Combining active learning suggestions

Alasdair Tran Corresp., [1, 2] , **Cheng Soon Ong** [1, 3] , **Christian Wolf** [4, 5]

[1] Research School of Computer Science, Australian National University, Canberra, ACT, Australia

[2] Data to Decisions Cooperative Research Centre, Adelaide, SA, Australia

[3] Machine Learning Research Group, Data61, CSIRO, Canberra, ACT, Australia

[4] Research School of Astronomy and Astrophysics, Australian National University, Canberra, ACT, Australia

[5] ARC Centre of Excellence for All-sky Astrophysics (CAASTRO), Australia

Corresponding Author: Alasdair Tran
Email address: alasdair.tran@anu.edu.au

We study the problem of combining active learning suggestions to identify informative training examples by empirically comparing methods on benchmark datasets. Many active learning heuristics for classification problems have been proposed to help us pick which instance to annotate next. But what is the optimal heuristic for a particular source of data? Motivated by the success of methods that combine predictors, we combine active learners with bandit algorithms and rank aggregation methods. We demonstrate that a combination of active learners outperforms passive learning in large benchmark datasets and removes the need to pick a particular active learner a priori. We discuss challenges to finding good rewards for bandit approaches and show that rank aggregation performs well.

Combining Active Learning Suggestions

Alasdair Tran[1, 2], **Cheng Soon Ong**[1, 3], **and Christian Wolf**[4, 5]

[1]**Research School of Computer Science, Australian National University**
[2]**Data to Decisions Cooperative Research Centre, Australia**
[3]**Machine Learning Research Group, Data61, CSIRO, Australia**
[4]**Research School of Astronomy and Astrophysics, Australian National University**
[5]**ARC Centre of Excellence for All-sky Astrophysics (CAASTRO)**

Corresponding author:
Alasdair Tran[1]

Email address: alasdair.tran@anu.edu.au

ABSTRACT

We study the problem of combining active learning suggestions to identify informative training examples by empirically comparing methods on benchmark datasets. Many active learning heuristics for classification problems have been proposed to help us pick which instance to annotate next. But what is the best heuristic for a particular source of data? Motivated by the success of methods that combine predictors, we combine active learners with bandit algorithms and rank aggregation methods. We demonstrate that a combination of active learners outperforms passive learning in large benchmark datasets and removes the need to pick a particular active learner a priori. We discuss challenges to finding good rewards for bandit approaches and show that rank aggregation performs well.

1 INTRODUCTION

Recent advances in sensors and scientific instruments have led to an increasing use of machine learning techniques to manage the data deluge. Supervised learning has become a widely used paradigm in many big data applications. This relies on building a training set of labeled examples, which is time-consuming as it requires manual annotation from human experts.

The most common approach to producing a training set is passive learning, where we randomly select an instance from a large pool of unlabeled data to annotate, and we continue doing this until the training set reaches a certain size or until the classifier makes sufficiently good predictions. Depending on how the underlying data is distributed, this process can be quite inefficient. Alternatively we can exploit the current set of labeled data to identify more informative unlabeled examples to annotate. For instance we can pick examples near the decision boundary of the classifier, where the class probability estimates are uncertain (i.e. we are still unsure which class the example belongs to).

Many active learning heuristics have been developed to reduce the labeling bottleneck without sacrificing the classifier performance. These heuristics actively choose the most informative examples to be labeled based on the predicted class probabilities. Section 2 describes two families of algorithms in detail: uncertainty sampling and version space reduction.

In this paper, we present a survey of how we can combine suggestions from various active learning heuristics. In supervised learning, combining predictors is a well-studied problem. Many techniques such as AdaBoost (Freund and Schapire, 1996) (which averages predictions from a set of models) and decision trees (Breiman et al., 1984) (which select one model for making predictions in each region of the input space) have been shown to perform better than just using a single model. Inspired by this success, we propose to combine active learning suggestions with bandit and rank aggregation methods in Section 3.

The use of bandit algorithms to combine active learners has been studied before (Baram et al., 2004; Hsu and Lin, 2015). Borda count, a simple rank aggregation method, has been used in the context of multi-task learning for linguistic annotations (Reichart et al., 2008), where we have one active learner selecting examples to improve the performance of multiple related tasks (e.g. part-of-speech tagging and name entity recognition). Borda count has also been used in multi-label learning (Reyes et al., 2018) to combine uncertainty information from multiple labels. As far as we know, other aggregation methods have

not been explored and our work is the first time that social choice theory is used to rank and aggregate suggestions from multiple active learners.

This paper makes the following two main contributions:

1. We empirically compare 4 bandit and 3 rank aggregation algorithms in the context of combining active learning heuristics. We apply these algorithms to 11 benchmark datasets from the UCI Machine Learning Repository (Lichman, 2013) and a large dataset from the Sloan Digital Sky Survey (SDSS) (Alam et al., 2015). The experimental setup and discussion are described from Section 4 to 6.

2. We propose two metrics for evaluation: the mean posterior balanced accuracy (MPBA) and the strength of an algorithm. The MPBA extends the metric proposed in Brodersen et al. (2010) from the binary to the multi-class setting. This is an accuracy measure that takes class imbalance into account. The strength measure is a variation on the deficiency measure used in Baram et al. (2004) which evaluates the performance of an active learner or combiner, relative to passive learning. The main difference between our measure and that of Baram et al. (2004) is that ours assigns a higher number for better active learning methods and ensures that it is upper-bounded by 1 for easier comparison across datasets.

2 OVERVIEW OF ACTIVE LEARNING

In this paper we consider the binary and multiclass classification settings where we would like to learn a classifier h, which is a function that maps some feature space $\mathcal{X} \subseteq \mathbb{R}^d$ to a probability distribution over a finite label space $\mathcal{Y}$:

$$h : \mathcal{X} \to p(\mathcal{Y}) \tag{1}$$

In other words, we require that the classifier produces class probability estimates for each unlabeled example. For instance, in logistic regression with only two classes, i.e. $\mathcal{Y} = \{0,1\}$, we can model the probability that an object with feature vector $\boldsymbol{x}$ belongs to the positive class with

$$h(\boldsymbol{x};\boldsymbol{\theta}) = \mathbb{P}(y = 1 \,|\, \boldsymbol{x};\boldsymbol{\theta}) = \frac{1}{1+e^{-\boldsymbol{\theta}^T\boldsymbol{x}}} \tag{2}$$

and the optimal weight vector $\boldsymbol{\theta}$ is learned in training. We can further consider kernel logistic regression, where the feature space $\mathcal{X}$ is the feature space corresponding to a given kernel, allowing for non-linear decision functions.

In active learning, we use the class probability estimates from a trained classifier to estimate a score of informativeness for each unlabeled example. In pool-based active learning, where we select an object from a pool of unlabeled examples at each time step, we require that some objects have already been labeled. In practice this normally means that we label a small random sample at the beginning. These become the labeled training set $\mathcal{L}_T \subset \mathcal{X} \times \mathcal{Y}$ and the rest form the unlabeled set $\mathcal{U} \subset \mathcal{X}$.

Now consider the problem of choosing the next example in $\mathcal{U}$ for querying. Labeling can be a very expensive task, because it requires using expensive equipment or human experts to manually examine each object. Thus we want to be smart in choosing the next example. This motivates us to come up with a rule $s(\boldsymbol{x};h)$ that gives each unlabeled example a score based only on their feature vector $\boldsymbol{x}$ and the current classifier h. Recall that the classifier produces $p(\mathcal{Y})$, a probability estimate for each class. We use these probability estimates from the classifier over the unlabeled examples to calculate the scores:

$$s : p(\mathcal{Y}) \to \mathbb{R} \tag{3}$$

The value of $s(\boldsymbol{x};h)$ indicates the informativeness of example $\boldsymbol{x}$, where bigger is better. We would then label the example with the largest value of $s(\boldsymbol{x};h)$. This will be our active learning rule r:

$$r(\mathcal{U};h) = \arg\max_{\boldsymbol{x}\in\mathcal{U}} s(\boldsymbol{x};h) \tag{4}$$

Algorithm 1 outlines the standard pool-based active learning setting.

Input: unlabeled set $\mathcal{U}$, labeled training set $\mathcal{L}_T$, classifier $h(\boldsymbol{x})$, and active learner $r(\mathcal{U};h)$.

repeat

Select the most informative candidate $\boldsymbol{x}_*$ from $\mathcal{U}$ using the active learning rule $r(\mathcal{U};h)$.
Ask the expert to label $\boldsymbol{x}_*$. Call the label y_*.
Add the newly labeled example to the training set: $\mathcal{L}_T \leftarrow \mathcal{L}_T \cup \{(\boldsymbol{x}_*, y_*)\}$.
Remove the newly labeled example from the unlabeled set: $\mathcal{U} \leftarrow \mathcal{U} \setminus \{\boldsymbol{x}_*\}$.
Retrain the classifier $h(\boldsymbol{x})$ using $\mathcal{L}_T$.

until we have enough training examples.

Algorithm 1: The pool-based active learning algorithm.

Coming up with an optimal rule is itself a difficult problem, but there have been many attempts to derive good heuristics. Five common ones, which we shall use in our experiments, are described in Section 2.1 and 2.2. They roughly fall into two categories: uncertainty sampling and version space reduction.

There are also heuristics that involve minimizing the variance or maximizing the classifier certainty of the model (Schein and Ungar, 2007), but they are computationally expensive. For example, in the variance minimization heuristic, the score of a candidate example is the expected reduction in the model variance if that example were in the training set. To compute this reduction, we first need to give the example each of the possible labels, add it to the training set, and update the classifier. This is expensive to run since in each iteration, the classifier needs to be retrained $k \times U$ times, where k is the number of classes and U is the size of the unlabeled pool. There are techniques to speed this up such as using online training or assigning a score to only a small subset of the unlabeled pool. Preliminary experiments showed that these heuristics do not perform as well as the simpler ones (Tran, 2015), so we do not consider them in this paper.

A more comprehensive treatment of these active learning heuristics can be found in Settles (2012).

2.1 Uncertainty Sampling

Lewis and Gale (1994) introduced uncertainty sampling, where we select the instance whose class membership the classifier is least certain about. These tend to be points that are near the decision boundary of the classifier. Perhaps the simplest way to quantify uncertainty is the least confidence heuristic (Culotta and McCallum, 2005), where we pick the candidate whose most likely label the classifier is most uncertain about:

$$r_{LC}(\mathcal{U};h) = \arg\max_{\boldsymbol{x}\in\mathcal{U}} \left\{ -\max_{y\in\mathcal{Y}} p(y|\boldsymbol{x};h) \right\} \tag{5}$$

where $p(y|\boldsymbol{x};h)$ is the probability that the object with feature vector $\boldsymbol{x}$ belongs to class y under classifier h. For consistency, we have flipped the sign of the score function so that the candidate with the highest score is picked.

A second option is to calculate the entropy (Shannon, 1948), which measures the amount of information needed to encode a distribution. Intuitively, the closer the class probabilities of an object are to a uniform distribution, the higher its entropy will be. This gives us the heuristic of picking the candidate with the highest entropy of the distribution over the classes:

$$r_{HE}(\mathcal{U};h) = \arg\max_{\boldsymbol{x}\in\mathcal{U}} \left\{ -\sum_{y\in\mathcal{Y}} p(y|\boldsymbol{x};h) \log \left[p(y|\boldsymbol{x};h) \right] \right\} \tag{6}$$

As a third option we can pick the candidate with the smallest margin, which is defined as the difference between the two highest class probabilities (Scheffer et al., 2001):

$$r_{SM}(\mathcal{U};h) = \arg\max_{\boldsymbol{x}\in\mathcal{U}} \left\{ -\left(\max_{y\in\mathcal{Y}} p(y|\boldsymbol{x};h) - \max_{z\in\mathcal{Y}\setminus\{y^*\}} p(z|\boldsymbol{x};h) \right) \right\} \tag{7}$$

where $y^* = \arg\max_{y\in\mathcal{Y}} p(z|\boldsymbol{x};h)$ and we again flip the sign of the score function. Since the sum of all probabilities must be 1, the smaller the margin is, the harder it is to differentiate between the two most likely labels.

An extension to the above three heuristics is to weight the score with the information density so that we give more importance to instances in regions of high density:

$$s_{ID}(\mathcal{U};h) = \left(\frac{1}{U}\sum_{k=1}^{E}\text{sim}(\boldsymbol{x},\boldsymbol{x}^{(k)})\right)s(\boldsymbol{x};h) \tag{8}$$

where h is the classifier, $s(\boldsymbol{x};h)$ is the original score function of the instance with feature vector $\boldsymbol{x}$, U is the size of the unlabeled pool, and $\text{sim}(\boldsymbol{x},\boldsymbol{x}^{(k)})$ is the similarity between $\boldsymbol{x}$ and another instance $\boldsymbol{x}^{(k)}$ using the Gaussian kernel with parameter γ:

$$\text{sim}(\boldsymbol{x},\boldsymbol{x}^{(k)}) = \exp(\gamma||\boldsymbol{x}-\boldsymbol{x}^{(k)}||^2) \tag{9}$$

The information density weighting was proposed by Settles and Craven (2008) to discourage the active learner from picking outliers. Although the class membership of outliers might be uncertain, knowing their labels would probably not affect the classifier performance on the data as a whole.

