ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-586

# 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 performance by successively selecting the most beneficial instances from a pool of unlabeled instances to be labeled by an oracle. In this work, we study the influence of the following factors for active learning: (1) an instance's impact, (2) its posterior, and (3) the reliability of this posterior. To do so, we propose 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 that the combination of the selected influence factors works best and that McPAL is superior 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 [23].

A particular and little researched challenge [26] 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 with support vector machines, thereby concentrating on instances close to the anticipated decision boundary [6, 12, 29], optionally extended by information about density or diversity [4, 14]. Others use expected error reduction by simulating the impact of a label acquisition on the whole dataset to determine the expected performance [13]. Both approaches have known limitations [7, 15]: the former fast, information-theoretic heuristic often fails in exploring the dataspace, the latter decision-theoretic method has high computation time.

[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

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 [17], the key idea is to compute the expectation over the true posterior by incorporating the number of labels in a neighborhood of the label candidate as a proxy for the posterior's reliability. The resulting score is weighted with the density which we use as a proxy for the new label's impact 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 that use the number of labels inside a candidate's neighborhood for multi-class active learning, which we show to has a strong impact on the learner's performance. Furthermore, by adding another decision-theoretic method to propositions in the comparative study of [14], 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 concluded 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 [23]. 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 [18]. 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 [23]. 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 [21]. 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. [16] argue that using posterior estimates directly in the expectation step leads to inaccuracies. They observed that these posterior estimates are highly unreliable especially having only few labeled instances. Probabilistic active learning [17] 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 [24].

For active learning with 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 [14] 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, 11, 28, 29]), (2) entropy-based sampling chooses the instance with highest posterior entropy (see also [30]), (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, 12]), 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. [13] 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 are evaluated for each labeling candidate and each class on an evaluation set. As these algorithms take long for execution, the authors 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 [25]), a density score that favors dense regions and a diversity score that enforces diversity among instances from the labeling set. More recently, Guo and Wang [6] developed a stepwise 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 score, a compatibility score and an uncertainty score, and propose rules on how these score have to be considered. Wang et al. [26] propose an ambiguity-based multi-class approach that uses possibilistic membership from One-vs-Rest SVMs. These membership values 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 $\gamma$ 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, 13]. Additionally, [13] also includes annotation cost, i.e. the cost of labeling one instance. The acquisition of instances can be done in a successive manner or in form of instance batches. Most approaches choose to acquire instances one-by-one, except for [4, 30]. Besides SVMs (often used with a probabilistic version), [14] used an ensemble of trees, [11] proposed a probabilistic version of the k-nearest-neighbor (pKNN) classifier, [5] tested their algorithms on a random forest, and [30] 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 [23]).

