
Online Choice of Active Learning Algorithms

Yoram Baram
Ran El-Yaniv
Kobi Luz

BARAM@CS.TECHNION.AC.IL
RANI@CS.TECHNION.AC.IL
KOBIL@CS.TECHNION.AC.IL

Department of Computer Science, Technion - Israel Institute of Technology

Abstract

This paper is concerned with the question of how to online combine an ensemble of active learners so as to expedite the learning progress during a pool-based active learning session. We develop a powerful active learning master algorithm, based a known competitive algorithm for the multi-armed bandit problem and a novel semi-supervised performance evaluation statistic. Taking an ensemble containing two of the best known active learning algorithms and a new algorithm, the resulting new active learning master algorithm is empirically shown to consistently perform almost as well as and sometimes outperform the best algorithm in the ensemble on a range of classification problems.

1. Introduction

The goal in *active learning* is to design and analyze learning algorithms that can effectively filter or choose the samples for which they ask the teacher for a label. The incentive in using active learning is mainly to expedite the learning process and reduce the labeling efforts required by the teacher. While there is lack of theoretical understanding of active learning (in particular, the generalization power of computationally practical active learning algorithms is not well understood), there are numerous empirical evidences showing that active learning can dramatically expedite the learning process.

We focus on incremental *pool-based* active learning of classifiers, which can be viewed as the following game consisting of *trials*. The learner is presented with a fixed pool of unlabeled instances and on each trial the learner chooses one instance from the pool to be labeled by the teacher. The teacher then provides the

learner with the true label of this instance and the learner induces a (new) classifier based on all the labeled samples seen so far and possibly based on unlabeled instances in the pool. Then a new trial begins, etc. Other variants of the active learning problem have been also considered. Two important variants are *stream-based* active learning (Freund et al., 1997) and learning with *membership queries* (Angluin, 1988). We do not consider these variants here.

Numerous active learning algorithms have been proposed in the literature. Here we mention two algorithms, which appear to be among the best performers, based on empirical studies. The first algorithm relies on kernel machines and was recently independently proposed by three research groups (Tong & Koller, 2001; Schohn & Cohn, 2000; Campbell et al., 2000). This algorithm, called SIMPLE in (Tong & Koller, 2001), uses the current SVM classifier to query the instance closest to the decision hyperplane (in kernel space). The second algorithm we discuss, proposed by (Roy & McCallum, 2001), is based on a different motivation: The algorithm chooses its next example to be labeled while attempting to reduce future generalization error probability. Since true future error rates are unknown, the learner attempts to estimate them using a “self-confidence” heuristic that utilized its current classifier for probability measurements. Throughout this paper we therefore call this algorithm SELF-CONF.

Seeking a top performing active learning algorithm we found that neither SIMPLE nor SELF-CONF is a consistent winner across problems. Moreover, both algorithms exhibit a severe pitfall that seems to appear in learning problems with a “XOR-like” structure (see Section 2). While perhaps no single active learning algorithm should be expected to consistently perform better than others on all problems, some problems clearly favor particular algorithms. This situation mo-

tivates an online learning approach whereby one attempts to online utilize an *ensemble* of algorithms so as to achieve a performance, which is close to the best algorithm in hindsight. This scheme has been extensively studied in computational learning theory, mainly in the context of ‘online prediction using expert advice’, see e.g. (Ceza-Bianchi et al., 1997). Our main contribution is an algorithm that actively learns by combining other active learners.

A reasonable approach for combining an ensemble of active learning algorithms (or ‘experts’) might be to evaluate their individual performance and dynamically switch to best performing expert so far. However, there are two obstacles for successfully implementing this scheme. First, standard classifier evaluation techniques such as cross-validation, leave-one-out or bootstrap tend to fail when used to estimate the performance of an active learner based on the labeled examples chosen by the learner. The reason is that the set of labeled instances selected by a good active learner tends to be acutely biased towards ‘hard’ instances that do not reflect the true underlying distribution. In Section 3 we show an example of this phenomenon. Second, even if we overcome the first problem, each time we choose to utilize a certain expert we only get to see the label of the example chosen by this expert and do not get to observe the consequence of choices corresponding to other experts.

We overcome these two obstacles using the following two ideas: Instead of using standard statistical techniques such as cross-validation we use a novel maximum entropy semi-supervised criterion, which utilizes the pool of unlabeled samples and can faithfully evaluate the relative progress of the various experts; second, we cast our problem as an instance of the *multi-armed bandit* problem, where each expert corresponds to one slot machine and on each trial we are allowed to play one machine (i.e. choose one active learning algorithm to generate the next query). We then utilize a known online multi-armed bandit algorithm of (Auer et al., 2002). This algorithm enjoys strong performance guarantees without any statistical assumptions.