2.2 Version Space Reduction

Instead of focusing on the uncertainty of individual predictions, we could instead try to constrain the size of the version space, thus allowing the search for the optimal classifier to be more precise. The version space is defined as the set of all possible classifiers that are consistent with the current training set. To quantify the size of this space, we can train a committee of B classifiers, $\mathcal{B} = \{h_1, h_2, ..., h_B\}$, and measure the disagreement among the members of the committee about an object's class membership. Ideally, each member should be as different from the others as possible but still be in the version space (Melville and Mooney, 2004). In order to have this diversity, we give each member only a subset of the training examples. Since there might not be enough training data, we need to use bootstrapping and select samples with replacement. Hence this method is often called Query by Bagging (QBB).

One way to measure the level of disagreement is to calculate the margin using the class probabilities estimated by the committee (Melville and Mooney, 2004):

$$r_{QBBM}(\mathcal{U};h) = \arg\max_{\boldsymbol{x}\in\mathcal{U}}\left\{-\left(\max_{y\in\mathcal{Y}}p(y|\boldsymbol{x};\mathcal{B}) - \max_{z\in\mathcal{Y}\setminus\{y^*\}}p(z|\boldsymbol{x};\mathcal{B})\right)\right\} \tag{10}$$

where

$$y^* = \arg\max_{y\in\mathcal{Y}}p(z|\boldsymbol{x};\mathcal{B}) \tag{11}$$

$$p(z|\boldsymbol{x};\mathcal{B}) = \frac{1}{B}\sum_{b\in\mathcal{B}}p(y|\boldsymbol{x};h_b) \tag{12}$$

This looks similar to one of the uncertainty sampling heuristics, except now we use $p(y|\boldsymbol{x};\mathcal{B})$ instead of $p(y|\boldsymbol{x};h)$. That is, we first average out the class probabilities predicted by the members before minimizing the margin. McCallum and Nigam (1998) offered an alternative disagreement measure which involves picking the candidate with the largest mean Kullback-Leibler (KL) divergence from the average:

$$r_{QBBKL}(\mathcal{U};h) = \arg\max_{\boldsymbol{x}\in\mathcal{U}}\left\{\frac{1}{B}\sum_{b=1}^{B}D_{\text{KL}}(p_b\|p_{\mathcal{B}})\right\} \tag{13}$$

where $D_{\text{KL}}(p_b\|p_{\mathcal{B}})$ is the KL divergence from $p_{\mathcal{B}}$ (the probability distribution that is averaged across the committee $\mathcal{B}$), to p_b (the distribution predicted by a member $b \in \mathcal{B}$):

$$D_{\text{KL}}(p_b\|p_{\mathcal{B}}) = \sum_{y\in\mathcal{Y}}p(y|\boldsymbol{x};h_b)\ln\frac{p(y|\boldsymbol{x};h_b)}{p(y|\boldsymbol{x};\mathcal{B})} \tag{14}$$

For convenience, we summarize the five heuristics discussed above in Table 1.

Table 1. Summary of active learning heuristics used in our experiments. Notations: $p(y|\boldsymbol{x};h)$ is the probability of that an object with feature vector $\boldsymbol{x}$ has label y under classifier h, $\mathcal{B}$ is the set of B classifiers $\{h_1, h_2, ..., h_B\}$, $\mathcal{Y}$ is the set of possible labels, y^* is the most certain label, $\mathcal{U}$ is the set of unlabeled instances, $D_{\mathrm{KL}}(p\|q)$ is the Kullback-Leibler divergence of p from q, and $p_{\mathcal{B}}$ is the class distribution averaged across classifiers in $\mathcal{B}$. For consistency, with heuristics that use minimization, we flip the sign of the score so that we can always take the argmax to get the best candidate.

Abbreviation	Heuristic	Objective Function
CONFIDENCE	Least Confidence	$\arg\max_{\boldsymbol{x}\in\mathcal{U}} \left\{-\max_{y\in\mathcal{Y}} p(y|\boldsymbol{x};h)\right\}$
ENTROPY	Highest Entropy	$\arg\max_{\boldsymbol{x}\in\mathcal{U}} \left\{-\sum_{y\in\mathcal{Y}} p(y|\boldsymbol{x};h)\log\left[p(y|\boldsymbol{x};h)\right]\right\}$
MARGIN	Smallest Margin	$\arg\max_{\boldsymbol{x}\in\mathcal{U}} \left\{-\left(\max_{y\in\mathcal{Y}} p(y|\boldsymbol{x};h) - \max_{z\in\mathcal{Y}\setminus\{y^*\}} p(z|\boldsymbol{x};h)\right)\right\}$
QBB-MARGIN	Smallest QBB Margin	$\arg\max_{\boldsymbol{x}\in\mathcal{U}} \left\{-\left(\max_{y\in\mathcal{Y}} p(y|\boldsymbol{x};\mathcal{B}) - \max_{z\in\mathcal{Y}\setminus\{y^*\}} p(z|\boldsymbol{x};\mathcal{B})\right)\right\}$
QBB-KL	Largest QBB KL	$\arg\max_{\boldsymbol{x}\in\mathcal{U}} \left\{\frac{1}{B}\sum_{b=1}^{B} D_{\mathrm{KL}}(p_b\|p_{\mathcal{B}})\right\}$

3 COMBINING SUGGESTIONS

Out of the five heuristics discussed, which one should we use in practice when we would like to apply active learning to a particular problem? There have been some attempts in the literature to do a theoretical analysis of their performance. Proofs are however scarce, and when there is one available, they normally only work under restrictive assumptions. For example, Freund et al. (1997) showed that the query by committee algorithm (a slight variant of our two QBB heuristics) guarantees an exponential decrease in the prediction error with the training size, but only when there is no noise. In general, whether any of these heuristics is guaranteed to beat passive learning is still an open question.

Even though we do not know which one is the best, we can still combine suggestions from all of the heuristics. This can be thought of as the problem of prediction with expert advice, where each expert is an active learning heuristic. In this paper we explore two different approaches: we can either consider the advice of only one expert at each time step (with bandit algorithms), or we can aggregate the advice of all the experts (with social choice theory).

3.1 Combining Suggestions with Bandit Theory

First let us turn our attention to the multi-armed bandit problem in probability theory (Berry and Fristedt, 1985). The colorful name originates from the situation where a gambler stands in front of a slot machine with R levers. When pulled, each lever gives out a reward according to some unknown distribution. The goal of the game is to come up with a strategy that can maximize the gambler's lifetime rewards. In the context of active learning, each lever is a heuristic with a different ability to identify the candidate whose labeling information is most valuable.

The main problem in multi-armed bandits is the trade-off between exploring random heuristics and exploiting the best heuristic so far. There are many situations in which we find our previously held beliefs to be completely wrong. By always exploiting, we could miss out on the best heuristic. On the other hand, if we explore too much, it could take us a long time to reach the desired accuracy.

Bandit algorithms do not need to know the internal workings of the heuristics, but only the reward received from using any of them. At each time step, we receive a reward from a heuristic, and based on the history of all the rewards, the bandit algorithm can decide on which heuristic to pick next. Formally, we need to learn the function

$$b : (J_{\mathcal{R}} \times [0,1])^n \to J_{\mathcal{R}} \tag{15}$$

where b is the bandit algorithm, $[0,1]$ is a normalized reward between 0 and 1, $J_{\mathcal{R}}$ is the index set over the set of heuristics $\mathcal{R}$, and n is the time horizon.

What would be an appropriate reward w in this setting? We propose using the incremental increase in the performance of the test set after the candidate is added to the training set. This, of course, means that

we need to keep a separate labeled test set around, just for the purpose of computing the rewards. We could, as is common practice in machine learning, use cross validation or bootstrap on $\mathcal{L}_T$ to estimate the generalization performance. However for simplicity of presentation we use a separate test set $\mathcal{L}_S$.

Figure 1 and Algorithm 2 outline how bandits can be used in pool-based active learning. The only difference between the bandit algorithms lies in the SELECT function that selects which heuristic to use, and the UPDATE function that updates the algorithm's selection parameters when receiving a new reward.

There have been some attempts to combine active learning suggestions in the literature. Baram et al. (2004) used the EXP4 multi-armed bandit algorithm to automate the selection process. They proposed a reward called the classification entropy maximization, which can be shown to grow at a similar rate to the true accuracy in binary classification with support vector machines (SVMs). We will not compare our results directly with those in Baram et al. (2004) since we would like to evaluate algorithms that can work with both binary and multi-class classification. Our experiments also use logistic regression which produces probability estimates directly, rather than SVMs which can only produce unnormalized scores. Hsu and Lin (2015) studied an improved version of EXP4, called EXP4.P, and used importance weighting to estimate the true classifier performance using only the training set. In this paper, we empirically compare the following four bandit algorithms: Thompson sampling, OC-UCB, kl-UCB, and EXP3++.

3.1.1 Thompson Sampling

The oldest bandit algorithm is Thompson sampling (Thompson, 1933) which solves the exploration-exploitation trade-off from a Bayesian perspective.

Let W_i be the reward of heuristic $r_i \in \mathcal{R}$. Observe that even with the best heuristic, we still might not score perfectly due to having a poor classifier trained on finite data. Conversely, a bad heuristic might be able to pick an informative candidate due to pure luck. Thus there is always a certain level of randomness in the reward received. Let us treat the reward W_i as a normally distributed random variable with mean v_i and variance τ_i^2:

$$(W_i \mid v_i) \sim \mathcal{N}(v_i, \tau_i^2) \tag{16}$$

If we knew both v_i and τ_i^2 for all heuristics, the problem would become trivially easy since we just need to always use the heuristic that has the highest mean reward. In practice, we do not know the true mean of the reward v_i, so let us add a second layer of randomness and assume that the mean itself follows a normal distribution:

$$v_i \sim \mathcal{N}(\mu_i, \sigma_i^2) \tag{17}$$

To make the problem tractable, let us assume that the variance τ_i^2 in the first layer is a known constant. The goal now is to find a good algorithm that can estimate μ_i and σ_i^2.

We start with a prior on μ_i and σ_i^2 for each heuristic r_i. The choice of prior does not usually matter in the long run. Since initially we do not have any information about the performance of each heuristic, the appropriate prior value for μ_i is 0, i.e. there is no evidence (yet) that any of the heuristics offers an improvement to the performance.

In each round, we draw a random sample v_i' from the normal distribution $\mathcal{N}(\mu_i, \sigma_i^2)$ for each i and select heuristic r_* that has the highest sampled value of the mean reward:

$$r_* = \arg\max_i v_i' \tag{18}$$

We then use this heuristic to select the object that is deemed to be the most informative, add it to the training set, and retrain the classifier. Next we use the updated classifier to predict the labels of objects in the test set. Let w be the reward observed. We now have a new piece of information that we can use to update our prior belief about the mean μ_* and the variance σ_*^2 of the mean reward. Using Bayes' theorem, we can show that the posterior distribution of the mean reward remains normal,

$$(v_* \mid W_* = w) \sim \mathcal{N}(\mu_*', \sigma_*'^2) \tag{19}$$

with the following new mean and variance:

$$\mu_*' = \frac{\mu_* \tau_*^2 + w\sigma_*^2}{\sigma_*^2 + \tau_*^2} \qquad \sigma_*'^2 = \frac{\sigma_*^2 \tau_*^2}{\sigma_*^2 + \tau_*^2} \tag{20}$$

Algorithm 3 summarizes the SELECT and UPDATE functions used in Thompson sampling.

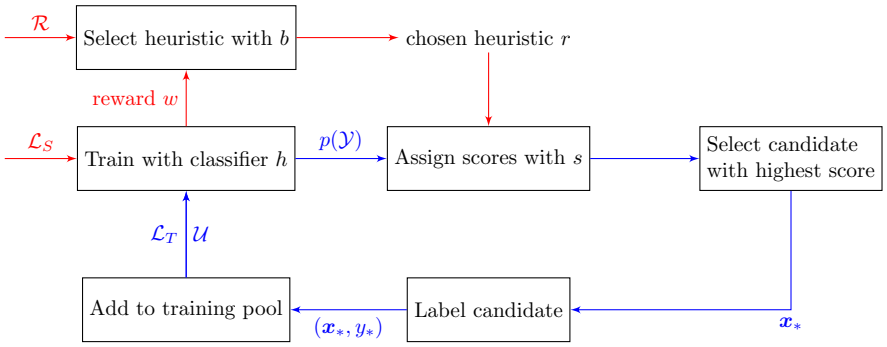

Figure 1. Active learning pipeline with bandit algorithms. We need to collect rewards w from the test set $\mathcal{L}_S$ in order to decide which heuristic to choose at each time step. This routine is indicated by the red arrows. Notations: $\mathcal{R}$ is the set of heuristics $\{r_1, ..., r_R\}$, $\mathcal{L}_T$ is the training set, $\mathcal{L}_S$ is the test set, $\mathcal{U}$ is the unlabeled set, and $p(\mathcal{Y})$ is the predicted class probabilities on the unlabeled data $\mathcal{U}$.

