Journal of Machine Learning Research 5 (2004) 255–291 Submitted 10/03; Published 3/04

Online Choice of Active Learning Algorithms

Yoram Baram BARAM@CS.TECHNION.AC.IL
Ran El-Yaniv RANI@CS.TECHNION.AC.IL
Kobi Luz KOBIL@CS.TECHNION.AC.IL
Department of Computer Science
Technion - Israel Institute of Technology
Haifa 32000
Israel

Editor: Manfred Warmuth

Abstract

This work is concerned with the question of how to combine online an ensemble of active learners so as to expedite the learning progress in pool-based active learning. We develop an active-learning master algorithm, based on a known competitive algorithm for the multi-armed bandit problem. A major challenge in successfully choosing top performing active learners online is to reliably estimate their progress during the learning session. To this end we propose a simple maximum entropy criterion that provides effective estimates in realistic settings. We study the performance of the proposed master algorithm using an ensemble containing two of the best known active-learning algorithms as well as a new algorithm. The resulting 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.

Keywords: Active learning, kernel machines, online learning, multi-armed bandit, entropy maximization

1. Introduction

The goal in *active learning* is to design and analyze learning algorithms that can effectively choose the samples for which they ask the teacher for a label. The incentive for using active learning is mainly to expedite the learning process and reduce the labeling efforts required by the teacher. While there is a lack of theoretical understanding of active learning (in particular, the generalization power of computationally practical active-learning algorithms is not well understood[1]), there is substantial empirical evidence that active learning can dramatically expedite the learning process.

We focus on *pool-based* active learning by classifiers, which can be viewed as the following game composed of *trials*. The learner is presented with a fixed pool of unlabeled instances. On each trial the learner chooses one instance from the pool to be labeled by the teacher. The teacher provides the learner with the true label of this instance and then the learner induces a (new) classifier based on all the labeled samples seen so far and possibly on unlabeled instances in the pool. Then a new trial begins, etc. Other variants of the active-learning problem have also been considered.

1. In noise-free settings, the Query-by-Committee (QBC) algorithm of Freund et al. (1997) can provably provide exponential speedups in the learning rate over a random selection. However, at this time a "practically efficient" implementation of this technique seems to be beyond reach (see, for example, Bachrach et al., 1999).

Two important variants are *stream-based* active learning (Freund et al., 1997) and learning with *membership queries* (Angluin, 1988); see Section 3 for more details. We do not consider these variants in this work.

Seeking top performing active-learning algorithms among the numerous algorithms proposed in the literature, we found two algorithms that appear to be among the best performers, based on empirical studies. The first algorithm relies on kernel machines and was independently proposed by three research groups (Tong and Koller, 2001, Schohn and Cohn, 2000, Campbell et al., 2000). The algorithm, called SIMPLE by Tong and Koller (2001), uses the current SVM classifier to query the instance closest to the decision hyperplane (in kernel space). A theoretical motivation for SIMPLE is developed by Tong and Koller (2001) in terms of version space bisection. The second algorithm we consider, proposed by Roy and McCallum (2001), is based on a different motivation: the algorithm chooses its next example to be labeled while attempting to reduce the generalization error probability. Since true future error rates are unknown, the learner attempts to estimate them using a "self-confidence" heuristic that utilizes its current classifier for probability measurements. Throughout this paper we, therefore, call this algorithm SELF-CONF. The original SELF-CONF proposed by Roy and McCallum (2001) bases its probability estimates (and classification) on Naive Bayes calculations and is shown to significantly outperform other known active-learning algorithms on text categorization tasks. In our studies we used an SVM-based variant of SELF-CONF, which appears to be stronger than the original.

Empirical studies we have conducted reveal 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 4.4). 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 motivates an online learning approach whereby one attempts to utilize an *ensemble* of algorithms online aiming to achieve a performance that 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, for example, Ceza-Bianchi et al., 1997). Our main contribution is an algorithm that actively learns by combining active learners.

A reasonable approach to combining an ensemble of active-learning algorithms (or 'experts') might be to evaluate their individual performance and dynamically switch to the best performer so far. However, two obstacles stand in the way of 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 6 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 cannot observe the consequence of choices corresponding to other experts without "wasting" more labeled examples.

We overcome these two obstacles by 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 (that is, choose one active-learning algorithm to generate the next query). We then utilize a known online multi-armed bandit algorithm proposed by 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 master 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 and Shmoys, 1985). Our master 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. Active Learning

In this section we first define pool-based active learning and then discuss performance measures for active learners. We consider a binary classification problem and are given a pool $\mathcal{U} = \{x_1, \ldots, x_n\}$ of unlabeled instances where each x_i is a vector in some space $\mathcal{X}$. 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 distribution $P(y|x)$. At the start, none of the labels is known to the learner. At each stage of the active-learning game, let $\mathcal{L}$ be the set of labeled instances already known to the learner.[2] An *active learner* consists of a classifier learning algorithm $\mathcal{A}$, and a *querying function* Q, which is a mapping $Q : \mathcal{L} \times \mathcal{U} \to \mathcal{U}$. The querying function determines one unlabeled instance in $\mathcal{U}$ to be labeled by the teacher. On each *trial* $t = 1, 2, \ldots$, 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 applies $\mathcal{A}$ to induce a new classifier C_t using $\mathcal{L}$ (and possibly $\mathcal{U}$) as a training set and a new trial begins, etc. Thus, the active learner generates a sequence of classifiers $C_1, C_2, \ldots$. Clearly, the maximal number of trials is the initial size of $\mathcal{U}$. In this paper we focus on active learners $(\mathcal{A}, Q)$ where $\mathcal{A}$ is always an SVM induction algorithm.

To the best of our knowledge, in the literature there is no consensus on appropriate performance measures for active learning. A number of performance measures for active-learning algorithms make sense. For example, some authors (for instance, Tong and Koller, 2001) test the accuracy achieved by the active learner after a predetermined number of learning trials. Other authors (for example, Schohn and Cohn, 2000, Campbell et al., 2000, Roy and McCallum, 2001) simply show active-learning curves to visually demonstrate advantages in learning speed. Here we propose the following natural performance measure, which aims to quantify the "deficiency" of the querying function with respect to random sampling from the pool (which corresponds to standard "passive learning"), while using a fixed 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 $\mathrm{Acc}_t(\mathrm{ALG})$ be the true average accuracy achievable by ALG using a training set of size t that is randomly and uniformly chosen from $\mathcal{U}$. A hypothetical $\mathrm{Acc}_t(\mathrm{ALG})$ is depicted by the lower learning curve in Figure 1. Let ACTIVE be an active-learning algorithm that uses ALG as its inductive learning component $\mathcal{A}$. Define $\mathrm{Acc}_t(\mathrm{ACTIVE})$ to be the average accuracy achieved by ACTIVE after t active-learning trials starting with the pool $\mathcal{U}$ (see Figure 1 for a hypothetical $\mathrm{Acc}_t(\mathrm{ACTIVE})$), which is depicted by the

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

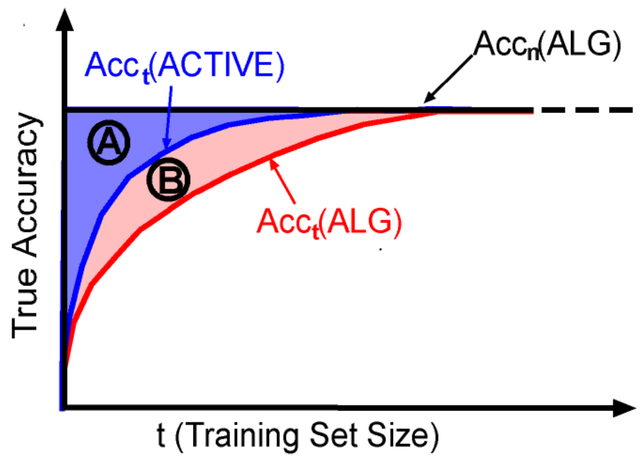


Figure 1: The definition of an active learner's *deficiency*. $\mathrm{Acc}_n(\textsc{alg})$ is the maximal achievable accuracy. $\mathrm{Acc}_t(\textsc{active})$ is the average accuracy achieved by hypothetical active learner ACTIVE. $\mathrm{Acc}_t(\textsc{alg})$ is the average accuracy achieved by a "passive" learner ALG (which queries randomly and uniformly from the pool). In this case (when the active-learning curve is above the passive learning curve) the *deficiency*, as defined in Equation (1), is the ratio of areas $\frac{A}{A+B}$.