We present an extensive empirical study of our new active learning meta algorithm and compare its performance to its ensemble members consisting of three algorithms: SIMPLE, SELF-CONF and a novel active learning heuristic based on “furthest-first traversals” (Hochbaum & Shmoys, 1985). The resulting meta algorithm is shown to consistently perform almost as well as the best algorithm in the ensemble and on some problems it outperforms the best algorithm.

2. On Active Learning and Three Algorithms

In this section we first define more formally ‘pool-based active learning’ and then briefly describe two known and one novel active learning algorithms. These algorithms form the ensemble on which our new “master” algorithm is later applied. The known algorithms we selected are SIMPLE (Tong & Koller, 2001; Schohn & Cohn, 2000; Campbell et al., 2000) and the algorithm of (Roy & McCallum, 2001), called here SELF-CONF. The reasons we selected these algorithms are that they are both reasonably well motivated and they achieve high performance on real-world data (and beat various other known algorithms). Nevertheless, both these algorithms are inefficient in learning problems with XOR-like structure. The third algorithm is novel. While this algorithm exhibits quite poor performance on problems with simple structure it particularly excels on XOR-like problems.

Pool-based active learning. We consider a binary classification problem and are given a pool $\mathcal{U} = \{x_1, \dots, x_m\}$ of unlabeled instances where each x_i is a vector in some Euclidean space. Instances are assumed to be i.i.d. distributed according to some unknown fixed distribution $P(x)$. Each instance x_i has a label $y_i \in \mathcal{Y}$ (where in our case $\mathcal{Y} = \{\pm 1\}$) distributed according to some unknown conditional $P(y|x)$. At each stage let \mathcal{L} be the set of labeled instances already known to the learner.¹ An *active learner* consists of a classifier learning algorithm, which has been trained on \mathcal{L} (and possibly on \mathcal{U}) and a *querying function* $Q : \mathcal{L} \times \mathcal{U} \rightarrow \mathcal{U}$. On each *trial* the active learner first applies Q to choose one unlabeled instance x from \mathcal{U} . The label y of x is then revealed and the pair (x, y) is added to \mathcal{L} and x is removed from \mathcal{U} . Then the learner induces a new classifier using \mathcal{L} as a training set and a new trial begins, etc.

Measuring active learning performance. To the best of our knowledge, in the literature there is no consensus on appropriate performance measures for active learning. We propose the following natural performance measure, which aims to quantify the “deficiency” of the querying function, while using a particular inductive learning algorithm ALG. Fix a particular classification problem. Let \mathcal{U} be a random pool of n instances. For each $1 \leq t \leq n$ let $\text{Acc}_t(\text{ALG})$ be the true average accuracy achievable by ALG using a training set of size t that is randomly and uniformly chosen from \mathcal{U} . Let ACTIVE be an active learning algorithm that uses ALG as its inductive learning component. De-

¹We assume that initially \mathcal{L} contains two examples, one from each class.

fine $\text{Acc}_t(\text{ACTIVE})$ to be the average accuracy achieved by ACTIVE after t active learning trials starting with the pool \mathcal{U} . Then, the *efficiency* of ACTIVE is defined to be

$$\text{Deff}_n(\text{ACTIVE}) = \frac{\sum_{t=1}^n \text{Acc}_n(\text{ALG}) - \text{Acc}_t(\text{ACTIVE})}{\sum_{t=1}^n \text{Acc}_n(\text{ALG}) - \text{Acc}_t(\text{ALG})}. \quad (1)$$

This measure captures the “global” performance of an active learner throughout the learning session. Notice that the numerator is simply the area between the “maximal” achievable accuracy $\text{Acc}_n(\text{ALG})$ using the entire pool², and the learning curve of the active learning algorithm. The denominator is the area between the same maximal accuracy and the learning curve of the “passive” algorithm. The purpose of the denominator is to normalize the measure so as to be “problem independent”. Thus, this measure is always non-negative and smaller values in $[0, 1)$ indicate more efficient active learning. $\text{Deff}_n(\text{ACTIVE})$ has the desired property that if n is sufficiently large so that $\text{Acc}_n(\text{ALG})$ (almost) achieves the maximal accuracy (for this classifier), then for any $n' > n$, $\text{Deff}_{n'}(\text{ACTIVE}) = \text{Deff}_n(\text{ACTIVE})$.

Algorithm simple. This algorithm uses an SVM as its induction component. The querying function of SIMPLE at trial t uses the already induced classifier C_{t-1} to choose an unlabeled instance, which is closest to the decision boundary of C_{t-1} . Following (Tong & Koller, 2001) in our applications of SVMs throughout this paper we use the “kernel trick” discussed in (Shaw-Taylor & Christianini, 2002), which guarantees linear separability of the training set in feature space.