Input: unlabeled set $\mathcal{U}$, labeled training set $\mathcal{L}_T$, labeled test set $\mathcal{L}_S$, classifier h, desired training size n, set of active learning heuristics $\mathcal{R}$, and bandit algorithm b with two functions SELECT and UPDATE.

while $|\mathcal{L}_T| < n$ **do**
- Select a heuristic $r_* \in \mathcal{R}$ according to SELECT.
- Select the most informative candidate $\boldsymbol{x}_*$ from $\mathcal{U}$ using the chosen heuristic $r_*(\mathcal{U}; h)$.
- Ask the expert to label $\boldsymbol{x}_*$. Call the label y_*.
- Add the newly labeled example to the training set: $\mathcal{L}_T \leftarrow \mathcal{L}_T \cup \{(\boldsymbol{x}_*, y_*)\}$.
- Remove the newly labeled example from the unlabeled set: $\mathcal{U} \leftarrow \mathcal{U} \setminus \{\boldsymbol{x}_*\}$.
- Retrain the classifier $h(\boldsymbol{x})$ using $\mathcal{L}_T$.
- Run the updated classifier on the test set $\mathcal{L}_S$ to compute the increase in the performance w.
- Update the parameters of b with UPDATE(w).

end

Algorithm 2: Pool-based active learning with bandit theory. Note that in addition to the set of active learning heuristics $\mathcal{R}$ and the test set $\mathcal{L}_S$, some bandit algorithms also need to know n, the maximum size of the training set, in the advance.

function SELECT*()*
- **for** $i \in \{1, 2, ..., R\}$ **do**
  - $v_i' \leftarrow$ draw a sample from $\mathcal{N}(\mu_i, \sigma_i^2)$
- **end**
- Select the heuristic with the highest sampled value: $r_* \leftarrow \arg\max_i v_i'$

function UPDATE*(w)*
- $\mu_* \leftarrow \dfrac{\mu_* \tau_*^2 + w\sigma_*^2}{\sigma_*^2 + \tau_*^2}$ $\qquad$ $\sigma_*^2 \leftarrow \dfrac{\sigma_*^2 \tau_*^2}{\sigma_*^2 + \tau_*^2}$

Algorithm 3: Thompson sampling with normally distributed rewards. Notations: $\mathcal{R}$ is the set of R heuristics, μ is the mean parameter of the average reward, σ^2 is the variance parameter of the average reward, τ^2 is the known variance parameter of the reward, and w is the actual reward received.

3.1.2 Upper Confidence Bounds

Next we consider the Upper Confidence Bound (UCB) algorithms which use the principle of "optimism in the face of uncertainty". In choosing which heuristic to use, we first estimate the upper bound of the reward (that is, we make an optimistic guess) and pick the one with the highest bound. If our guess turns out to be wrong, the upper bound of the chosen heuristic will decrease, making it less likely to get selected in the next iteration.

There are many different algorithms in the UCB family, e.g. UCB1-TUNED & UCB2 (Auer et al., 2002a), V-UCB (Audibert et al., 2009), OC-UCB (Lattimore, 2015), and kl-UCB (Cappé et al., 2013). They differ only in the way the upper bound is calculated. In this paper, we only consider the last two. In Optimally Confident UCB (OC-UCB), Lattimore (2015) suggests that we pick the heuristic that maximizes the following upper bound:

$$r_* = \arg\max_i \left(\overline{w_i} + \sqrt{\frac{\alpha}{T_i(t)} \ln\left(\frac{\psi n}{t}\right)} \right) \tag{21}$$

where $\overline{w_i}$ is the average of the rewards from r_i that we have observed so far, t is the time step, $T_i(t)$ is the number times we have selected heuristic r_i before step t, and n is the maximum number of steps that we are going to take. There are two tunable parameters, α and ψ, which the author suggests setting to 3 and 2, respectively.

In kl-UCB, Cappé et al. (2013) suggest that we can instead consider the KL-divergence between the distribution of the current estimated reward and that of the upper bound. In the case of normally distributed rewards with known variance σ^2, the chosen heuristic would be

$$r_* = \arg\max_i \left(\overline{w_i} + \sqrt{2\sigma^2 \frac{\ln\left(T_i(t)\right)}{t}} \right) \tag{22}$$

Algorithms 4 and 5 summarize these two UCB approaches. Note that the size of the reward w is not used in UPDATE(w) of UCB, except to select the best arm.

function SELECT()
  $r_* \leftarrow \arg\max_i \overline{w_i} + \sqrt{\frac{3}{T_i(t)} \ln\left(\frac{2n}{t}\right)}$
function UPDATE(w)
  $t \leftarrow t + 1$
  $T_*(t) \leftarrow T_*(t-1) + 1$

Algorithm 4: Optimally Confident UCB. Notations: n is the time horizon (maximum number of time steps), t is the current time step, $T_i(t)$ counts of how many times heuristic i has been selected before step t, w is the reward received, and $\overline{w_i}$ is the average of the rewards from r_i so far.

function SELECT()
  $r_* \leftarrow \arg\max_i \overline{w_i} + \sqrt{2\sigma^2 \frac{\ln\left(T_i(t)\right)}{t}}$
function UPDATE(w)
  $t \leftarrow t + 1$
  $T_*(t) \leftarrow T_*(t-1) + 1$

Algorithm 5: kl-UCB with normally distributed rewards. Notations: σ is the variance of the rewards, t is the current time step, $T_i(t)$ counts of how many times heuristic i has been selected before step t, w is the reward received, and $\overline{w_i}$ is the average of the rewards from r_i so far.

3.1.3 EXP3++

The exponential-weight algorithm for exploration and exploitation (EXP3) was first developed by Auer et al. (2002b) to solve the non-stochastic bandit problem where we make no statistical assumptions about the reward distribution. This is also often known as the adversarial setting, where we have an adversary who generates an arbitrary sequence of rewards for each heuristic in advance. Like Thompson sampling, the algorithm samples from a probability distribution at each step to pick a heuristic. Here however, we construct the distribution with exponential weighting (hence the name EXP3). We shall test Seldin and Slivkins (2014)'s EXP3++ algorithm (see Algorithm 6). This is a generalization of the original EXP3 and it has been shown to perform well in both the stochastic (where the reward of each heuristic follows an unknown reward distribution) and the adversarial regime.

function SELECT*()*

$\beta = \frac{1}{2}\sqrt{\frac{\ln R}{tR}}$

for $i \in \{1,2,...,R\}$ **do**

$\xi_i = \frac{18\ln(t)^2}{t\min(1,\frac{1}{t}(L_i - \min(L)))^2}$

$\varepsilon_i = \min(\frac{1}{2R}, \beta, \xi_i)$

$\rho_i = \frac{e^{-\beta * L_i}}{\sum_j e^{-\beta * L_j}}$

end

$r_* \leftarrow$ draw a sample from $\mathcal{R}$ with probability distribution ρ.

function UPDATE*(w)*

$t \leftarrow t+1$

$T_*(t) \leftarrow T_*(t-1)+1$

$L_* \leftarrow \frac{L_* + (1-w)}{(1-\sum_j \varepsilon_j)W_* + \varepsilon_*}$

Algorithm 6: EXP3++ algorithm. Notations: $\mathcal{R}$ is the set of R heuristics, t is the current time step, β is a parameter used to weight the heuristics for selection, ξ_i and ε_i are used to compute the loss L_i, ρ is the distribution from which a heuristic is sampled, and w is the reward received.

3.2 Combining Suggestions with Social Choice Theory

A drawback of the above bandit methods is that at each iteration, we could only use one suggestion from one particular heuristic. EXP4 and EXP4.P algorithms can take advice from all heuristics by maintaining a weight on each of them. However, being a bandit method, they require designing a reward scheme. If the reward is the performance on a test set, we would need to keep around a separate subset of the data, which is expensive and sometimes impossible to obtain in practice. This leads us to social choice theory, which can combine suggestions like EXP4 and EXP4.P, while not needing the concept of a reward. Originally developed by political scientists like Nicolas de Condorcet and Jean-Charles de Borda, this field of study is concerned with how we aggregate preferences of a group of people to determine, for example, the winner in an election (List, 2013). It has the nice property that everyone (or in our context, every active learning heuristic) has a voice.

For each heuristic, we assign a score to every candidate with the score function $s(\boldsymbol{x}, h)$ like before. We are neither interested in the actual raw scores nor the candidate with the highest score. Instead, we only need a ranking of the candidates, which is achieved by a function $k(s, \mathcal{U})$ that provides a ranking of the unlabeled examples according to their scores. For example, k could assign the candidate with the highest score a rank of 1, the next best candidate a rank of 2, and so on. An aggregation function c will then combine all the rankings into a combined ranking,

$$c : \sigma(J_{\mathcal{U}})^R \rightarrow \sigma(J_{\mathcal{U}}) \tag{23}$$

where $\sigma(J_{\mathcal{U}})$ is a permutation over the index set of the unlabeled pool $\mathcal{U}$ and R is the number of heuristics. From these we can pick the highest-ranked candidate to annotate. See Table 2 for an example.

Table 2. An example of how to convert raw scores into a ranking.

Score	$s(\boldsymbol{x};h)$	0.1	0.9	0.3	0.8
Rank	$k(s,\mathcal{U})$	4	1	3	2

The main difference between this approach and the bandit algorithms is that we do not consider the reward history when combining the rankings. Here each heuristic is assumed to always have an equal weight. A possible extension, which is not considered in this paper, is to use the past performance to re-weight the heuristics before aggregating at each step. Figure 2 and Algorithm 7 provide an overview of how social choice theory is used in pool-based active learning.

The central question in social choice theory is how we can come up with a good preference aggregation rule. We shall examine three aggregation rules: Borda count, the geometric mean, and the Schulze method.

In the simplest approach, Borda count, we assign an integer point to each candidate. The lowest-ranked candidate receives a point of 1, and each candidate receives one more point than the candidate below. To aggregate, we simply add up all the points each candidate receives from every heuristic. The candidate with the most points is declared the winner and is to be labeled next. We can think of Borda count, then, as ranking the candidate according to the arithmetic mean.

An alternative approach is to use the geometric mean, where instead of adding up the points, we multiply them. Bedő and Ong (2016) showed that the geometric mean maximizes the Spearman correlation between the ranks. Note that this method requires the ranks to be scaled so that they lie strictly between 0 and 1. This can be achieved by simply dividing the ranks by $(U+1)$, where U is the number of candidates.

The third approach we consider is the Schulze method (Schulze, 2011). Out of the three methods considered, this is the only one that fulfills the Condorcet criterion, i.e. the winner chosen by the algorithm is also the winner when compared individually with each of the other candidates. However, the Schulze method is more computationally intensive since it requires examining all pairs of candidates. First we compute the number of heuristics that prefer candidate x_i to candidate x_j, for all possible pairs (x_i,x_j). Let us call this $d(x_i,x_j)$. Let us also define a path from candidate x_i to x_j as the sequence of candidates, $\{x_i,x_1,x_2,...,x_j\}$, that starts with x_i and ends with x_j, where, as we move along the path, the number of heuristics that prefer the current candidate over the next candidate must be strictly decreasing. Intuitively, the path is the rank of a subset of candidates, where x_i is the highest-ranked candidate and x_j is at the lowest-ranked.

Associated with each path is a strength p, which is the minimum of $d(x_i,x_j)$ for all consecutive x_i and x_j along the path. The core part of the algorithm involves finding the path of the maximal strength from each candidate to every other. Let us call $p(x_i,x_j)$ the strength of strongest path between x_i and x_j. Candidate x_i is a potential winner if $p(x_i,x_j) \geq p(x_j,x_i)$ for all other x_j. This problem has a similar flavor to the problem of finding the shortest path. In fact, the implementation uses a variant of the Floyd–Warshall algorithm to find the strongest path. This is the most efficient implementation that we know of, taking cubic time in the number of candidates.

We end this section with a small illustration of how the three aggregation algorithms work in Table 3.

4 EXPERIMENTAL PROTOCOL

We use 11 classification datasets taken from the UCI Machine Learning Repository[1] (Lichman, 2013), with a large multiclass classification dataset which we extracted from the SDSS project[2] (Alam et al., 2015). The code for the experiments can be found on our GitHub repository[3]. Table 4 shows the size and the number of classes in each dataset, along with the proportion of the samples belonging to the majority class and the maximum achievable performance using logistic regression. These datasets were chosen such that we have an equal number of binary and multiclass datasets, and a mixture of small and large datasets.

[1] https://archive.ics.uci.edu/ml/
[2] https://doi.org/10.5281/zenodo.58500
[3] https://github.com/chengsoonong/mclass-sky

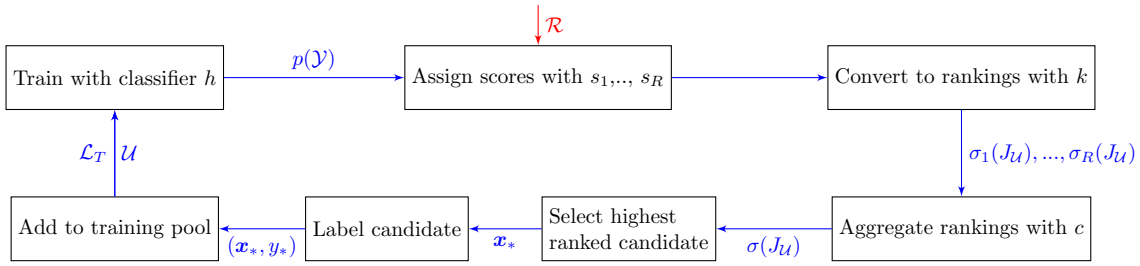

Figure 2. Active learning pipeline with rank aggregation methods. Unlike the bandit pipeline, there is only one cycle in which we aggregate information from all heuristics. Additional notation: $\sigma(J_\mathcal{U})$ is a permutation (i.e. rank) on the index set of the unlabeled data.