higher learning curve). Then, the *deficiency* of ACTIVE is defined to be

$$\mathrm{Def}_n(\textsc{active}) = \frac{\sum_{t=1}^{n}(\mathrm{Acc}_n(\textsc{alg}) - \mathrm{Acc}_t(\textsc{active}))}{\sum_{t=1}^{n}(\mathrm{Acc}_n(\textsc{alg}) - \mathrm{Acc}_t(\textsc{alg}))}. \quad (1)$$

In words, the deficiency is simply the ratio of areas $\frac{A}{A+B}$, as depicted in Figure 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"[3] achievable accuracy $\mathrm{Acc}_n(\textsc{alg})$ using the entire pool, and the learning curve of the active-learning algorithm (area A in Figure 1). The denominator is the area between the same maximal accuracy and the learning curve of the "passive" algorithm (the sum of areas A and B in Figure 1). 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. $\mathrm{Def}_n(\textsc{active})$ has the desired property that if n is sufficiently large so that $\mathrm{Acc}_n(\textsc{alg})$ achieves the maximal accuracy (for this classifier), then for any $n' > n$, $\mathrm{Def}_{n'}(\textsc{active}) = \mathrm{Def}_n(\textsc{active})$.

3. Related Work

This paper is the first to consider a utilization of an ensemble of active learners. Here we discuss selected results related to the present work. Our main focus is on techniques for devising querying functions and methods that can be used to evaluate the progress of an active learner. Note that, in

3. In some cases (see for example, Schohn and Cohn, 2000), better accuracy can be achieved using "early stopping" (see Section 3).

general, a method estimating the "value" of an *unlabeled point* can be used to construct a querying function, and a method that evaluates the gain of a newly acquired *labeled point* can be used as an estimator for an active learner's progress.

We begin with a presentation of various techniques used to construct querying functions. Although our work focuses on pool-based active learning, a number of interesting and relevant ideas appear within other active-learning frameworks. In addition to the pool-based active-learning setting introduced in Section 2, we consider two other settings:

- *Stream-based* active learning (see, for example, Freund et al., 1997): the learner is provided with a stream of unlabeled points. On each trial, a new unlabeled point is drawn and introduced to the learner who must decide whether or not to request its label. Note that the stream-based model can be viewed as an online version of the pool-based model.

- Active learning with *membership queries* (Angluin, 1988), also called *selective sampling*: On each trial the learner constructs a point in input space and requests its label. This model can be viewed as a pool-based game where the pool consists of *all possible points* in the domain.

Within the *stream-based* setting, Seung et al. (1992) present the Query by Committee (QBC) algorithm. This algorithm is based on a seminal theoretical result stating that by halving the version space after each query, the generalization error decreases exponentially (relative to random active learning). To approximately bisect the version space, the proposed method randomly samples the version space and induces an even number of classifiers. The label of a (stream) point is requested whenever a voting between the classifiers on this point's label results in a tie. The main obstacle in implementing this strategy is the sampling of the version space. Currently, it is not known how to efficiently sample version spaces uniformly at random for many hypothesis classes of interest (see also discussions in Bachrach et al., 1999, Herbrich et al., 2001).

A variation of the QBC algorithm was proposed by McCallum and Nigam. Expectation Maximization (EM) is used to create a committee of classifiers to implement the querying function using the QBC voting idea: A diversified committee of classifiers is created by sampling a population of Naive Bayes distributions. Then an EM-like procedure is used to iteratively classify the unlabelled data and rebuild the classifiers until the process converges. Experimental results show that this algorithm has an advantage over random sampling and QBC.[4]

Interesting ideas were also considered within the *learning with membership queries* setting of Angluin (1988). Cohn et al. (1994) use the following version space reduction strategy. Two "distant" hypotheses are selected from the version space by finding a "most specific" concept (that classifies as 'negative' as much of the domain as possible) and a "most general" one (that classifies as 'positive' as many points as possible).[5] Selective sampling is done by randomly searching for a domain point on which the most specific and most general hypotheses disagree. This method ensures that the version space size is reduced after every query, though it might be reduced by only one hypothesis. The algorithm was implemented using a hypothesis class consisting of feed-forward neural networks trained with the back-propagation algorithm. According to the authors, the method works well only for "simple" concepts.

4. The QBC used for comparison utilizes the committee created by sampling of the Naive Bayes distributions without employing the EM-like procedure.
5. The most specific (resp. general) concept is found by classifying many (unlabeled) examples as 'negative' (resp. 'positive').

Again, within the selective sampling model, Lindenbaum et al. (2004) propose an active-learning algorithm for the nearest-neighbor classifier. The paper proposes using a random field model to estimate class probabilities. Using the class probabilities of the unlabeled examples a "utility function" of a training set is defined. The querying function of this algorithm is constructed using a game-tree that models a game between the learner and the teacher. For each unlabeled example, its expected utility is measured using the utility function on the training set and using expected probabilities for the possible classes of the unlabeled example. A set of domain points is constructed randomly and the example with highest expected utility is chosen. This algorithm suffers from extensive time complexity that depends both on the depth of the game-tree and on the number of near examples taken to construct the random field.

In the *pool-based* setting, three independent working groups (Schohn and Cohn, 2000, Campbell et al., 2000, Tong and Koller, 2001) propose the same querying function for active learners based on support vector machines (SVMs). This querying function chooses, for the next query, the unlabeled point closest to the decision hyperplane in kernel space; namely, the point with the smallest margin. Some theoretical motivation is given to this function in terms of version space reduction along the lines of QBC ideas (Tong and Koller, 2001); see Section 4.1 for further details. Experimental results presented in these papers show that the resulting active learner can provide significant sample complexity speedups compared to random sampling. Schohn and Cohn (2000) encountered a phenomenon where the true accuracy of the learner *decreases* (after reaching a certain peak) as the active session progresses. This phenomenon motivates the idea of "early stopping" and the authors suggest to stop querying when no example lies within the SVM margin. Campbell et al. (2000) offer to use this stopping criterion as a trigger for running a test for the current classifier. In this test they randomly and uniformly choose a small subsample of (unlabeled) pool points, request the labels of these points from the teacher and then test the current classifier on these points. The algorithm decides to stop if the error (as estimated using this test) is "satisfying" according to a user defined threshold.

Zhang and Oles (2000) analyze the value of unlabeled data in both active and transductive learning settings focusing on SVM learning. The value of unlabeled data is evaluated using Fisher information matrices. It is proved that selecting unlabeled data *with low confidence* (i.e. small margin) that is not subsumed by (i.e., not too close to) previous choices is likely to cause a large change in the model (once the true label is known). A few numerical examples presented in this paper show that an SVM-based active learner using this criterion is superior to random selection.

Roy and McCallum (2001) offer a pool-based active-learning algorithm that attempts to "directly" optimize the true generalization error rate (rather than reduce the version space). The algorithm chooses to query points that strengthen the belief of the current classifier about the pool classification. The learner estimates its error on the unlabeled pool, where the current classifier's posterior class probabilities are taken as proxies for the true ones. For each example in the pool and for each possible label, the expected loss of a classifier trained after adding this example to the training set is calculated. The example from the pool with the lowest total expected loss is chosen as the next query. The computational complexity required to implement this scheme is "hopelessly impractical," as noted by the authors. However, various heuristic approximations and optimizations are suggested. Some of these optimizations (for example, subsampling) are general and some are designed for the specific implementation provided by Roy and McCallum (2001), which is based on Naive Bayes. The authors report that the resulting active learner exhibits excellent performance over text categorization problems.[6] In Section 4 we further elaborate on the two pool-based algorithms mentioned above.

Active learning has also been explored in the contexts of regression and probabilistic estimation. Saar-Tsechansky and Provost (2002) examine the use of active learning for *class probability estimation* (in contrast to exact, or 'hard' classification). The algorithm proposed is based on creating a number of subsamples from the training set and training one soft classifier (capable of giving class probability estimates) on each subsample. All these generated classifiers form an "ensemble". The querying function works by drawing a pool point according to a probability distribution that is proportional to the variance of the probability estimates computed by the various ensemble members. This variance is normalized by the average probability of the minority class. Empirical results indicate that this algorithm performs better than random sampling (both as a soft as well as a hard classifier).