Algorithm self-conf. At the start of each trial this algorithm already holds a trained probabilistic (soft) classifier given by $\hat{P}(y|x)$. For each $x \in \mathcal{U}$ and $y \in \mathcal{Y}$ the algorithm trains a new classifier \hat{P}' over $\mathcal{L}'(x, y) = \mathcal{L} \cup \{(x, y)\}$ and estimates the resulting “self-estimated expected log-loss” defined to be $E(\hat{P}'_{\mathcal{L}'(x,y)}) = -\frac{1}{|\mathcal{U}|} \sum_{y' \in \mathcal{Y}, x' \in \mathcal{U}} \hat{P}'(y'|x') \log \hat{P}'(y'|x')$. Then, for each $x \in \mathcal{U}$ it calculates the self-estimated average expected loss $\sum_{y \in \mathcal{Y}} \hat{P}(y|x) E(\hat{P}'_{\mathcal{L}'(x,y)})$. The x with the lowest expected loss is then chosen to be queried. The original SELF-CONF algorithm uses Naive Bayes probability estimates. In the experiments described below we implement SELF-CONF using soft (confidence rated) SVMs. Probabilistic estimates are obtained in a standard way using the logistic regression transform. The algorithm as presented is extremely inefficient. Various optimizations and approximations

²Note that in some cases, see e.g. (Schohn & Cohn, 2000), better accuracy can be achieved using “early stopping”.

are proposed in (Roy & McCallum, 2001) to make its running time practically feasible. From all these methods we only use random sub-sampling in our implementation of the algorithm: on each trial we estimate $E(\hat{P}'_{\mathcal{L}'(x,y)})$ for only a random subset of \mathcal{U} .³

Algorithm Kernel Farthest-First (KFF) We propose a simple active learning heuristic based on “farthest-first” traversal sequences in kernel space. *Farthest-first (FF)* sequences have been previously used for computing provably approximately optimal clustering for k -center problems (Hochbaum & Shmoys, 1985). The FF querying function is defined as follows: Given the current set \mathcal{L} of labeled instances, we choose as our next query an instance $x \in \mathcal{U}$, which is farthest from \mathcal{L} (where the distance of a point from a set is defined to be the minimum distance to a point in the set). Using $\mathcal{L} \cup \{(x, y)\}$ as a training set we then induce a classifier. This heuristic has a nice intuitive appeal in our context: The next instance to query is the farthest (and in some sense the most dissimilar) instance in the pool from those we have already learned. Unlike SIMPLE (and other algorithms) whose querying function is based on the classifier, the above FF querying function can be applied with any classifier learning algorithm. We apply it with an SVM and compute distances (for the FF traversals) in kernel space.

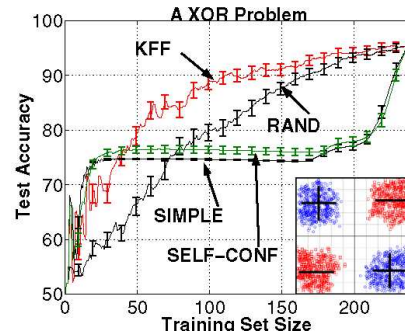


Figure 1. **Lower right panel:** a 1000 points XOR problem (250 in each “cluster”); 250 points form the unlabeled pool and the rest are the test set. **Main figure:** Learning curves for SIMPLE, SELF-CONF, KFF and RAND. Each point on each learning curve represents an average over 100 folds (measured over a test set); each error bar represents one standard error of the mean. Error bars are diluted to enhance visibility.

Example. Figure 1 shows the learning curves of SIMPLE, SELF-CONF, KFF and of a random sampling “active learning” (which we call RAND)⁴ for

³First sub-sample contains 100 points; on each iteration we decrement the sub-sample down to 10 points.

⁴The querying function of RAND chooses the next exam-

the “XOR” problem of Figure 1(Lower right panel). Notice that in this case, the overall performance of SIMPLE and SELF-CONF is significantly worse than that of random sampling (the deficiency of these algorithms is given in Table 2). On the other hand, KFF clearly shows that active learning can expedite learning also in this problem. This weakness of both SIMPLE and SELF-CONF is typical in many problems with XOR-like structure. However, despite the apparent advantage of KFF over SIMPLE and SELF-CONF in XOR problems, we later show a number of examples where KFF is considerably weaker than these algorithms and even than random sampling. The main advantage in considering KFF is its use in an ensemble of active learning algorithms. Our master algorithm utilizes KFF to benefit in XOR-like problems without significant compromises in problems where KFF is weaker.

3. Combining Active Learners Online

In this section we describe an online algorithm for combining active learners. The combination algorithm, called here for short COMB, is based on a known competitive algorithm for the multi-armed bandit problem and on a novel model selection criterion. After describing these two components we present algorithm COMB.