Input: unlabeled set $\mathcal{U}$, labeled training set $\mathcal{L}_T$, classifier h, set of active learning suggestions $\mathcal{R}$, ranking function k, and rank aggregator c.

repeat
  for $r \in \mathcal{R}$ **do**
    Rank all the candidates in $\mathcal{U}$ with k.
  end
  Aggregate all the rankings into one ranking using the aggregator c.
  Select the highest-ranked candidate $\boldsymbol{x}_*$ from $\mathcal{U}$.
  Ask the expert to label $\boldsymbol{x}_*$. Call the label y_*.
  Add the newly labeled example to the training set: $\mathcal{L}_T \leftarrow \mathcal{L}_T \cup \{(\boldsymbol{x}_*, y_*)\}$.
  Remove the newly labeled example from the unlabeled set: $\mathcal{U} \leftarrow \mathcal{U} \setminus \{\boldsymbol{x}_*\}$.
  Retrain the classifier $h(\boldsymbol{x})$ using $\mathcal{L}_T$.
until we have enough training examples.

Algorithm 7: Pool-based active learning with social choice theory.

Table 3. An example of how social choice theory algorithms rank candidates by aggregating three heuristics: r_1, r_2, and r_3. There are four candidates in the unlabeled pool: A, B, C, and D.

Heuristic	Ranking
r_1	B A C D
r_2	A C B D
r_3	B D C A

(a) An example of how the three heuristics rank four candidates A, B, C, and D. For instance, heuristic r_1 considers B to be the highest rank candidate, followed by A, C, and D.

Candidate	Borda count	Geometric mean
A	$3+4+1=8$	$3\times4\times1=12$
B	$4+2+4=10$	$4\times2\times4=32$
C	$2+3+2=7$	$2\times3\times2=12$
D	$1+1+3=5$	$1\times1\times3=3$

(b) Aggregated ranking with Borda count and geometric mean. The scores are determined by the relative ranking in each heuristic. For example, A is ranked second by r_1, first by r_1, and last by r_3, thus giving us a score of 3, 4 and 1, respectively. In both methods, candidate B receives the highest aggregated score.

From / To	A	B	C	D
A	–	1	2	2
B	2	–	2	3
C	1	1	–	2
D	2	0	1	–

(c) Aggregated ranking with the Schulze method. The table shows the strongest path strength $p(x_i, x_j)$ between all pairs of candidates. For example, $p(B,D) = 3$ because the path $B \to D$ is the strongest path from B to D, where three heuristics prefer B over D. Candidate B is the winner since $p(B,A) > p(A,B)$, $p(B,C) > p(C,B)$, and $p(B,D) > p(D,B)$.

Table 4. Overview of datasets. The following datasets are from the UCI Machine Learning Repository: glass, ionosphere, iris, magic, miniboone, pageblock, pima, sonar, vehicle, wine, and wpbc. In particular, the vehicle dataset comes from the Turing Institute, Glasgow, Scotland. The sdss dataset was extracted from Data Release 12 of SDSS-III.

Dataset	Size	No. of Classes	No. of Features	Majority Class	Max Performance (MPBA)
glass	214	6	10	33%	65%
ionosphere	351	2	34	64%	89%
iris	150	3	4	33%	90%
magic	19 020	2	11	65%	84%
miniboone	129 596	2	50	72%	88%
pageblock	5 473	5	10	90%	79%
pima	733	2	8	66%	71%
sdss	2 801 002	3	11	61%	90%
sonar	208	2	60	53%	78%
vehicle	846	4	18	26%	81%
wine	178	3	13	40%	94%
wpbc	194	2	34	76%	58%

For each dataset, we use Scikit-learn (Pedregosa et al., 2011) to train a logistic regression model using a 10-fold stratified shuffled cross-validation. Here 'stratified' means that the proportion of the classes remains constant in each split. We standardize all features to have zero mean and unit variance. Although all examples have already been labeled, we simulate the active learning task by assuming that certain examples do not have any labels. For each fold, the unlabeled pool size is 70% of data up to a maximum of 10,000 examples, and the test pool consists of the remaining examples up to a maximum of 20,000. We assume all test examples are labeled. We initialize the classifier by labeling 10 random instances and using them as the initial training set. The heuristics are fast enough such that we can assign a score to every unlabeled instance at every time step. We use logistic regression with a Gaussian kernel approximation and an L2 regularizer. In the binary case, the loss function is

$$L = \frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta} + C\sum_{i=1}^{n} \ln\left(1+\exp(-y_i(\boldsymbol{\theta}^T f(\boldsymbol{x}_i)))\right) \tag{24}$$

where $\boldsymbol{x}_i$ is the feature vector of the ith example, $y_i \in \{0,1\}$ is the label of $\boldsymbol{x}_i$, and n is the training size. The term $\frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta}$ is the regularization term to ensure that the weight vector $\boldsymbol{\theta}$ is not too large, and C is a regularization hyperparameter in $[10^{-6}, 10^{6}]$ which we find using grid search. To speed up the training time while using the Gaussian kernel, we approximate the feature map of the kernel with Random Kitchen Sinks (Rahimi and Recht, 2008), transforming the raw features $\boldsymbol{x}_i$ into a fixed 100-dimensional feature vector $f(\boldsymbol{x}_i)$. In the multiclass case, we use the One-vs-Rest strategy, where for every class we build a binary classifier that determines whether a particular example belongs to that class or not. For the QBB algorithms, we train a committee of 7 classifiers, where each member is given a sample of maximum 100 examples that have already been labeled.

For the bandit algorithms, we use the increase in the mean posterior balanced accuracy (MPBA) on the test set as the reward. The MPBA can be thought of as the expected value of the average recall, where we treat the recall as a random variable that follows a Beta distribution. Compared to the raw accuracy score, this metric takes into account class imbalance. This is because we first calculate the recall in each class and then take the average, thus giving each class an equal weight. Refer to Appendix A for the derivation of the MPBA, which extends Brodersen et al. (2010)'s formula from the binary to the multiclass setting.

In total, we test 17 query strategies. This includes passive learning, 8 active learning heuristics, 5 bandit algorithms, and 3 aggregation methods. The bandit algorithms include the four described in Section 3.1 and a baseline called EXPLORE which simply selects a random heuristic at each time step. In other words, we ignore the rewards and explore 100% of the time. For all bandit and rank aggregation methods, we take advice from six representative experts: PASSIVE, CONFIDENCE, MARGIN, ENTROPY, QBB-MARGIN, and QBB-KL. We have not explored how adding the heuristics with information density weighting to the bandits would impact the performance. See Table 5 for a list of abbreviations associated with the methods.

Given that there are 12 datasets, each with 17 learning curves, we need a measure that can summarize in one number how well a particular heuristic or policy does. Building on Baram et al. (2004)'s deficiency measure, we define the strength of an active learner or a combiner relative to passive learning as

$$\text{Strength}(h;m) = 1 - \frac{\sum_{t=1}^{n}\left(m(\max) - m(h,t)\right)}{\sum_{t=1}^{n}\left(m(\max) - m(\text{passive},t)\right)} \tag{25}$$

where m is a chosen metric (e.g. accuracy rate, MPBA), $m(\max)$ is the best possible performance[4], and $m(h,t)$ is the performance achieved using the first t examples selected by heuristic h. We can think of the summation as the area between the best possible performance line and the learning curve of h. The better the heuristic is, the faster it would approach this maximum line, and thus the smaller the area. Finally, so that we can compare the performance across datasets, we normalize the measure with the area obtained from using just passive learning. Refer to Figure 3 for a visualization of the strength measure.

We evaluate the algorithm performance with two metrics: the accuracy score and the MPBA. The accuracy score is the percentage of instances in the test set where the predicted label matches the true label. If a dataset has a dominant class, then the accuracy score of instances within that class will also

[4]The best possible performance in each trial is obtained by the higher of: 1) the performance achieved by using all the labeled examples in the training set; and 2) the maximum value of the learning curves of all the methods.

Table 5. Summary of active learning heuristics and combiners used in the experiments.

Abbreviation	Type	Description
PASSIVE	Heuristic	Passive learning
CONFIDENCE	Heuristic	Least confidence heuristic
W-CONFIDENCE	Heuristic	Least confidence heuristic with information density weighting
MARGIN	Heuristic	Smallest margin heuristic
W-MARGIN	Heuristic	Smallest margin heuristic with information density weighting
ENTROPY	Heuristic	Highest entropy heuristic
W-ENTROPY	Heuristic	Highest entropy heuristic with information density weighting
QBB-MARGIN	Heuristic	Smallest QBB margin heuristic
QBB-KL	Heuristic	Largest QBB KL-divergence heuristic
EXPLORE	Bandit	Bandit algorithm with 100% exploration
THOMPSON	Bandit	Thompson sampling
OCUCB	Bandit	Optimally Confidence UCB algorithm
KLUCB	Bandit	kl-UCB algorithm
EXP3++	Bandit	EXP3++ algorithm
BORDA	Aggregation	Aggregation with Borda count
GEOMETRIC	Aggregation	Aggregation with the geometric mean
SCHULZE	Aggregation	Aggregation with the Schulze method

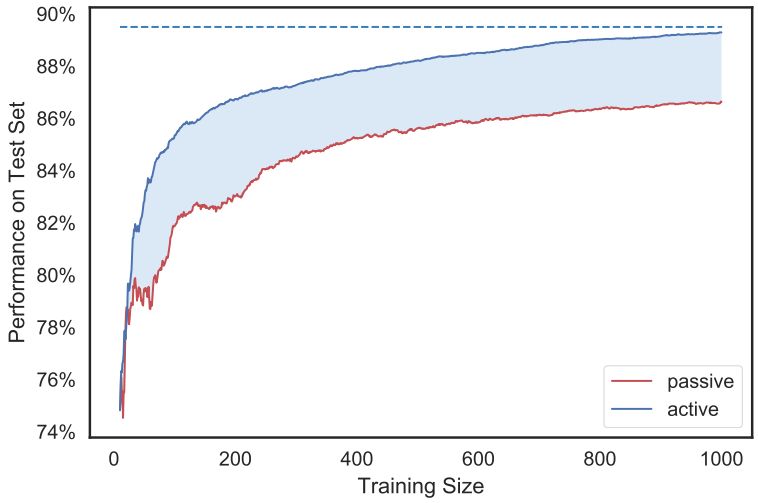

Figure 3. An illustration of the MPBA strength measure. It is proportional to the shaded area between the (red) passive learning curve and the (blue) active learning curve. The bigger the area is, the more the active learner outperforms the passive learner. The top dotted line indicates the maximum performance achieved.

dominate the overall accuracy score. The MPBA, on the other hand, puts an equal weight on each class and thus favors algorithms that can predict the label of all classes equally well.

The heuristics with information density weighting and Thompson sampling have a few additional hyperparameters. To investigate the effect of these hyperparameters, we pick one binary dataset (glass) and one multiclass dataset (ionosphere) to investigate. Both of these are small enough to allow us to iterate through many hyperparameter values quickly. With W-CONFIDENCE, W-MARGIN, and W-ENTROPY, we set γ in the Gaussian kernel to be the inverse of the 95$^{\text{th}}$ percentile of all pairwise distances. This appears to work well, as shown in Figure 4. For THOMPSON, the prior values for μ, σ^2 and the value of τ^2 seem to have little effect on the final performance (see Figure 5). We set the initial μ to 0.5, the initial σ^2 to 0.02, and τ^2 to 0.02.

5 RESULTS

Figures 6 and 7 show the strengths of all methods that we consider, while Figures 8 and 9 provide selected learning curves. Plots for the 6 small datasets with fewer than 500 examples (glass, ionosphere, iris, sonar, wine, and wpbc) are shown in Figures 6 and Figure 8. Plots for the 2 medium-sized datasets (pima and vehicle) and the 4 large datasets (magic, miniboone, pageblocks, and sdss) are shown in Figure 7 and Figure 9. Each figure contains two subfigures, one reporting the raw accuracy score, while the other showing the MPBA score.

Active learning methods generally beat passive learning in four of the six small datasets—glass, ionosphere, iris, and wine. This can be seen by the fact that the boxplots are mostly above the zero line in Figure 6. For sonar and wpbc, the results are mixed—active learning has little to no effect here. The wpbc dataset is particularly noisy—our classifier cannot achieve an MPBA score greater than 60% (see Figure 8). Thus it is not surprising that active learning does not perform well here since there is not much to learn to begin with.

The advantage of active learning becomes more apparent with the larger datasets like magic, miniboone, pageblocks, and sdss. Here there is a visible gap between the passive learning curve and the active learning curve for most methods. For instance, using a simple heuristic such as CONFIDENCE in the pageblocks dataset results in an average MPBA score of 74% after 1,000 examples, while passive learning can only achieves 67% (see Figure 9F).

Out of the 8 active learning heuristics tested, the heuristics with the information density weighting (W-CONFIDENCE, W-MARGIN, and W-ENTROPY) generally perform worse than the ones without the weighting. QBB-KL performs the best in pageblocks while it can barely beat passive learning in other datasets. The remaining heuristics—CONFIDENCE, MARGIN, ENTROPY, and QBB-MARGIN—perform equally well in all datasets.

We find no difference in performance between the bandit algorithms and the rank aggregation methods. Combining active learners does not seem to hurt the performance, even if we include a poorly performing heuristic such as QBB-KL.