## 3 OUR METHOD

In this section, we propose probabilistic active learning for multiple classes, an extension of the binary version proposed in [16]. 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 active learner successively selects the most useful instances $\vec{x}^*$ from the candidate pool $\mathcal{U}$ and requests its label $y$ from the oracle. After retraining the classifier with the new labeled set $\mathcal{L} \cup (\vec{x}^*, y)$, this procedure 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 [19], which is updated according to the contents of $\mathcal{L}$.

```
function al_framework(U){
  L = {}
  cl = init_classifier()
  for(i=1; i<=b; i++){
    x* = active_learning(U, cl, L)
    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 positively. 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. $\hat{p}_i = \hat{P}(y = i \mid \vec{x}), 1 \leq i \leq C$. 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 [22]. 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 [14]. 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 [14, 23]). In Fig. 2, we show a ternary heatmap plot where the darker shades indicate higher usefulness. This is a barycentric coordinate system, where each position 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) [23].

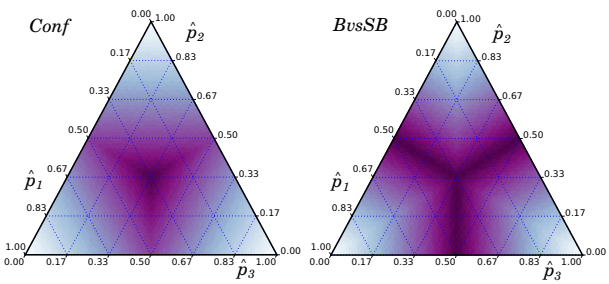


**Figure 2.** Ternary heatmap plot of the usefulness of confidence-based (Conf) and Best-vs-Second-Best (BvsSB) sampling. Dark color indicates high usefulness of a posterior in that barycentric coordinate system.

In the next section, we propose our method, which combines all three influence factors in a decision-theoretic way. Then, we visualize the behavior of McPAL (without the density weight) 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) |
| $\hat{\vec{p}} = \vec{k}/n$ | - | 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 |
| $\vec{l} = (l_1, \ldots, l_C) \in \mathbb{N}^C$ | - | Vector representing the number of hypothetic labels per class ($\sum l_i = m$) |

**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 [17]. 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 ($\vec{k} = n \cdot \hat{\vec{p}}$). We use the generalized multinomial coefficient for non-integer arguments containing the $\Gamma$ function by Legendre [20].

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

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

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 [16] 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 $\vec{p} = \hat{\vec{p}}$ and the variance decreases by increasing $n = \sum k_i$.

Given a performance measure like accuracy, a Bayesian optimal classifier [16] 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.

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

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

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

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

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

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

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

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

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

$$\vec{k} = \text{cl}\left(\vec{x} \mid \mathcal{L}\right); \quad k_i = \sum_{\{(\vec{x'},y')\in\mathcal{L}:\ y'=i\}} K(\vec{x},\vec{x'}) \tag{12}$$

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

We determine the performance gain in Eq. 14 by the difference between the expected performance considering $m$ new labels and the expected current performance. The latter is simply calculated as in Eq. 9, the more general expected performance (see Eq. 15) considers multiple possibilities of a labeling. 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, after receiving a labeling $\vec{l}$, the classifier output 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 the current label candidate $\vec{x}$ (similarity of $\vec{x}$ and $\vec{x'}$ should be 1 to be exact). However, as we only select one instance for labeling at each step, this effect is negligible. Finally, we divide the gain by $m$ to have the average gain per label acquisition.