The Multi-Armed Bandit (MAB) Problem. In this problem a gambler must choose one of n non-identical slot machines to play in a sequence of trials. Each machine can yield rewards whose distribution is unknown to the gambler, and the gambler’s goal is to maximize his total reward over a sequence of trials. This classical problem is one of the most basic problems whose essence is the tradeoff between *exploration* and *exploitation*: Sticking to any single machine may prevent discovering a better machine. On the other hand, continually seeking a better machine will prevent achieving the best possible total reward.

We make the following straightforward analogy between the problem of online combining an ensemble of active learners and the MAB problem. The n active learning algorithms in our ensemble are the n slot machines. On each trial, choosing a query generated by one active learning algorithm corresponds to choosing one slot machine. The true (generalization) accuracy achieved (by the combined algorithm) using the augmented training set (which includes the newly queried data point) corresponds to the gain achieved by the chosen machine. Of course, this rough analogy does not immediately provide a solution for combining active learners. Later on we show how to fill in all the

missing details.

The *adversarial* MAB results of (Auer et al., 2002), provide MAB algorithms that are guaranteed to extract a total gain close to that of the best slot machine (in hindsight) without *any* statistical assumptions. Two particular MAB algorithms from (Auer et al., 2002) are potentially useful for implementing online choice of active learners. The first algorithm, called EXP3.1, directly matches the above analogy. The second algorithm, called EXP4, appears to be more suitable for our purposes. In particular, EXP4 is designed to deal with a more sophisticated MAB game than the standard MAB game described above: The goal is to combine and utilize a number k of *strategies* or *experts*, each giving an advice on how to play n slot machines. On each trial t , each expert j , $j = 1, \dots, k$, provides a weighting $\mathbf{b}^j(t) = (b_1^j(t), \dots, b_n^j(t))$ with $\sum_i b_i^j(t) = 1$, where $b_i^j(t)$ represents the recommended probability of expert j for playing the i th machine, $i = 1, \dots, n$, on trial t . Denoting the vector of rewards for the n machines, on trial t by $\mathbf{g}(t) = (g_1(t), \dots, g_n(t))$ where $g_i(t)$ is non-negative and bounded, the expected reward of expert j on trial t is $\mathbf{b}^j(t) \cdot \mathbf{g}(t)$. In the MAB game only one reward from $\mathbf{g}(t)$ is revealed to the online player after the player chooses one machine in trial t . For a game consisting of T trials define $G_{\max} = \max_{1 \leq j \leq k} \sum_{t=1}^T \mathbf{b}^j(t) \cdot \mathbf{g}(t)$, the expected reward of the best expert in hindsight. The goal of the online player in this game is to utilize the advice given by the experts so as to achieve reward as close as possible to G_{\max} . Algorithm EXP4 from (Auer et al., 2002) achieves this goal and this paper proves that the “regret” of EXP4, defined to be G_{\max} minus the expected reward of EXP4, is bounded above by $O(\sqrt{G_{\max} n \ln k})$. This regret bound holds for any number of trials T , provided that one of the experts in the ensemble is the “uniform expert”, which always provides the uniform recommendation vector for the n slot machines.

To employ EXP4 in our context we associate the k experts with the ensemble of k active learning algorithms (note that for the bound to hold we must include RAND in our pool of active learners, which corresponds to the “uniform expert”). The slot machines are associated with the unlabeled instances in our pool \mathcal{U} . This way, the expert advice vectors are probabilistic recommendations on instances of \mathcal{U} . It is thus required that each algorithm in the ensemble will provide “ratings” for the entire pool on each trial. In practice, the three algorithms we consider naturally provide such ratings: SIMPLE uses the kernel distance from the decision hyperplane, SELF-CONF by using expected loss and KFF using the kernel distance from the training set. The main missing element is the definition of rewards. We

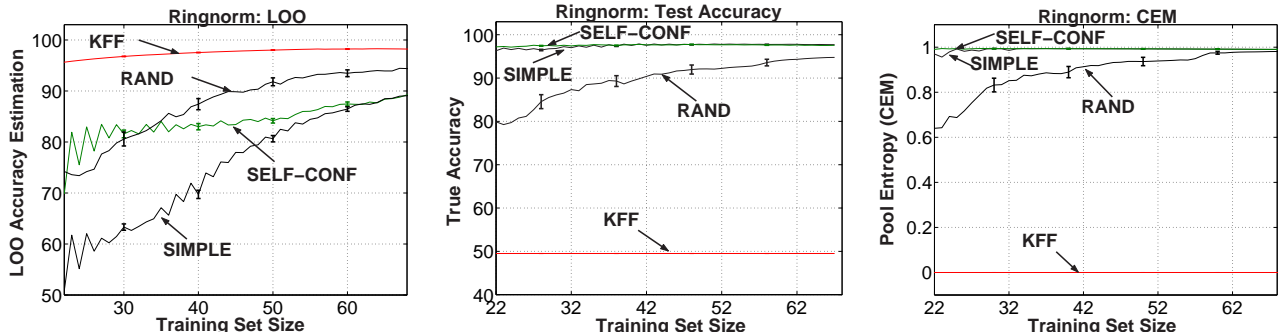


Figure 2. Left: LOO estimates of active learning sessions of SIMPLE, SELF-CONF, KFF and RAND over the ‘Ringnorm’ dataset; Middle: The “true” accuracy of these algorithms, as estimated using a test set; Right: CEM entropy scores of the algorithms. All estimates are averages of 100 folds. Error bars (diluted to reduce clutter) represent one standard error of the mean.

deal with this issue in the next subsection.