For bandit algorithms, it is interesting to note that THOMPSON favors certain heuristics a lot more than others, while the behavior of EXP3++, OCUCB, and KLUCB is almost indistinguishable from EXPLORE, where we explore 100% of the time (see Figure 10). Changing the initial values of μ, σ^2, and τ^2 changes the order of preference slightly, but overall, which heuristics THOMPSON picks seems to correlate with the heuristic performance. For example, as shown in Figure 11, PASSIVE and QBB-KL tend to get chosen less often than others in the ionosphere dataset.

6 DISCUSSION

The experimental results allow us to answer the following questions:

1. **Can active learning beat passive learning?** Yes, active learning can perform much better than passive learning, especially when the unlabeled pool is large (e.g. sdss, miniboone, pageblock). When the unlabeled pool is small, the effect of active learning becomes less apparent, as there are now fewer candidates to choose from. This can be seen in Figure 12, where we show that artificially reducing the unlabeled pool results in a reduction in the final performance. At the same time, having a small test set also makes the gap between the active learning curve and the passive learning curve smaller (see the rightmost subplots of Figure 12). This further contributes to the

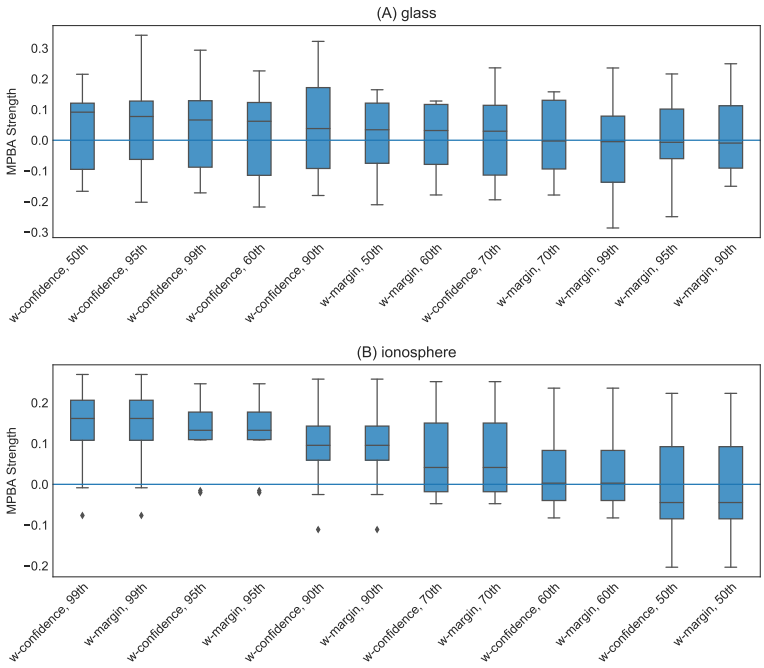

Figure 4. Effect of γ on W-CONFIDENCE and W-MARGIN using the glass and ionosphere datasets. We examine six different values for γ: the 50^{th}, 60^{th}, 70^{th}, 90^{th}, 95^{th}, and 99^{th} percentile of the of the pairwise L1-distance between the data points. For the glass dataset, changing value of γ has minimal effect on the results, while for the ionosphere dataset, using the 90^{th} percentile and above seems to work well.

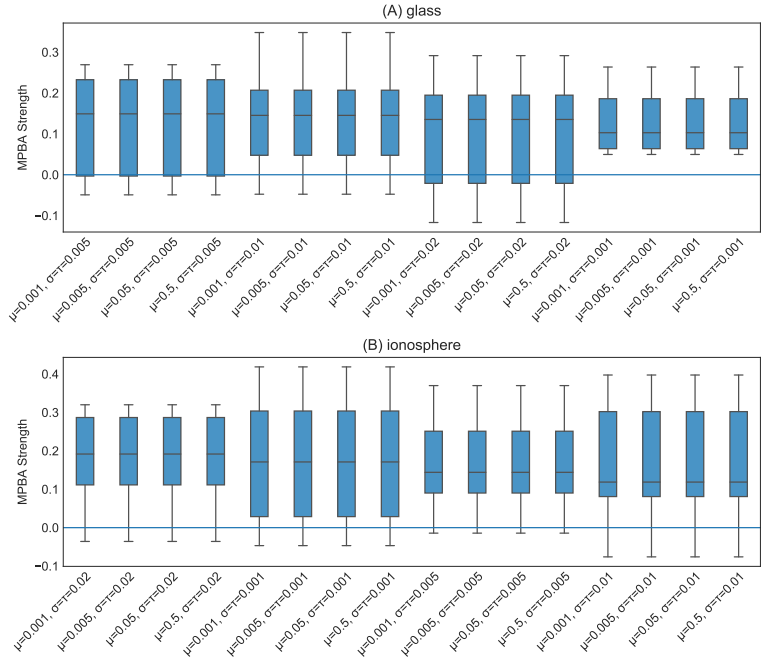

Figure 5. Effect of the initial values of the parameters in THOMPSON. We test 16 combinations of μ, σ^2, and τ^2 on the glass and ionosphere dataset. Varying these values does not seem to affect the final performance.

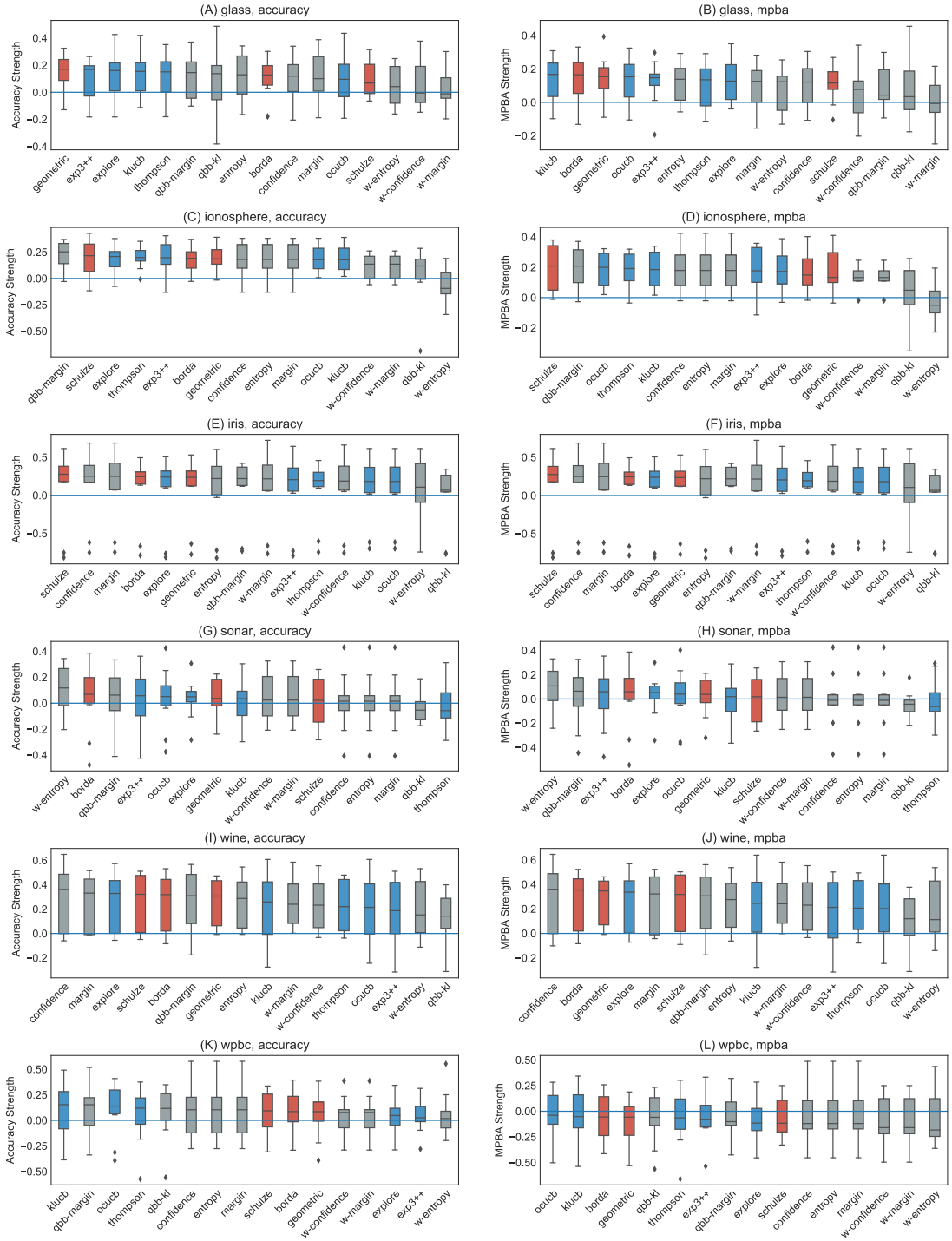

Figure 6. Boxplots of the accuracy and MPBA strength of the 16 active learning strategies, relative to passive learning, using the small datasets (glass, ionosphere, iris, sonar, wine, and wpbc). The more positive the strength is, the better the heuristic/combiner is. Gray boxes represent individual heuristics; blue boxes represent bandit algorithms, and red boxes are for rank aggregation methods. A strategy that is above the zero line is better than passive learning. Each boxplot contains 10 trials. The accuracy score is a simple metric that simply counts up the number of correct predictions. The MPBA score, being the weighted average of the recall and precision, gives an equal representation to each class. The boxes represent the quartiles and the whiskers extend to 1.5 times of the interquartile range.

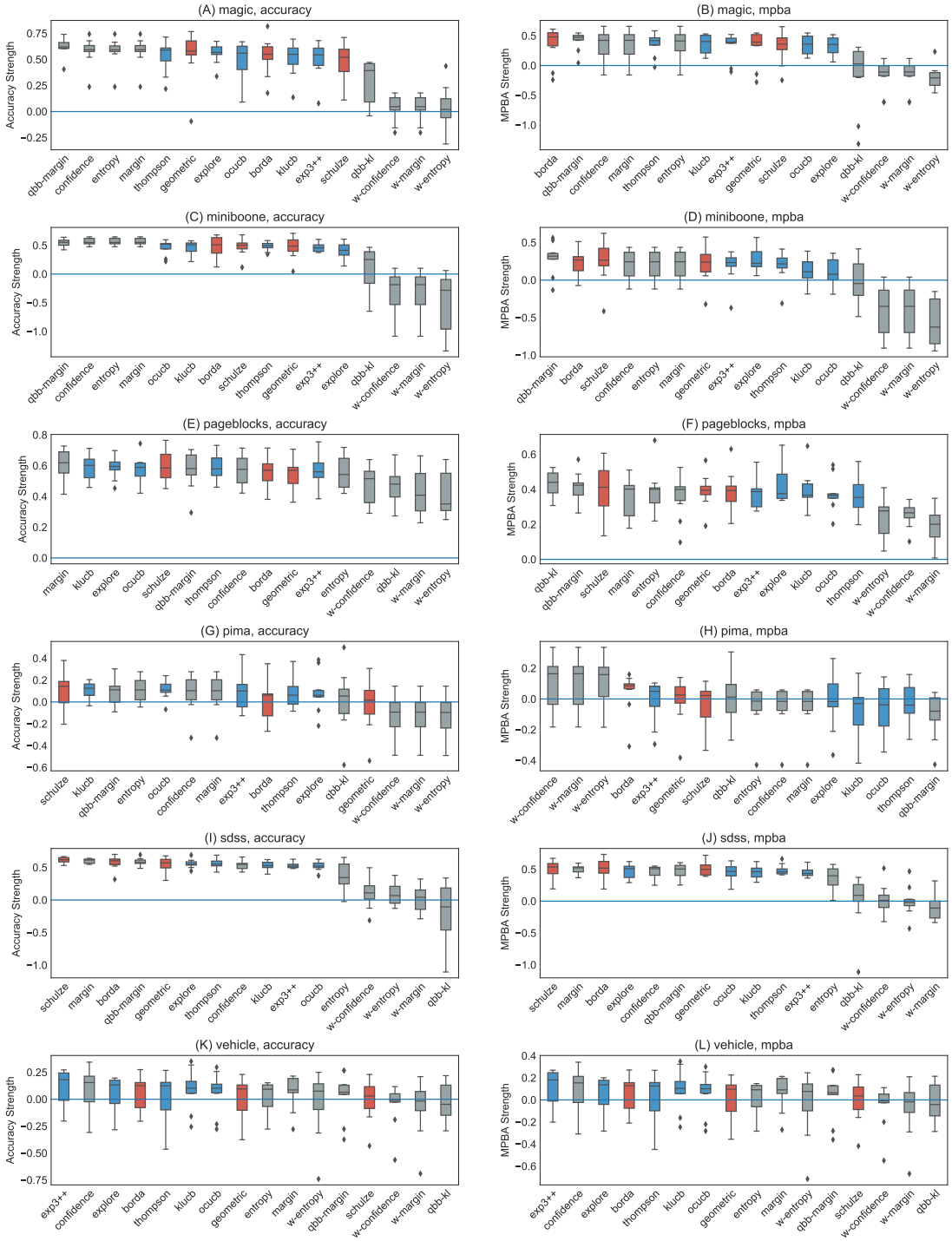

Figure 7. Boxplots of the accuracy and MPBA strength of the 16 active learning strategies, relative to passive learning, using medium to the large datasets (magic, miniboone, pageblocks, pima, sdss, and vehicle). The more positive the strength is, the better the heuristic/combiner is. Gray boxes represent individual heuristics; blue boxes represent bandit algorithms, and red boxes are for rank aggregation methods. A strategy that is above the zero line is better than passive learning. Each boxplot contains 10 trials. The accuracy score is a simple metric that simply counts up the number of correct predictions. The MPBA score, being the weighted average of the recall and precision, gives an equal representation to each class. The boxes represent the quartiles and the whiskers extend to 1.5 times of the interquartile range.

