & \hat{p} < 0.5 \\ p & otherwise \end{cases} \tag{4}$$

In the static, pool-based setting, this probabilistic gain is weighted with the candidate's density to incorporate the information about the influence of the accuracy gain for the whole dataset. In a stream environment, any generative classifier gives us information about the label statistics of an incoming instance. As these label statistics are the only input parameters to calculate the probabilistic gain, it is easily applied. In a datastream and especially at the beginning, it is difficult to estimate the influence of a label for the whole dataset reliably. Hence, we here set the density weight to one.

3.2 Balanced Incremental Quantile Filter

Using the probabilistic gain, we extend the stream of feature vectors (from the instances) by a stream of scalars (spatial usefulness values). As higher values mean higher benefit for the classification task, the next step is to select highest values over time. There exist two related problem formulations in lit ature: Either to collect a batch and to choose the best within, or to determ immediately which instances are the best. The first strategy is easier but ne additional resources to store the data and delays learning to the end of e batch, thus we decided for the second one. The challenges for this stream scalars are: (C1) to decide immediately whether to acquire the label or not, (the values are distributed arbitrarily, (C3) acquiring a label changes the cla fication model, hence the distribution of spatial usefulness values might cha (as classification performance should improve over time, the spatial usefuln should decrease), and (C4) the classification model changes due to evolution the data.

In this section, we propose a new algorithm to determine the most use instances respecting a predefined budget (b) over time (temporal usefulne called *Balanced Incremental Quantile Filter (BIQF)*. It is based on an inc mental quantile filter to determine a threshold for the spatial usefulness va and a threshold-adjustment-component that ensures that the predefined bud is met. In streams, the relative budget $b \in [0, 1]$ is usually defined as the share labels that are acquired over time. Additionally, we try to distribute the bud constantly over time such that this enables to detect drift as we always expl the data, and we have constant and predictable annotation cost.

Incremental Quantile Filter. Given a budget $b \in [0, 1]$ and a stream of spa usefulness values $(u_1, u_2, \dots)$, the incremental quantile filter aims to determ the best values such that a share of b labels is acquired. Thus, it stores the l w (w denotes the window size) values of this input stream in a queue Q a representation of the most current value distribution. The decision to acq the label of an instance x_i with its spatial usefulness value u_i is based on rank (rank_{u_i}) in Q. If Eq. 5 is true, the label is acquired [1]:

$$\text{rank}_{u_i} \leq \lceil \text{len(Q)} \cdot b \rceil$$

The rank_{u_i} describes the position of the new value u_i in the list Q, e.g. highest value has a rank of 1, the second highest one has a rank of 2, and so

Figure 1 visualizes this process for a window size $w = 6$ and a budget $b =$ (Additionally to the chronologically sorted queue Q (not shown in the figu the method stores a value-sorted duplicate Q_s. In the first step, the algorit gets the first usefulness value $u_1 = 1$ from the stream. As Eq. 5 returns t ($\text{rank}_{u_1} = 1 \leq 1 = \lceil 1 \cdot 0.5 \rceil$), the label y_1 of the instance x_1 is acquired. N $u_2 = 6$ is added with $\text{rank}_{u_2} = 1$ and Eq. 5 is again true. Hence, the label y acquired, too. The same happens with value $u_3 = 8$. As value $u_4 = 5$ is add Eq. 5 results in $\text{rank}_{u_4} = 3 \not\leq 2 = \lceil 4 \cdot 0.5 \rceil$, which means that the correspond label is not acquired. Value $u_5 = 3$ and $u_6 = 4$ are not added, too. Adding va 9 would result in a list length of 7, which is higher than the window size $w =$ Thus, the oldest value, determined from the original queue Q, is removed, the formula is applied again.

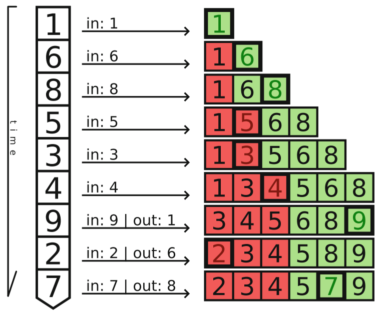


Fig. 1. Scheme of the Incremental Quantile Filter (IQF) for window size $w = 6$ and budget $b = 0.5$. Each usefulness value u_i (left stream) is inserted into the sorted list Q_s. If the incoming value is in the green area, the corresponding label is acquired.

Instead of calculating the rank, it is also possible to determine the usefulness threshold (θ) and check if the current value is higher or equal (Eq. 7). Referring to Fig. 1, the threshold is the most left green value. Depending on the queue's length ($|Q|$) and the predefined budget (b), it is calculated by Eq. 6 (using Eq. 5).

$$\text{thresIdx} = \lfloor |Q| \cdot (1 - b) \rfloor; \qquad \theta = Q_s[\text{thresIdx}] \tag{6}$$

$$u_i \geq \theta \tag{7}$$

The implementation of this algorithm was optimized using a B-tree [6] data structure to store and update the sorted list of usefulness values (Q_s). This reduced the computational complexity of sorting a whole list ($\mathcal{O}(w \log(w))$) into inserting (resp. deleting) an element ($\mathcal{O}(\log(w))$). This optimization needs the threshold index description. A complete pseudocode, a Python implementation and a detailed description of this optimization is given at our companion website[1].

Summarizing, this method decides immediately about a label acquisition (C1), works with arbitrary distributions (C2) but is only applicable when the distribution of the incoming usefulness values does not change over time (neither C3, nor C4). The simplest counterexample is a stream of monotonously decreasing values. In this case, no labels will be acquired because the rank is always at the very last position. With no new labels, we are not able to detect changes and the constant budget constraint is violated. This requires a solution which is described in the next subsection.

Balancing. We solve challenges C3 and C4 by a balancing approach that ensures that the predefined budget will be met within a given tolerance window. The tolerance window (w_{tol}) defines the maximal absolute difference between the number of actually acquired labels and the number of labels that should have been acquired so far. This target number of label acquisitions is the result of multiplying the predefined relative budget (b) and the number of processed stream instances. Counting the number of already acquired labels, we determine

[1] Companion website: http://kmd.cs.ovgu.de/res/pals.

the number of label acquisitions that should be spent to reach the predefi
budget by Eq. 8.

$$acq_{\text{left}} = \#\{\text{processed instances}\} \cdot b - \#\{\text{acquired labels}\}$$

Using this equation, the number of left labels for acquisition is real-valı
and possibly negative (in case that the number of labels is higher than desire
If this value is positive, the acquisition threshold (θ) should be decreased to m
the threshold less restrictive and vice versa. The amount of adaptation depe
on the predefined tolerance window (w_{tol}) and the range of the most rec
usefulness values (denoted as Δ). We use the difference between the first a
last element of the sorted queue to calculate the range ($\Delta = Q_s[|Q_s|-1]-Q_s[$
Hence, the new threshold is determined by Eq. 9.

$$\theta_{\text{bal}} = \theta - \Delta \cdot \frac{acq_{\text{left}}}{w_{\text{tol}}}$$

Next, we show that the next label will be acquired if the tolerance winc
is reached ($w_{\text{tol}} = acq_{\text{left}}$). To calculate the range, we determine the maxi
and minimal usefulness values stored in Q_s ($\Delta = u_{\max} - u_{\min}$). Therefore,
threshold is between these values ($\theta \in [u_{min}, u_{max}]$).

$$\theta_{\text{bal}} = \theta - \Delta \cdot \frac{acq_{\text{left}}}{w_{\text{tol}}} = \theta - (u_{\max} - u_{\min}) \cdot \frac{w_{\text{tol}}}{w_{\text{tol}}} \leq u_{\min} \qquad ($$

Hence, the new threshold θ_{bal} is below or equal all currently observed use
ness values. As the current usefulness value is already added to Q, we ensu
that the corresponding label will be acquired because for all $u \in Q : u \geq u_{\text{r}}$
Analogously, one can show that the next label will not be acquired for the op
site case $w_{\text{tol}} = -acq_{\text{left}}$.

3.3 Pseudocode

Algorithm 1 shows the complete stream active learning procedure using Pr
abilistic Active Learning (PAL) and the Balanced Incremental Quantile Fi
(BIQF). The user defined parameters are the budget (b), the IQF window s
(w), and the tolerance window size (w_{tol}). From lines 5–19, the instances
processed one by one. The probabilistic gain is calculated in lines 6–8, follov
by the processing of the Incremental Quantile Filter (IQF) (lines 9–12). In l
13, the threshold is adapted by the proposed balancing approach. If the use
ness value (u_i) is greater or equal this balanced threshold (θ_{bal}) in line 14,
label is acquired and this labeled instance is forwarded to the classifier (line
and the label acquisition counter (c_{acq}) is increased (line 16).

4 Experiments

The experimental evaluation section consists of two components. First, we sh
that the BIQF algorithm is able to select the best instances over time and seco
we evaluate our algorithm that combines Probabilistic Active Learning (PA
with BIQF against current baselines on seven datasets.

Algorithm 1. Probabilistic Active Learning in Streams

```
 1: $b \in [0,1]; w, w_{\text{tol}} \in \mathbb{N}$ {Predefined budget, IQF window size, balancing window size}
 2: $C \leftarrow \{\}$ {Generative Classifier}
 3: $Q \leftarrow \{\}$ {Queue for IQF algorithm}
 4: $i \leftarrow 1, c_{acq} \leftarrow 0$ {Instance counter, counter of acquired labels}
 5: while Stream delivers new instance $x_i$ do
 6:   {determine spatial usefulness value}
 7:   $\hat{p} \leftarrow P_C(+|x_i)$;   $n \leftarrow \text{KFE}_C(x_i)$
 8:   $u_i \leftarrow \text{pgain}(\hat{p}, n)$
 9:   {determine BIQF threshold}
10:   $Q$.push($u_i$);   if $|Q| > w$: $Q$.pop()
11:   $Q_s \leftarrow \text{sort}(Q)$
12:   $\theta \leftarrow Q_s[\lfloor |Q| \cdot (1-b) \rfloor]$
13:   $\theta_{bal} \leftarrow \theta - \frac{Q_s[|Q_s|-1] - Q_s[0]}{w_{\text{tol}}} \cdot (b \cdot (i - c_{acq}))$
14:   if $u_i \geq \theta_{bal}$ then
15:     $C$.retrain($x_i$, getLabel($x_i$))
16:     $c_{acq} \leftarrow c_{acq} + 1$
17:   end if
18:   $i \leftarrow i + 1$
19: end while
```

4.1 Performance of BIQF

To evaluate the Balanced Incremental Quantile Filter (BIQF), we test BIQF on static, synthetic usefulness streams. Therefore, we generate single-valued streams of different distributions (uniform, normal, gamma and a mixture of two normal distributions). The task of BIQF is to select the highest values as they appear without knowing the future values of that stream. As the distributions of these synthetic streams do not change over time, the optimal solution for a predefined budget b is determined by sorting the values of the whole stream and selecting the highest instances until the budget b is reached. To quantify the performance of BIQF, we calculated the mean of all selected values (resp. to b) and determined the reached percentage compared to the optimal solution. The window size is set to $w = 100$ and the tolerance window to $w_{\text{tol}} = 50$.

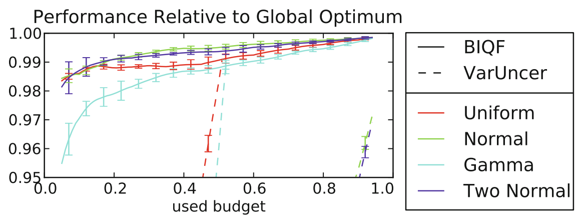


Fig. 2. Comparison of BIQF and the variable threshold method for different distributions.

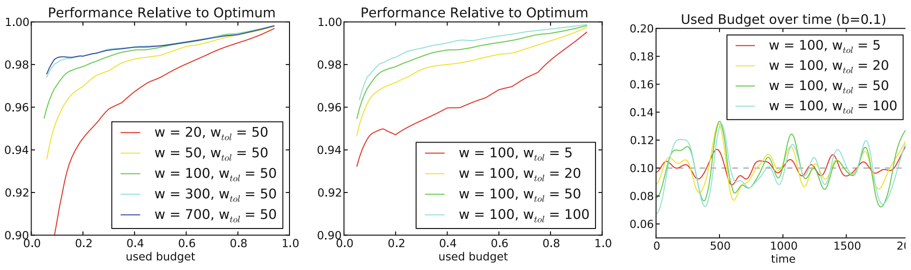


Fig. 3. Performances (left, middle) and visualization of really used budget (right) of BIQF algorithm for different parameters on a static Gamma-distributed value stre

In Fig. 2, we show the results in terms of the reached percentage of optimum as the mean and standard deviation for five streams underlying same distribution. Additionally, we executed the budget control mechanism fr the Variable Uncertainty method (VarUncer), proposed in [26]. The results sh that VarUncer does not reach a competing performance as its results mostly below 95 % compared to the optimal solution. In contrast, the BIQF is alw better than 95 % for every budget b which is completely enough for the demar of stream active learning.

For static data, increasing the window size w improves the results especia for low budgets (see Fig. 3 left) but also increases the execution time sligh ($\mathcal{O}(\log(w))$). Even more relevant, the average age of the queue rises beca more old values are considered. Hence, setting the window size to higher val reduces the currency of the model, which impairs the performance in non-sta data. Hence, the window size should be set to the highest acceptable delay recognizing a possibly appearing drift. In our case, the window size $w = 100$ v a good trade-off.

Additionally, Fig. 3 shows the performance of our algorithm for different t erance window sizes (w_{tol}) on the same value streams. Again, the performa increases for higher tolerance windows as the data is static. Nevertheless, a h variable budget distribution possibly does not recognize drift early enough a does not use the resources of an oracle efficiently as its workload should be c stant. The right plot shows the distribution of the really used budget over tir As expected, the variance of $w_{\text{tol}} = 5$ is the smallest. Hence, it met the bud restrictions the best in average. We suggest to set the tolerance window to half of the window size. If the resulting variance is too high for the oracle process the incoming data, one should reduce it to an acceptable level.

4.2 Stream Active Learning Performance

In this section, we compare our proposed algorithm that combines Probabi tic Active Learning (PAL) with the new Balanced Incremental Quantile Fil (BIQF) against other algorithms in stream active learning: a randomly sa pling method (Random), Split and Variable Uncertainty (VarUncer), propo in [26]. As we noticed some problems with the temporal selection strategy of [26], we further combined their ideas with our method: uncertainty sampling + BIQF (Uncer + BIQF) and Split + BIQF that selects one half of the instances randomly for exploration and the other half by uncertainty sampling for exploitation. The window sizes are the same as above. The generative classifier is a Parzen window classifier with pre-tuned bandwidths. To be able to react to drift, we add a sliding window with a size of 300 instances. All experiments run on a compute cluster running the (Neuro)Debian operating system [8].

The electricity dataset (27 k instances) [9] and the abalone dataset (4 k instances) [2] come from a real-world application. The checker dataset (motivated in [3]) consists of a 4 × 4 checkerboard (10 k instances) that switches all labels gradually after 50 % of the instances have been processed. The farcluster and movplane (10 k instances) are motivated in [26]. In farcluster an additional cluster appears far the decision boundary. In movplane, the decision boundary rotates slowly after 50 % of the instances have been processed. For the latter three datasets, 10 % of the labels are flipped to add noise. Bars and wave (10 k instances) are synthetic datasets without noise and a well-formed decision boundary. For each datastream, we created 100 random train/test-stream partitions. The results are averaged with respect to the actually used budget. To evaluate the algorithms, we provide learning curves in Fig. 4 for three datasets and an overview of mean accuracies for all datasets in Table 1.[2]

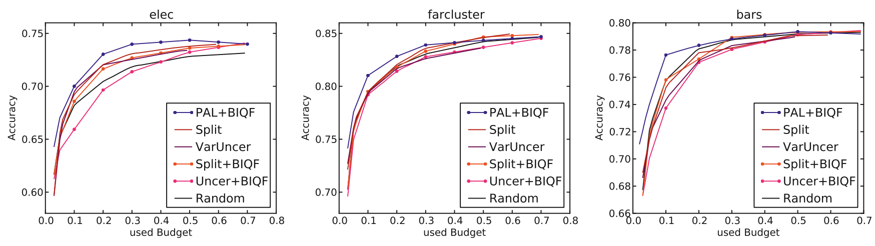


Fig. 4. Accuracy learning curves for datastreams elec, farcluster and bars.

For small budgets, PAL + BIQF is clearly dominating the other algorithms. Except for abalone, this approach always receives higher accuracy values given a budget of $b = 0.1$. For a budget of $b = 0.2$, PAL + BIQF is solely defeated on the wave dataset. This is expected because wave has a very simple and well defined decision boundary with small Bayesian error. Here, it is not necessary to explore the dataset (as PAL does), but to exploit the decision boundary (as uncertainty sampling methods do). Setting the budget to $b = 0.5$, the dominance of PAL + BIQF diminishes. On the one hand, this effect is not surprising in active learning because all sampling techniques should converge to the same level in the end. On the other hand, this might be caused by a problem of PAL with many labels: Especially for high n values, the probabilistic gain can get zero if one single

[2] More learning curves are available on http://kmd.cs.ovgu.de/res/pals.