In the context of regression, Cohn et al. (1996) assume that the learner is approximately unbiased and propose an active-learning algorithm that chooses to query the instance that minimizes the average expected variance of the learner (integrated over the input domain). This algorithm is outlined for general feed-forward neural networks, a mixture of Gaussians and locally weighted regression. For the latter two learning schemes, this method is efficient. However, this method ignores the learner's bias. The algorithm requires a closed form calculation of the learner's variance, which has been developed only for the three learning schemes considered.

In the context of ensemble methods for regression, Krogh and Vedelsby (1995) show a nice equality giving the (squared error loss) regression generalization error as a (weighted) sum of the generalization errors of the individual members minus a diversity measure called 'ambiguity', which is related to the anti-correlation between ensemble members. Increasing this ambiguity (without increasing the weighted sum of individual errors) will reduce the overall error. Increasing the ambiguity is done by dividing the training set into a number of disjoint subsets and training one ensemble member over a distinct training subset. One nice property of the ambiguity measure is that it can be computed without the labels. Krogh and Vedelsby (1995) apply this relation (which guarantees that the weighted ambiguity is a lower bound on the weighted sum of individual ensemble generalization errors) in the context of pool-based active learning as follows. Assuming that the most beneficial pool point to sample is the one that increases the weighted sum of individual ensemble errors, they search for the point whose contribution to the ensemble ambiguity is maximal.

We now turn to discuss some approaches to measuring the "value" of *labeled* points. Such methods are potentially useful for obtaining information on the progress of active learners, and may be useful for implementing our approach that combines an ensemble of active learners. Within a Bayesian setting, MacKay (1992) attempts to measure the information that can be gained about the unknown target hypothesis using a new *labeled* point. Two gain measures are considered, both based on Shannon's entropy. It is assumed that given a training set, a probability distribution over the possible hypotheses is defined. The first measure is how much the entropy of the hypothesis distribution has decreased using a new labeled point. A decrease in this entropy can indicate how much the support of this hypothesis distribution shrinks. The second measure is the cross-entropy between the hypothesis distributions before and after the new labeled point is added. An increase in this cross-entropy can indicate how much the support has changed due to the new information. It is proved that these two gain measures are equivalent in the sense that their expectations are equal.

6. Our implementation of this algorithm, as discussed in Section 4.2, uses SVMs rather than Naive Bayes and only utilizes the subsampling approximation idea suggested by Roy and McCallum (2001).

Guyon et al. (1996) propose methods to measure the information gain of labeled points in a data set. The information gain of a labeled point with respect to a trained probabilistic (soft) classifier is defined as the probability that the classification of this point is correct. This probability equals the Shannon "information gain" of the class probability of that point as measured by the probabilistic classifier. On a given labeled data set, the information gain of each point can be measured using the above definition with a leave-one-out estimator. The paper also addresses the information gain of labeled points in the context of minimax learning algorithms such as SVM. The SVM measures the information gain of a labeled point by its ("alpha") coefficient in the induced weights vector. The coefficient of a point that is not a support vector is zero, and so is its information gain. Support vectors with larger coefficients are more informative. These ideas are discussed in the context of "data cleaning" of large databases. We note that, in the context of SVM active learning it is not in general the case that support vectors identified during early trials will remain support vectors thereafter.

Finally, we note that active learning is only one way of exploiting the availability of unlabeled data. In recent years the broader topic of "semi-supervised" or "learning with labeled-unlabeled examples" has been gaining popularity. One can distinguish between inductive semi-supervised methods and transductive ones (Vapnik, 1998); see a recent survey by Seeger (2002) for a recent review.

4. On Three Active-Learning Algorithms

The main purpose of this section is to motivate the challenge considered in this work. In contrast to what may be implied in various recent papers in the field, when closely examining state-of-the-art active-learning algorithms on various learning problems, there is no consistent single winner. While not necessarily surprising, it remains unclear how to choose the best active-learning algorithm for the problem at hand.

To demonstrate this point we focus on two known active-learning algorithms: SIMPLE (Tong and Koller, 2001, Schohn and Cohn, 2000, Campbell et al., 2000) and the algorithm of Roy and McCallum (2001), which we call here SELF-CONF. We selected these algorithms because they are both reasonably well motivated, achieve high performance on real-world data (and surpass various other known algorithms) and appear to complement each other. In particular, as we show below, neither algorithm consistently tops the other.[7] Moreover, both algorithms fail in learning problems with "XOR-like" structures (in the sense that they perform considerably worse than random sampling).[8] We thus consider another novel algorithm, which excels on 'XOR' problems but exhibits quite poor performance on problems with simple structures. Altogether, these three algorithms form the ensemble on which our new "master" algorithm is later experimentally applied (Section 8). Throughout the rest of the paper we assume basic familiarity with SVMs.[9]

4.1 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

7. To the best of our knowledge, no previous attempts to compare these two algorithms have been conducted.
8. This deficiency (of SIMPLE) was already observed by Tong and Koller (2001).
9. Good textbooks on the subject were written by Cristianini and Shawe-Taylor (2000) and Schölkopf and Smola (2002).

decision boundary of C_{t-1}. Thus, the querying function can be computed in time linear in $|\mathcal{U}|$ (assuming the number of support vectors is a constant).

An intuitive interpretation of SIMPLE is that it chooses to query the instance whose label is the "least certain" according to the current classifier. A more formal theoretical motivation for SIMPLE is given by Tong and Koller (2001) in terms of *version space* bisection. This argument utilizes a nice duality between instances and hypotheses. The *version space* (Mitchell, 1982) is defined as the set of all hypotheses consistent with the training set. Let $\mathcal{L} = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ be a training set, and let $\mathcal{H}$ be the set of all hypotheses (in our case $\mathcal{H}$ is the set of all hyperplanes in kernel space). The version space $\mathcal{V}$ is defined as

$$\mathcal{V} = \{h \in \mathcal{H} | \forall (x,y) \in \mathcal{L}, h(x) = y\}.$$

Let w^* be a hyperplane constructed in kernel space by training an SVM on the entire pool with the true labeling (clearly w^* cannot be computed by an active learner without querying the entire pool). A good strategy for an active learner would be to choose a query that bisects the version space as it brings us faster to w^*. The question is how to find the instance that approximately bisects the version space. The version space can be viewed as a subsurface of the unit hypersphere in parameter space (the entire surface contains all possible hypotheses). Every unlabeled instance x corresponds to a hyperplane $\Phi(x)$ in the parameter space. The hypotheses $\{w|(w \cdot x) > 0\}$ that classify x 'positively' lie on one side of this hyperplane and the hypotheses $\{w|(w \cdot x) < 0\}$ that classify x 'negatively' lie on the other side. Let w be a hyperplane constructed in kernel space by training an SVM on the current set of labeled instances. Tong and Koller (2001) show that w is the center of the largest hypersphere that can be placed in version space and whose surface does not intersect the hyperplanes corresponding to the labeled instances. Moreover, the radius of this hypersphere is the margin of w and the hyperplanes that "touch" this hypersphere are those corresponding to the support vectors. Therefore, in cases where this hypersphere sufficiently approximates the version space, w approximates the center of mass of the version space. The closer an unlabeled instance is to w, the closer the hyperplane $\Phi(x)$ is to the center of mass of the version space. This motivates the strategy of querying the closest instance to w, as it can bisect the version space and quickly advance the algorithm toward the (unknown) target hypothesis w^*.

Tong and Koller (2001) require that each of the classifiers (i.e., SVMs) computed during the operation of SIMPLE is consistent with its training set. This is essential to guarantee a non-vacuous version space. This requirement is easy to fulfil using the "kernel trick" discussed by Shaw-Taylor and Christianini (1999), which guarantees that the training set is linearly separable in kernel space. Further details are specified in Appendix A.

When considering simple binary classification problems, this strategy provides rapid refinement of the decision boundary and results in very effective active learning (Tong and Koller, 2001, Schohn and Cohn, 2000, Campbell et al., 2000). In our experience, algorithm SIMPLE is a high-performance active-learning algorithm, which is hard to beat on a wide range of real-world problems. In particular, this algorithm performs quite well on text categorization problems (Tong and Koller, 2001).