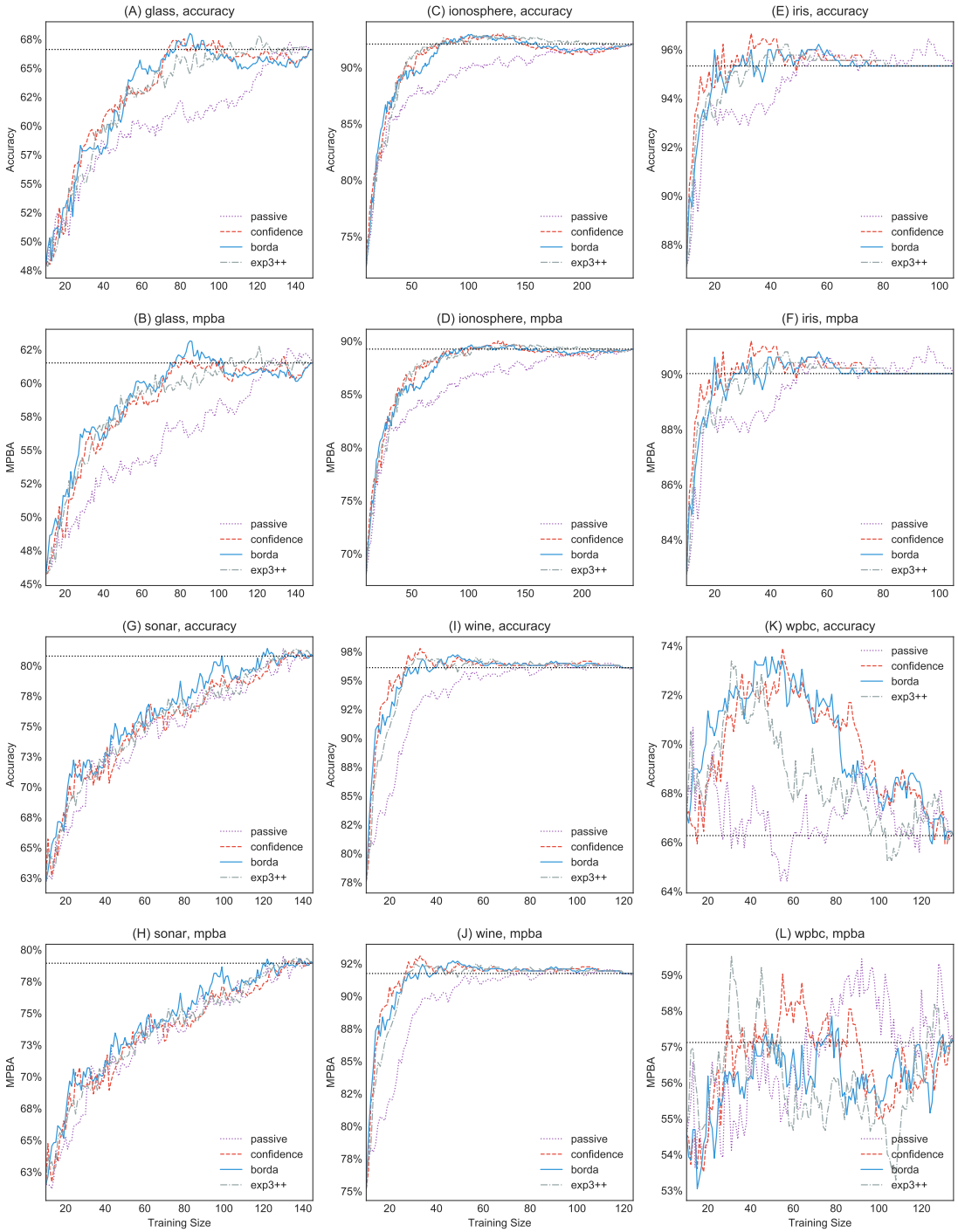

Figure 8. Selected accuracy and MPBA learning curves for the small datasets (glass, ionosphere, iris, sonar, wine, and wpbc). As it would get too cluttered to plot 17 learning curves, we only show the learning curve for PASSIVE, CONFIDENCE, EXP3++, and BORDA. The learning curves are averaged over 10 trials. The dotted horizontal line shows the performance obtained from using the whole training data.

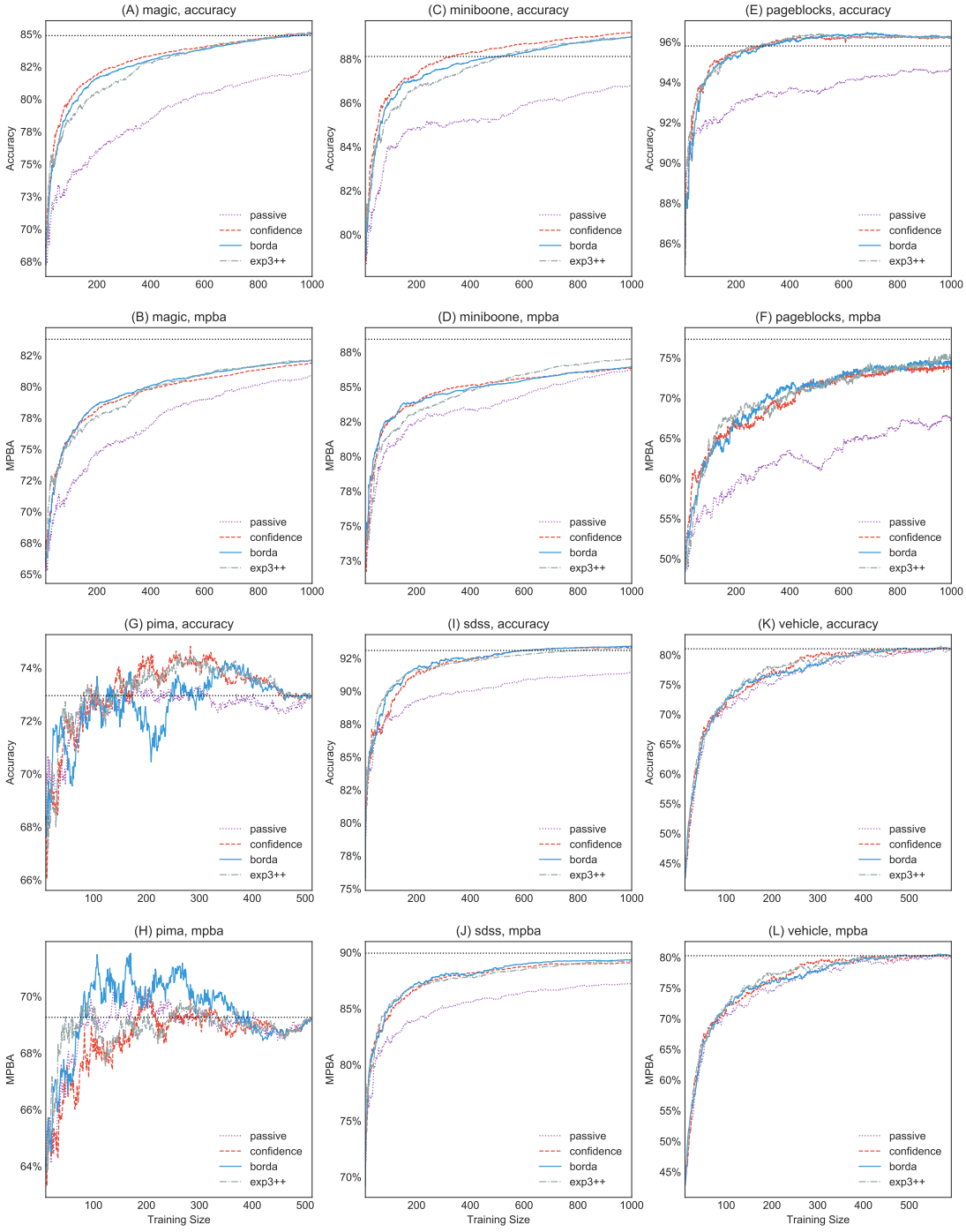

Figure 9. Selected accuracy and MPBA learning curves for the medium to large datasets (magic, miniboone, pageblocks, pima, sdss, and vehicle). As it would get too cluttered to plot 17 learning curves, we only show the learning curve for PASSIVE, CONFIDENCE, EXP3++, and BORDA. The learning curves are averaged over 10 trials. The dotted horizontal line shows the performance obtained from using the whole training data.

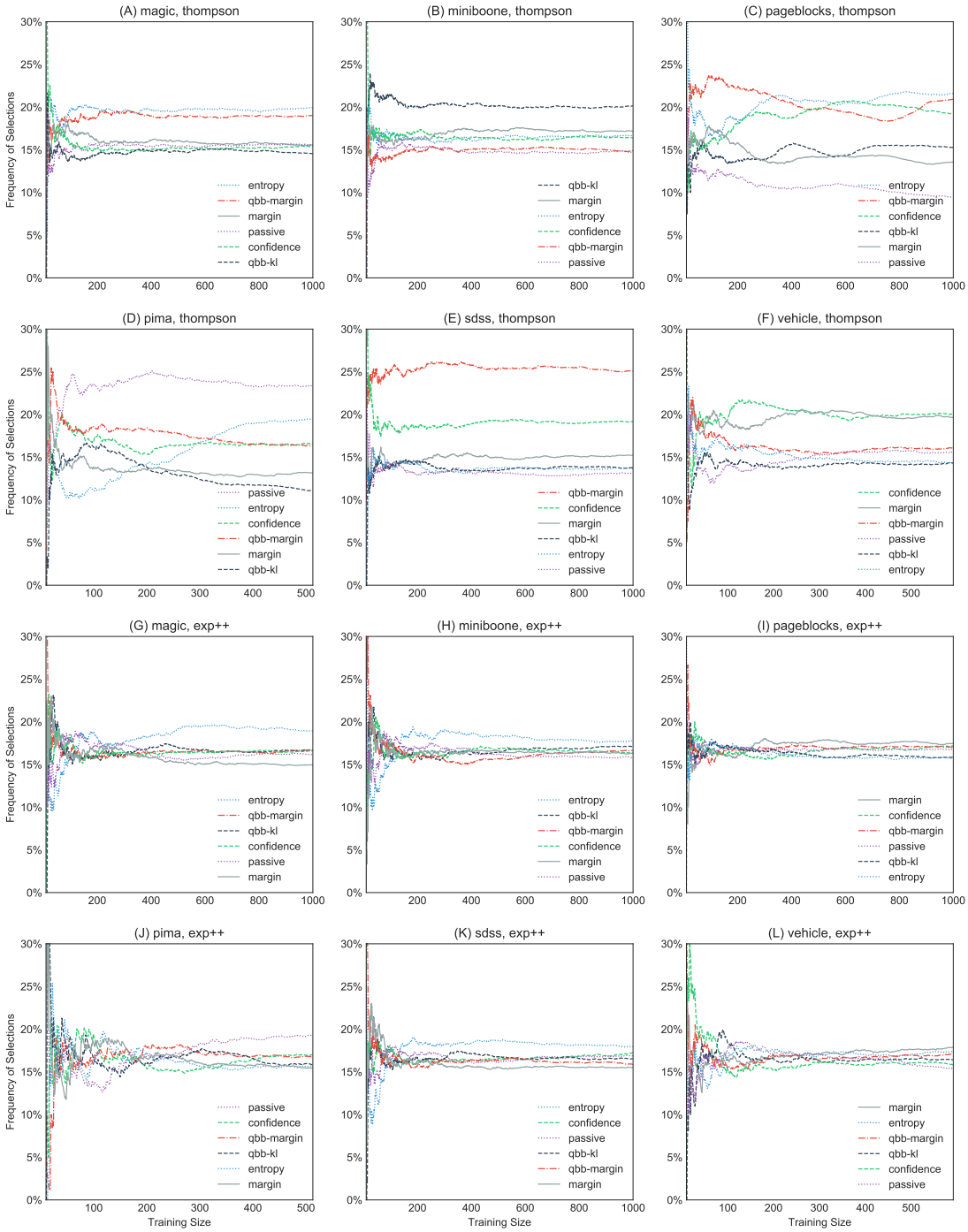

Figure 10. Selection frequencies of heuristics in THOMPSON and EXP3++, with the large datasets (magic, miniboone, pageblocks, pima, sdss, and vehicle). The plots show how often each of the heuristics gets selected over time. The selection frequencies are averaged over 10 trials. THOMPSON favors certain heuristics more strongly than others. In contrast, EXP3++ favors uniform exploration more, sampling each heuristic with roughly equal weights. The plots for OCUCB and KLUCB are not shown here, but they are similar to EXP3++.