Table 1. Mean accuracy for each algorithm on each dataset for the used budg
$0.1, 0.2, 0.5$ including standard deviation. Higher values are better and the best al
rithm is printed in bold.

b = 0.1	PAL+BIQF	Split	VarUncer	Split+BIQF	Uncer+BIQF	Random
abalone	0.721 ±0.02	0.716 ±0.02	0.721 ±0.02	**0.729** ±0.02	0.689 ±0.02	0.709 ±0.01
bars	**0.776** ±0.02	0.752 ±0.02	0.743 ±0.01	0.758 ±0.01	0.737 ±0.01	0.757 ±0.01
checker	**0.775** ±0.01	0.717 ±0.01	0.706 ±0.02	0.727 ±0.01	0.674 ±0.02	0.742 ±0.01
elec	**0.700** ±0.01	0.692 ±0.01	0.695 ±0.01	0.686 ±0.01	0.659 ±0.01	0.682 ±0.01
farcluster	**0.810** ±0.02	0.795 ±0.01	0.793 ±0.01	0.795 ±0.01	0.792 ±0.01	0.794 ±0.01
movplane	**0.775** ±0.01	0.759 ±0.02	0.756 ±0.01	0.762 ±0.01	0.743 ±0.02	0.760 ±0.01
wave	**0.920** ±0.01	0.912 ±0.01	0.911 ±0.01	0.908 ±0.01	0.904 ±0.01	0.900 ±0.01
b = 0.2	PAL+BIQF	Split	VarUncer	Split+BIQF	Uncer+BIQF	Random
abalone	**0.761** ±0.02	0.738 ±0.02	0.747 ±0.02	0.755 ±0.02	0.728 ±0.02	0.739 ±0.02
bars	**0.783** ±0.02	0.778 ±0.01	0.772 ±0.01	0.773 ±0.01	0.771 ±0.01	0.781 ±0.01
checker	**0.803** ±0.02	0.797 ±0.02	0.790 ±0.02	0.801 ±0.02	0.772 ±0.02	0.798 ±0.02
elec	**0.730** ±0.01	0.720 ±0.01	0.720 ±0.01	0.716 ±0.01	0.697 ±0.01	0.705 ±0.01
farcluster	**0.828** ±0.01	0.820 ±0.01	0.817 ±0.01	0.818 ±0.01	0.814 ±0.01	0.819 ±0.01
movplane	**0.791** ±0.01	0.785 ±0.01	0.786 ±0.01	0.788 ±0.01	0.779 ±0.01	0.778 ±0.01
wave	0.929 ±0.01	0.930 ±0.01	**0.935** ±0.01	0.931 ±0.01	0.934 ±0.01	0.923 ±0.01
b = 0.5	PAL+BIQF	Split	VarUncer	Split+BIQF	Uncer+BIQF	Random
abalone	0.768 ±0.02	0.766 ±0.02	0.761 ±0.02	**0.770** ±0.02	0.757 ±0.02	0.763 ±0.02
bars	**0.794** ±0.01	0.791 ±0.01	nan $\pm nan$	0.793 ±0.01	0.791 ±0.01	0.792 ±0.01
checker	0.839 ±0.01	0.840 ±0.01	nan $\pm nan$	**0.841** ±0.01	0.831 ±0.01	**0.841** ±0.01
elec	**0.744** ±0.01	0.738 ±0.01	nan $\pm nan$	0.736 ±0.01	0.732 ±0.01	0.728 ±0.01
farcluster	0.843 ±0.02	0.846 ±0.01	nan $\pm nan$	**0.847** ±0.01	0.837 ±0.01	0.842 ±0.01
movplane	0.804 ±0.01	0.803 ±0.01	nan $\pm nan$	**0.805** ±0.01	0.804 ±0.01	0.799 ±0.01
wave	0.941 ±0.01	0.941 ±0.01	nan $\pm nan$	0.943 ±0.01	**0.945** ±0.01	0.940 ±0.01

additional label would not change the classifier's decision. Nevertheless, resu
with small budgets are more important as we aim to save label acquisitions.

Very interesting is the fact that Uncer + BIQF could not improve the unc
tainty sampling method with the adaptive threshold (VarUncer) of [26]. Her
we also could confirm that excluding exploration (the adaptive threshold meth
solely excludes very certain samples and therefore does exploration) for unc
tainty sampling is malicious. Using BIQF for the idea of combining unc
tainty and random sampling shows a slight advantage of Split + BIQF agai
Split. Hence, the idea of random samples for uncertainty sampling is benefic
although its performance is below the one from PAL. We assume that the su
riority of PAL is caused by its direct integration of exploration and exploitati

5 Conclusion

In this paper, we proposed a new active learning algorithm for datastrea
that combines Probabilistic Active Learning to measure the spatial usefuln
of each instance, and the new Balanced Incremental Quantile Filter (BIQ
that selects the best over time. Through threshold adaptation, BIQF is a
to ensure that the predefined budget is met within a tolerance window. O
experimental evaluation on seven datasets and five competing algorithms show
the superiority of PAL + BIQF, especially for small budgets. We suggest t
the reasons are the implicit consideration of exploration and exploitation of
spatial usefulness measure using the probabilistic gain and the selection of

highest spatial values in its temporal context by BIQF. For future work, we will investigate if these effects are also true for the application scenarios mentioned in the introduction, and we will apply our framework in combination with different generative classifiers.

References

1. Arasu, A., Manku, G.S.: Approximate counts and quantiles over sliding windows. In: 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 286–296. ACM, New York (2004)
2. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository (2007)
3. Chapelle, O.: Active learning for parzen window classifier. In: International Workshop on Artificial Intelligence and Statistics, pp. 49–56 (2005)
4. Cheng, Y., Chen, Z., Liu, L., Wang, J., Agrawal, A., Choudhary, A.: Feedback-driven multiclass active learning for data streams. In: Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, CIKM 2013, San Francisco, California, USA, pp. 1311–1320. ACM, New York (2013). doi:10.1145/2505515.2505528
5. Chu, W., Zinkevich, M., Li, L., Thomas, A., Tseng, B.: Unbiased online active learning in data streams. In: 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, California, USA (2011)
6. Comer, D.: Ubiquitous b-tree. ACM Comput. Surv. **11**(2), 121–137 (1979)
7. Freund, Y., Seung, H.S., Shamir, E., Tishby, N.: Selective sampling using the query by committee algorithm. Mach. Learn. **28**(2–3), 133–168 (1997)
8. Halchenko, Y.O., Hanke, M.: Open is not enough. Let's take the next step: an integrated, community-driven computing platform for neuroscience. Front. Neuroinf. **6**, 22 (2012)
9. Harries, M.B., Sammut, C., Horn, K.: Extracting hidden context. Mach. Learn. **32**, 101–126 (1998)
10. Huang, S., Dong, Y.: An active learning system for mining time-changing data streams. Intell. Data Anal. **11**, 401–419 (2007)
11. Ienco, D., Bifet, A., Žliobaitė, I., Pfahringer, B.: Clustering based active learning for evolving data streams. In: Fürnkranz, J., Hüllermeier, E., Higuchi, T. (eds.) DS 2013. LNCS, vol. 8140, pp. 79–93. Springer, Heidelberg (2013)
12. Krempl, G., Ha, C.T., Spiliopoulou, M.: Clustering-based optimised probabilistic active learning (copal). In: 18th International Conference on Discovery Science (DS), Banff (2015)
13. Krempl, G., Kottke, D., Spiliopoulou, M.: Probabilistic active learning: towards combining versatility, optimality and efficiency. In: Džeroski, S., Panov, P., Kocev, D., Todorovski, L. (eds.) DS 2014. LNCS, vol. 8777, pp. 168–179. Springer, Heidelberg (2014)
14. Krempl, G., Zliobaite, I., Brzezinski, D., Hllermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M., Stefanowski, J.: Open challenges for data stream mining research. SIGKDD Explor. **16**(1), 1–10 (2014)
15. Lewis, D.D., Gale, W.A.: A sequential algorithm for training text classifiers. In: 17th Annual Intenational ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1–10 (1994)
16. Lindstrom, P., Delany, S.J., Namee, B.M.: Handling concept drift in a text data stream constrained by high labelling cost. In: FLAIRS Conference (2010)

17. Ng, A.Y., Jordan, M.I.: On discriminative vs. generative classifiers: a compariso logistic regression and naive bayes. In: Advances in Neural Information Process Systems 14, pp. 841–848. MIT Press (2002)
18. Roy, N., McCallum, A.: Toward optimal active learning through sampling esti tion of error reduction. In: International Conference on Machine Learning, IC 2001, pp. 441–448. Morgan Kaufmann Publishers Inc., San Francisco (2001)
19. Ryu, J.W., Kantardzic, M.M., Kim, M.-W., Ra Khil, A.: An efficient metho building an ensemble of classifiers in streaming data. In: Srinivasa, S., Bhatna V. (eds.) BDA 2012. LNCS, vol. 7678, pp. 122–133. Springer, Heidelberg (201
20. Settles, B.: Active Learning Literature Survey. University of Wisconsin, Madi (2010)
21. Settles, B.: Active Learning. Synthesis Lectures on Artificial Intelligence Machine Learning, vol. 6, no. 1, pp. 1–114 (2012)
22. Tomanek, K., Olsson, F.: A web survey on the use of active learning to sup annotation of text data. In: NAACL HLT Workshop on Active Learning for Nat Language Processing, Stroudsburg, PA, USA, pp. 45–48 (2009)
23. Wang, L., Luo, G., Yi, K., Cormode, G.: Quantiles over data streams: an exp mental study. In: Proceedings of the 2013 ACM SIGMOD International Confere on Management of Data, SIGMOD 2013, pp. 737–748. ACM, New York (2013
24. Wang, P., Zhang, P., Guo, L.: Mining multi-label data streams using ensem based active learning. In: SIAM Conference on Data Mining, pp. 1131–1140 (20
25. Zhu, X., Zhang, P., Lin, X., Shi, Y.: Active learning from stream data using opti weight classifier ensemble. IEEE Trans. Syst. Man Cybern. Part B Cybern. **40** 1607–1621 (2010)
26. Zliobaite, I., Bifet, A., Pfahringer, B., Holmes, G.: Active learning with drift streaming data. IEEE Trans. Neural Netw. Learn. Syst. **25**(1), 27–39 (2014)

Chapter 8

Clustering-Based Optimised Probabilistic Active Learning

The following chapter has been published as:

Clustering-Based Optimised Probabilistic Active Learning
Georg Krempl, Tuan Cuong Ha, and Myra Spiliopoulou.

Nathalie Japkowicz and Stan Matwin. Proceedings of the 18th Int. Conf. on Discovery Science, DS 2015, Springer Lecture Notes in Computer Science, vol. 9356, pages 101–115, 2015. ISBN 978-3-319-24281-1.
DOI: `10.1007/978-3-319-24282-8_10`

The original publication is available at Springer International Publishing
`http://link.springer.com/chapter/10.1007/978-3-319-24282-8_10`

Clustering-Based Optimised Probabilistic Acti Learning (COPAL)

Georg Krempl( ), Tuan Cuong Ha, and Myra Spiliopoulou

Knowledge Management and Discovery, Otto-von-Guericke University,
Universitätsplatz 2, 39106 Magdeburg, Germany
georg.krempl@iti.cs.uni-magdeburg.de
http://kmd.ovgu.de/res/pal

Abstract. Facing ever increasing volumes of data but limited human annotation capacities, active learning approaches that allocate these capacities to the labelling of the most valuable instances gain in importance. A particular challenge is the active learning of arbitrary, user-specified adaptive classifiers in evolving datastreams.We address this challenge by proposing a novel clustering-based optimised probabilistic active learning (COPAL) approach for evolving datastreams. It combines established clustering techniques, inspired by semi-supervised learning, which are used to capture the structure of the unlabelled data, with the recently introduced probabilistic active learning approach, which is used for the selection among clusters. The labels actively selected by COPAL are then available for training an arbitrary adaptive stream classifier. The performance of our algorithm is evaluated on several synthetic and real-world datasets. The results show that it achieves a better accuracy for the same budget than other recently proposed active learning approaches for such evolving datastreams.

Keywords: Probabilistic active learning · Selective sampling · Evolving datastreams · Nonstationary environments · Concept drift · Adaptive classification · Clustering

1 Introduction

In the face of ever increasing volumes of data [6] that contrast limited hun
annotation and supervision capacities, approaches for the efficient allocatio
these capacities are of increasing interest. Active machine learning approac
address this by providing strategies for determining and selecting the most va
able information. In classification tasks, this corresponds to selecting the insta
from a set of candidates, whose label is expected to improve a classifier's p
formance the most [23]. Active learning is considered a particularly challeng
problem [15] in evolving datastreams, where instances arrive continuously o
time and distributions may change and require adaptation.

We address this challenge by proposing a novel active learning approach for evolving datastreams. Inspired from semi-supervised learning, our clustering-based optimised probabilistic active learning (COPAL) approach uses established clustering techniques to capture the structure of the unlabelled data. We combine this with the recently introduced optimised probabilistic active learning [13] approach that we use for selecting the cluster, and with a diversity-maximising criterion for selecting the instance within that cluster. This actively selected label is then used for the training of a user-specified adaptive stream classifier.

We contribute two such clustering-based probabilistic active learning approaches. The first is an incremental clustering variant that maintains and adapts its clustering model over time. The second is an amnesic clustering variant that iteratively learns new clustering models from scratch on each chunk and discards it after updating the classifier. We evaluate both variants by comparing them against each other and several competitors on six different datastreams, among them four real-world datasets. The results of the experimental evaluation indicate an overall superior performance of our incremental clustering-based probabilistic active learning approach.

We first review the related work in Sect. 2, before presenting our clustering-based optimised probabilistic active learning approach in its incremental and amnesic clustering variant in Sect. 3. These and other recently proposed AL-approaches are evaluated in Sect. 4, followed by a conclusion in Sect. 5.

2 Background and Related Work

Active machine learning [23] aims to optimise the selection of labels when they are costly to obtain. The scenario addressed in this paper is stream-based selective sampling [22], where instances arrive continuously and an active classifier has to decide for each instance upon its arrival once-and-forever whether to acquire its label. Compared to the rich literature on selective sampling in streams in general, active learning in *nonstationary*, *evolving* datastreams has received far less attention, although it is considered as a challenging, relevant task [7,15].

One line of research [16,17,21,24,25] has investigated ensemble-based active learning approaches for evolving datastreams. The approach in [24] processes instances in chunks, such that in each chunk a certain initial percentage of instances are labelled. These initial labels in the chunk are used to learn a new base classifier, which is added to the ensemble. The disagreement within the updated ensemble is then used to select iteratively a given number of instances within the remaining unlabelled ones in the chunk. This is extended in [25] by a criterion that determines when to stop the active learning process on a chunk, and by an adaptive weighting of the base classifiers in the ensemble. The ActMiner-algorithm proposed in [17] processes data also in chunks, but clusters the data into spherical micro-clusters, which represent base classifiers of an ensemble. New instances that are not covered by any micro-cluster (so-called F-outliers) or instances with disagreeing micro-clusters are labelled and saved in

a buffer for later inspection. If the instances in the buffer form a new clus
this cluster is added to the ensemble. The approach suggested in [21] exte
this in two directions. First, by processing the stream instance-wise, and seco
by using decision trees as base classifiers. Like [17], it also summarises the
tribution of each base classifier's training data by a spherical cluster centred
its mean. This clustering is then used for the weighting of base classifiers and
the identification of suspicious instances outside all clusters. The labels of th
suspicious instances are then requested to train a new base classifier. In contr
to these works, a different combination of query-by-committee and cluster
for instance-wise active learning is proposed in [16]: upon the arrival of a
instance, a new ensemble of Gaussian mixture models is created by sampl
from a normal inverse Wishart distribution, such that each Gaussian compon
corresponds to one class. The GMMs in the ensemble converge as the numbe
acquired labels increases, reducing the areas of disagreement between the GM
and balancing exploration and exploitation.

More recently, other authors [1, 10, 11, 20] have investigated the idea of co
bining clustering and stream-based active learning further. They extend
older clustering-based active learning approach in [19], which addressed a po
based setting but already used the clustering information to select the m
representative instances for labelling, thereby reducing the required numbe
labelled instances in each cluster. In contrast, the newer StreamAR approach
[1] is actually a semi-supervised stream classification approach that uses a mic
clustering ensemble to assign labels and uses active learning solely to resolve
due to votes from opposing classes in the ensemble. Thus, while reducing
requested number of labels, it provides no means for controlling its budget.

In [20], a clustering-based approach is proposed for evolving datastreams,
so-called Concurrent Semi-supervised Learning of Data Streams (CSL-Strea
It maintains a clustering and assumes the posterior distribution within a clus
to be homogeneous, i.e. statistically independent of the feature position gi
the cluster membership. Its active learning step differentiates between clust
with and without any labelled instances. In the latter case, the algorithm se
to obtain the label of the centremost instance. In the former case, the algorit
checks for a skewed label distribution: if all labels are on one side of the clus
an additional label at the opposite side of the cluster is requested. Otherw
if the class of the labels differs between the sides of the cluster, the cluste
split. If the distribution of labels is homogeneous, the cluster is kept as it
Concept drift is addressed by using a fading model such that instances age o
time. Unfortunately, the author's informed us that an implementation for t
algorithm is no longer available.