We note that two more querying functions (called "Maxmin" and "Maxratio") are proposed by Tong and Koller (2001). These are also based on the version space bisection principle and achieve more efficient version space reduction than SIMPLE. The main drawback of these functions is their computational intensity, which makes them impractical. Specifically, for each query, the computation of each of these functions requires the induction of two SVMs for each unlabeled point in the pool.[10]

4.2 Algorithm SELF-CONF

The second algorithm we discuss, proposed by Roy and McCallum (2001), is based on a different motivation. The algorithm chooses its next example to be labeled while "directly" 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 utilizes its current classifier for probability measurements. Throughout this paper we therefore call this algorithm SELF-CONF.[11]

At the start of each trial this algorithm already holds a trained probabilistic (soft) classifier denoted 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"[12] defined to be

$$E(\hat{P}'_{\mathcal{L}'(x,y)}) = -\frac{1}{|\mathcal{U}|} \sum_{y' \in \mathcal{Y}, x' \in \mathcal{U}} \hat{P}'(y'|x') \log \hat{P}'(y'|x'). \tag{2}$$

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 self-estimated expected loss is then chosen to be queried.

The original SELF-CONF algorithm proposed by Roy and McCallum (2001) bases its probability estimates (and classification) on Naive Bayes calculations[13] and is shown to outperform other known active-learning algorithms on text categorization tasks. In our studies, we used an SVM-based variant of SELF-CONF, which appears to be stronger than the original Naive Bayes algorithm.[14] Thus, in our case, probabilistic estimates are obtained in a standard way, using logistic regression.[15] Also note that the algorithm, as presented, is extremely inefficient, since for each query, two SVMs are induced for each unlabeled point in the pool. Various optimizations and approximations are proposed by Roy and McCallum (2001), to make its running time practically feasible. From all these methods we only use random subsampling in our implementation of the algorithm: On each trial we estimate the expression (2) for only a random subset of $\mathcal{U}$. The subsample in the first active session trial contains 100 points; on each subsequent trial we decrement the subsample size by one point until we reach a minimum of 10 points, which we keep for the remaining trials.

Observe that the motivation for SELF-CONF and SIMPLE is in some sense complementary. While SIMPLE queries instances in $\mathcal{U}$ with the least confidence, as measured by its current model, SELF-CONF queries instances that provide the maximal "reassurance" of its current model.

10. Approximation heuristics such as subsampling of the pool as well as using incremental/decremental SVM algorithms (Cauwenberghs and Poggio, 2000) can be used to speed up these algorithms. We have not explored these possibilities in the present work.
11. This self-confidence approach is somewhat similar to the one proposed by Lindenbaum et al. (2004).
12. Roy and McCallum (2001) also considered using zero-one loss instead of log-loss.
13. The empirical results of Roy and McCallum (2001) considered multinomial models (in particular, text categorization), which enabled the required calculations.
14. However, we have not performed an extensive comparison between the Naive Bayes and SVM variants of this algorithm.
15. If $h(x)$ is the (confidence-rated) SVM value for a point x whose label is y, we take $\hat{P}(y|x) = \frac{1}{1+\exp\{-yh(x)\}}$.

4.3 Algorithm Kernel Farthest-First (KFF)

Here 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 approximate optimal clustering for k-center problems (Hochbaum and Shmoys, 1985). The FF traversal of the points in a data set is defined as follows. Start with any point x and find the farthest point from x. Then find the farthest point from the first two (where the distance of a point from a set is defined to be the minimum distance to a point in the set), etc. In any metric space, FF traversals can be used for computing 2-approximation solutions to the *k-center clustering* problem in which one seeks an optimal k-clustering of the data and optimality is measured by the maximum diameter of the clusters. In particular, by taking the first k elements in the FF traversal as "centroids" and then assigning each other point to its closest "centroid", one obtains a k-clustering whose cost is within a factor 2 of the optimal (Hochbaum and Shmoys, 1985).

We use FF traversals for active learning in the following way. 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 $\mathcal{L}$. Using $\mathcal{L} \cup \{(x,y)\}$ as a training set, the active learner then induces the next 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 observed. 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. For compatibility with the SVM, we compute the FF traversals in kernel space as follows. Given any kernel K, if x and y are instances in input space, and $\Phi(x)$ and $\Phi(y)$ are their embedding in kernel space (so that $\langle \Phi(x), \Phi(y) \rangle = K(x,y)$), then we measure $d_K(x,y)$, the distance between x and y in kernel space, as $d^2(x,y) = \|\Phi(x) - \Phi(y)\|^2 = K(x,x) + K(y,y) - 2K(x,y)$. We call the resulting algorithm KFF.

4.4 Some Examples

Figure 2 shows the learning curves of SIMPLE, SELF-CONF, KFF and of a random sampling "active learner" (which we call RAND)[16] for the artificial 'XOR' problem of Figure 2 (top left) as well as two other UCI problems - 'Diabetis' and 'Twonorm'. These three learning problems demonstrate that none of the three active-learning algorithms discussed here is consistently better than the rest. In the two UCI problems SIMPLE performs better than SELF-CONF on the 'Twonorm' data set, while SELF-CONF performs better than SIMPLE on the 'Diabetis' data set. KFF is inferior to the other two in both cases. In the 'XOR' problem SIMPLE's and SELF-CONF's performances are significantly worse than that of random sampling. 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 typically occurs when confronting problems with "XOR-like" structure. The inferiority of KFF relative to SIMPLE and SELF-CONF (and even RAND) typically occurs in learning problems with a "simple" structure. The main advantage in considering KFF is its use within an ensemble of active-learning algorithms. The "master" active algorithm we present later benefits from KFF in "XOR-like" problems without significant compromises in problems where KFF is weaker.

16. The querying function of RAND chooses the next example to be labeled uniformly at random from $\mathcal{U}$.

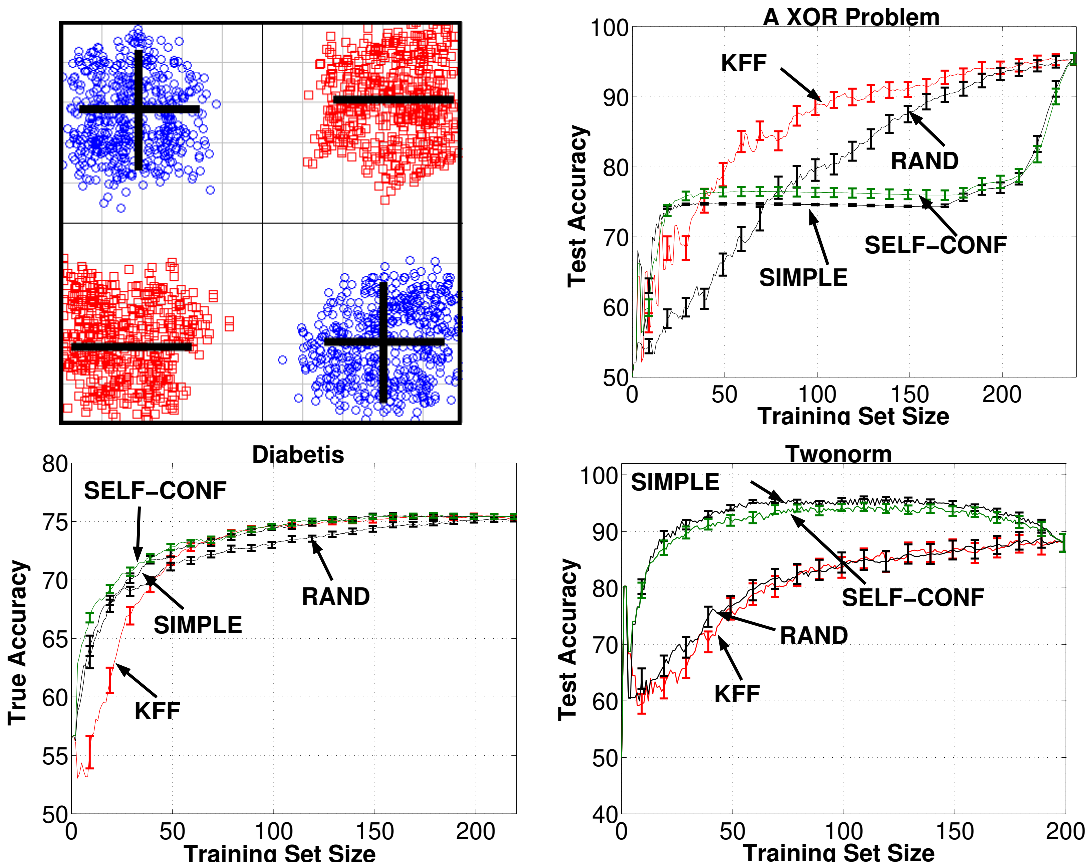


Figure 2: **Top Left**: 'XOR' problem consisting of 1000 points (250 in each "cluster"); 250 points form the pool and the rest form the test set. **Top Right**: Learning curves for SIMPLE, SELF-CONF, KFF and RAND for the 'XOR' problem where KFF exhibits superior performance. **Bottom left**: Learning curves for the same four learners on the 'Diabetis' data set in which SELF-CONF is superior. **Bottom Right**: Learning curves for the same four learners on the 'Twonorm' data set in which SIMPLE is superior. 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.