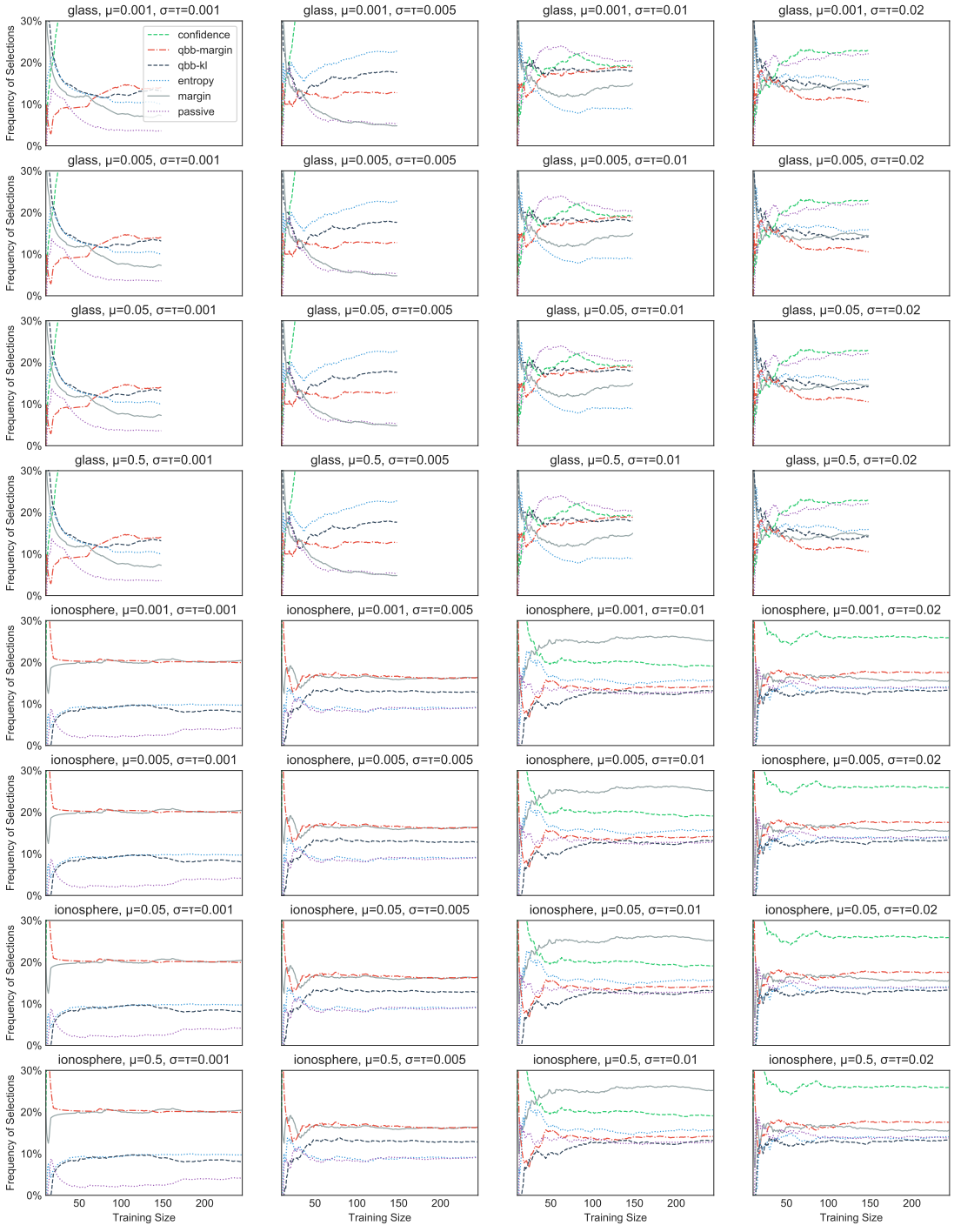

Figure 11. The effect of the initial values of the parameters in THOMPSON on the heuristic selection frequencies. We test 16 combinations of μ, σ^2, and τ^2 on the glass and ionosphere dataset. Which heuristics THOMPSON picks seems to correlate with the heuristic performance. For example, in ionosphere, PASSIVE (the dotted purple line) and QBB-KL (the dashed dark blue line) tend to get picked less often than others.

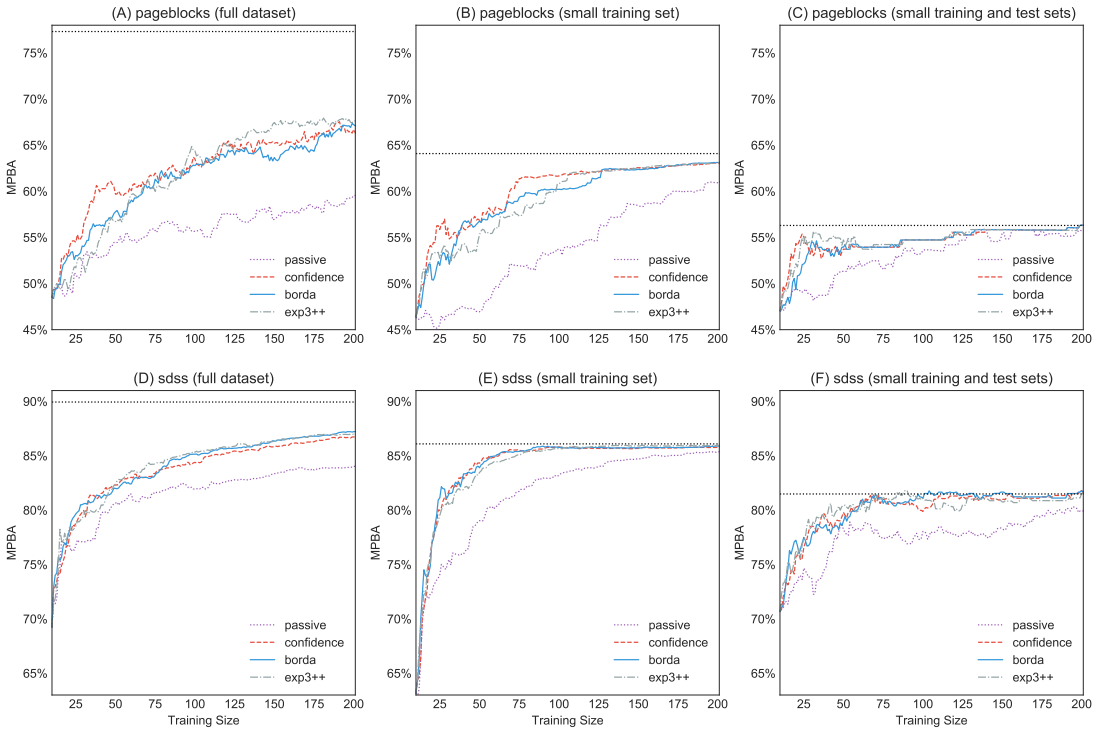

Figure 12. Effect of the pool size on the learning curves. We pick two large datasets—pageblocks and sdss—to investigate how the size of the pool affects the performance. The leftmost figures (A and D) are the original learning curves from Figures 9F and 9J (we only show the first 200 examples so that all figures have the same scale). For the middle figures, we use the same test pool, but the unlabeled pool now only has a maximum of 300 candidates. Finally for the rightmost figures, the combined test pool and training pool have a size of 300.

poorer performance on the smaller datasets. In any case, when a dataset is small, we can label everything so active learning is usually not needed.

2. **Can active learning degrade performance?** Yes, there is no guarantee that active learning will always beat passive learning. For example, W-ENTROPY actually slows down the learning in the many datasets. However, this only happens with certain heuristics, like those using the information density weighting.

3. **What is the best single active learning heuristic?** All of CONFIDENCE, MARGIN, ENTROPY, and QBB-MARGIN have a similar performance. However CONFIDENCE is perhaps the simplest to compute and thus is a good default choice in practice.

4. **What are the challenges in using bandit algorithms?**

   (a) Designing a good reward scheme is difficult. This paper uses the increase in the classifier performance as the reward. However this type of reward is non-stationary (i.e. it gets smaller after each step as learning saturates) and the rewards will thus eventually go to zero.

   (b) In practice, we do not have a representative test set that can be used to compute the reward. As a workaround, Hsu and Lin (2015) computed the reward on the training set and then used importance weighting to remove any potential bias. For this to work, we need to ensure that every training example and every active learning suggestion have a non-zero probability of being selected in each step.

   (c) Finally, some bandit algorithms such as Thompson sampling assumes that the reward follows a certain distribution (e.g. Gaussian). However, this assumption is unrealistic.

5. **What are the challenges in using rank aggregation algorithms?**

   (a) We need to compute the scores from all heuristics at every time step. This might not be feasible if there are too many heuristics or if we include heuristics that require a large amount of compute power (e.g. variance minimization).

   (b) The Schulze method uses $O(n^2)$ space, where n is the number of candidates. This might lead to memory issues if we need to rank a large number of candidates from the unlabeled pool.

   (c) Before aggregating the rankings, we throw away the score magnitudes, which could cause a loss of information.

   (d) Unlike bandit algorithms, all of the rank aggregators always give each heuristic an equal weight.

6. **Which method should I use in practice to combine active learners?** Since there is no difference in performance between various combiners, we recommend using a simple rank aggregator like Borda count or geometric mean if we do not want to select a heuristic a priori. Rank aggregators do not need a notion of a reward—we simply give all suggestions an equal weight when combining. Thus we neither need to a keep a separate test set, nor do we need to worry about designing a good reward scheme.

Our investigation has a few limitations. Firstly, we empirically compare algorithms that only work with single-label classification problems. Nowadays, many problems require multi-label learning, in which each example is allowed to be in more than one class. Our methods can be extended to work with multi-label datasets with the following modifications. We first need a multi-label classifier. This can be as simple as a collection of binary classifiers, each of which produces the probability that an example belongs to a particular class. For each class, we can use an active learning heuristic to assign a score to each unlabeled example as before. However now we need to aggregate the scores among the classes. As suggested by Reyes et al. (2018), we can use any aggregation method like Borda count to combine these scores. In effect, the multi-label learning problem adds an extra layer of aggregation into the pipeline.

Another limitation of our methods is that our active learning methods are myopic. That is, in each iteration, we only pick one instance to give to a human expert for labeling. In many practical applications like astronomy, batch-mode active learning is preferred, as it is much more cost efficient to obtain multiple labels simultaneously. One naive extension is to simply choose the m highest ranked objects using our current methods. However, it is possible to have two unlabeled objects whose class membership we are currently uncertain about, but because they have very similar feature vectors, labeling only one of them would allow us to predict the label of the other one easily. More sophisticated batch-mode active learning approaches have been proposed to take into account other factors such as the diversity of a batch and the representiveness of each batch example. These approaches include looking at the angles between hyperplanes in support vector machines (Brinker, 2003), using cluster analysis (Xu et al., 2007), and using an evolutionary algorithm (Reyes and Ventura, 2018). How to aggregate suggestions from these approaches is an interesting problem for future work.

7 CONCLUSION

In this paper we compared 16 active learning methods with passive learning. Our three main findings are: active learning is better than passive learning; combining active learners does not in general degrade the performance; and social choice theory provides more practical algorithms than bandit theory since we do not need to design a reward scheme.