While the clustering-based approaches above integrate clustering and cla
fication, the aim of Clustering Based Active Learning for Evolving Data Strea
(ACLStream) proposed in [10] is to be usable with any stream classifier techn
ogy. On each arriving chunk of instances a new clustering is performed and
most informative instances from each cluster are selected for labelling. For t
selection, the approach distinguishes between a macro and a micro step. T
macro step is used to rank clusters according to their homogeneity in terms

their predicted class distribution. This distribution is estimated by the model learnt from all the labelled instances from previous chunks of instances. The later micro step determines the most useful instance within a given cluster. Thus, it ranks instances by combining geometrical information inside their cluster and the maximum a posteriori classification probability. After selected instances are labelled, the clustering information is discarded.

The most recent active learning approach for evolving datastreams is DBAL-Stream [11]. This instance-wise approach combines density-weighting with uncertainty sampling. The density-weighting is used in a preselection step, such that solely instances within dense areas are considered as labelling candidates. Among those preselected candidates from dense regions, a margin-based uncertainty sampling approach is used to select one-by-one instances for labelling. This is done by comparing an instance's margin against a threshold, which is adjusted depending on the consumed and available budget and combined with random noise to improve exploration. This approach was reported to perform best in the evaluation by [11], making it an interesting candidate for our experimental evaluation.

The active learning algorithms for evolving datastreams discussed above are all based either on the disagreement in a query-by-committee approach, or the uncertainty in an uncertainty sampling approach, with known shortcomings [14,23]. Recently, the probabilistic active learning (PAL) approach has been proposed to overcome these shortcomings in the pool-based setting [14]. PAL summarises the labelled information in an instance's neighbourhood and evaluates the impact of acquiring a label therein in terms of the expected performance change. Expectation is not only done over the possible realisations of a candidate's label as in error reduction, but also over the true posterior in the candidate's neighbourhood. In [12], combining this approach with budgeting for datastreams is investigated. In [13], a fast closed-form solution is proposed that combines the qualities of uncertainty sampling and error reduction, namely being fast and optimising directly a performance measure. Thus, it seems worth exploring this fast approach in combination with clustering in a stream-based setting.

3 Clustering-Based Probabilistic Active Learning

Our approach combines ideas from clustering-based semi-supervised learning and probabilistic active learning. More precisely, we use the clustering model to define the neighbourhoods for the label statistics in a probabilistic active learning approach [14]. Our **C**lustering-based **O**ptimised **P**robabilistic **A**ctive **L**earning (COPAL) algorithm consists of four steps, which are pre-clustering, macro and micro selection, and updating. To complete the big picture, we briefly summarise them before providing their details in the Subsects. 3.1 to 3.3. Finally, in Subsect. 3.4, we present two variants of COPAL with their pseudocode.

The *pre-clustering* step starts with a pool of unlabelled instances. In this step, all instances are divided into some initial clustering. While more elaborate clustering algorithms can be used, we opted for conventional K-means because it builds spherical clusters and because our focus is on assessing the neighbo of a data point and not on achieving a good partitioning of the data space.

The task of the *macro* step is to determine the most important cluster select instances from for labelling. Therefore, we need an approach to measure value of additional labels for a cluster. For that purpose, we adapt the OPAL-g formula from probabilistic active learning [13] to our clustering model. The *mi* step then selects an instance from the previously chosen cluster for labelling, s that the *diversity* among the labelled instances within a cluster is maximis After a new instance is labelled, the class distribution in the selected cluster n have changed. Thus, an *updating* step is used to adjust the clustering model. this last step, we examine the homogeneity of the posterior distribution wit the selected cluster. We split the cluster in case it has become inhomogeneo

3.1 Macro Step: Determining the Most Valuable Cluster

The OPAL-gain formula in [13] is designed to compute the expected average n classification loss reduction from obtaining m additional labels within a can date's neighbourhood. It relies on label statistics ℓs, which summarise the num of already obtained labels n within its neighbourhood and the share of positi therein $\hat{p}$. For COPAL, the cluster of an instance defines its neighbourhood, t n equals the number of labels acquired in that cluster, and $\hat{p}$ equals the share positives therein. Because all instances in the cluster share the same neighbo hood, their probabilistic gains are equal. Following [13], the resulting expec average misclassification loss reduction in the cluster (G_{OPAL}) is calculated

$$G_{OPAL}(n,\hat{p},\tau,m) \ = \frac{(n+1)}{m} . \binom{n}{n.\hat{p}} . \Big(I_{ML}(n,\hat{p},\tau,0,0) \Big) - \sum_{k=0}^{m} I_{ML}(n,\hat{p},\tau,m$$

Here, $\tau \in [0,1]$ is given by the application and corresponds to the relat cost of each false positive classification, normalised such that the costs of a f positive and a false negative sum to one. For example, when the objectiv maximising the classifier's accuracy, $\tau = 0.5$ and G_{OPAL} is proportional to gain in accuracy. Likewise, $m > 0$ is the application-given remaining budget the currently processed chunk. Thus, τ and m are the same for each clus Equation 1 uses the function I_{ML}, introduced in [13], to compute a value tha proportional to the expected misclassification loss within a cluster, given tha additional positives among the m additional labels are sampled:

$$I_{ML}(n,\hat{p},\tau,m,k) = \binom{m}{k} . \begin{cases} (1-\tau) & . \frac{\Gamma(1-k+m+n-n\hat{p})\Gamma(2+k+n\hat{p})}{\Gamma(3+m+n)} & \frac{n\hat{p}+k}{m+n} < \tau \\ (\tau-\tau^2) & . \frac{\Gamma(1-k+m+n-n\hat{p})\Gamma(1+k+n\hat{p})}{\Gamma(2+m+n)} & \frac{n\hat{p}+k}{m+n} = \tau \\ \tau & . \frac{\Gamma(2-k+m+n-n\hat{p})\Gamma(1+k+n\hat{p})}{\Gamma(3+m+n)} & \frac{n\hat{p}+k}{m+n} > \tau \end{cases}$$

In a clustering model, the expected misclassification loss reduction for a cl ter depends not only on the probabilistic gain, but also on the size of the clus Since a larger cluster affects more future classifications than a smaller one, larger is favoured if their probabilistic gains are (nearly) equal. Therefore, for estimating the importance of a cluster $Cluster_i$ with $N_{Cluster_i}$ (labelled and unlabelled) instances therein, we propose to compute a cluster-size-weighted probabilistic gain $G_{Cluster_i}$ by the following formula:

$$G_{Cluster_i} = G_{OPAL_i} \cdot \frac{N_{Cluster_i}}{\sum_{j=1}^{N} N_{Cluster_j}} \tag{3}$$

The algorithm in Algorithm 1 describes this macro step in detail. For each cluster c, we calculate the values of n and $\hat{p}$ for this cluster, before computing its weighted gain by using the formulas 1, 2 and 3 (line 3–5). Finally, we select the cluster with the largest weighted gain and return it as the output (line 7–8).

Algorithm 1. Select the Best Cluster for Budget m and Cost-Ratio τ

```
1: procedure SELECTCLUSTER($C$, $m$, $\tau$)                    ▷ C: Pool of clusters
2:     for $c \in C$ do
3:         $(n, \hat{p}) \leftarrow labelstatistic\,(c)$
4:         $G_{OPAL} \leftarrow getOPALGain\,(n, \hat{p}, m, \tau)$          ▷ Use Eq. 1
5:         $G_c \leftarrow getWeightedGain\,(G_{OPAL}, c)$               ▷ Use Eq. 3
6:     end for
7:     $c^* \leftarrow \arg\max_{c \in C}(G_c)$
8:     return $c^*$
9: end procedure
```

3.2 Micro Step: Selecting an Instance Within the Cluster

The micro step selects an instance within the cluster c^* that was previously chosen in the macro step. We aim to maximise diversity among the label that are requested within that cluster. Thus, by using Eq. 4, we select the instance for labelling, whose nearest labelled neighbour is the furthest away. Here, $c^*_{\mathcal{U}}$ is the subset of the current chunk's unlabelled instances within c^*, $c^*_{\mathcal{L}}$ is the subset of labelled ones, and $|\cdot|_2$ is the l^2-Norm:

$$x^* \leftarrow \arg\max_{x_i \in c^*_{\mathcal{U}}} \left(\min_{x_l \in c^*_{\mathcal{L}}} |x_i - x_l|_2 \right). \tag{4}$$

3.3 Updating Step: Adjusting the Clustering Model

Upon having obtained the label for the instance selected in the micro step, the cluster it belongs to is updated. In this step, two alternating hypotheses are considered: The first hypothesis H_1 is that the cluster is homogeneous with respect to its posterior distribution and, as a consequence, should not be split further. The second, alternative hypothesis H_2 is that the cluster is inhomogeneous and the instances therein originate from two spatially separable subpopulations with different posteriors. In the second case, splitting the cluster should improve

homogeneity. Therefore, we calculate the current error rate E_1 (under H_1) a
compare it to the error rate after splitting[1](E_2 under H_2). However, due
the limited number of remaining labels in each subcluster, the simple appro
to use directly the training error is prone to overfitting. Instead, we perform
leave-one-out cross-validation on the labelled instances and calculate E_2 as
average error rate over each fold. In the case of ties due to an equal number
positives and negatives, we use an error rate of 50 %. If the error rate decrea
by splitting (i.e. $E_1 > E_2$), we retrain the classifier on all labels in the clus
and partition the instances based on their assigned labels into two new clust

3.4 Variants of COPAL

We propose two variants of COPAL for combining the modules above. The f
uses a sliding window and an incremental clustering, the second an amnesic cl
tering that forgets the clustering model after processing its chunk. The lat
is inspired by the discussions of the authors of [10], who observed good perf
mance with their amnesic clustering approach. However, COPAL worked bet
with an incremental rather than an amnesic clustering in our experiments.

Incremental Clustering Variant (COPAL-I). The pseudocode of the p
posed incremental variant *COPAL-I* is provided in Algorithm 2. It uses the f
steps above in combination with an incremental clustering model to activ
select instances for labelling, which are then passed to an arbitrary incremen
classifier. Using a sliding window approach, a fixed number of the most rec
instances is kept in a *Cache*. These instances are used to maintain the clus
model. Consequently, the pre-clustering step is only applied for the first chu
(line 6), which also initialises the *Cache* (line 7). For subsequent chunks, the n
instances are matched to the closest cluster of the current model (lines 9–1
and appended to the *Cache*, eventually replacing the oldest ones therein (li
13–17). Afterwards, the macro, micro and update steps are applied iterativ
to select the most valuable instances for labelling (lines 19–25). The update
each iteration comprises updating the dedicated classifier by the new label (l
23), and updating the clustering model (line 24).

Amnesic Clustering Variant (COPAL-A). This variant of COPAL uses
amnesic clustering model, as outlined in Algorithm 3. As above, after a chu
has been processed, the labels selected by *COPAL-A* therein are used to tr
an incremental classifier. The clustering model, however, is forgotten. Therefo
the pre-clustering step is repeated on each chunk (line 4). Afterwards, the ma
and micro steps are applied to get the most valuable instance for labelling (li
6–7). Its label is used to update the incremental classifier (line 9), and the proc
is repeated until the budget for this chunk is exhausted. Then, if necessary,
clustering is updated by splitting the cluster of the new instance (line 10).

[1] For speed, we used logistic regression for determining the preliminary splits.

Algorithm 2. Incremental Clustering Variant COPAL-I

```
Require: S: Stream of Instances
Require: b: Budget (per Chunk)
Require: w: Window Size (of Cache)
 1: $cl \leftarrow initClassifier$
 2: $Cache \leftarrow null$                                  ▷ Initialise cache of recent instances
 3: while $hasMoreInstances(S)$ do
 4:     $S_t \leftarrow nextChunk(S)$
 5:     if $Cache == null$ then
 6:         $C \leftarrow preClustering\,(S_t)$              ▷ $C$: Pool of clusters with centroids $\bar{c}$
 7:         $Cache \leftarrow S_t$
 8:     else                                             ▷ Cluster pool and cache maintainance
 9:         for $x_i \in S_t$ do
10:             $c^* \leftarrow \arg\min_{\bar{c} \in C} |x_i - \bar{c}|_2$    ▷ $l^2$-Norm(instance $x_i$,centroid $\bar{c}$ of C)
11:             addInstance($c^*, x_i$)                  ▷ Add $x_i$ to cluster $c^*$
12:         end for
13:         $Cache.append(S_t)$                          ▷ Add new instances to cache
14:         while $Cache.size() > w$ do
15:             $x \leftarrow Cache.removeOldest()$      ▷ Remove oldest instance from cache
16:             removeInstance($C, x$)                   ▷ Remove oldest instance from clustering
17:         end while
18:     end if
19:     for $k \in \{1, 2, \cdots, b\}$ do
20:         $c^* \leftarrow selectCluster\,(C, b+1-k, \tau)$     ▷ Marco step, Alg. 1
21:         $x_i \leftarrow selectInstance\,(c^*)$               ▷ Micro step, Eq. 4
22:         $y_i \leftarrow askLabel(x_i)$
23:         $trainClassifier(cl, x_i, y_i)$                     ▷ Classifier update
24:         $updateCluster(c^*, x_i, y_i)$                      ▷ Cluster update
25:     end for
26: end while
```

Algorithm 3. Amnesic Clustering Variant COPAL-A

```
Require: S: Stream of instances
Require: b: Budget (per chunk)
Require: $\tau$: false positive misclassification cost
 1: $cl \leftarrow initClassifier$
 2: while $hasMoreInstances(S)$ do
 3:     $S_t \leftarrow nextChunk(S)$
 4:     $C \leftarrow preClustering\,(S_t)$                  ▷ $C$: Pool of clusters with centroids $\bar{c}$
 5:     for $k \in \{1, 2, \cdots, b\}$ do
 6:         $c^* \leftarrow selectCluster\,(C, b+1-k, \tau)$     ▷ Marco step, Alg. 1
 7:         $x_i \leftarrow selectInstance\,(c^*)$               ▷ Micro step, Eq. 4
 8:         $y_i \leftarrow askLabel(x_i)$
 9:         $trainClassifier(cl, x_i, y_i)$                     ▷ Classifier update
10:         $updateCluster(c^*, x_i, y_i)$                      ▷ Cluster update
11:     end for
12: end while
```

4 Experimental Evaluation

In the following Subsect. 4.1, we describe the setting for our experimental eva
ation, including the datasets and the compared active learning approaches. T
is followed by a presentation and discussion of the results in Subsect. 4.2.

4.1 Experimental Setup

The objective in active learning is the selection of the most beneficial lal
for the training of the classifier, such that for a given budget the classificat
performance is maximised. How well a strategy handles this trade-off betw
classification performance and consumed budget is usually evaluated in learn
curves, which plot the performance in dependence of the budget. For strea
based scenarios, this requires to aggregate the performance over time, as d
for example in [10,11]. However, for evolving datastreams the variance in the p
formance over time is also an important aspect, as it indicates whether and h
quickly an algorithm adapts to drift. For passive stream classifiers, the stand
approach is prequential evaluation [5], which uses newly arrived instances f
for testing the current classifier, before using them for updating the classifie

We consider both aspects in our experimental evaluation: following the p
quential evaluation paradigm, we evaluate the classifier first on newly arriv
instances, before we consider them as candidates for the active learning a
classifier updating step. For studying the active learning strategies' effect on
adaptation of the classifier to drift, we provide curves that show the accuracy
the algorithms over time. For evaluating how well the strategies perform in
trade-off between accuracy and budget size, we provide learning curves that p
the aggregated accuracy over time for different budget shares. This is the m
informative common evaluation method, as there is no consensus on an appro
for statistically testing such active learning results in evolving datastreams y

Using this setup, we compare the incremental variant ***COPAL-I*** and
amnesic variant ***COPAL-A*** of our approach against several other active lea
ing strategies: first, we use complete labelling (denoted as ***Complete***), wh
requests all labels and serves as a proxy for the upper bound of the achieva
performance, thereby indicating the complexity of the datastream. Second,
use random selection (denoted as ***Random***) as a baseline, where instances
chosen randomly with equal selection probabilities. Third, we compare our a
roach to ***ACLStream***, the most recently proposed [10] *clustering-based* act
learning strategy for evolving datastreams. Finally, we compare against ***DBA***
Stream, to our knowledge the *most recently proposed* active learning strat
for *evolving* datastreams. This strategy was reported in [11] to outperform s
eral other active learning strategies for evolving datastreams, including the o
proposed in [26]. Other active learning approaches for evolving datastreams d
cussed in Sect. 2 integrate a specific classifier into their algorithm. Since this c
flicts with a differentiated evaluation between the impact of the AL-compon
alone and the used classifier technology, they were not included into the ev
uation. Furthermore, to ensure a fair evaluation, all algorithms are run wit

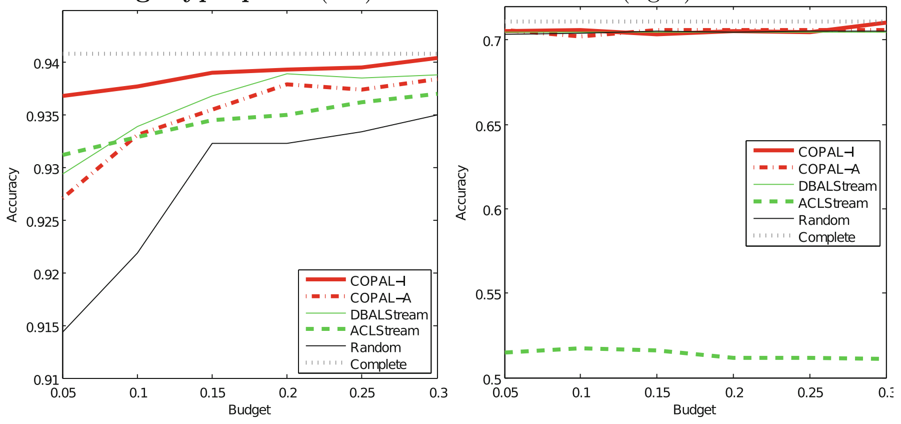
the MOA framework in Java, using the original implementations and recommended parameter settings of their authors. For the non-deterministic strategies ACLStream and Random, we average the performance over 5 runs. For better comparison, we use for COPAL the same k-means pre-clustering technique with $k = 5$ as in ACLStream, and the same type of classifier (adaptive Naive Bayes with drift detection, see [4]) that was proposed for DBALStream in the evaluation of all approaches. The chunk- and sliding window size is set to 100 instances for all approaches. We measure accuracy gain in COPAL by setting $\tau = 0.5$.