Classification Entropy Maximization. In order to utilize the above MAB algorithm in our context we need to receive after each trial the gain of the instance x that was chosen to be queried (corresponding to one slot machine). While the ultimate reward in our context is the expected *true* accuracy of the classifier resulting from training over $\mathcal{L} \cup \{(x, y)\}$ (where y is the label of x), this quantity is not available to us. At the outset it may appear that standard error (or accuracy) estimation methods could be useful. However, this is not the case: Unless \mathcal{L} is sufficiently large, standard methods like cross-validation or leave-one-out fail to provide reliable true error estimates for an active learner’s performance. This fact, which was already pointed out by (Schohn & Cohn, 2000), is a result of the biased sample acquired by the active learner: In order to progress quickly, the learner must focus on “hard” or more “informative” samples. Consider Figure 2 (left). The figure shows leave-one-out (LOO) estimates of four (active) learning algorithms: SIMPLE, SELF-CONF, KFF and RAND. For each training set size, each point on a curve is generated using the LOO estimate based on the currently available labeled set \mathcal{L} . On Figure 2 (middle) we see the “true accuracy” of these algorithms as estimated using a test set. Not only that LOO severely fails to estimate the true accuracy, it even fails to order the algorithms according to their relative success as active learners. This unfortunate behavior is typical to LOO on most datasets and is suffered by other standard estimation techniques including cross-validation and bootstrap. Thus, these techniques cannot be reliably used for receiving feedback on the (relative) progress of active learners.

Instead we use the following semi-supervised estimator, which we call *Classification Entropy Maximization*

(*CEM*). We define the *CEM score* of a classifier with respect to an unlabeled set of points to be the binary entropy of the classification it induces on the unlabeled set. If C is a binary classifier giving values in $\{\pm 1\}$, let $C^{+1}(\mathcal{U})$ and $C^{-1}(\mathcal{U})$ be the positively and negatively classified subsets of some unlabeled set \mathcal{U} , respectively. Then, the CEM score of C is the binary entropy $H\left(\frac{|C^{+1}(\mathcal{U})|}{|\mathcal{U}|}\right)$. Thus, the CEM score is larger if the classification of the pool is more balanced. Figure 2 (right) provides CEM curves for the three active learners discussed above. Clearly, the CEM measure orders the algorithms in accordance with their true accuracy (middle). This behavior of CEM is typical in most of our empirical examinations and somewhat surprisingly, CEM succeeds to correctly evaluate performance even when the positive and negative priors are not balanced (see below). While formal connections between CEM and generalization are currently unknown the following informal discussion provides farther insights into CEM and attempts to characterize conditions for its effectiveness.

A sequence of sets S_1, S_2, \dots is called an *inclusion sequence* if $S_1 \subset S_2 \subset \dots$. Consider the inclusion sequence of training sets generated by an active learner. Let $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$ be any binary labeled set of samples where one of the classes (either +1 or -1) is a majority and its empirical proportion is r (i.e. the size of the majority class over m is r). Consider any classifier C giving the label $C(x)$ for $x \in S$. We say that the classification $S_C = (x_1, C(x_1)), \dots, (x_m, C(x_m))$ is *majority-biased* (with respect to S) if the majority class in S is also a majority class in S_C and its proportion is larger than or equal r . Let $I = S_1 \subset \dots \subset S_T$ be an inclusion sequence of labeled samples. We say that a *learning algorithm* ALG is *majority-biased* (with respect to