5. The Multi-Armed Bandit (MAB) Problem

A major ingredient in our online learning approach to active learning is a known competitive algorithm for the multi-armed bandit problem. In this section we present this problem, its connection to online choice of active learners and the particular multi-armed bandit algorithm we use.

In the ***multi-armed bandit (MAB) problem***, a gambler must choose one of k 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 the sequence. This classic problem represents a fundamental predicament whose essence is the tradeoff between *exploration* and *exploitation*: Sticking with any single machine may prevent discovering a better machine; on the other hand, continually seeking a better machine will prevent achieving the best total reward.

We make the following straightforward analogy between the problem of online combining of an ensemble of active learners and the MAB problem. The k active-learning algorithms in our ensemble are the k 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. In particular, one of the main obstacles in using MAB algorithms for combining active learners is how to define the reward of a query. The optimal reward might be the true accuracy achieved by the learner using the augmented training set, however this accuracy is unknown to us. We therefore have to estimate it in some way. The reward estimation technique we use is developed in Section 6. For the rest of this section we assume we have some (non-negative and bounded) reward function.

Most known MAB algorithms and their analyses assume that rewards are distributed according to certain statistical models. Typically, simple statistical models including independence and time invariance assumptions are taken (for example, i.i.d. Gaussian rewards). In our active-learning context (using, for example, the above analogy), overly simplistic statistical assumptions on reward distributions are likely to be violated. Therefore, it is particularly useful to use, in our active-learning context, the *adversarial* MAB results of Auer et al. (2002), which 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 developed by Auer et al. (2002) are potentially useful for implementing online choice of active learners. The first algorithm is called EXP3 (see Section 3 in Auer et al., 2002) and directly matches the above analogy. The second algorithm, called EXP4 (see Section 7 in Auer et al., 2002), appears to be more suitable for our purposes (see below). We first describe the EXP3 algorithm and then describe the EXP4 algorithm which we have chosen to use.

EXP3 is an algorithm for the standard MAB game, where n slot machines are played. On each trial t, one slot machine $i, i = 1, \ldots, n$, is chosen and played, yielding a reward $g_i(t)$, where $g_i(t)$ is non-negative and bounded. For a game consisting of T trials, define $G_{\text{MAX}} = \max_{1 \le i \le n} \sum_{t=1}^{T} g_i(t)$, the expected reward of the best slot machine in hindsight. The goal of the online player in this game is to achieve a reward as close as possible to G_{MAX}. Let G_{EXP3} be the expected reward of this algorithm. The "regret" of EXP3 is defined to be $G_{\text{MAX}} - G_{\text{EXP3}}$. Given an upper bound $g \ge G_{\text{MAX}}$, it is guaranteed (Auer et al., 2002) that the regret of EXP3 is bounded above by

$$G_{\text{MAX}} - G_{\text{EXP3}} \le 2.63\sqrt{gn\ln n}.$$

This bound holds for any assignment of rewards and for any number of trials T. We also note that this bound holds for rewards in the range of $[0, 1]$. In our algorithm we also used rewards in this range (see Section 7).

The problem of providing the upper bound $g \ge G_{\text{MAX}}$ is solved using a standard doubling algorithm that guesses a bound and runs the EXP3 algorithm until this bound is reached. Once reached, the guessed bound is doubled and EXP3 is restarted with the new bound. The guaranteed bound on the regret of this algorithm is

$$G_{\text{MAX}} - G_{\text{DBL-EXP3}} \le 10.5\sqrt{G_{\text{MAX}} n \ln n} + 13.8n + 2n\ln n,$$

where $G_{\text{DBL-EXP3}}$ is the expected reward of the doubling algorithm. Once again, this bound holds for any reward function and for any number of trials.

Combining active learners using EXP3 can be done in the following manner. Each one of the n active-learning algorithms in the ensemble is modelled as a slot machine. On each trial t, one active learner is chosen to provide the next query. The reward of this query is associated with the chosen learner.[17] In this modeling approach, the points are not at all considered by EXP3 and each active learner is a slot machine "black box". Thus, this approach does not directly utilize some useful information on the unlabeled points that may be provided by the active learners.

The other algorithm of Auer et al. (2002) called EXP4 is designed to deal with a more sophisticated MAB variant than the above (standard) MAB game. Here the goal is to combine and utilize a number k of *strategies* or *experts*, each giving "advice" on how to play the n slot machines. To employ EXP4 in our context, we associate the k experts with the ensemble of k active-learning algorithms. The slot machines are associated with the unlabeled instances in our pool $\mathcal{U}$. Using this approach for modeling active learning has a considerable benefit since now the choice of the next query is directly based on the opinions of all ensemble members.

Algorithm EXP4 operates as follows. On each trial t, each expert j, $j = 1,\ldots,k$, provides a weighting $\mathbf{b}^j(t) = (b_1^j(t),\ldots,b_n^j(t))$ with $\sum_i b_i^j(t) = 1$, where $b_i^j(t)$ represents the confidence (probability) of expert j for playing the ith machine, $i = 1,\ldots,n$, on trial t. Denoting the vector of rewards for the n machines on trial t by $\mathbf{g}(t) = (g_1(t),\ldots,g_n(t))$, where $g_i(t)$ is non-negative and bounded, the expected reward of expert j on trial t is $\mathbf{b}^j(t)\cdot\mathbf{g}(t)$. Note that 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 $G_{\text{MAX}} = \max_{1\le j\le k}\sum_{t=1}^{T}\mathbf{b}^j(t)\cdot\mathbf{g}(t)$, 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}. Let G_{EXP4} be the expected reward of this algorithm. Given an upper bound $g \ge G_{\text{MAX}}$, the "regret" of EXP4, defined to be $G_{\text{MAX}} - G_{\text{EXP4}}$, is bounded above by

$$G_{\text{MAX}} - G_{\text{EXP4}} \le 2.63\sqrt{gn\ln k}.$$

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 confidence vector for n slot machines. The problem of providing the upper bound $g \ge G_{\text{MAX}}$ is solved by employing the same 'guess and double' technique used with EXP3. The guaranteed bound on the regret of this algorithm is

$$G_{\text{MAX}} - G_{\text{DBL-EXP4}} \le 10.5\sqrt{G_{\text{MAX}}n\ln k} + 13.8n + 2n\ln k,$$

where $G_{\text{DBL-EXP4}}$ is the expected reward of the doubling algorithm.

As mentioned above, we use EXP4 for active learning by modeling the k active-learning algorithms as k experts and the unlabeled pool points in $\mathcal{U}$ as the slot machines. Note that for the performance bound of EXP4 to hold we must include RAND in our pool of active learners, which corresponds to the "uniform expert". This modeling requires that expert advice vectors are probabilistic recommendations on points in $\mathcal{U}$. It is thus required that each algorithm in the ensemble will provide "rating" for the entire pool on each trial. In practice, the three algorithms we consider naturally provide such ratings, as discussed in Section 7.

17. The alternative possibility of modeling the pool samples as the slot machines clearly misses the 'exploitation' dimension of the problem since once we choose a sample, we cannot utilize it again. In addition, it is not clear how to utilize an ensemble of learners using this approach.