The experimental evaluation is done on six datastreams, including four real-world ones. The first synthetic datastream is based on the *Moving Hyperplane* generator proposed in [9]. The concept therein is based on a hyperplane, which rotates over time to generate drift. The implementation of the HyperplaneGenerator class in MOA was used with default settings to generate the data. The second synthetic datastream, random radial basis function (*Random RBF*), is based on the randomRBFGeneratorDrift class in MOA [3]. It uses a mixture of Gaussians with a fixed number of components, such that each component generates instances from a single class. Drift is induced by moving the centroids of the components in the featurespace. Except for the number of components, which was set to 20, the default parameter settings were used. The first real-world datastream is the *Airline* dataset by the US Bureau of Transportation Statistics, Research and Innovative Technology Administration (RITA), with the task being to predict whether a flight will be delayed based on the information of its scheduled departure. The second one is the *Bank Marketing* dataset by [18], where the task is to predict whether the client will subscribe to a term deposit subsequently to a direct marketing campaign. The third one is the *Electricity* dataset by [8], with the task to predict an increase or decline of the electricity prices in New South Wales (Australia). The fourth datastream is the *EEG Eye State* dataset from [2], with the task to repeatedly predict over the experiment's duration of 117 s whether a proband's eyes are opened or closed.

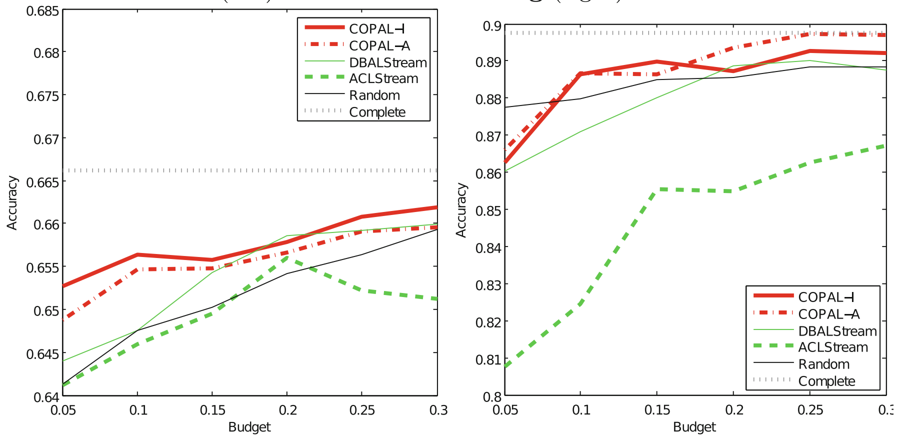
4.2 Results and Discussion

We first discuss the results of the evaluation of the active learning strategies' performances under different budgets. These are shown in the learning curves in Fig. 1, which plot the accuracy (aggregated over time) for different budget shares. Overall, *COPAL-I* performs best for the most datastreams and budget sizes. It is always better than *ACLStream*, better than *Random* except for a budget of 0.05 % on Bank Marketing, and better than *DBALStream* except for a single budget share on the Airline, the Bank Marketing, the EEG Eye State and the Electricity datastreams. Compared to its amnesic counterpart *COPAL-A*, it performs better on Airline and Moving Hyperplane, and comparably on the remaining datastreams, except for its worse performance on the EEG Eye State. Compared to their competitors, *COPAL-A* performs also well, being better than *Random* on all datastream-budget combinations except for one particular budget share on Electricity and Bank Marketing, and performing always better than *ACLStream* except for the budget share of 0.05 on the

Moving Hyperplane (left) and **Random RBF** (right) Datastreams:



Airline (left) and **Bank Marketing** (right) Datastreams:



EEG (left) and **Electricity** (right) Datastreams:

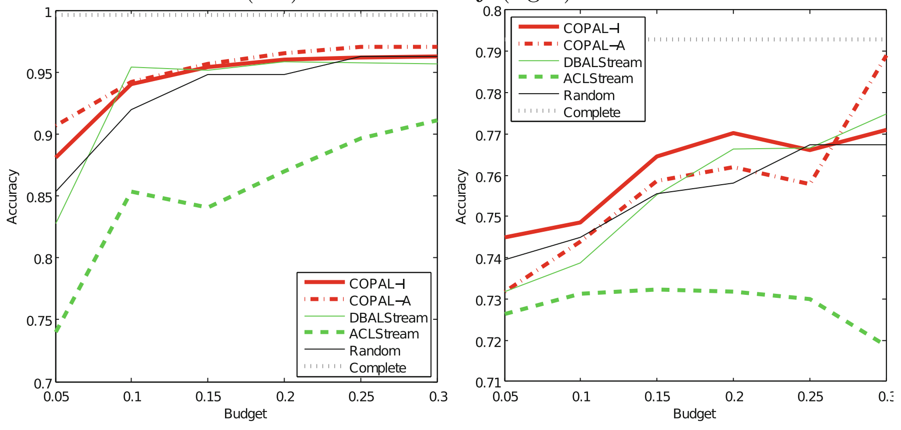
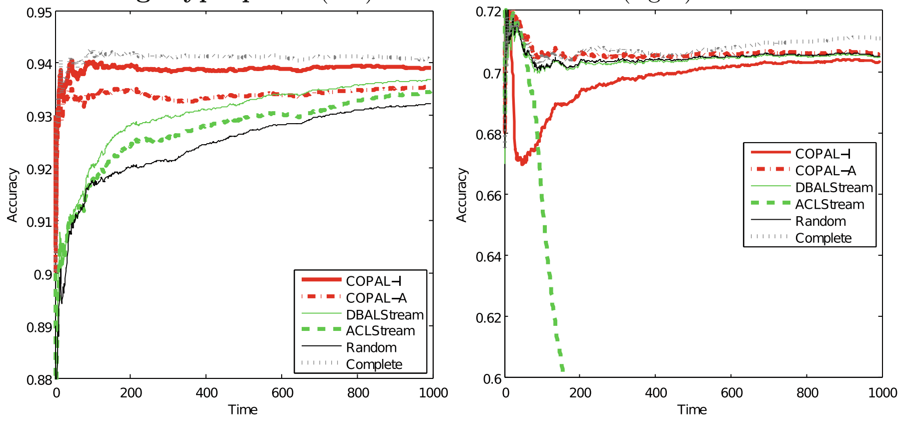
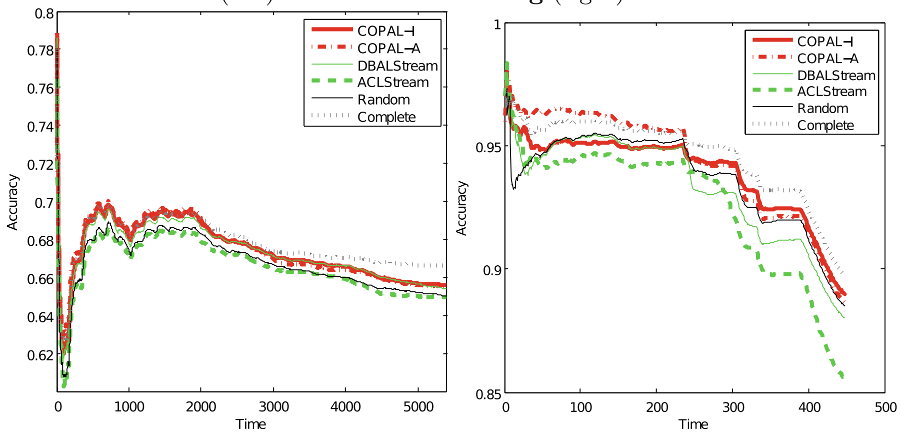


Fig. 1. Learning curves in budget share against accuracy for different datastrea
Complete (black dotted line) corresponds to an upper bound of the performance w
all instances being labelled. Early convergence to high values is favourable.

Moving Hyperplane (left) and **Random RBF** (right) Datastreams:



Airline (left) and **Bank Marketing** (right) Datastreams:



EEG (left) and **Electricity** (right) Datastreams:

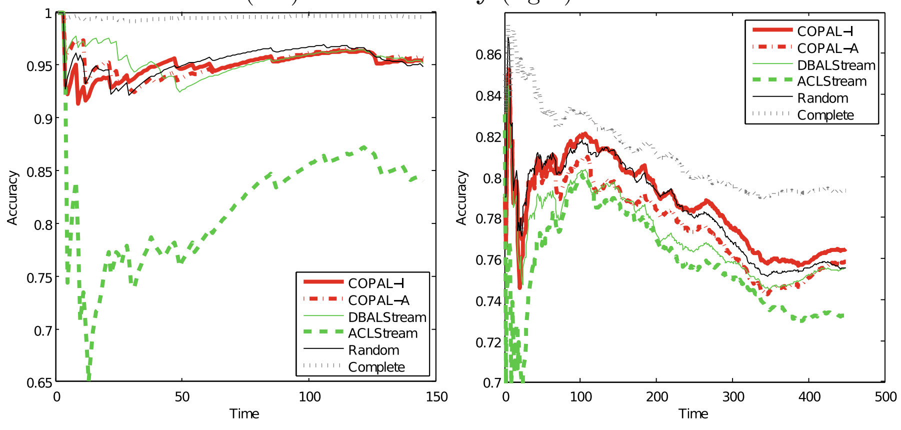


Fig. 2. Performance (in accuracy) over time (in steps of 100 instances). On all datastreams prequential evaluation and a budget share of 15 % were used. Complete (black dotted line) corresponds to an upper bound of the performance with all instances being labelled. Higher values are favourable.

Moving Hyperplane datastream. However, on the latter datastream, *COPAI*
is worse than *DBALStream*, while being on the other datastreams still bette
the majority of tested budget shares. Concerning the superiority of *DBALStre*
over *ACLStream*, which was indicated in [11], our results confirm that overall
former is the better strategy of the two. Due to the label sets becoming more a
more similar with increasing budgets, one would expect the differences betw
the strategies to diminish with increasing budget shares. This is indeed the c
in most of our results, except for *ACLStream* on Electricity.

The performance and adaptivity over time are reported in Fig. 2, which sh
the active classifier's accuracy for the budget share of 0.15. The trend of
black-doted curve of the *Complete*-baseline indicates changes in the classifi
tion task's complexity over time. Except for initially low performance (compa
to *Complete*) on the Random RBF and Electricity datastreams, the curves
COPAL-I and *COPAL-A* follow this trend, indicating a quick adaptation. H
ever, on the Random RBF datastream all approaches initially perform poo
and the *ACLStream* approach completely fails to improve over time (for bet
visibility of the other strategies' performance, its curve was cut below an ac
racy of 0.6, but its downward trend continued). Thus, except for *ACLStream*
active learning approaches were able to recover from drift.

In summary, our experimental evaluation indicates a mostly superior p
formance of the incremental variant *COPAL-I* compared to all other tes
approaches including its amnesic counterpart *COPAL-A*, while the latter sh
comparable performance to the most recently proposed *DBALStream* approa
In our experiments, the clustering-based active learning strategy *ACLStre*
proposed in [10] performed in most test-cases not better than random sampli

5 Conclusion

In this paper, we have proposed a clustering-based optimised probabilistic act
learning approach (COPAL) for selective sampling in evolving datastrea
Inspired from semi-supervised learning, it combines established clustering te
niques, which it uses to capture the structure within the unlabelled data, w
the recently proposed probabilistic active learning [14] approach, which ser
for selecting the best among the clusters. Our approach is designed for select
labels actively in nonstationary environments, and is usable to actively train a
adaptive stream classifier. We studied two variants of this approach: *COP*
I uses incremental clustering and windowing to maintain and adapt a si
clustering model over time. *COPAL-A* is an amnesic clustering variant that i
atively learns a new clustering model on each chunk and discards it after cla
fier training. The experimental evaluation against competitors that include
recently proposed approaches for evolving datastreams shows an overall su
rior performance of the proposed COPAL approach. The incremental vari
performs overall the best, while the amnesic variant performs at least on
with competitors and in three out of six datasets best for large budget si
While for better comparison with competitors the same combination of clus
ing and classifier technique was used in this paper, the performance with ot

Furthermore, COPAL uses the obtained clustering model solely in the active sampling process. Thus, the information from the structure of the unlabelled data is not considered explicitly during classifier training. Future work will focus on extending COPAL by semi-supervised techniques in the classification step, for example by self-labelling of the unlabelled instances or by using the clustering directly in the classification process.

Acknowledgments. We thank our colleagues, in particular Daniel Kottke, from University of Magdeburg, Christian Beyer from IBM Germany, and Vincent Lemaire from Orange Labs France, as well as Dino Ienco, Albert Bifet and Bernhard Pfahringer and the anonymous reviewers.

References

1. Abdallah, Z., Gaber, M., Srinivasan, B., Krishnaswamy, S.: Streamar: incremental and active learning with evolving sensory data for activity recognition. In: Proceedings of the 24th IEEE International Conference on Tools with Artificial Intelligence (2012)
2. Asuncion, A., Newman, D.J.: UCI machine learning repository (2015)
3. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: massive online analysis. J. Mach. Learn. Res. **11**, 1601–1604 (2010)
4. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. In: Bazzan, A.L.C., Labidi, S. (eds.) SBIA 2004. LNCS (LNAI), vol. 3171, pp. 286–295. Springer, Heidelberg (2004)
5. Gama, J., Sebastião, R., Rodrigues, P.P.: On evaluating stream learning algorithms. Mach. Learn. **90**, 317–346 (2013)
6. Gantz, J., Reinsel, D.: The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east, December 2012
7. Gopalkrishnan, V., Steier, D., Lewis, H., Guszcza, J.: Big data, big business: Bridging the gap. In: Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications, BigMine 2012, pp. 7–11. ACM, New York (2012)
8. Harries, M.: Splice-2 comparative evaluation: Electricity pricing. University of New South Wales, Australia, Technical report (1999)
9. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: KDD 2001: Proceedings of the seventh ACM SIGKDD International Conference on Knowledge discovery and data mining, pp. 97–106. ACM, New York (2001)
10. Ienco, D., Bifet, A., Žliobaitė, I., Pfahringer, B.: Clustering based active learning for evolving data streams. In: Fürnkranz, J., Hüllermeier, E., Higuchi, T. (eds.) DS 2013. LNCS (LNAI), vol. 8140, pp. 79–93. Springer, Heidelberg (2013)
11. Ienco, D., Pfahringer, B., Zliobaitė, I.: High density-focused uncertainty sampling for active learning over evolving stream data. In: Proceedings of the 3rd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications, pp. 133–148 (2014)
12. Kottke, D., Krempl, G., Spiliopoulou, M.: Probabilistic active learning in data streams. In: De Bie, T., Fromont, E. (eds.) Advances in Intelligent Data Analysis XIV - 14th International Symposium (IDA 2015). LNCS. Springer (2015)

13. Krempl, G., Kottke, D., Lemaire, V.: Optimised probabilistic active learn (OPAL) for fast, non-myopic, cost-sensitive active classification. Mach. Lea Spec. Issue ECML PKDD **2015**, 1–28 (2015)
14. Krempl, G., Kottke, D., Spiliopoulou, M.: Probabilistic active learning: towa combining versatility, optimality and efficiency. In: Džeroski, S., Panov, P., Koc D., Todorovski, L. (eds.) DS 2014. LNCS, vol. 8777, pp. 168–179. Sprin Heidelberg (2014)
15. Krempl, G., Zliobaitė, I., Brzeziński, D., Hüllermeier, E., Last, M., Lemaire, Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M., Stefanowski, J.: Open challen for data stream mining research. SIGKDD Explor. **16**(1), 1–10 (2014). special Is on Big Data
16. Loy, C.C., Hospedales, T.M., Xiang, T., Gong, S.: Stream-based joint explorati exploitation active learning. In: 2012 IEEE Conference on Computer Vision Pattern Recognition (CVPR), pp. 1560–1567 (2012)
17. Masud, M.M., Gao, J., Khan, L., Han, J., Thuraisingham, B.: Classification novel class detection in data streams with active mining. In: Zaki, M.J., J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010. LNCS, vol. 6119, pp. 311– Springer, Heidelberg (2010)
18. Moro, S., Laureano, R., Cortez, P.: Using data mining for bank direct marl ing: an application of the crisp-dm methodology. In: Novais, P. (ed.) Proceedi of the European Simulation and Modelling Conference (ESM'2011), pp. 117– EUROSIS, Guimarães (2011)
19. Nguyen, H.T., Smeulders, A.: Active learning using pre-clustering. In: Proceedi of the 21st International Conference on Machine Learning, ICML 2004, Ba Alberta, Canada, pp. 79–86. ACM Press (2004)
20. Nguyen, H.-L., Ng, W.-K., Woon, Y.-K.: Concurrent semi-supervised learn with active learning of data streams. In: Hameurlain, A., Küng, J., Wagner, Cuzzocrea, A., Dayal, U. (eds.) TLDKS VIII. LNCS, vol. 7790, pp. 113– Springer, Heidelberg (2013)
21. Ryu, J.W., Kantardzic, M.M., Kim, M.-W., Ra Khil, A.: An efficient methoc building an ensemble of classifiers in streaming data. In: Srinivasa, S., Bhatna V. (eds.) BDA 2012. LNCS, vol. 7678, pp. 122–133. Springer, Heidelberg (201
22. Settles, B.: Active learning literature survey. Computer Sciences Technical Rep 1648, University of Wisconsin-Madison, Madison, Wisconsin, USA (2009)
23. Settles, B.: Active Learning. Synthesis Lectures on Artificial Intelligence Machine Learning, vol. 18. Morgan and Claypool Publishers, San Rafael (201
24. Zhu, X., Zhang, P., Lin, X., Shi, Y.: Active learning from data streams. In: Proce ings of the 2007 Seventh IEEE International Conference on Data Mining, IC 2007, pp. 757–762. IEEE Computer Society, Washington, DC (2007)
25. Zhu, X., Zhang, P., Lin, X., Shi, Y.: Active learning from stream data using opti weight classifier ensemble. IEEE Trans. Syst. Man. Cybern. Part B Cybern. **40** 1607–1621 (2010)
26. Zliobaitė, I., Bifet, A., Pfahringer, B., Holmes, G.: Active learning with drift streaming data. IEEE Trans. Neural Netw. Learn. Syst. **25**(1), 27–39 (2013)

Chapter 9

Using Probabilistic Active Learning for Active Class Selection to Find Difficult Classes

A paper based on this chapter has been published as:

Probabilistic Active Learning for Active Class Selection.
Daniel Kottke, Georg Krempl, Marianne Stecklina, Cornelius Styp von Rekowski, Tim Sabsch, Tuan Pham Minh, Matthias Deliano, Myra Spiliopoulou and Bernhard Sick.