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

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

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. Achieving 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 get rid of numerical integration, it is sufficient to simplify the expected performance, as the expected current performance is a special case of the former (see Eq. 17ff.).

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

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

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

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

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

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

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.

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

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

The first and second factors 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

performance, a value within $[0, 1]$.

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

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

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

$$\text{expPerf}\left(\vec{k}, 0\right) = \sum_{\vec{\imath}} \left( \prod_{j=\sum(k_i+1)}^{\left(\sum(k_i+1)+\left(\sum l_i\right)+\left(\sum d_i\right)\right)-1} \frac{1}{j} \right) \cdot \prod \left( \prod_{j=k_i+1}^{k_i+l_i+d_i} j \right) \cdot \frac{\Gamma\left(\left(\sum l_i\right)+1\right)}{\prod\left(\Gamma\left(l_i+1\right)\right)} \tag{35}$$

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

### 3.4 Characteristics of McPAL

As briefly discussed in Sec. 3.1, there are different ways to combine the posterior estimates $\hat{\vec{p}}$ 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 $\hat{\vec{p}}$ but also includes the reliability $n = \sum k_i$, which is summarized in the frequency vector $\vec{k} = n \cdot \hat{\vec{p}}$. 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.

The number of hypothetical label acquisitions $M$ in the neighborhood of a labeling candidate is bounded by the globally available budget. In the beginning, it is sufficient to have $M = 1$, as one instance has the highest average benefit for the classification task. Over time, we need more hypothetical labels to achieve this benefit. In our experiments, it was sufficient to set $M = 2$. Applications with more labels should adjust the $M$ to greater values accordingly.

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.

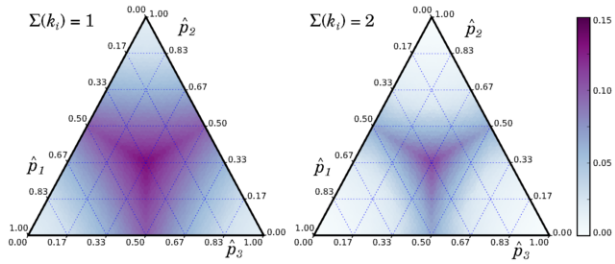


**Figure 3.** Ternary plot for performance gain for situations with $n = \sum k_i = 1$ (left) and $n = 2$ (right).

## 4 EVALUATION

The goals of our evaluation are twofold: on the one hand, we show the advantage of combining 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 tested on six datasets, labeling instances successively until the available budget of $b = 60$ label acquisitions has been exhausted. 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 of misclassification cost across all splits. Additionally, we compared each algorithm on all datasets against our method McPAL to determine if our method is significantly better. Therefore, we used a Wilcoxon signed rank test [27] at a p-value of $0.05$ and performed the Hommel procedure [10] to prevent the results from errors induced by multiple testing.

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 ($x^* =$

$\arg\min_{x\in\mathcal{U}} \max_{y\in Y} \hat{p}_y$) [11]. The next approach uses the shannon entropy to model the uncertainty of an instance (`Entr`) [13]. *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 [12, 13, 14]. *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. [13] 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 [19], namely the Parzen window classifier (PWC) [3], and a probabilistic variant of the k-nearest-neighbor classifier (pKNN, with $k = 9$; received good results for our classification tasks (between 3 and 9 classes)) proposed by Jain and Kapoor [11]. 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 | 846 | 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 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].