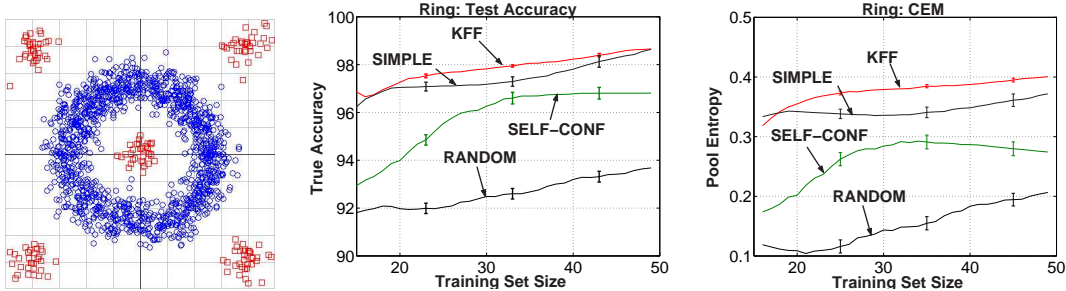


Figure 3. **Left:** A synthetic “Ring” problem consisting of 3500 points, 90% in the Ring subset (3150 points) and 10% equally divided within the 5 small clusters (70 points in each “cluster”); 1500 points were taken to be the pool of unlabeled points and the rest are the test set. **Middle:** True accuracy of four active learners - KFF, SIMPLE, SELF-CONF and RAND on the “Ring” dataset (left) as estimated on an independent test set. **Right:** Corresponding pool classification entropies of these four learners. All estimates are averages of 100 folds. Error bars (diluted to reduce clutter) represent one standard error of the mean.

I) if the classification of S_T by each of the classifiers C_1, \dots, C_T (induced by ALG) is majority-biased, where C_i is induced by ALG using S_i as a training set.

Our main (empirical) observation is that whenever the learning algorithm is majority biased with respect to the inclusion sequence of training sets (generated by the learner) CEM’s growth rate corresponds to the growth rate of the generalization accuracy in which case the CEM criterion can be used to online compare the performance of active learners. In other words, by comparing the growth of a learner’s CEM score (pool classification entropy) we can get a useful indication on the growth of its true accuracy. Consider Figure 3(left) depicting a synthetic “ring” example in which the positive class (the ‘ring’) is significantly larger than that of the negative class (the five small clusters). In particular the proportion of the positive class is $r = 0.9$. When running our three active learners (as well as RAND) on this dataset we observe that all four learners are majority biased. Specifically, undiscovered (negative) small clusters are classified as positive (like the ring). Therefore, for a majority biased learner, better performance in this example corresponds to more quickly “discovering” the 5 small clusters. Clearly by discovering the small clusters such a learner improves its true accuracy and simultaneously increases its CEM score.

Figure 3(middle) compares the true accuracy of the learners as estimated on an independent test set. Figure 3(right) compares the corresponding pool classification entropies (CEM scores). The two graphs exhibit a striking correspondence. Clearly, this example shows that the CEM criterion (as measured by the pool proportion) correctly ranks the true accuracy of these four learners. To summarize, the CEM criterion

should correctly rank active learners whose learning algorithms are majority biased. We note that SVMs tend to be majority biased (in particular, when applied with RBF kernels).

Algorithm comb. On Figure 4 we provide a commented pseudocode of our master active learning algorithm, called COMB. The algorithm utilizes an ensemble of active learning algorithms and online tracks the best algorithm in the ensemble. Many of the steps in this code are adapted from the MAB algorithm EXP4 of (Auer et al., 2002) (in particular, steps 1,3,4,5,10,11). Due to space limitations we refer the reader to (Auer et al., 2002) for a detailed explanation (and the proof guarantee) of EXP4. Here we explain steps 2 and 9, which in our experience are essential for efficient active learning using our scheme (the remaining steps, 6, 7 and 8 are self-explanatory). In step 2, after receiving (in step 1) the advice probability vectors from the active learners, we project the pool \mathcal{U} over high probability candidate instances. This projection is controlled by the parameter α and an instance x in \mathcal{U} remains in \mathcal{U}_e if at least one active learners assigns to x a probability mass greater than α . In all the experiments described below we used $\alpha = 0.05$.⁵ In step 9 we apply a utility function to the entropic reward calculated on step 8. The contribution of the last queried instance is measured by the change in the entropy. H_t is the entropy of the last partition of \mathcal{U} generated using the last queried instance in the training set. H_{t-1} is the entropy of the partition of the same pool generated without using the last queried instance. The outcome

⁵Initially this value was arbitrarily chosen without tuning. Sensitivity analysis in the range $[0.01, 0.1]$ showed that the algorithm is not sensitive to choices in this range (and other values were not examined). More details will be presented in the full paper.

of the utility function is normalized to be within $[0,1]$.