Mathewson, Kory and Subramanian, Kaushik and Loftin, Robert (eds.). Future of Interactive Learning Machines Workshop at the 30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

The final version of the resulting paper is available at
`http://www.filmnips.com/wp-content/uploads/2016/11/FILM-NIPS2016_paper_10.pdf`

Using Probabilistic Active Learning for Active Class Selection to Find Difficult Classes

Daniel Kottke, Georg Krempl, Marianne Stecklina, Cornelius Styp von Rekowski, Tim Sabsch, Tuan Pham Minh, Matthias Deliano, and Myra Spiliopoulou

Knowledge Management and Discovery Lab,
Otto von Guericke University Magdeburg, Germany.
{daniel.kottke,georg.krempl}@ovgu.de,
{tuan.pham,tim.sabsch,marianne.stecklina,cornelius.styp}@st.ovgu.de,
deliano@lin-magdeburg.de,myra@iti.cs.uni-magdeburg.de
http://www.kmd.ovgu.de/

Abstract. In machine learning, active class selection (ACS) algorithms aim to intelligently ask for instances of specific classes to optimize a classifier's performance while minimizing the number of instances. The challenge is to find the most appropriate sampling proportion for classes according to their difficulty at runtime. In this paper, we show the influence of the sampling proportion on the overall classification performance. Our proposed algorithm (PAL-ACS) applies the approach of probabilistic active learning to the active class selection scenario. We introduce the concept of pseudo instances, which are used to estimate in expectation the classifier's benefit from additional information. Weighting that score with the pseudo instance's density and its class conditional probability yields our final class selection score. Our experimental evaluation (on synthetic and real data) shows the advantages of our algorithm compared to state-of-the-art algorithms. Through determining the difficulty of classes, it adapts its sampling proportion and thereby improves its classification performance the most.

Keywords: Active class selection, probabilistic active learning, sampling proportion, difficult classes

1 Introduction

Methods with active interaction currently receive much attention as economy, medicine and research benefit from human knowledge for machine learning algorithms. Such methods are researched in the field of active machine learning [16]. Instead of asking for labels, active class selection (ACS) addresses classification problems, where *instances* are actively acquired or generated. In detail, an ACS algorithm selects classes subsequently from which an instance is added to the training set. The objective is to distribute the proportion of sampled instances for each class such that well-separable classes are sampled less than classes with complex decision boundaries.

One example for active class selection is the development of brain computer interfaces for motoric prostheses in the neurobiology research domain [5]. To train such a prosthesis, the impaired patient has to imagine motoric movements, such as finger movements, while brain activity is captured by EEG. The resulting data is evaluated to construct a classification hypothesis, which is used to assign new incoming brain data to a motion. As the learning process is highly exhausting and costly, one aims to reduce the length of that training phase, i.e. to create a good classification hypothesis within only few iterations. Especially for disabled patients, the use of BCI systems is critical, as they provide the possibility of performing otherwise impossible tasks [5]. There exist further examples, e.g. in arousal detection [18].

In this paper, we present a new Active Class Selection (ACS) approach that uses the benefits of the recently proposed multi-class probabilistic active learning [6] to select those classes that improve the classification performance the most. The adaptation of this approach uses pseudo instances to enable the applicability of the probabilistic active learning approach for active class selection. For each pseudo instance, we estimate the expected gain in performance and weight it with its density and class conditional probability. The density describes the influence of the hypothetic information gain for the overall classification performance. The class conditional probability is the probability that this class could be sampled when choosing that class. These values together build a comparative score, which we use to choose the class of the next instance.

The rest of the paper is structured as follows. In Sec. 2, we discuss the literature on active class selection, followed by a section discussing the influence of the sampling proportion for ACS experiments. In Sec. 4, we propose our new method PAL-ACS, show a pseudo code and discuss the approach with an exemplary ACS task. After an evaluation on multiple datasets in Sec. 5, we finally conclude our work.

2 Related Work

Active classification systems have the ability to request relevant information from external sources. With respect to the type of requested information, different approaches are distinguished [2]. The most intensively researched ones actively select instances for labeling from an oracle. The aim of these so-called active learning methods is to select those instances whose labels will improve the classification performance the most [16]. Scope of this paper is the inverse setting of active class selection [15]: here, the active component is able to select a class from which subsequently an unknown instance (feature vector) is generated.

The idea of active class selection (ACS) is to distribute the number of instances per class such that a certain level of classification performance is reached with the lowest number of requested instances [2, p. 29]. The work presented in [12] (see also [11]) proposes different techniques to determine this class distribution for acquisition chunks. First, they propose to use a *uniform* distribution and the *Original Proportion* (that usually is not known) as baselines. Performing what they called f-fold cross validation on the already seen chunks, they propose to use the results for the subsequent chunk: the approach *Inverse* distributes the information according to the inverse of the class accuracy. An extension of this is called *Accuracy Improvement*. It distributes the values according to the accuracy difference between the two most current chunks. The *Redistricting* method counts the number of labels that have been flipped (these instances are marked as redistricted) by adding the most recent chunk to the training set. Here, the upcoming instances are distributed with respect to the number of redistricted instances of the true classes.

Wu and Parsons [18] applied the previous algorithms *Inverse* and *Accuracy Improvement* for arousal classification. Later, they extended this paper in [17] and improved the approach *Inverse* to be applicable for incremental stream acquisitions and added a constraint that two consecutive new training examples are from different classes. In her PhD thesis [10], Lomasky extended her work by two more methods: *Risk* estimates the sensitivity of error that is induced by adding new instances of a certain class, and *Sensitivity* measures the stability of class decisions. As these methods are only mentioned in the PhD thesis yielding mixed results, we only consider the former ones.

As there are applicable aspects mentioned in active learning literature, which we partly use for our approach, we shortly summarize the most relevant works. To determine the most valuable label, most active learning methods propose a model to determine the usefulness of a label acquisition. Uncertainty sampling [9] suggests to acquire labels from instances that are near the decision boundaries. Expected error reduction methods [14] simulate every possible labeling for each unlabeled instance and train new classifiers. The instance whose new classifier reduces the error the most on an evaluation set is then selected for labeling. In [3], Chapelle observed that these error reduction estimates have issues with unreliable posterior estimates at the beginning. Thus, he suggests using a beta-prior to shift posterior values with less labeled information towards equal posterior probabilities. Recently, we proposed a method that overcomes the issues of unstable posteriors in a theoretically founded approach by modeling the expectation over the true posterior, called probabilistic active learning [6, 7]. Probabilistic methods directly optimize a performance measure and incorporate the amount of labeled information to notice unseen areas and areas with high Bayesian error.

3 Finding the Best Sampling

Active class selection (ACS) algorithms vary in their proportion of selected instances per class. As this is the only difference in their resulting training set, we can reformulate the problem which class to select next as the question of the most beneficial sampling proportion. To substantiate the idea of preferring a certain class over others, we designed an experiment to be able to describe the optimal behavior of such a method. Therefore, we generated three datasets consisting of four classes. The decision boundary between classes 2, 3 and 4 are similarly difficult, whereas the complexity of the boundary between class 1 and the other three varies between difficult, normal and easy.

To simplify the readability, we say that a class is easy, resp. difficult. A difficult class is characterized by having more complex decision boundaries. This implies that a classifier benefits from having more instances from that class. Nevertheless, to determine the shape of a decision boundary a classifier needs data from the class on its opposite site as well. Hence, we can conclude that the best sampling for a binary classification task is a uniform one. Another aspect is the kind of complexity of a decision boundary. While some decision boundaries might have a complex shape that can be learned, others might suffer from high Bayesian error. In the latter case, even learning this decision boundary perfectly might not improve the classification performance.

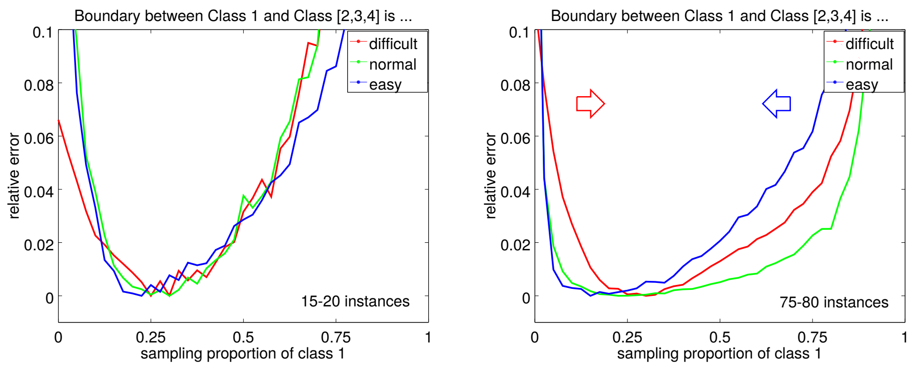


Fig. 1: Performance regarding sampling proportion on a synthetic dataset with a varying difficulty for the decision boundary between class 1 vs. classes 2,3,4.

In our experiment, we used active class selection algorithms that select instances according to a predefined sampling proportions. As we defined the data such that classes 2, 3 and 4 are equally complex, we used 41 versions of this sampler with different proportion values for class 1. The remaining proportions are distributed equally across the classes 2, 3 and 4. To achieve reliable results, we repeated this experiment using 300 randomly generated datasets of the same distribution. For evaluation, we generated 100 hold-out instances per class. During training, the ACS method has access to 100 data points per class, stored in an acquisition stack. In that way, we ensure that two samplers with the same proportion would receive the same result on the same dataset. This reduces the result's variance. The evaluation methodology is similar to the one we used in the later experiments and is described in more detail in Sec. 5.2.

In Fig. 1, we show the performance of the different active class selection algorithms in terms of mean performance on the hold out data with respect to the share of instances that are from class 1. All curves have been shifted to their best performance to be able to compare them across the datasets. Hence, this relative error describes the loss in performance by not finding the best sampling proportion. The left plot shows the results after 15-20 instances have been acquired, the right one after 75-80 instances. Using a range here reduces variance. In result, each of the 41 data points in the plots is calculated as a mean of $6 \cdot 300$ performance values.

The plots show that it is beneficial to prefer difficult classes over others. In case class 1 is more difficult (red curve), the performance curve shifts to the right with increasing number of instances, whereas in the opposite case (blue curve), it shifts to the left. If all three classes are equally difficult (green curve), the best sampling is a uniform one. Comparing both plots, we see that the best sampling at the beginning is nearly a uniform one and drifts towards the left, resp. right, with a higher number of instances. Furthermore, we see that missing the optimal sampling proportion has a smaller effect with increasing number of instances because the feature space is already covered better and classifier estimates become more reliable. Additionally, finding the best sampling proportions is more critical for cases where the decision boundaries are unequally difficult.

Summarizing, a non-uniform sampling proportion makes a difference in performance if that classes are unequally difficult. Therefore, finding difficult classes improves the learning task and helps reducing the number of training samples to achieve an appropriate classification model.

4 Our Method

In this section, we propose a new method called *Probabilistic Active Learning for Active Class Selection (PAL-ACS)*. The main idea is to estimate the benefit of acquiring an instance of class $y \in Y = \{1, \dots, C\}$ in terms of classification performance, and to select the class with best expected improvement. In the next section, we present the algorithm, followed by a discussion of its properties. The last section describes implementation aspects including a pseudo code.

4.1 Probabilistic Active Learning for Active Class Selection (PAL-ACS)

In an active class selection (ACS) task, instances are generated from the selected classes and stored in the set of training data that solely consists of labeled data $(x, y) \in \mathcal{L}$. To select the most beneficial classes, our approach *PAL-ACS* aims to estimate the usefulness of selecting the next instance from class $y \in \{1, \dots, C\}$ for the overall classification performance.

Works on probabilistic active learning [6, 7] have shown that an important aspect to consider is the reliability of the posterior estimates used to classify unseen data. Therefore, these probabilistic approaches model the distribution of the *true* posterior probability building on the observed posterior vector $\hat{\boldsymbol{p}}$, and the number of nearby labels n. The information about the number of nearby labels is used as a proxy for the reliability of that observed posterior. Multiple labels around an instance x provide evidence for the true posterior being close to the observed one. In contrast, posteriors based on few observed labels are doubtful. As a mathematical model, probabilistic approaches calculate the probabilities of the true posteriors $\boldsymbol{p}$ using the normalized likelihood function of a Multinomial distribution [6]. The gain in performance according to [6] is then calculated as given in Eq. 1-2.

$$\operatorname{perfGain}\big(x \mid \mathcal{L}\big) = \max_{m \leq M} \left(\frac{1}{m} \big(\operatorname{expExpPerf}\big(\boldsymbol{k}, m\big) - \operatorname{expExpPerf}\big(\boldsymbol{k}, 0\big) \big) \right) \tag{1}$$

$$\operatorname{expExpPerf}\big(\boldsymbol{k}, m\big) = \mathop{\mathbb{E}}_{\boldsymbol{p}} \left[\mathop{\mathbb{E}}_{\boldsymbol{l}} \left[\operatorname{perf}\big(\boldsymbol{k} + \boldsymbol{l} \mid \boldsymbol{p}\big) \right] \right] \tag{2}$$

Here, the parameter M is a maximal local budget for non-myopic applications with small constant (e.g. 3) as a recommended default [6]. The labeling vector $\boldsymbol{l}$ represents all possible labeling combinations when acquiring m labels (e.g., having a local budget of $m = 2$ and $C = 3$ classes, $\boldsymbol{l} \in \mathbb{N}^C$, with each column l_i representing the number of labels that are hypothetically acquired from class i, $\sum(l_i) = m$, e.g., $\boldsymbol{l} \in \{(2,0,0),(1,1,0),(0,1,1),\dots\}$). The vector $\boldsymbol{k} = \hat{\boldsymbol{p}} \cdot n$ represents the currently observed label frequencies (number of nearby labels of each class C) of an instance x. Hence, the $\operatorname{expExpPerf}\big(\boldsymbol{k}, m\big)$ calculates the expected performance having observed the labels in $\boldsymbol{k}$ near x allowing to select m more labels.

ACS methods do not have access to a candidate pool like active learning methods. Thus for using this model in active class selection, we propose to generate pseudo instances x_p over the whole feature space. Using these pseudo instances, we transform the active class selection problem into an active learning task. Hence, we have to additionally weight each instance with its probability from being from the currently considered class $P(x \mid y)$. Calculating the expectation over all pseudo instances yields the selection criterion for the *PAL-ACS* algorithm as follows.

$$y^* = \arg\max_{y} \left(\mathop{\mathbb{E}}_{x_p} \left[P(x_p \mid y, \mathcal{L}) \cdot \operatorname{perfGain}(x_p \mid \mathcal{L}) \right] \right) \tag{3}$$

Sampling the pseudo instances uniformly over the whole feature space, the probability of observing it in the data $\mathcal{L}$ can be estimated by its density $P(x|\mathcal{L})$ [8, 19].

$$y^* = \arg\max_{y} \left(\sum_{x_p} P(x_p \mid \mathcal{L}) \cdot P(x_p \mid y, \mathcal{L}) \cdot \operatorname{perfGain}(x_p \mid \mathcal{L}) \right) \tag{4}$$

A general tool to calculate the label frequencies $\boldsymbol{k}$ is kernel frequency estimation. These can also be used for classification by selecting the most frequent class. This classifier is similar to the very generic Parzen window classifier [3, 13]. In this paper, we use unnormalized Gaussian kernels with bandwidth σ as they provide a robust and expressive estimates.

$$k_y = \sum_{(x',y') \in \mathcal{L},\, y=y'} \exp\left(-\frac{||x-x'||^2}{2\sigma^2}\right) \tag{5}$$

4.2 Characteristics of PAL-ACS and Example

We now discuss PAL-ACS's approach in two exemplary active class selection situations shown in Fig. 2. Both situations are based on a three-class-classification task with a one-dimensional feature space. One class (blue) is well separated from the other two classes (red and green) and can therefore be considered to be easy. Due to an overlap of the other two classes, finding the best decision boundary between them is more difficult.