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

## 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. Thereby, we aim to measure the importance of the different influence factors *posterior*, *reliability*, and *impact*. In addition to the original `McPAL` algorithm, we show variants that exclude information either (1) about the reliability by normalizing the $\vec{k}$ vector to $\sum \vec{k} = n = 1$ (denoted `w/o reliability`), or (2) about 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`), or (3) about 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 impact 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 late in the learning process in terms of classification error. Even on the iris dataset `Conf`, `BvsSB` and `VoI` struggle to perform better than random selection.

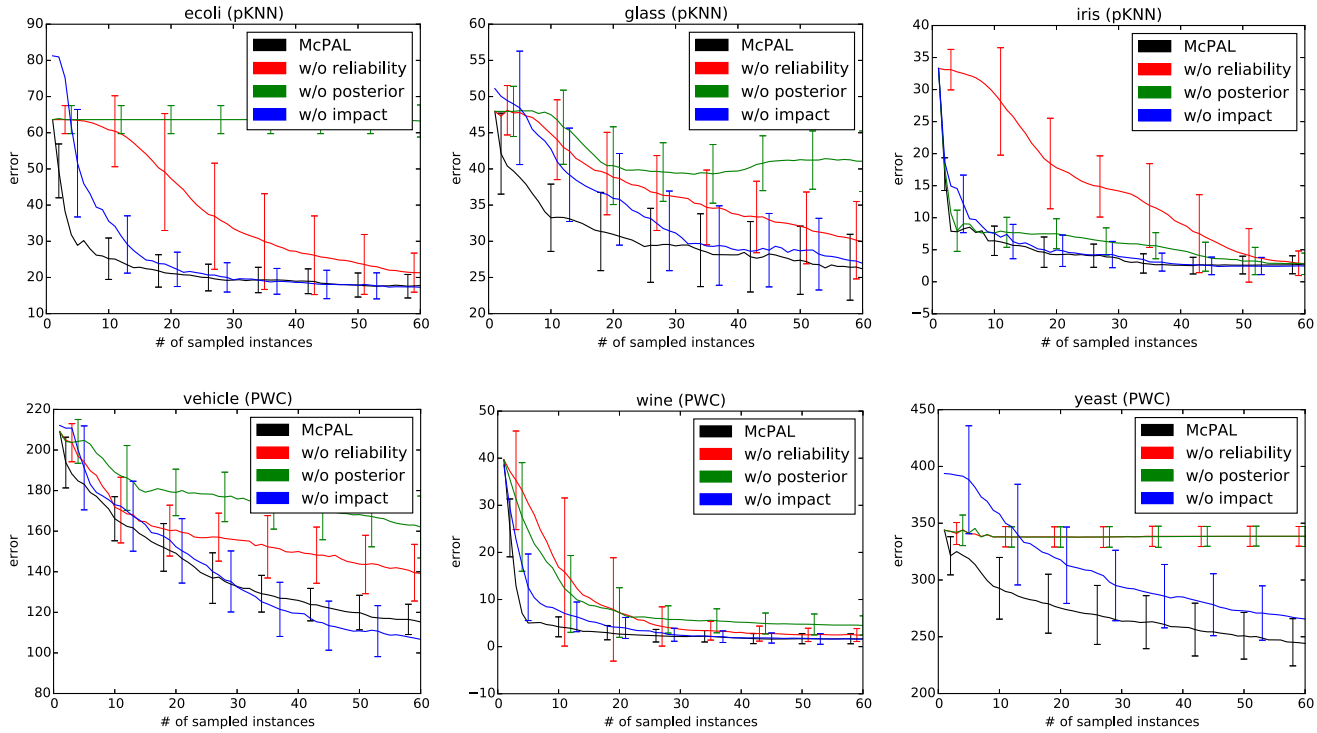


**Figure 4.** Learning curves of mean misclassification cost (including standard deviation as error bars) different variants of the McPAL algorithm on all six datasets. The upper plots show results from the pKNN classifier, the lower ones with the PWC.

**Table 3.** Mean execution time for each algorithm for choosing one instance for labeling on the specified dataset in $s$ (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 |

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 reviewers, Michael Hanke, Alex Waite and our colleague Pawel Matuszyk for all discussions. Ternary plots are generated with python-ternary [9].

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

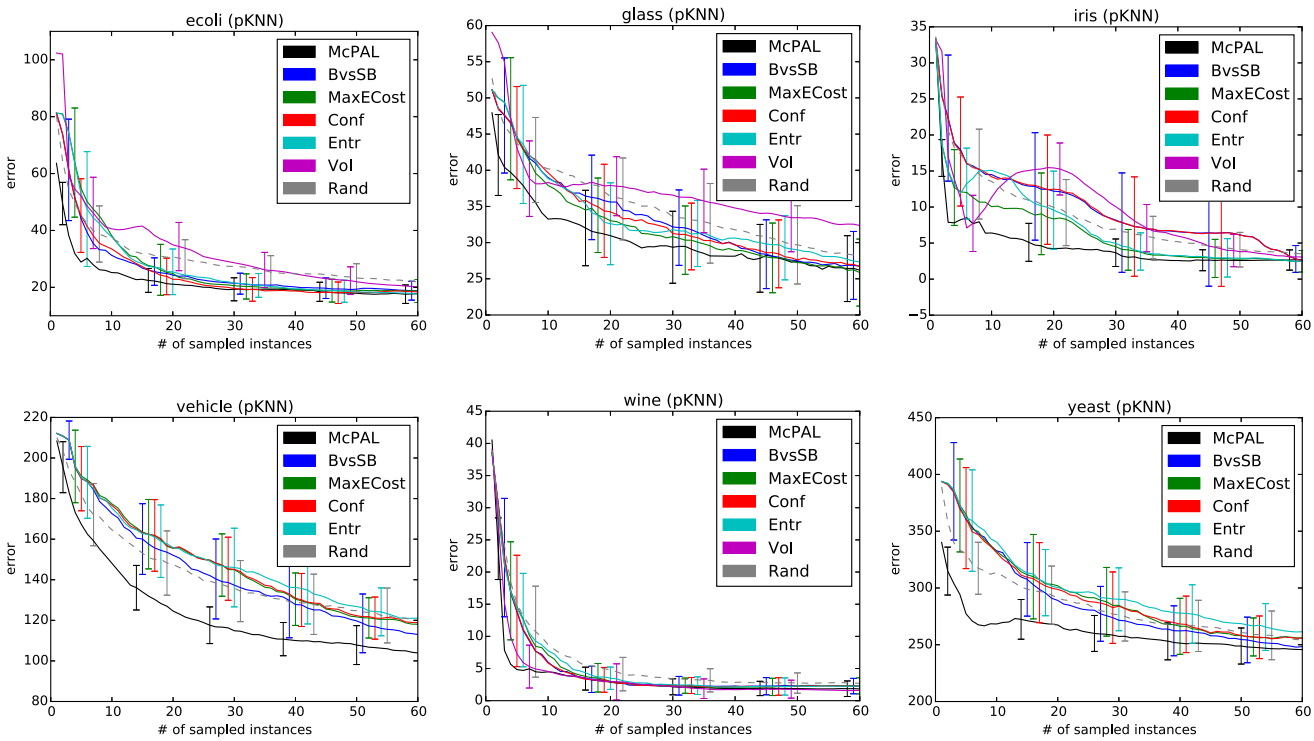


**Figure 5.** Learning curves of mean misclassification cost (including standard deviation as error bars) of McPAL and its competitors on all six datasets using the pKNN classifier.

**Table 4.** Mean misclassification cost and its standard deviation of the all algorithms on all 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. Results showing significant superiority of McPAL against other algorithms are indicated with *.

| 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 *Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pp. 13–18, (2013).

[6] Husheng Guo and Wenjian Wang, 'An active learning-based svm multi-class 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*, **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] Gerhard Hommel, 'A stage-wise rejective multiple test procedure based on a modified bonferroni test', *Biometrika*, **75**, 383–386, (2010).

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

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

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

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

[15] 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).

[16] 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), 449–476, (2015).

[17] 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*, pp. 1049–1050. IOS Press, (2014).

[18] 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.

[19] 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)*, pp. 841–848, (2001).

[20] 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.

[21] 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.

[22] 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).

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

[24] 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), September 1-4, 2009, Redondo Beach, California, USA*, eds., Yolanda Gil and Natasha Fridman Noy, pp. 105–112. ACM, (2009).

[25] 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).

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

[27] Frank Wilcoxon, 'Individual comparisons by ranking methods', *Biometrics bulletin*, **1**(6), 80–83, (1945).

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

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

[30] 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).