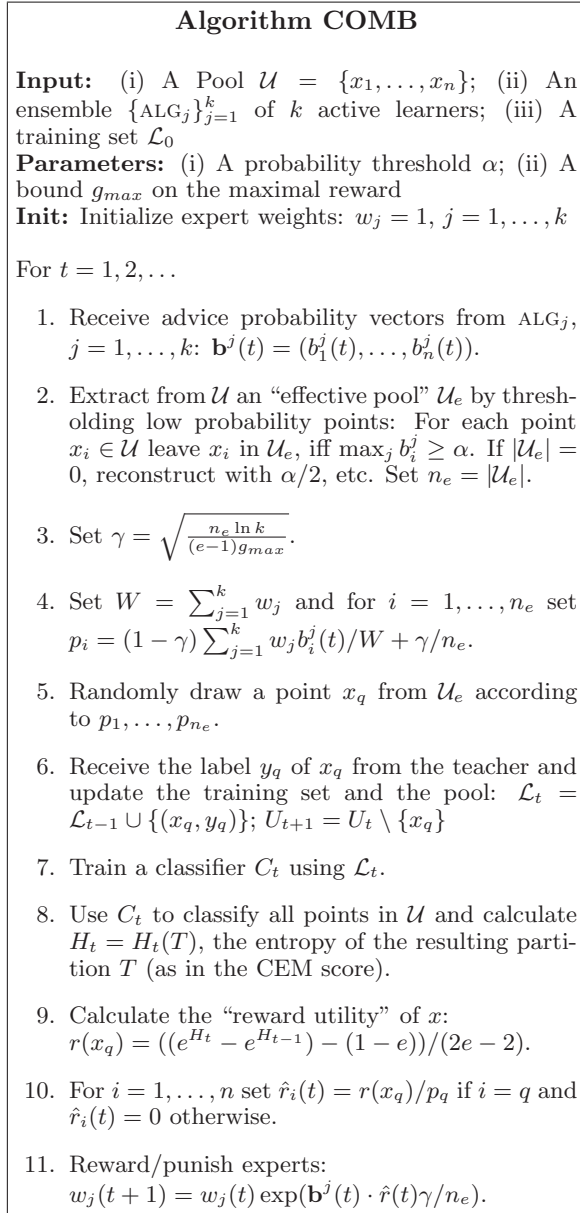


Figure 4. Algorithm COMB

The reward bound parameter g_{max} , used for setting an optimal value for γ (step 3) can and is eliminated in all our experiments using a standard “guess and double” technique. In particular we operate algorithm COMB in rounds $r = 1, 2, \dots$, where in round r we set the reward limit to be $g_r = (n_e \ln k / (e - 1)) 4^r$ and restart algorithm COMB with $g_{max} = g_r$. The round continues until the maximal reward reached by one of the ensemble algorithms exceeds $g_r - n_e / \gamma_r$. For more details, the reader is referred to algorithm EXP3.1 in (Auer et al., 2002).

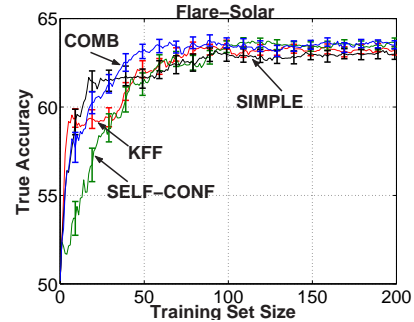


Figure 5. Learning curves of COMB and its ensemble members for Dataset ‘Flare-Solar’. All estimates are averages of 100 folds. Error bars (diluted to reduce clutter) represent one standard error of the mean.

4. Empirical Evaluation of COMB

We evaluated the performance of COMB on the entire dataset collection of (Rätsch, 1998), consisting of 13 binary classification problems extracted from the UCI repository. For almost all problems, this collection includes 100 folds each consisting of a fixed 60%/40% training/test partition. The use of this set is particularly useful as it allows for easier experiment replications. We also added our artificial XOR problem (see Figure 1) to the set of problems.⁶ Some essential characteristics of these datasets appear on Table 1: For each problem we provide the size, the dimension, the bias (proportion of largest class), the maximal accuracy achieved using the entire pool and the (rounded up) number of instances required by the worst active learner in the ensemble to achieve this maximal accuracy.

Table 2 shows the average *efficiency* of COMB and the various active learners in its ensemble for all the datasets. Recall the definition of efficiency of an active learner as given in Equation (1): Smaller values represent higher active learning efficiency. Each of these averages are calculated over the 100 folds.⁷ It is evident, that none of the ensemble algorithms is consistently winning over all sets (SELF-CONF, SIMPLE and KFF win in 7, 4 and 3 cases, respectively). Except for one case where KFF is significantly better, it is an overall loser (which is inferior to RAND). In 10 cases out of the 14 presented COMB is either the winner (5 cases) or a close runner up (5 cases). In 3 cases (‘Heart’, ‘Thyroid’ and ‘Breast-Cancer’) which are among the smallest datasets, COMB is not the winner and not even

⁶For this dataset we have created folds by randomly splitting the data to 80%/20% test/training partition and generated 100 such random partitions.

⁷Two datasets, Image and Splice, have only 20 folds.

the runner up, although even in these cases it is not far from the winner. This behavior is reasonable because COMB need sufficient “time” for both exploration and exploitation. There is only one case, the ‘Diabetis’ dataset, that demonstrates a problem where our entropy criterion fails in online choosing the best ensemble member. In this case, COMB performed significantly worse than both SELF-CONF (the winner) and SIMPLE. On the other extreme, of particular interest are cases where COMB is the winner. Figure 5 shows such a case and depicts the learning curves of COMB and its ensemble members on the ‘Flare-Solar’ dataset.