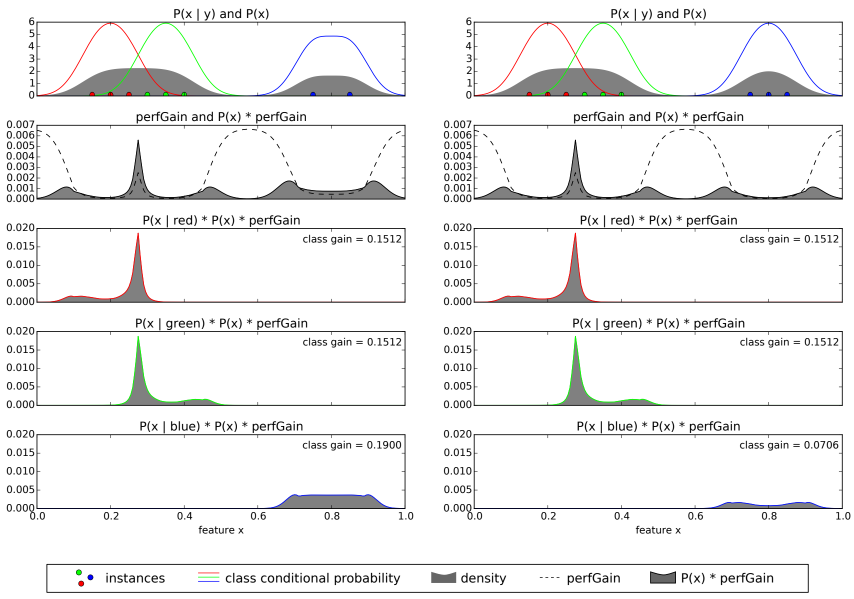


Fig. 2: Visualization of *PAL-ACS* for different situations.

The situations shown in the left and right columns are from consecutive selection steps. On the left, 8 instances (3 red, 3 green, 2 blue) have already been acquired, and subsequently one additional blue instance on the right. The upper plots show the labelings (red, green, and blue colored dots on the x-axis, corresponding to the classes' instances). Furthermore, they show the class conditional distributions in the corresponding color and the density as the gray area. The plots in the second horizontal row show the perfGain function over the whole feature space as a black dashed line, and the density weighted perfGain as a solid line with gray area. The lower three plots show the density weighted perfGain (solid black curve from above) weighted with the corresponding class conditional probabilities. The numbers in the upper right corners represent the sum of the corresponding values. The class with the maximal value is chosen for the next instance generation.

The relevant difference between both situations is the smaller number of blue instances on the left. Thus, there is less evidence for the blue class' posterior to be one. In terms of the weighted performance gain, this outweights the complex boundary of the red and green classes (see left lower three plots). Thus, *PAL-ACS* selects the blue class to validate the posterior.After generation of another blue instance (right), the red and green class have a similarly good score which is higher than the blue one. Hence, the algorithm detects that these classes are more beneficial. Thus, in analogy to [6], the perfGain function balances here exploration and exploitation by using the number of nearby labels. It also decreases the usefulness in regions with many labels and a high uncertain posterior. This prevents the algorithm from getting stuck in areas with high Bayesian errors.

4.3 Implementation and Pseudo Code

Fig. 3 provides the pseudo code of our approach, beginning with the sampling of pseudo instances $\mathcal{X}^p$. Sampling the whole feature space to get pseudo instances is time consuming, especially for high-dimensional data.

```
1: $n_p \leftarrow 25 \cdot C, \quad M \leftarrow 5$                    ▷ Init params to defaults
2: $\mathcal{L} \leftarrow \{\}$
3: while instance acquisitions left do
4:     $\mathcal{X}^p \leftarrow$ SampleFromDensity$(\mathcal{L}, n_p)$    ▷ Sample pseudo points (PP)
5:     for $i \in \{1, \dots, |\mathcal{X}^p|\}$ do                ▷ Calculate PP's perfGains
6:         $k_{i,\cdot} \leftarrow$ getKVector$(x_i^p, \mathcal{L})$
7:         $pg_i \leftarrow$ perfGain$(\boldsymbol{k}_i, m)/|\mathcal{L}, n_p)|$
8:     end for
9:     for $y \in \{1, \dots, C\}$ do                          ▷ Summarize weighted perfGains
10:        $g_y \leftarrow 0$
11:        for $i \in \{1, \dots, |\mathcal{X}^p|\}$ do
12:            $g_y \leftarrow g_y + pg_i \cdot k_{i,y}/(\sum_{j=1}^{|\mathcal{X}^p|} k_{j,y})$
13:        end for
14:    end for
15:    $y^* \leftarrow \arg\max_y(g_y)$                           ▷ Select optimal class
16:    $x \leftarrow$ requestInstance$(y^*)$
17:    $\mathcal{L} \leftarrow \mathcal{L} \cup (x, y^*)$
18: end while
```

Fig. 3: Pseudo code of the probabilistic active learning for active class selection (PAL-ACS) method.

Hence, we use a Monte-Carlo approach. In active class selection it is generally assumed that each class per se is similarly important (albeit not all are necessarily equally difficult). Therefore, we sample the same number of pseudo instances from each class. The distribution to sample from is determined by a kernel density estimation similarly to the frequency estimate's kernel. This function is called SampleFromDensity in the pseudo code (line 4).

In the for-loop (ll. 5-8), we estimate the kernel frequency vector as defined in Eq. 5 and calculate the corresponding performance gain (see Eq. 1) for each pseudo instance. As all values are generated from the data, each pseudo instance is now equally probable. Hence, the density weight is a simple division by the number of pseudo points. In lines 9-14, we weight this density weighted performance gain with the class conditional probability and sum all values for each class separately. Finally, we select the best class gain g_y and request a corresponding instance (ll. 15-17).

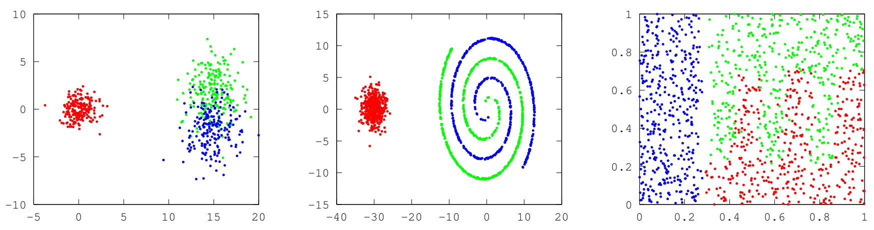
5 Evaluation

In this section, we evaluate the probabilistic active learning for active class selection (PAL-ACS)-approach against other methods on multiple datasets in experimental comparisons. After describing our evaluation setup, we provide learning curves, error tables, as well as proportion plots and discuss the results.

5.1 Evaluation Setup

The methods are evaluated on six different datasets. Thereof, three datasets are synthetic, having one class that is easily distinguishable from the others and two classes with a more complex decision boundary. A visualization for these two-dimensional datasets, called 3Clusters, Spirals and Bars, is given in Fig. 4a–4c. Additionally, we used three real-world datasets from the UCI machine learning repository [1], namely Vehicle, Vertebral Column, and Yeast. In Vehicle, the task is to distinguish four different vehicle classes. The task in Vertebral Column is to classify patients to one of the classes Normal, Disk Hernia or Spondylolisthesis. For Yeast, we selected four classes for our application: CYT vs. NUC vs. ME1 or ME2 vs. ME3. On all datasets, features are normalized to a $[0, 1]$ range. The datasets differ in their complexity. This requires a different number of instance acquisitions needed for analysis for each dataset. Thus, the maximum number of learning steps was set to 60 for 3Clusters, Vertebral, and Yeast, to 80 for Vehicle, and to 120 for Bars and Spirals. As a baseline approach, we implemented a randomly selecting strategy (*Random*) that selects each class with equal probability. Furthermore, we compare against the state-or-the-art approaches *Inverse* and *Redistricting* published in [12].

To classify unseen data from a hold-out test set, we use a Parzen window classifier with the same kernel used in the kernel frequency estimation with bandwidth $\sigma = 0.05$. Due to the feature normalization, we use the same bandwidth



(a) 3Clusters (b) Spirals (c) Bars

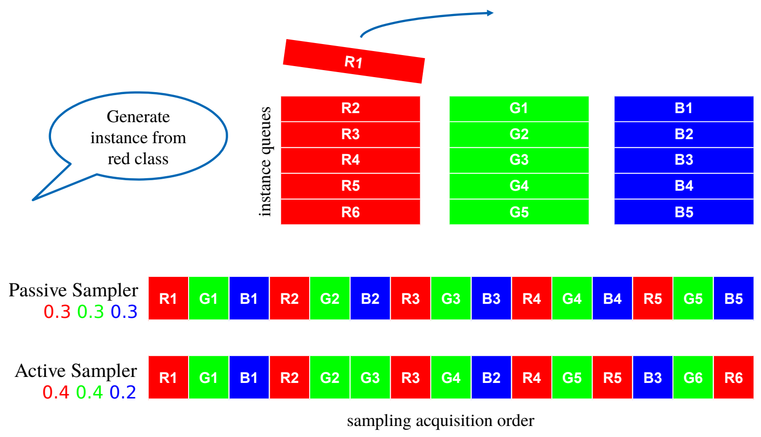
Fig. 4: Scatterplots of the synthetic datasets.

for all datasets. Changing the bandwidth slightly had only little influence on the order of the algorithms' performance. Error rate is used as performance measure.

According to [6], we set the maximal local budget of the performance gain function to a small number that does not change the result. To have a good tradeoff between performance and speed, we set $M = 5$. Especially for more classes, we recommend to reduce that value (e.g., $M = 3$). We used different values for the number of the pseudo instances to set a final value of $n_p = 25 \cdot C$. All experiments were performed on a computer cluster running the NeuroDebian operating system [4].

5.2 Evaluation Methodology

For each dataset, we generated 500 random test-training-set combinations (trials). The test set consists of 50 instances from each class. The training set is a stack containing all remaining instances in a random but fixed order.



R1
Generate instance from red class
instance queues
R2
R3
R4
R5
R6
G1
G2
G3
G4
G5
B1
B2
B3
B4
B5
Passive Sampler
0.3 0.3 0.3
R1 G1 B1 R2 G2 B2 R3 G3 B3 R4 G4 B4 R5 G5 B5
Active Sampler
0.4 0.4 0.2
R1 G1 B1 R2 G2 G3 R3 G4 B2 R4 G5 R5 B3 G6 R6
sampling acquisition order

Fig. 5: Evaluation methodology for active class selection

When an algorithm requests an instance, the first instance of this class is returned (see Fig. 5). At the beginning, each algorithm is initialized with one instance from each class. This setup reduces noise in the results, ensuring that only the algorithm's sampled class distribution influences the results. Thus, two selection strategies with exactly the same class distributions obtain precisely the same set of training instances. As seen in Fig. 5, this also means that the training data of samplers with different sampling proportions might overlap largely. In the example, we acquired 15 instances from 3 classes according to a uniform sampling proportion as a passive sampler does (like random), and according to a proportion of 40%, 40%, 20%. In that case, 13 instances of their training set are completely equal. Hence, the resulting classifiers might be very similar.

5.3 Results and Discussion

To compare the algorithms, we provide learning curves in Fig. 6. These learning curves show the mean error and the variance of all algorithms with respect to the number of acquired instances. The best algorithm is the one that converges fastest to the lowest error. Additionally, we provide quantitative values for the algorithms' performances in Tab. 2. Here, we separated the learning process into four phases, in order to determine how fast algorithms get the the structure of the learning problem. Each phase contains 25% of the learning steps. In the first phase, we exclude the results of the initialization phase as they are similar for each algorithm.

For each phase, we determine the mean accuracy for each algorithm on each dataset and calculate the ratio of won trials. Note, that these ratios do not sum to one because some trials have multiple, ex-equo winners due to the aspects discussed in Sec. 5.2. Furthermore, we provide information on the methods' sampling proportions. In Tab. 1, the final sampling proportions are summarized.

Their detailed development over the learning steps is shown for 3Clusters as exemplary dataset in Fig. 7. For the other datasets, these plots are provided at our companion website[1]. We now analyze different aspects of the result.

As visible from the learning curves and tables, *PAL-ACS* is constantly better than both competing active class selection methods, except for Bars. In the Bars dataset, it became only the best towards the very end. This might be due to the non-Gaussian structure of the data and the Gaussian classifier. Although its performance is low, *PAL-ACS* finds the easy class even in early phases (see companion webpage and Tab. 1). Comparing *PAL-ACS* with *Random*, we see that the superiority of our method depends on the structure of the data. The higher the differences in the complexity of the classes, the more beneficial is PAL-ACS. In Vehicle, for example, each class is equally difficult. Here, *Random* has the advantage that it assumes the classes to be equally difficult per default. Hence, its performance is slightly better, although *PAL-ACS* detects the best sampling proportion. Here, the advantages of *PAL-ACS* compared to Restricting and *Inverse* get visible, as it also detects regions of high Bayesian error.

[1] See `http://kmd.cs.ovgu.de/res/palacs`

Method	3Clusters	Bars	Spirals	Vehicle	Vertebral	Yeast
PAL-ACS	17,42,41	38,41,21	05,49,46	25,25,25,25	30,35,34	23,27,27,23
Inverse	29,35,36	35,36,30	28,36,36	27,27,24,23	38,36,25	28,28,22,23
Redist.	25,37,38	38,37,24	19,41,39	26,26,23,25	38,37,25	29,27,20,24
Random	33,34,33	33,33,34	33,34,33	25,25,25,25	33,34,33	25,25,25,25

Table 1: Final sampling proportions (for all classes) in percent.

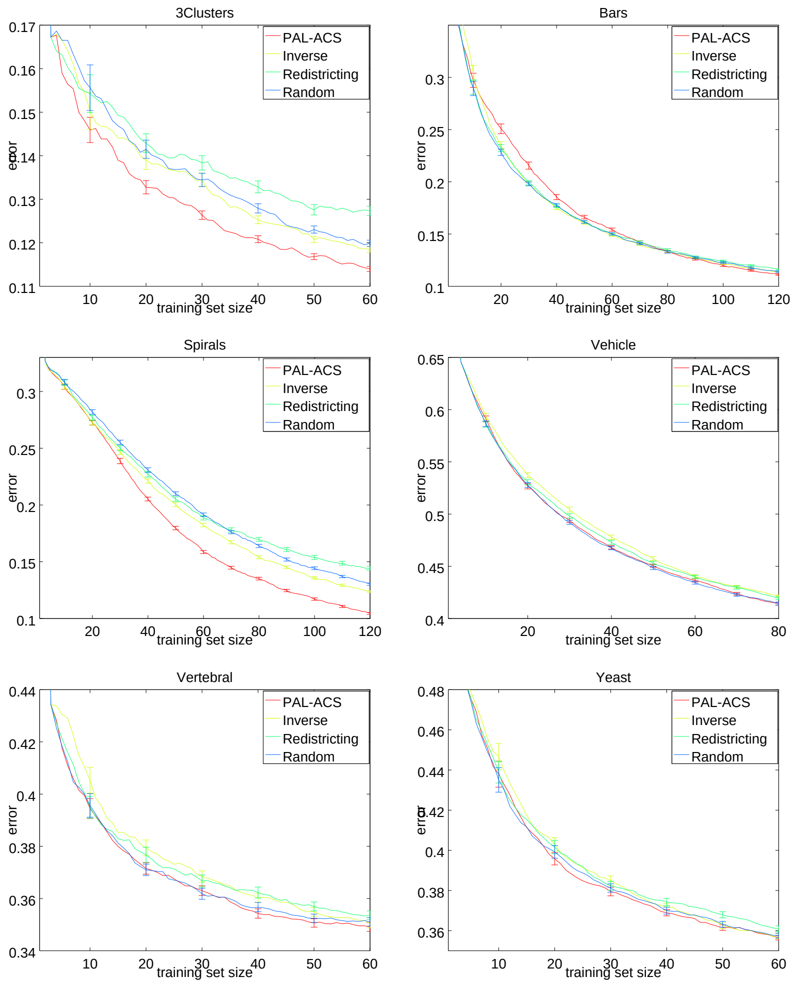


Fig. 6: Learning Curves for each algorithm on every dataset. Each curve shows the mean error and standard deviation.