6. Classification Entropy Maximization

In order to utilize the MAB algorithm (EXP4, see Section 5) in our context, we need to receive, after each trial, the gain of the instance $x \in \mathcal{U}$ that was chosen to be queried (corresponding to one slot machine). While the ultimate reward in our context should be based on the *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 entirely the case and, unless $\mathcal{L}$ (and $\mathcal{U}$) is sufficiently large, standard methods such as cross-validation or leave-one-out provide substantially biased estimates of an active learner's performance. This fact, which was already pointed out by Schohn and 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 3 (left). The figure shows leave-one-out (LOO) estimates of four (active) learning algorithms: SIMPLE, SELF-CONF, KFF and RAND on the UCI 'Ringnorm' data set. For each training set size, each point on a curve is generated using the LOO estimate based on the currently available labeled set $\mathcal{L}$. In Figure 3 (middle) we see the "true accuracy" of these algorithms as estimated by an independent test set. Not only does LOO severely fail to estimate the true accuracy, it even fails to order the algorithms according to their relative success as active learners. This unfortunate behavior is quite typical of LOO on many data sets, and afflicts other standard error estimation techniques, including cross-validation. These techniques should, therefore, be avoided in general for receiving feedback on the (relative) progress of active learners, especially if one cares about the active learner's performance using a small number of labeled points.

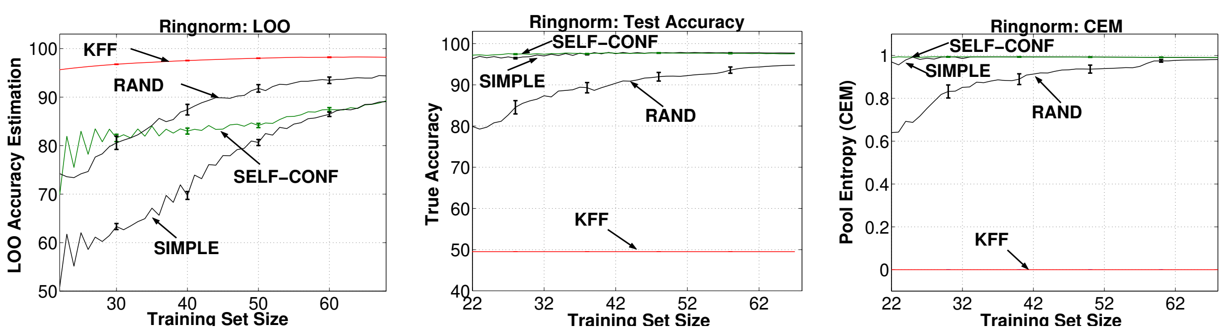


Figure 3: **Left**: LOO estimates of active-learning sessions of SIMPLE, SELF-CONF, RAND and KFF over the 'Ringnorm' data set; **Middle**: The "true" accuracy of these algorithms, as estimated using a test set; **Right**: CEM entropy scores of these algorithms. All estimates are averages of 100 folds. Error bars (diluted to reduce clutter) represent one standard error of the mean.

We propose the following semi-supervised estimator, which we call *Classification Entropy Maximization (CEM)*. 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. Specifically, 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, as determined by C. Then, the CEM score of C (with respect to $\mathcal{U}$) is the binary entropy $H(\frac{|C^{+1}(\mathcal{U})|}{|\mathcal{U}|})$.[18] It is not difficult to see that the CEM score is larger if the division of the pool classification between classes is more balanced. Figure 3 (right) provides CEM curves for the four active learners discussed above. Clearly, the CEM measure orders the algorithms in accordance with their true accuracy as depicted in Figure 3 (middle), and moreover, if we ignore the scales, all CEM curves in this example appear surprisingly similar to the corresponding true accuracy curves. This behavior of CEM is typical in many of our empirical examinations, and somewhat surprisingly, CEM succeeds in correctly evaluating performance even when the positive and negative priors are not balanced. A more comprehensive discussion and analysis of the CEM criterion is provided later in Section 9.

Remark 1 *It is interesting to note that the CEM criterion is also useful for SVM model selection in (semi-)supervised learning settings where the (labeled) training set is small and some set of unlabeled points is available. In Appendix B we present an empirical comparison of CEM and LOO in this setting.*

7. Combining Active Learners Online

In this section we describe our master algorithm for combining active learners. The combination algorithm, called here for short COMB, is based on the EXP4 multi-armed bandit (MAB) algorithm discussed in Section 5 and on the CEM criterion presented in Section 6. In Figure 4 we provide an annotated pseudocode of COMB. The algorithm utilizes an ensemble of active-learning algorithms and tracks online the best algorithm in the ensemble. Many of the steps in this code are adapted from the EXP4 MAB algorithm of Auer et al. (2002) (in particular, steps 4,5,6,11,12). We refer the reader to Auer et al. (2002) for a more detailed exposition (and the proof of the performance guarantee) of EXP4. Here we elaborate on steps 1,2,3 and 10, which require further explanation (the remaining steps, 7,8 and 9, are self-explanatory).

In steps 1 and 2, we compute advice probability vectors of the active learners as required by the original EXP4 algorithm. Each algorithm in the ensemble provides (in Step 1) a scoring vector that "rates" each point in the pool $\mathcal{U}$. In practice, the three algorithms we consider naturally provide such ratings: SIMPLE uses the kernel distance from the decision hyperplane, SELF-CONF uses the expected loss and KFF uses the kernel distance from the current training set. We scale these scoring vectors (in Step 2) using a scaling parameter β and a Gibbs probability function $\exp\{-\beta x\}$. In all our experiments we used $\beta = 100$ (a sensitivity analysis of this parameter is provided in Section 8).

In Step 3, after producing (in Step 2) the advice probability vectors for the active learners, we project the pool $\mathcal{U}$ over high probability candidate instances. The projected pool is denoted $\mathcal{U}_e$. 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 learner assigns to x a probability mass greater than α. In all the experiments described below, we used $\alpha = 0.05$ (here again, a sensitivity analysis of this parameter is provided in Section 8).

In Step 6, the learner chooses one (unlabeled) point x_q from $\mathcal{U}_e$ as the next query. According to EXP4, this choice should be random according to the distribution computed in Step 5. In practice, in the experiments described in Section 8, we greedily picked the point with the largest probability

18. The binary entropy of a (Bernoulli) random variable with bias p is $H(p) = H(1-p) = -p\log(p) - (1-p)\log(1-p)$.

Algorithm COMB

Input: (i) A Pool $\mathcal{U} = \{x_1, \ldots, x_n\}$; (ii) An ensemble $\{\text{ALG}_j\}_{j=1}^k$ of k active learners;
(iii) An initial training set $\mathcal{L}_0$
Parameters: (i) A probability threshold α; (ii) A probability scaling factor β;
(iii) A bound g_{max} on the maximal reward
Init: Initialize expert weights: $w_j = 1$, $j = 1, \ldots, k$

For $t = 1, 2, \ldots$

1. Receive advice scoring vectors from ALG_j, $j = 1, \ldots, k$: $\mathbf{e}^j(t) = (e_1^j(t), \ldots, e_n^j(t))$. $e_i^j(t)$ is the score of the ith point in the pool. The scores are normalized to lie within [0,1].
2. For each ALG_j, $j = 1, \ldots, k$, we compute an advice probability vector $\mathbf{b}^j(t) = (b_1^j(t), \ldots, b_n^j(t))$ by scaling the advice scoring vectors. For each $\mathbf{b}^j(t)$, $j = 1, \ldots, k$, for each $b_i^j(t)$, $i = 1, \ldots, n$: $b_i^j(t) = (\exp\{-\beta(1 - e_i^j(t))\})/Z$, where Z normalizes $\mathbf{b}^j(t)$ to be a probability vector.
3. Extract from $\mathcal{U}$ an "effective pool" $\mathcal{U}_e$ by thresholding low probability points: For each point $x_i \in \mathcal{U}$ leave x_i in $\mathcal{U}_e$, iff $\max_j b_i^j \geq \alpha$. If $|\mathcal{U}_e| = 0$, reconstruct with $\alpha/2$, etc. Set $n_e = |\mathcal{U}_e|$.
4. Set $\gamma = \sqrt{\frac{n_e \ln k}{(e-1)g_{max}}}$.
5. Set $W = \sum_{j=1}^k w_j$ and for $i = 1, \ldots, n_e$, set $p_i = (1 - \gamma) \sum_{j=1}^k w_j b_i^j(t)/W + \gamma/n_e$.
6. Randomly draw a point x_q from $\mathcal{U}_e$ according to $p_1, \ldots, p_{n_e}$.
7. Receive the label y_q of x_q from the teacher and update the training set and the pool: $\mathcal{L}_t = \mathcal{L}_{t-1} \cup \{(x_q, y_q)\}$; $U_{t+1} = U_t \setminus \{x_q\}$
8. Train a classifier C_t using $\mathcal{L}_t$.
9. Use C_t to classify all points in $\mathcal{U}$ and calculate $H_t = H_t(\frac{|C^{+1}(\mathcal{U})|}{|\mathcal{U}|})$, the entropy of the resulting partition $C^{+1}(\mathcal{U}), C^{-1}(\mathcal{U})$ (as in the CEM score; see Section 6).
10. Calculate the "reward utility" of x_q:
$r(x_q) = ((e^{H_t} - e^{H_{t-1}}) - (1 - e))/(2e - 2)$.
11. For $i = 1, \ldots, n$, set $\hat{r}_i(t) = r(x_q)/p_q$ if $i = q$ and $\hat{r}_i(t) = 0$ otherwise.
12. Reward/punish experts:
$w_j(t+1) = w_j(t) \exp(\mathbf{b}^j(t) \cdot \hat{r}(t)\gamma/n_e)$.