Table 1. The datasets: some essential characteristics.

| Dataset | Size | Dim | Bias | Max Acc. (Sample Size) |
|----------|------|-----|-------|---------------------------|
| Banana | 5300 | 2 | 0.551 | 88.23 (200) |
| Breast | 277 | 9 | 0.707 | 73.45 (100) |
| Diabetis | 768 | 8 | 0.651 | 75.60 (220) |
| Flare | 1066 | 9 | 0.552 | 63.87 (200) |
| German | 1000 | 20 | 0.700 | 75.50 (330) |
| Heart | 270 | 13 | 0.555 | 82.16 (82) |
| Image | 2310 | 18 | 0.571 | 95.17 (500) |
| Ringnorm | 8300 | 20 | 0.549 | 97.87 (200) |
| Splice | 3175 | 60 | 0.519 | 85.86 (450) |
| Thyroid | 215 | 5 | 0.697 | 95.51 (69) |
| Titanic | 2201 | 3 | 0.676 | 73.14 (74) |
| Twonorm | 7400 | 20 | 0.500 | 95.65 (200) |
| Waveform | 5000 | 21 | 0.670 | 81.40 (200) |
| XOR | 3000 | 2 | 0.500 | 96.35 (240) |

Table 2. Average *efficiency* (see Equation (1)) achieved by COMB and its ensemble members. For each dataset the winner appears in boldface and marked with a star. The runner-up appears in boldface.

| Dataset | SIMPLE | KFF | SELF-CONF | COMB |
|-------------|--------------|--------------|--------------|--------------|
| Banana | 1.13 | 0.73* | 0.74 | 0.74 |
| Breast | 1.06 | 1.28 | 0.95* | 1.09 |
| Diabetis | 0.64 | 1.07 | 0.48* | 0.82 |
| Flare-solar | 1.13 | 1.09 | 1.39 | 0.79* |
| German | 0.71 | 0.85 | 0.67 | 0.64* |
| Heart | 0.57 | 1.04 | 0.50* | 0.64 |
| Image | 0.54 | 0.76 | 0.45* | 0.47 |
| Ringnorm | 0.34* | 4.70 | 0.38 | 0.36 |
| Splice | 0.58 | 2.54 | 0.60 | 0.57* |
| Thyroid | 0.55 | 2.34 | 0.47* | 0.64 |
| Titanic | 0.76 | 0.92 | 0.68 | 0.65* |
| Twonorm | 0.24* | 1.03 | 0.32 | 0.26 |
| Waveform | 0.58* | 0.97 | 0.66 | 0.60 |
| XOR | 1.24 | 0.63 | 1.19 | 0.47* |

5. Conclusions and future work

We presented an online algorithm that effectively combines an ensemble of active learners. The algo-

rith successfully utilizes elements from both statistical learning and online (adversarial) learning. Extensive empirical results strongly indicate that our algorithm can track the best algorithm in the ensemble on real world problems. Quite surprisingly our algorithm can quite often outperform the best ensemble member.

Some questions require further investigations. In our experience, the ‘classification entropy maximization (CEM)’ semi-supervised criterion for tracking active learning progress is essential and cannot be replaced by standard error estimation techniques. Further studies of this overly simple but effective criterion may be revealing. It would also be interesting to examine alternative (semi-supervised) estimators. Farther improvements can be achieved by developing MAB bounds which depend on the game duration. Such bounds can help in controlling the tradeoff between exploration and exploitation when using very small datasets.

References

- Angluin, D. (1988). Queries and concept learning. *Machine Learning*, 2, 319–342.
- Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. (2002). The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32, 48–77.
- Campbell, C., Cristianini, N., & Smola, A. (2000). Query learning with large margin classifiers. *ICML* (pp. 111–118).
- Ceza-Bianchi, N., Freund, Y., Haussler, D., Helmbold, D., Schapire, R., & Warmuth, M. (1997). How to use expert advice. *Journal of the ACM*, 44, 427–485.
- Freund, Y., Seung, H., Shamir, E., & Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning*, 28, 133–168.
- Hochbaum, D., & Shmoys, D. (1985). A best possible heuristic for the K-center problem. *Mathematics of Operations Research*, 10, 180–184.
- Rätsch, G. (1998). Benchmark data sets. URL: <http://ida.first.gmd.de/raetsch/data/benchmarks.htm>.
- Roy, N., & McCallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. *ICML* (pp. 441–448).
- Schohn, G., & Cohn, D. (2000). Less is more: Active learning with support vector machines. *ICML* (pp. 839–846).
- Shaw-Taylor, J., & Christianini, N. (2002). On the generalization of soft margin algorithms. *IEEE Transactions on Information Theory*, 48(10), 2721–2735.
- Tong, S., & Koller, D. (2001). Support vector machine active learning with applications to text classification. *JMLR*, 2, 45–66.