Dataset	Method	phase 1		phase 2		phase 3		phase 4	
		error	win ratio	error	win ratio	error	win ratio	error	win ratio
3Clusters	PAL-ACS	**0.1498**	**40.97%**	**0.1316**	**42.69%**	**0.1215**	**45.85%**	**0.1161**	**47.93%**
	Inverse	0.1543	38.97%	0.1382	33.67%	0.1271	34.28%	0.1206	34.17%
	Redistricting	0.1557	36.10%	0.1418	30.25%	0.1339	31.28%	0.1281	31.69%
	Random	0.1575	35.18%	0.1390	31.13%	0.1294	30.93%	0.1217	32.68%
Bars	PAL-ACS	0.2705	25.97%	0.1773	28.82%	0.1378	32.47%	**0.1177**	**38.22%**
	Inverse	0.2636	31.12%	**0.1686**	33.14%	**0.1364**	32.24%	0.1196	31.85%
	Redistricting	0.2564	34.73%	0.1697	**33.69%**	0.1384	32.74%	0.1218	30.67%
	Random	**0.2539**	**35.59%**	0.1694	32.65%	0.1371	**32.88%**	0.1202	31.61%
Spirals	PAL-ACS	**0.2816**	**38.82%**	**0.1927**	**58.07%**	**0.1397**	**66.61%**	**0.1139**	**66.81%**
	Inverse	0.2831	34.47%	0.2103	24.53%	0.1611	21.94%	0.1328	20.74%
	Redistricting	0.2861	30.61%	0.2165	21.17%	0.1735	15.96%	0.151	15.64%
	Random	0.2897	26.16%	0.2205	13.66%	0.1701	13.09%	0.1404	13.97%
Vehicle	PAL-ACS	0.5783	**34.45%**	0.4931	29.19%	0.4499	29.89%	0.4238	32.06%
	Inverse	0.5851	31.49%	0.5041	23.60%	0.4572	23.04%	0.4301	23.75%
	Redistricting	0.5783	33.62%	0.4981	26.00%	0.4536	28.84%	0.4290	26.78%
	Random	**0.5776**	34.32%	**0.4920**	**33.42%**	**0.4486**	**32.75%**	**0.4230**	**33.89%**
Vertebral	PAL-ACS	**0.3989**	34.00%	0.3696	31.55%	**0.3566**	**33.44%**	**0.3506**	**33.63%**
	Inverse	0.4088	30.53%	0.3764	27.61%	0.3625	27.97%	0.3536	28.99%
	Redistricting	0.4009	34.13%	0.3737	30.12%	0.363	28.57%	0.3557	27.25%
	Random	0.3993	**34.38%**	**0.3695**	**32.43%**	0.3578	32.51%	0.3522	31.85%
Yeast	PAL-ACS	0.4439	37.38%	**0.3909**	**29.99%**	**0.3716**	**29.59%**	**0.3606**	31.20%
	Inverse	0.4495	32.48%	0.3967	27.23%	0.3744	28.80%	0.3612	**31.64%**
	Redistricting	0.4444	35.07%	0.3958	26.99%	0.3762	27.39%	0.3659	23.67%
	Random	**0.4417**	**38.40%**	0.3931	28.48%	0.3731	27.95%	0.3617	28.21%

Table 2: Quantitative Comparison of ACS methods on all datasets. To show the learning process, the mean of errors has been calculated for different learning phases. Additionally, the ratio of won trials for each algorithm is shown.

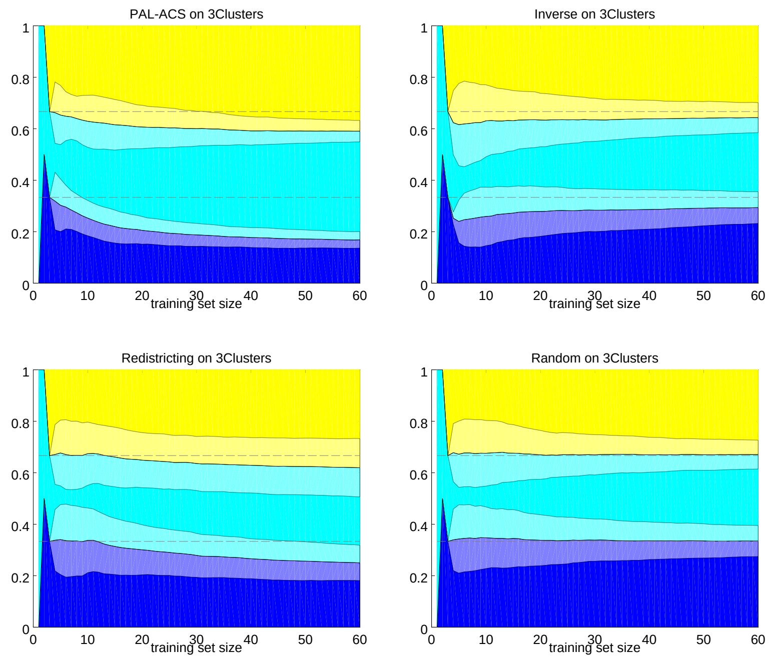


Fig. 7: Sampling proportions on 3Clusters dataset for all four sampling strategies.

The results on 3Clusters, Spirals, and Bars in Tab. 1 show, that *PAL-ACS* contributes a smaller sampling proportion to the easier class (1st in 3Clusters and Spirals, 3rd in Bars) than to the more difficult ones. *Inverse* and *Redistricting* show the same tendency, but to a much lesser extend, resulting in weaker performance in 3Clusters and Spirals. Furthermore, *PAL-ACS* also chose the best final sampling proportion in cases when all classes are equally difficult, as in the Vehicle dataset. Here, the best-performing methods are *PAL-ACS* and *Random*, both using uniform sampling proportions. In contrast, *Inverse* and *Redistricting* perform worse by undersampling classes.

Overall, *PAL-ACS* always identifies the difficult classes and samples accordingly. As a result, its performance is best (in cases some classes are more difficult than others) or ex-equo with the best competitor *Random* (in cases all classes are equally difficult). Furthermore, PAL-ACS' sampling rate over the learning process shows a stable convergence with lower variance, as shown in Fig. 7 for the exemplary 3Clusters dataset. This is similar for the other datasets (see companion website).

6 Conclusion

In this paper, we introduced an approach to subsequently choose the class where we expect a newly requested instance to improve the classification performance the most. Therefore, we generate pseudo instances to simulate possible instance acquisitions. Using a probabilistic model, we estimate the expected gain in performance and weight it with the instance's density and class conditional probability. The class with the highest class gain is the next to be selected.

The experimental evaluation shows our method's superiority on datasets where a non-uniform sampling improves the learning process. On datasets with equally complex decision boundaries, our method identifies uniform sampling as the best strategy. Thus, in contrast to other active class selection methods, it also performs on such datasets comparably well as the winning random sampling strategy.

Acknowledgements

We thank the Psychoinformatics lab, esp. Michael Hanke and Alex Waite, from Magdeburg University to let us use their cluster, and our colleague Pawel Matuszyk for various inspiring discussions.

References

1. A. Asuncion and D. J. Newman. Uci machine learning repository, 2007.
2. Josh Attenberg, Prem Melville, Foster Provost, and Maytal Saar-Tsechansky. Selective data acquisition for machine learning. In Balaji Krishnapuram, Shipeng Yu, and R. Bharat Rao, editors, *Cost-Sensitive Machine Learning*, chapter 5. CRC Press, 2011.

3. Olivier Chapelle. Active learning for parzen window classifier. In *Int. Workshop on Artificial Intelligence and Statistics*, pages 49–56, 2005.
4. Yaroslav O Halchenko and Michael Hanke. Open is not enough. let's take the next step: an integrated, community-driven computing platform for neuroscience. *Frontiers in neuroinformatics*, 6, 2012.
5. Johannes Höhne, Elisa Holz, Pit Staiger-Sälzer, Klaus-Robert Müller, Andrea Kübler, and Michael Tangermann. Motor imagery for severely motor-impaired patients: Evidence for brain-computer interfacing as superior control solution. *PLoS ONE*, 9(8), 08 2014.
6. Daniel Kottke, Georg Krempl, Dominik Lang, Johannes Teschner, and Myra Spiliopoulou. Multi-class probabilistic active learning. In *Proc. of the European Conf. on Artificial Intelligence 2016 (ECAI)*. IOS Press, 2016. under review.
7. Georg Krempl, Daniel Kottke, and Vincent Lemaire. Optimised probabilistic active learning (OPAL) for fast, non-myopic, cost-sensitive active classification. *Machine Learning*, Special Issue of ECML PKDD 2015, 2015.
8. Georg Krempl, Daniel Kottke, and Myra Spiliopoulou. Probabilistic active learning: Towards combining versatility, optimality and efficiency. In Saso Dzeroski, Pance Panov, Dragi Kocev, and Ljupco Todorovski, editors, *Proc. of the 17th Int. Conf. on Discovery Science*, volume 8777 of *LNCS*, pages 168–179. Springer, 2014.
9. DD Lewis and WA Gale. A sequential algorithm for training text classifiers. In *Proc. of the 17th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 1–10, 1994.
10. Rachel Lomasky. *Active Acquisition of Informative Training Data.* PhD thesis, Tufts Univ., 2009.
11. Rachel Lomasky, Carla E. Brodley, Matthew Aernecke, Sandra Bencic, and David Walt. Guiding class selection for an artificial nose. In *NIPS Workshop on Testing of Deployable Learning and Decision Systems*, 2006.
12. Rachel Lomasky, Carla E. Brodley, Matthew Aernecke, David Walt, and Mark Friedl. Active class selection. In *Machine Learning: ECML 2007*, volume 4701 of *LNCS*, pages 640–647. Springer, 2007.
13. Emmanuel Parzen. On Estimation of a Probability Density Function and Mode. *Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.
14. Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Int. Conf. on Machine Learning, ICML*, ICML '01, pages 441–448, San Francisco, CA, USA, 2001. Morgan Kaufmann.
15. Burr Settles. Active learning literature survey. *Univ. of Wisconsin, Madison*, 2010.
16. Burr Settles. *Active Learning*, volume 18 of *Synthesis Lectures on Artificial Intelligence and Machine Learning.* Morgan & Claypool, June 2012.
17. Dongrui Wu, Brent J. Lance, and Thomas D. Parsons. Collaborative filtering for brain-computer interaction using transfer learning and active class selection. *PLoS ONE*, 8(2):e56624, 2013.
18. Dongrui Wu and Thomas D. Parsons. Active class selection for arousal classification. In *Proc. of the 4th Int. Conf. on Affective Computing and Intelligent Interaction - Volume Part II*, ACII'11, pages 132–141. Springer, 2011.
19. Jingbo Zhu, Huizhen Wang, Tianshun Yao, and Benjamin K. Tsou. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *COLING 2008, 22nd Int. Conf. on Computational Linguistics, 18-22 August 2008, Manchester, UK*, pages 1137–1144, 2008.

Chapter 10

Appendix

10.1 Contribution Sheets

Contribution Sheet

Open Challenges for Data Stream Mining Research
by Georg Krempl, Indrė Zliobaitė, Dariusz Brzeziński, Eyke Hüllermeier, Mark Last, Vincent Lemaire, Tino Noack, Ammar Shaker, Sonja Sievi, Myra Spiliopoulou, and Jerzy Stefanowski.

In: SIGKDD Explorations, Special Issue on Big Data, 2014, 16, 1-10, ACM.

Contributions to the Field of Computer Science

This position paper discusses open challenges for mining streaming data in real-world applications. It points out gaps between current research and meaningful applications, highlight open problems, and defines new application-relevant research directions for data stream mining. The identified challenges cover the whole cycle of knowledge discovery, including the issues of data privacy protection, dealing with legacy systems, handling incomplete and delayed information, analysis of complex data, and evaluation of stream mining algorithms. Thus, it provides general suggestions concerning lines of future research in data stream mining.

Individual Contributions by the Authors

The authors made the following contributions to this joint work:

Georg Krempl and Indrė Zliobaitė jointly organized the RealStream-2013 workshop preceding this paper (together with G. Forman and Y. Wang). The two proposed the idea of this joint position paper, proposed its structure, and coordinated its writing. They jointly contributed section 1 (Introduction) and section 8 (Concluding Remarks).

Georg Krempl contributed section 2 (Data Stream Mining) and subsection 4.2 (Timing and Availability of Information).

Indrė Zliobaitė contributed section 3 (Protecting Privacy and Confidentiality), and subsection 4.1 (Streamed Preprocessing).

Eyke Hüllermeier, Mark Last, Ammar Shaker, and Myra Spiliopoulou jointly contributed section 5 (Mining Entities and Events). Eyke Hüllermeier, Mark Last, and Ammar Shaker jointly contributed in particular subsection 5.2 (Analyzing Event Data), Myra Spiliopoulou contributed in particular subsection 5.1 (Entity Stream Mining).

Dariusz Brzeziński and Jerzy Stefanowski contributed section 6 (Evaluation of Data Stream Algorithms).

Vincent Lemaire, Tino Noack, and Sonja Sievi contributed section 7 (From Algorithms to Decision Support Systems). Vincent Lemaire contributed in particular subsection 7.1 (Making models simpler, more reactive, and more specialized). Tino Noack and Sonja Sievi contributed in particular subsection 7.2 (Dealing with Legacy Systems)

All authors contributed by reviewing, discussing and editing the final version of the paper.

As acknowledged in the work, the authors would like to thank the participants of the RealStream2013 workshop at ECML PKDD 2013 in Prague, and in particular Bernhard Pfahringer and George Forman, for suggestions and discussions on the challenges in stream mining.

Georg Krempl	Indrė Zliobaitė
Dariusz Brzeziński	Eyke Hüllermeier
Mark Last	Vincent Lemaire
Tino Noack	Ammar Shaker
Sonja Sievi	Myra Spiliopoulou
Jerzy Stefanowski	

Contribution Sheet

Probabilistic Active Learning:
Towards Combining Versatility, Optimality & Efficiency.
by Georg Krempl, Daniel Kottke, Myra Spiliopoulou.

In: S. Deroski, P. Panov, D. Kocev, and L. Todorovski (eds.). Proceedings of the 17th International Conference on Discovery Science (DS), October 8–10, 2014, Bled, Slovenia. Lecture Notes in Computer Science, Springer. ISSN: 0302-9743.

Contributions to the Field of Computer Science

This paper addresses the problem of active machine learning in a pool-based setting. In particular, it aims to offer an approach that combines versatility, direct optimisation of a performance measure, and computational efficiency. It contributes a novel probabilistic active learning (PAL) approach for binary classification that combines these qualities. This is a decision-theoretic approach that computes the expected gain in performance from labelling a candidate from a pool. To this end, PAL models both the candidate's label realisation and the true posterior in its neighbourhood as random variables. In contrast to expected error reduction, PAL then performs expectation not only over this candidate's label realisation, but also over the true posterior of the positive class in the candidate's neighbourhood. The latter directly incorporates the likelihoods of different possible posteriors under the already labelled data. This advances the state-of-the-art decision-theoretic active learning literature, which considers solely the most likely (or most pessimistic) posterior. Subsequently, PAL weights this expected gain by the density over labelled and unlabelled instances at the candidate's position. Thereby, it approximates the importance of the candidate's neighbourhood in the classification task. Finally, PAL selects the candidate for labelling that will improve the classification performance in expectation the most. The experimental evaluation on several data sets shows that PAL yields comparable or better classification performance than error-reduction, uncertainty-sampling or random active learning strategies, while requiring less computational time than error-reduction.

Individual Contributions by the Authors

The authors made the following contributions to this joint work:

Georg Krempl contributed the idea of this approach, i.e. to compute the gain in classification performance using expectation over both the candidate's label realisation and the true posterior of the positive class in its neighbourhood. He derived and formulated the model and the asymptotic bound for its computational complexity. He contributed the review of the related work and the idea for Figure 1 on the different exemplary active learning situations. He contributed the idea for Figure 3 and the discussion of PAL's properties. He suggested the setup for the experimental evaluation and advised Daniel Kottke in the implementation and evaluation, which was done jointly. He texted drafts for all sections but Section 4, which was drafted by Daniel Kottke.

Daniel Kottke contributed the draft for the text of the experimental evaluation, for which he implemented the compared algorithms and an evaluation framework in MATLAB jointly with Georg Krempl. In addition to the experiments reported in the paper, Daniel Kottke and Georg Krempl jointly performed extensive preliminary experimental evaluations, which led to the final formulation of PAL as presented in the paper.

Myra Spiliopoulou contributed in discussions and written comments her expertise in semi-supervised learning. She helped in defining the scope of the work, and provided insights that led to PAL's formulation. Furthermore, she pointed to some of the related work.

All authors contributed by reviewing, discussing and editing the final version of the paper.

As acknowledged in the work, the authors would like to thank in particular Vincent Lemaire, who helped in discussions in shaping the scope of this paper, and furthermore the anonymous reviewers for their comments to improve the paper. The authors,

Georg Krempl

Daniel Kottke

Myra Spiliopoulou

Contribution Sheet

Optimised Probabilistic Active Learning (OPAL)
For Fast, Non-Myopic, Cost-Sensitive Active Classification
by Georg Krempl, Daniel Kottke, Vincent Lemaire.

In: C. Bielza, J. Gama, A. Jorge, I. Zliobaite (eds.). Machine Learning, Special Issue of the Journal Track of ECML/PKDD 2015, Springer, 2015. ISSN: 1573-0565.