Figure 4: Algorithm COMB

(and in case of ties we randomly chose one of the points). This deterministic implementation is useful for reducing the additional variance introduced by these random choices.[19]

In Step 10 we calculate the "utility" of the last query. This utility is defined using the (convex) function e^x on the entropic reward calculated in Step 9 (that is, the CEM score discussed in

19. We also experimented with random query choices, as prescribed by EXP4. A slight advantage of the deterministic variant of the algorithm was observed.

Section 6). The utility function is essentially the difference

$$\Delta_t = e^{H_t} - e^{H_{t-1}},$$

where 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. Clearly, this function emphasizes entropy changes in the upper range of possible entropy values. The rationale behind this utility function is that it is substantially harder to improve CEM scores (and also accuracy) that are already large. The additional transformation $(\Delta_t - (1-e))/(2e-2)$ normalizes the utility to be in [0,1].

The reward bound parameter g_{max}, used for setting an optimal value for γ (Step 4) can be and is eliminated in all our experiments using a standard "guess and double" technique. In particular we operate the COMB algorithm in rounds $r = 1, 2, \ldots$, where in round r we set the reward limit to be $g_r = (n_e \ln k/(e-1))4^r$ and restart the COMB algorithm 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 the discussion on the EXP3.1 algorithm in Section 4 of Auer et al. (2002).

8. Empirical Evaluation of the COMB Algorithm

We evaluated the performance of the COMB algorithm on the entire benchmark collection selected and used by Rätsch et al. (2001), consisting of 13 binary classification problems extracted from the UCI repository. For almost all problems, this collection includes fixed 100 folds each consisting of a fixed 60%/40% training/test partition.[20] The use of this set is particularly convenient as it allows for easier experimental replication. To this collection we also added our artificial 'XOR' problem (see Section 4.4).[21] Some essential characteristics of all these data sets appear on Table 1. For each problem we specify its size, its dimension, the bias (proportion of largest class), the maximal accuracy achieved using the entire pool as a training set, the (rounded up) number of instances required by the *worst* learner in the ensemble to achieve this maximal accuracy and, finally, the average fraction of support vectors (from the entire pool size) utilized by an SVM trained over the entire pool (this average is computed over 100 folds).

In all the experiments described below, each active learner is provided with two initial examples (one from each class) for each learning problem. These two examples were randomly chosen among all possible pairs. The same initial training set (pair) was given to all learners. All active-learning algorithms applied in our experiments used an SVM with RBF kernel as their learning component ($\mathcal{A}$). More particular implementation details, which are essential for replication, are given in Appendix A.

In Figure 5 we depict the learning curves of COMB and its ensemble members obtained on four data sets. Notice that for three of these data sets a different ensemble member is the winner. In the 'Image' data set it is SELF-CONF, in the 'XOR' data set it is KFF and in the 'Waveform' data set it is SIMPLE.[22] However, in all of these cases COMB tracks the winner. In the fourth data set

20. Two data sets in this collection ('Image' and 'Splice') include only 20 folds.
21. For this 'XOR' data set, consisting of 1000 points, we constructed 100 folds by randomly and uniformly splitting the data to 25%/75% training/test partitions.
22. In the 'Waveform' data set the accuracy decreases from a certain point. When there is noise in the data one may benefit by stopping early. See a discussion of this phenomenon by Schohn and Cohn (2000).

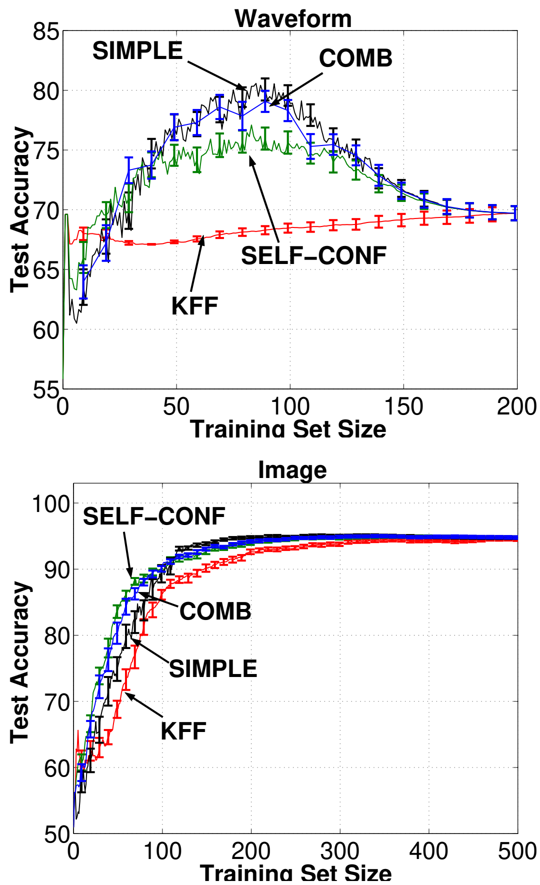


Figure 5: Learning curves of COMB and its ensemble members on four data sets. All estimates are averages over 100 folds. Error bars (diluted to reduce clutter) represent one standard error of the mean. **Top Left**: 'Waveform' data set, where COMB performs almost as well as the winner SIMPLE. **Top Right**: 'XOR' data set, where COMB performs better than the winner KFF. **Bottom Left**: 'Image' data set, where COMB performs almost as well as the winner SELF-CONF. **Bottom Right**: 'Flare-Solar' data set, where COMB is the overall winner and significantly beats its ensemble members.

presented, the 'Flare-Solar' data set, COMB is the overall winner and significantly beats its three ensemble members.

Table 2 shows the average *deficiency* of COMB and its ensemble members for all these data sets. Recall the definition of deficiency of an active learner as given in Equation (1), where smaller values represent a larger active-learning *efficiency*. Each of these averages is calculated over the corresponding 100 folds. It is evident that none of the ensemble algorithms is consistently outdoing all sets (SELF-CONF, SIMPLE, and KFF win in 7, 4 and 3 cases, respectively). Nevertheless, SELF-CONF is the most dominant algorithm, winning on half of the data sets. In many cases KFF performs poorly and it is often inferior to RAND. Overall, this algorithm is significantly worse than

Data Set	Size	Dim	Bias	Max Acc. (%) (Sample Size)	SV proportion (%)
Banana	5300	2	0.551	88.22 (200)	14.3 ± 0.2
Breast-Cancer	277	9	0.707	73.60 (100)	45.8 ± 1.07
Diabetis	768	8	0.651	74.56 (220)	38.5 ± 1.11
Flare-Solar	1066	9	0.552	63.76 (200)	15.08± 0.4
German	1000	20	0.700	75.48 (330)	53.00 ± 0.56
Heart	270	13	0.555	82.33 (82)	41.29 ± 0.68
Image	2310	18	0.571	95.00 (500)	46.32 ± 1.17
Ringnorm	8300	20	0.549	97.96 (200)	79.0±0.3
Splice	3175	60	0.519	85.86 (450)	74.1±2.13
Thyroid	215	5	0.697	95.53 (69)	26.2±0.8
Titanic	2201	3	0.676	74.14 (74)	17.09±0.32
Twonorm	7400	20	0.500	95.94 (200)	87.45±0.81
Waveform	5000	21	0.670	80.58 (200)	97.78±0.55
XOR	3000	2	0.500	96.35 (240)	2.19±0.01

Table 1: The data sets: some essential characteristics. For each problem we provide the size, the dimension, the bias (proportion of largest class), the maximal accuracy achieved using the entire pool, the (rounded up) number of instances required by the worst learner in the ensemble to achieve this maximal accuracy and the average fraction of the number of support vectors (from the pool size) utilized by an SVM trained over the entire pool (the average is computed over 100 folds).