REFERENCES

Alam, S., Albareti, F. D., Prieto, C. A., Anders, F., Anderson, S. F., Andrews, B. H., Armengaud, E., Éric Aubourg, Bailey, S., Bautista, J. E., Beaton, R. L., Beers, T. C., Bender, C. F., Berlind, A. A., Beutler, F., Bhardwaj, V., Bird, J. C., Bizyaev, D., Blake, C. H., Blanton, M. R., Blomqvist, M., Bochanski, J. J., Bolton, A. S., Bovy, J., Bradley, A. S., Brandt, W. N., Brauer, D. E., Brinkmann, J., Brown, P. J., Brownstein, J. R., Burden, A., Burtin, E., Busca, N. G., Cai, Z., Capozzi, D., Rosell, A. C., Carrera, R., Chen, Y., Chiappini, C., Chojnowski, S. D., Chuang, C., Clerc, N., Comparat, J., Covey, K., Croft, R. A. C., Cuesta, A. J., Cunha, K., da Costa, L. N., Rio, N. D., Davenport, J. R. A., Dawson, K. S., Lee, N. D., Delubac, T., Deshpande, R., Dutra-Ferreira, L., Dwelly, T., Ealet, A., Ebelke, G. L., Edmondson, E. M., Eisenstein, D. J., Escoffier, S., Esposito, M., Fan, X., Fernández-Alvar, E., Feuillet, D., Ak, N. F., Finley, H., Finoguenov, A., Flaherty, K., Fleming, S. W., Font-Ribera, A., Foster, J., Frinchaboy, P. M., Galbraith-Frew, J. G., García-Hernández, D. A., Pérez, A. E. G., Gaulme, P., Ge, J., Génova-Santos, R., Ghezzi, L., Gillespie, B. A., Girardi, L., Goddard, D., Gontcho, S. G. A., Hernández, J. I. G., Grebel, E. K., Grieb, J. N., Grieves, N., Gunn, J. E., Guo, H., Harding, P., Hasselquist, S., Hawley, S. L., Hayden, M., Hearty, F. R., Ho, S., Hogg, D. W., Holley-Bockelmann, K., Holtzman, J. A., Honscheid, K., Huehnerhoff, J., Jiang, L., Johnson, J. A., Kinemuchi, K., Kirkby, D., Kitaura, F., Klaene, M. A., Kneib, J., Koenig, X. P., Lam, C. R., Lan, T., Lang, D., Laurent, P., Goff, J. L., Leauthaud, A., Lee, K., Lee, Y. S., Licquia, T. C., Liu, J., Long, D. C., López-Corredoira, M., Lorenzo-Oliveira, D., Lucatello, S., Lundgren, B., Lupton, R. H., III, C. E. M., Mahadevan, S., Maia, M. A. G., Majewski, S. R., Malanushenko, E., Malanushenko, V., Manchado, A., Manera, M., Mao, Q., Maraston, C., Marchwinski, R. C., Margala, D., Martell, S. L., Martig, M., Masters, K. L., McBride, C. K., McGehee, P. M., McGreer, I. D., McMahon, R. G., Ménard, B., Menzel, M., Merloni, A., Mészáros, S., Miller, A. A., Miralda-Escudé, J., Miyatake, H., Montero-Dorta, A. D., More, S., Morice-Atkinson, X., Morrison, H. L., Muna, D., Myers, A. D., Newman, J. A., Neyrinck, M., Nguyen, D. C., Nichol, R. C., Nidever, D. L., Noterdaeme, P., Nuza, S. E., O'Connell, J. E., O'Connell, R. W., O'Connell, R., Ogando, R. L. C., Olmstead, M. D., Oravetz, A. E., Oravetz, D. J., Osumi, K., Owen, R., Padgett, D. L., Padmanabhan, N., Paegert, M., Palanque-Delabrouille, N., Pan, K., Parejko, J. K., Park, C., Pâris, I., Pattarakijwanich, P., Pellejero-Ibanez, M., Pepper, J., Percival, W. J., Pérez-Fournon, I., Pérez-Ràfols, I., Petitjean, P., Pieri, M. M., Pinsonneault, M. H., de Mello, G. F. P., Prada, F., Prakash, A., Price-Whelan, A. M., Raddick, M. J., Rahman, M., Reid, B. A., Rich, J., Rix, H., Robin, A. C., Rockosi, C. M., Rodrigues, T. S., Rodríguez-Rottes, S., Roe, N. A., Ross, A. J., Ross, N. P., Rossi, G., Ruan, J. J., Rubiño-Martín, J. A., Rykoff, E. S., Salazar-Albornoz, S., Salvato, M., Samushia, L., Sánchez, A. G., Santiago, B., Sayres, C., Schiavon, R. P., Schlegel, D. J., Schmidt, S. J., Schneider, D. P., Schultheis, M., Schwope, A. D., Scóccola, C. G., Sellgren, K., Seo, H., Shane, N., Shen, Y., Shetrone, M., Shu, Y., Sivarani, T., Skrutskie, M. F., Slosar, A., Smith, V. V., Sobreira, F., Stassun, K. G.,

Steinmetz, M., Strauss, M. A., Streblyanska, A., Swanson, M. E. C., Tan, J. C., Tayar, J., Terrien, R. C., Thakar, A. R., Thomas, D., Thompson, B. A., Tinker, J. L., Tojeiro, R., Troup, N. W., Vargas-Magaña, M., Vazquez, J. A., Verde, L., Viel, M., Vogt, N. P., Wake, D. A., Wang, J., Weaver, B. A., Weinberg, D. H., Weiner, B. J., White, M., Wilson, J. C., Wisniewski, J. P., Wood-Vasey, W. M., Yèche, C., York, D. G., Zakamska, N. L., Zamora, O., Zasowski, G., Zehavi, I., Zhao, G., Zheng, Z., Zhou, X., Zhou, Z., Zhu, G., and Zou, H. (2015). The Eleventh and Twelfth Data Releases of the Sloan Digital Sky Survey: Final Data from SDSS-III. *The Astrophysical Journal Supplement Series*, 219:12.

Audibert, J.-Y., Munos, R., and Szepesvári, C. (2009). Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19):1876 – 1902. Algorithmic Learning Theory.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002a). Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (2002b). The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77.

Baram, Y., El-Yaniv, R., and Luz, K. (2004). Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5:255–291.

Bedő, J. and Ong, C. S. (2016). Multivariate spearman's ρ for aggregating ranks using copulas. *Journal of Machine Learning Research*, 17(201):1–30.

Berry, D. A. and Fristedt, B. (1985). Bandit problems: sequential allocation of experiments (monographs on statistics and applied probability). *London: Chapman and Hall*, 5:71–87.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Brinker, K. (2003). Incorporating diversity in active learning with support vector machines. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pages 59–66. AAAI Press.

Brodersen, K. H., Ong, C. S., Stephan, K. E., and Buhmann, J. M. (2010). The balanced accuracy and its posterior distribution. In *Proceedings of the 2010 20th International Conference on Pattern Recognition*, ICPR '10, pages 3121–3124, Washington, DC, USA. IEEE Computer Society.

Cappé, O., Garivier, A., Maillard, O.-A., Munos, R., and Stoltz, G. (2013). Kullback-leibler upper confidence bounds for optimal sequential allocation. *Annals of Statistics*, 41(3):1516–1541.

Culotta, A. and McCallum, A. (2005). Reducing labeling effort for structured prediction tasks. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2*, AAAI'05, pages 746–751. AAAI Press.

Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, pages 148–156, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Freund, Y., Seung, H. S., Shamir, E., and Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine learning*, 28(2-3):133–168.

Hsu, W.-N. and Lin, H.-T. (2015). Active learning by learning. In *AAAI 15*.

Lattimore, T. (2015). Optimally confident UCB : Improved regret for finite-armed bandits. *CoRR*, abs/1507.07880.

Lewis, D. D. and 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*, pages 3–12. Springer-Verlag New York, Inc.

Lichman, M. (2013). UCI machine learning repository.

List, C. (2013). Social choice theory. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, winter 2013 edition.

McCallum, A. and Nigam, K. (1998). Employing EM and pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 350–358, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Melville, P. and Mooney, R. J. (2004). Diverse ensembles for active learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 74. ACM.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning*

Research, 12:2825–2830.

Rahimi, A. and Recht, B. (2008). Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184.

Reichart, R., Tomanek, K., Hahn, U., and Rappoport, A. (2008). Multi-task active learning for linguistic annotations. *Proceedings of ACL-08: HLT*, pages 861–869.

Reyes, O., Morell, C., and Ventura, S. (2018). Effective active learning strategy for multi-label learning. *Neurocomputing*, 273:494 – 508.

Reyes, O. and Ventura, S. (2018). Evolutionary strategy to perform batch-mode active learning on multi-label data. *ACM Trans. Intell. Syst. Technol.*, 9(4):46:1–46:26.

Scheffer, T., Decomain, C., and Wrobel, S. (2001). Active hidden markov models for information extraction. In Hoffmann, F., Hand, D., Adams, N., Fisher, D., and Guimaraes, G., editors, *Advances in Intelligent Data Analysis*, volume 2189 of *Lecture Notes in Computer Science*, pages 309–318. Springer Berlin Heidelberg.

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

Schulze, M. (2011). A new monotonic, clone-independent, reversal symmetric, and condorcet-consistent single-winner election method. *Social Choice and Welfare*, 36(2):267–303.

Seldin, Y. and Slivkins, A. (2014). One practical algorithm for both stochastic and adversarial bandits.

Settles, B. (2012). Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114.

Settles, B. and Craven, M. (2008). An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1070–1079. Association for Computational Linguistics.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294.

Tran, A. (2015). Photometric classification with thompson sampling.

Xu, Z., Akella, R., and Zhang, Y. (2007). Incorporating diversity and density in active learning for relevance feedback. In *European Conference on Information Retrieval*, pages 246–257. Springer.

APPENDIX A: POSTERIOR BALANCED ACCURACY

Most real-world datasets are unbalanced. In the SDSS dataset, for example, there are 4.5 times as many galaxies as quasars. The problem of class imbalance is even more severe in the pageblocks dataset, where one class makes up 90% of the data and the remaining four classes only make up 10%. An easy fix is to undersample the dominant class when creating the training and test sets. This, of course, means that the size of these sets are limited by the size of the minority class.

When we do not want to alter the underlying class distribution or when larger training and test sets are desired, we need a performance measure that can correct for the class imbalance. Brodersen et al. (2010) show that the posterior balanced accuracy distribution can overcome the bias in the binary case. We now extend this idea to the multi-class setting.

Suppose we have k classes. For each class i between 1 and k, there are N_i objects in the universe. Given a classifier, we can predict the label of every object and compare our prediction with the true label. Let G_i be the number of objects in class i that are correctly predicted. Then we define the recall A_i of class i as

$$A_i = \frac{G_i}{N_i} \tag{26}$$

The problem is that it is not feasible to get the actual values of G_i and N_i since that would require us to obtain the true label of every object in the universe. Thus we need a method to estimate these quantities when we only have a sample. Initially we have no information about G_i and N_i, so we can assume that each A_i follows a uniform prior distribution between 0 and 1. This is the same as a Beta distribution with shape parameters $\alpha = \beta = 1$:

$$A_i \sim \text{Beta}(1,1) \tag{27}$$

The probability density function (PDF) of A_i is then

$$\begin{aligned} f_{A_i}(a) &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} a^{\alpha-1}(1-a)^{\beta-1} \\ &\propto a^{1-1}(1-a)^{1-1} \end{aligned} \tag{28}$$

where $\Gamma(\alpha)$ is the gamma function.

After we have trained the classifier, suppose we have a test set containing n_i objects in class i. Running the classifier on this test set is the same as conducting k binomial experiments, where, in the ith experiment, the sample size is n_i and the probability of success is simply A_i. Let g_i be the number of correctly labeled objects belonging to class i in the test set. Then, conditional on the recall rate A_i, g_i follows a binomial distribution:

$$(g_i \mid A_i) \sim \text{Bin}(n_i, A_i) \tag{29}$$

The probability mass function of $(g_i \mid A_i = a)$ is thus

$$\begin{aligned} p_{g_i|A_i}(g_i) &= \binom{n_i}{g_i} a^{g_i}(1-a)^{n_i-g_i} \\ &\propto a^{g_i}(1-a)^{n_i-g_i} \end{aligned} \tag{30}$$

In the Bayesian setting, Eq. (28) is the prior and Eq. (30) is the likelihood. To get the posterior PDF, we simply multiply the prior with the likelihood:

$$f_{A_i|\boldsymbol{g}}(a) \propto f_{A_i}(a) \times f_{g_i|A_i}(g_i) \tag{31}$$

$$\propto a^{1-1}(1-a)^{1-1} \times a^{g_i}(1-a)^{n_i-g_i} \tag{32}$$

$$= a^{1+g_i-1}(1-a)^{1+n_i-g_i-1} \tag{33}$$

Thus, with respect to the binomial likelihood function, the Beta distribution is conjugate to itself. The posterior recall rate A_i also follows a Beta distribution, now with parameters

$$(A_i \mid g_i) \sim \text{Beta}(1+g_i, 1+n_i-g_i) \tag{34}$$

Our goal is to have a balanced accuracy rate, A, that puts an equal weight in each class. One way to achieve this is to take the average of the individual recalls:

$$A = \frac{1}{k}\sum_{i=1}^{k} A_i \tag{35}$$

$$= \frac{1}{k} A_T \tag{36}$$

Here we have defined A_T to be the sum of the individual recalls. We call $(A \mid \boldsymbol{g})$ the posterior balanced accuracy, where $\boldsymbol{g} = (g_1, ..., g_k)$. Most of the time, we simply want to calculate its expected value:

$$\mathbb{E}[A \mid \boldsymbol{g}] = \frac{1}{k}\mathbb{E}[A_T \mid \boldsymbol{g}] \tag{37}$$

$$= \frac{1}{k}\int a \cdot f_{A_T|\boldsymbol{g}}(a)\, da \tag{38}$$

Let us call this the mean posterior balanced accuracy (MPBA). Note that there is no closed form solution for the PDF $f_{A_T|\boldsymbol{g}}(a)$. However assuming that A_T is a sum of k independent Beta random variables, $f_{A_T|\boldsymbol{g}}(a)$ can be approximated by numerically convolving k Beta distributions. The independence assumption is reasonable here, since there should be little to no correlation between the individual recall rates. For example, knowing that a classifier is really good at recognizing stars does not tell us much about how well that classifier can recognize galaxies.

Having the knowledge of $f_{A|\boldsymbol{g}}(a)$ will allow us to make violin plots, construct confidence intervals and do hypothesis tests. To get an expression for this, let us first rewrite the cumulative distribution function (CDF) as

$$F_{A|\boldsymbol{g}}(a) = \mathbb{P}(A \leq a \mid \boldsymbol{g}) \tag{39}$$

$$= \mathbb{P}\Big(\frac{1}{k}A_T \leq a \,\Big|\, \boldsymbol{g}\Big) \tag{40}$$

$$= \mathbb{P}(A_T \leq ka \mid \boldsymbol{g}) \tag{41}$$

$$= F_{A_T|\boldsymbol{g}}(ka) \tag{42}$$

Differentiating (42) with respect to a, we obtain the PDF of $(A \mid \boldsymbol{g})$:

$$f_{A|\boldsymbol{g}}(a) = \frac{\partial}{\partial a} F_{A|\boldsymbol{g}}(ka) \tag{43}$$

$$= \frac{\partial}{\partial a}(ka) \cdot \frac{\partial}{\partial ka} F_{A_T|\boldsymbol{g}}(ka) \tag{44}$$

$$= k \cdot f_{A_T|\boldsymbol{g}}(ka) \tag{45}$$

A Python implementation for the posterior balanced accuracy can be found on our GitHub repository[5].

[5] https://github.com/chengsoonong/mclass-sky