Contributions to the Field of Computer Science

This paper addresses the problem of active machine learning in a pool-based setting. Building on the probilistic active learning (PAL) framework introduced in [1], this paper contributes an optimised approach (OPAL) that, in addition to PAL, (1) is also usable in **cost-sensitive** applications, as it optimises the candidate selection for minimisation of the expected misclassification loss, (2) includes a **non-myopic** variant, that considers the available budget for subsequent label acquisitions when computing the value of a candidate, (3) is **fast** due to a closed-form computation. This combination of properties in a single active learning approach, that is not limited to a particular classifier technique, is novel and of practical relevance, due to limited human annotation capacities but ever increasing volumes of automatically generated data.

Individual Contributions by the Authors

The authors made the following contributions to this joint work:

Georg Krempl contributed the idea to extend the approach proposed by considering several label acquisitions at once, thereby making it non-myopic, and to derive a closed-form solution for misclassification loss, thereby making it fast and cost-sensitive. He derived this closed-form solution jointly with Daniel Kottke. He proposed scope, structure and organisation of the paper. He has written the introduction, related work, conclusion, and most parts of the method section. He contributed the idea for Figure 3 and the discussion thereof. He advised Daniel Kottke in the setup for experimental evaluation.

Daniel Kottke contributed the experimental evaluation, for which he implemented the compared algorithms and an evaluation framework in MATLAB. In addition to the experiments reported in the paper, Daniel Kottke and Georg Krempl jointly performed extensive preliminary experimental evaluations, which led to the final formulation of the OPAL presented in the paper, for which they jointly derived a closed-form solution. Daniel Kottke has written the experimental evaluation section, he has designed the presentation of results on the companion Website, and he contributed the idea for Figure 2 and the discussion thereof.

Vincent Lemaire contributed in discussions and written comments his expertise in active learning. He helped in defining the scope of the work, pointed to some of the related work, and helped in designing the setup for the experimental evaluation.

As acknowledged in the work, the authors would like to thank in particular Myra Spiliopoulou, Pawel Matuszyk and Christian Braune, and furthermore the anonymous reviewers for their comments to improve the paper. The authors,

Georg Krempl

Daniel Kottke

Vincent Lemaire

References

[1] Georg Krempl, Daniel Kottke, and Myra Spiliopoulou. Probabilistic active learning: Towards combining versatility, optimality and efficiency. In Saso Dzeroski, Pance Panov, Dragi Kocev, and Ljupco Todorovski, editors, *Proceedings of the 17th Int. Conf. on Discovery Science (DS), Bled*, volume 8777 of *Lecture Notes in Computer Science*, pages 168–179. Springer, 2014.

Contribution Sheet

Multi-Class Probabilistic Active Learning
by Daniel Kottke, Georg Krempl, Dominik Lang, Johannes Teschner, Myra Spiliopoulou.

Under Review for: European Conference on Artificial Intelligence ECAI 2016, IOS Press, 2016.

Contributions to the Field of Computer Science

This work addresses active learning for multi-class classification in pools. The first contribution of this work is the identification of different influence factors that positively affect active learning. These factors are (1) an instance's impact, (2) its posterior, and (3) the reliability of this posterior. Furthermore, this work contributes a new approach, called multi-class probabilistic active learning (McPAL). It builds on the probabilistic active learning framework[1], is non-myopic, fast, and directly optimises a performance measure, e.g. accuracy. Considering all influence factors, McPAL determines the expected gain in performance to compare the usefulness of instances. For this purpose, it calculates the density-weighted expectation over the true posterior and over all possible labeling combinations in a closed-form solution. Thus, in contrast to other multi-class algorithms, it considers the posterior's reliability which improved the performance. The experimental evaluation of this paper shows the reasonability of the selected influence factors and the superiority of McPAL in comparison to various other multi-class active learning algorithms on six datasets.

Contributions by the Authors

The authors made the following contributions to this joint work:

Daniel Kottke contributed the scope, the formulation of the approach and the derivation of the closed-form solution to the paper. He has written the drafts for the section introduction, related work, method, and conclusion and implemented the functions for the McPAL method. He co-supervised (together with Georg Krempl) Dominik Lang and Johannes Teschner and specified the experimental setup. He participated in the review of the drafts and finalizing the paper.

Georg Krempl advised in the planning and in the definition of the scope and the structure of the work. Together with Daniel Kottke, he co-supervised Dominik Lang and Johannes Teschner. He participated in the review of the drafts and finalizing the paper.

Dominik Lang and Johannes Teschner contributed the drafts of the experimental evaluation, for which they implemented the compared algorithms and an evaluation framework. They participated in the review of the draft and finalizing the paper.

Myra Spiliopoulou initiated the research on multi-class active learning on opinionated data. She advised in the planning and contributed her expertise in discussions and comments on multi-class classification tasks.

As acknowledged in the work, the authors would like to thank the Pycho-informatics Labs at OVG-University Magdeburg, in particular Michael Hanke and Alex Waite, for providing their cluster for the experiments, as well as Pawel Matuszyk for all discussions.

Daniel Kottke	Georg Krempl
Dominik Lang	Johannes Teschner
Myra Spiliopoulou	

References

[1] Georg Krempl, Daniel Kottke, and Myra Spiliopoulou. Probabilistic active learning: Towards combining versatility, optimality and efficiency. In Saso Dzeroski, Pance Panov, Dragi Kocev, and Ljupco Todorovski, editors, *Proceedings of the 17th Int. Conf. on Discovery Science (DS), Bled*, volume 8777 of *Lecture Notes in Computer Science*, pages 168–179. Springer, 2014.

Contribution Sheet

How to Select Information That Matters: A Comparative Study on Active Learning Strategies for Classification
by Christian Beyer, Georg Krempl, and Vincent Lemaire.

In: Stefanie Lindstaedt, Tobias Ley, and Harald Sack. Proceedings of the 15th Int. Conf. on Knowledge Technologies and Data-driven Business, i-KNOW 2015, ACM, 2015. ISBN: 978-1-4503-3721-2.

Contributions to the Field of Computer Science

This paper addresses the problem of active machine learning in a pool-based setting. It contributes a comparison between different combinations of active learning strategies and classification algorithms. In particular, the compared active learning strategies include random sampling as pure exploration strategy, uncertainty sampling as exploitation strategy, a semi-random sampling strategy as hybrid between the previous two, and probabilistic active learning. As classification algorithms, it includes Hoeffding trees, Naive Bayes, logistic regression, k-nearest neighbour and Parzen Window classifiers. Its results confirm the finding of previous studies that neither pure exploration nor pure exploitation strategies perform consistently well. It underlines the importance of handling the trade-off between *exploration* and *exploitation*. Furthermore, it shows that probabilistic active learning approach significantly outperforms uncertainty-sampling-based strategies when used with Bayes, Naive Bayes or Decision-Tree Classifiers, but works not well on k-Nearest Neighbour or Logistic Regression Classifiers. In addition, it shows that better results for most classifier technologies are obtained when using label statistics that are directly based on the probabilistic classifier's estimates.
This paper was awarded as best paper at the i-KNOW conference 2015.

Individual Contributions by the Authors

This paper builds on experimental work from a preceding master thesis. This master thesis was written by Christian Beyer, supervised by Georg Krempl, and reviewed by Vincent Lemaire. For this joint paper, the authors made the following contributions:

Christian Beyer and Georg Krempl jointly wrote a first version of this paper.

Christian Beyer wrote the master thesis preceding this paper, for which he performed the experimental evaluation used in this paper. For this evaluation, he implemented the compared algorithms as well as an evaluation framework in Java.

Georg Krempl supervised Christian Beyer's master thesis. He contributed the idea and scope of the master thesis and paper, provided related work and proposed the comparison to the problem of label reusability, and defined the setup for the experimental evaluation.

Vincent Lemaire was second reviewer of Christian Beyer's master thesis. He contributed in discussions and written comments his expertise in active learning. He helped in defining the scope of the work, suggested among others the comparison between a hybrid between random sampling for exploration and uncertainty sampling for exploitation.

All authors contributed jointly in reviewing and editing the paper.

As acknowledged in the work, the authors would like to thank Myra Spiliopoulou and Daniel Kottke for their comments and suggestions.

Christian Beyer

Georg Krempl

Vincent Lemaire

Contribution Sheet

Probabilistic Active Learning in Datastreams
by Daniel Kottke, Georg Krempl, Myra Spiliopoulou.

In: E. Fromont, T. De Bie, M. van Leeuwen. Advances in Intelligent Data Analysis XIV, Proc. of the 14th Int. Symposium (IDA 2015), Lecture Notes in Computer Science, vol. 9385, pp.145–157, Springer, 2015.

Contributions to the Field of Computer Science

This work addresses active learning in evolving data streams. It contributes a new algorithm for stream-based active learning that decides immediately whether to acquire a label (selective sampling). To this end, the work extends the recently proposed pool-based Probabilistic Active Learning framework [1] for data streams. In particular, the work complements the notion of usefulness within a topological space ("spatial usefulness") with the concept of "temporal usefulness". In this paper, the so-called Balanced Incremental Quantile Filter (BIQF) is introduced for active selection of instances for labelling. This BIQF algorithm assesses the usefulness of instances in a sliding window, ensuring that the predefined budget restrictions will be met within a given tolerance window. The experimental evaluation of this approach against other active learning approaches for evolving data streams shows its competitiveness.

Contributions by the Authors

The authors made the following contributions to this joint work:

Daniel Kottke wrote his Master's thesis about "Budget Optimization for Active Learning in Data Streams". This thesis was supervised by Georg Krempl, who provided the topic. The idea of the approach is a result of multiple intense discussions between Daniel Kottke and Georg Krempl. The paper presents the main results from this thesis. Hence, Daniel Kottke provided raw drafts from the original manuscript of the thesis. He wrote the draft of the paper jointly with Georg Krempl.

Georg Krempl provided the topic for the underlying Master's thesis and defined in joint work with Daniel Kottke the work packages. He supervised the planning and the definition of the scope and the structure of this paper. He participated in writing the draft of the paper, based on raw text from the thesis, together with Daniel Kottke.

Myra Spiliopoulou advised in the planning and scope and contributed her expertise in data stream mining and comments on structure and presentation.

All authors contributed in editing the final version of this paper.

The authors would like to thank the Pychoinformatics Labs at OVG-University Magdeburg, in particular Michael Hanke and Alex Waite, for providing their cluster for the experiments.

Daniel Kottke

Georg Krempl

Myra Spiliopoulou

References

[1] Georg Krempl, Daniel Kottke, and Myra Spiliopoulou. Probabilistic active learning: Towards combining versatility, optimality and efficiency. In Saso Dzeroski, Pance Panov, Dragi Kocev, and Ljupco Todorovski, editors, *Proceedings of the 17th Int. Conf. on Discovery Science (DS), Bled*, volume 8777 of *Lecture Notes in Computer Science*, pages 168–179. Springer, 2014.

Contribution Sheet

Clustering-Based Optimised Probabilistic Active Learning
by Georg Krempl, Tuan Cuong Ha, and Myra Spiliopoulou.

In: Nathalie Japkowicz and Stan Matwin. Proceedings of the 18$^{\text{th}}$ Int. Conf. on Discovery Science, DS 2015, Springer Lecture Notes in Computer Science, vol. 9356, pages 101–115, 2015. ISBN 978-3-319-24281-1.

Contributions to the Field of Computer Science

This paper addresses the problem of active machine learning in evolving data streams. It proposes a so-called clustering-based optimised probabilistic active learning (COPAL) approach. This approach combines the idea of clustering-based active learning with the idea of probabilistic active learning. Two variants of this COPAL-approach are proposed. The first uses an incremental clustering variant, where the clustering model is maintained over the arriving chunks of data. The second, amnesic clustering variant builds the clustering model anew from scratch on each chunk. The proposed approaches are evaluated on several synthetic and real-world datasets against two state-of-the-art active learning approaches for evolving data streams.

Individual Contributions by the Authors

This paper uses the results of the experiments that were performed for a master thesis written shortly after the paper was submitted for review. The master thesis was written by Tuan Cuong Ha, supervised by Georg Krempl, and reviewed by Myra Spiliopoulou. For this joint paper, the authors made the following contributions:

Georg Krempl contributed the idea and design of the COPAL approach and texted this paper. He proposed the combination of clustering and OPAL, and to split the algorithm in the repeatedly performed four steps (initial clustering or clustering update from unlabelled data, macro selection, micro selection, clustering update from the acquired label). He proposed variants of COPAL that combine these steps with a semi-supervised learning step using self-labelling (not included in this paper), and a variant that uses the clustering model as ensemble classifier (not included in this paper). He proposed the design of the experiments, including the selection of ACLStream and DBALStream as baselines.

Tuan Cuong Ha contributed the results of the experimental evaluation, which he performed for his master thesis. Therein, he implemented the COPAL variants and an evaluation framework in Java, and performed the experimental evaluation, providing the plots for Figures 1 and 2. Furthermore, he proposed to select labels within the micro step such that the distance between newly requested labels and existing ones is maximised. His suggested modification of the micro step was used in the final version of this paper.

Myra Spiliopoulou contributed in discussions and written comments her expertise in active learning, semi-supervised learning and stream clustering. She helped in defining the scope of the work and helped in editing the camera ready of this paper.

As acknowledged in the work, the authors would like to thank Daniel Kottke and Christian Beyer for suggestions on the Java implementation, Vincent Lemaire for discussions on active learning, Dino Ienco, Albert Bifet and Bernhard Pfahringer for discussions and providing their source code of the ACLStream and DBALStream approaches.

Georg Krempl

Tuan Cuong Ha

Myra Spiliopoulou

Contribution Sheet

Using Probabilistic Active Learning for Active Class Selection to Find Difficult Classes
by Daniel Kottke, Georg Krempl, Marianne Stecklina, Cornelius Styp von Rekowski, Tim Sabsch, Tuan Pham Minh, Matthias Deliano, and Myra Spiliopoulou.

Under Review for: Discovery Science, DS 2016, Springer, 2016.

Contributions to the Field of Computer Science

This work is in the subfield of active learning, within the field of machine learning. It addresses the problem active class selection (ACS) for multi-class classification. Active class selection algorithms aim to intelligently ask for instances of specific classes to optimize a classifier's performance while minimizing the number of instances. This paper shows the influence of the sampling proportion on the overall classification performance. Thus, the challenge is to find the most appropriate sampling proportion for classes according to their difficulty at runtime. This paper proposes to apply the approach of probabilistic active learning [1] to the active class selection scenario. A novel algorithm (PAL-ACS) that uses this approach is contributed in this work. This algorithm introduces the concept of pseudo instances, which are used to estimate in expectation the classifier's benefit from additional information. This expected value is weighted with the pseudo instance's density and its class conditional probability, yielding the final class selection score. The experimental evaluation shows the advantages of the PAL-ACS algorithm compared to state-of-the-art algorithms on several synthetic and real data. These experiments indicate that PAL-ACS adapts its sampling proportion according to the difficulty of classes, thereby optmizing its classification performance.

Contributions by the Authors

The authors made the following contributions to this joint work:

Daniel Kottke contributed the final PAL-ACS approach as presented in this paper. He implemented this algorithm and performed the experimental evaluation. He defined the scope of this work and complemented the related work. He drafted the sections (1) introduction, (3) finding the best sampling, and (4) our method.

Georg Krempl contributed in discussions with Matthias Deliano the initial idea of using the probabilistic approach for the active class selection problem. He provided initial references and drafted the sections (5) evaluation and (6) conclusion.

Marianne Stecklina, Cornelius Styp von Rekowski, Tim Sabsch, and Tuan Pham Minh contributed the implementation of reference algorithms and the evaluation framework, and a draft of section (2) related work. The implementations for the experiments in section 3 were done by Cornelius Styp von Rekowski, the text of this section was drafted by Daniel Kottke. Marianne Stecklina contributed Figure 5 visualizing the evaluation methodology. Tim Sabsch contributed in the review and editing of the draft for the submission.

Matthias Deliano contributed in discussions with Georg Krempl the problem definition and motivation in BCI applications. Furthermore, he provided data and support for preliminary experiments (not shown in the final version of this paper).

Myra Spiliopoulou contributed her expertise in discussions and comments. She helped in shaping the scope and presentation of this work. Inputs from these discussions motivated Figure 7 and related experiments.

Daniel Kottke, Georg Krempl, and Matthias Deliano jointly supervised the different student projects leading to this paper. Thereby, Daniel Kottke was the primary contact person.

All authors contributed in editing the final version of this paper.

As acknowledged in the work, the authors would like to thank the Psychoinformatics Labs at OVG-University Magdeburg, in particular Michael Hanke and Alex Waite, for providing their cluster for the experiments, and Pawel Matuszyk for all discussions.

Daniel Kottke	Georg Krempl
Marianne Stecklina	Cornelius Styp von Rekowski
Tim Sabsch	Tuan Pham Minh
Matthias Deliano	Myra Spiliopoulou

References

[1] Georg Krempl, Daniel Kottke, and Myra Spiliopoulou. Probabilistic active learning: Towards combining versatility, optimality and efficiency. In Saso Dzeroski, Pance Panov, Dragi Kocev, and Ljupco Todorovski, editors, *Proceedings of the 17th Int. Conf. on Discovery Science (DS), Bled*, volume 8777 of *Lecture Notes in Computer Science*, pages 168–179. Springer, 2014.

10.2 Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich keine früheren Habilitationsversuche unternommen habe, dass ich die vorliegende Habilitationsschrift selbständig gemäß den Angaben auf den beiliegenden Contribution Sheets verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Magdeburg, den 15. Juni 2016

Georg Krempl