Data Set	SIMPLE	KFF	SELF-CONF	COMB
Banana	1.13 ± 0.02	**0.73**∗ ± 0.01	**0.74** ± 0.02	**0.74** ± 0.01
Breast-Cancer	**1.06** ± 0.02	1.28 ± 0.03	**0.95**∗ ± 0.03	1.09 ± 0.01
Diabetis	**0.64** ± 0.04	1.07 ± 0.02	**0.48**∗ ± 0.05	0.82 ± 0.04
Flare-Solar	1.13 ± 0.01	**1.09** ± 0.05	1.39 ± 0.07	**0.79**∗ ± 0.05
German	0.71 ± 0.04	0.85 ± 0.01	**0.67** ± 0.02	**0.64**∗ ± 0.02
Heart	**0.57** ± 0.03	1.04 ± 0.01	**0.50**∗ ± 0.03	0.64 ± 0.02
Image	0.54 ± 0.02	0.76 ± 0.01	**0.45**∗ ± 0.01	**0.47** ± 0.01
Ringnorm	**0.34**∗ ± 0.01	4.70 ± 0.4	0.38 ± 0.01	**0.36** ± 0.01
Splice	**0.58** ± 0.01	2.54 ± 0.11	0.60 ± 0.01	**0.57**∗ ± 0.01
Thyroid	**0.55** ± 0.01	2.34 ± 0.09	**0.47**∗ ± 0.01	0.64 ± 0.03
Titanic	0.76 ± 0.04	0.92 ± 0.02	**0.68** ± 0.06	**0.65**∗ ± 0.05
Twonorm	**0.24**∗ ± 0.01	1.03 ± 0.01	0.32 ± 0.02	**0.26** ± 0.01
Waveform	**0.58**∗ ± 0.05	0.97 ± 0.01	0.66 ± 0.04	**0.60** ± 0.05
XOR	1.24 ± 0.07	**0.63** ± 0.02	1.19 ± 0.04	**0.47**∗ ± 0.03

Table 2: Average deficiency (± standard error of the mean) achieved by COMB and its ensemble members. For each data set the winner appears in boldface and is marked with a star. The runner-up appears in boldface.

all the other algorithms. However, as noted above, KFF usually excels in "XOR-like" problems (for example, 'Banana', 'Flare-Solar' and 'XOR').

In 10 cases out of the 14 presented, COMB is either the winner or a close runner-up. In five cases out of these 10 ('Flare-Solar', 'German', 'Splice', 'Titanic' and 'XOR') COMB is the overall winner. A striking feature of COMB is that it does not suffer from the presence of KFF in the ensemble even in cases where this algorithm is significantly worse than RAND, and can clearly benefit from KFF in those cases where KFF excels.

In three cases ('Heart', 'Thyroid' and 'Breast-Cancer'), which are among the smallest data sets, COMB is not the winner and not even the runner-up, although even in these cases it is not far behind the winner. This behavior is reasonable because COMB needs a sufficient number of trials for exploration and then a sufficient number of trials for exploitation. There is only one case, the 'Diabetis' data set, in which COMB completely failed. A closer inspection of this case revealed that our entropy criterion failed in the online choosing of the best ensemble member.

We now turn to examine the possibility of using standard error estimation techniques instead of our CEM. In Table 3 we compare the deficiencies of COMB using (for computing the query rewards) both CEM and standard 10-fold cross validation (10-CV) on the training set. The 10-CV results are obtained by running the COMB algorithm such that in Step 9 of the pseudocode in Figure 4 we calculate the 10-CV of the current training set $\mathcal{L}_t$.[23] Denote this quantity by CV_t. Then in Step 10 we used the same utility function (applied on the CEM estimator) over the CV outcome, $r(x_q) = ((e^{CV_t} - e^{CV_{t-1}}) - (1-e))/(2e-2)$. The table indicates that CEM is more reliable than 10-CV as an online estimator for active learners' performance. 10-CV is very unreliable as it performs well in some cases and very poorly in others. In particular, CEM beats 10-CV on 12 out of 14 data sets, where in some data sets ('Ringnorm' and 'Heart' for example) the difference is quite large. The 10-CV estimator outperforms CEM on two data sets, 'Titanic' and 'Breast-Cancer', where in the first one the difference is small.

Data Set	COMB using CEM	COMB using 10-CV
Banana	$\mathbf{0.74} \pm 0.01$	0.78 ± 0.01
Breast-Cancer	1.09 ± 0.01	$\mathbf{0.99} \pm 0.05$
Diabetis	$\mathbf{0.82} \pm 0.04$	0.88 ± 0.04
Flare-Solar	$\mathbf{0.79} \pm 0.05$	0.90 ± 0.02
German	$\mathbf{0.64} \pm 0.02$	0.66 ± 0.03
Heart	$\mathbf{0.64} \pm 0.02$	0.93 ± 0.01
Image	$\mathbf{0.47} \pm 0.01$	0.48 ± 0.01
Ringnorm	$\mathbf{0.36} \pm 0.01$	0.68 ± 0.03
Splice	$\mathbf{0.57} \pm 0.01$	0.63 ± 0.01
Thyroid	$\mathbf{0.64} \pm 0.03$	0.74 ± 0.03
Titanic	0.65 ± 0.05	$\mathbf{0.61} \pm 0.05$
Twonorm	$\mathbf{0.26} \pm 0.01$	0.45 ± 0.01
Waveform	$\mathbf{0.60} \pm 0.05$	0.79 ± 0.06
XOR	$\mathbf{0.47} \pm 0.03$	0.53 ± 0.02

Table 3: COMB's *deficiency* when operated using CEM and 10-fold cross-validation (10-CV) for computing the query rewards. For each data set the winner appears in boldface.

As defined, the COMB algorithm has two parameters (see Figure 4): the probability threshold α, and the probability scaling factor β. The experimental results presented above were obtained using

23. For training sets of size smaller than 20, we used a leave-one-out (LOO) estimator.

the assignments $\alpha = 0.05$ and $\beta = 100$. These assignments were not optimized and in fact were a priori set to these values, which appeared "reasonable" to us. In the rest of this section we provide a sensitivity analysis of these parameters. This analysis indicates that better performance may be obtained by optimizing the β parameter.

In Table 4 we show a comparison of COMB's performance obtained with different values of the probability threshold parameter α (which determines the size of the "effective pool" computed in Step 3 of the pseudo-code in Figure 4). For the presentation here we selected the four data sets appearing in Figure 5. Recall that each of the first three data sets favors (using our "standard" $\alpha = 0.05$ value) a different ensemble member and in the fourth one (data set 'Flare-Solar'), COMB significantly beats its ensemble members. When applying COMB with $\alpha = 0.01, 0.05, 0.1$ to these four data sets, there is no significant difference in the deficiencies demonstrated by COMB (see Table 4). This result indicates that COMB is not overly sensitive to the choices of α values within this range.

Data Set	COMB with $\alpha = 0.05$ (used in this paper)	COMB with $\alpha = 0.01$	COMB with $\alpha = 0.1$
Flare-Solar	**0.79** $\pm$ 0.05	0.83 $\pm$ 0.02	0.80 $\pm$ 0.04
Image	0.47 $\pm$ 0.01	0.47 $\pm$ 0.01	**0.46** $\pm$ 0.01
Waveform	0.60 $\pm$ 0.05	**0.59** $\pm$ 0.05	0.61 $\pm$ 0.05
XOR	0.47 $\pm$ 0.03	**0.45** $\pm$ 0.03	0.48 $\pm$ 0.03

Table 4: COMB's *deficiency* with different values of the probability threshold parameter α (see Figure 4). For each data set the winner appears in boldface.

In Table 5 we show a comparison of COMB performance with three values of the probability scaling factor parameter β (see Step 2 in Figure 4). Here, again, we use the same four data sets presented in Table 4, to check the sensitivity of COMB with respect to this β parameter (recall that in the experiments above we use $\beta = 100$). Observing the deficiency of COMB operated with $\beta = 10$ and $\beta = 1000$, we see that the performance is dependent on this parameter. However, it is clear that the value we use in the experiments above ($\beta = 100$) was not optimized and the table indicates that $\beta = 10$ yields better results. Hence, here there is room for further improvements, which we have not pursued in this paper.

Data Set	COMB with $\beta = 100$ (used in this paper)	COMB with $\beta = 10$	COMB with $\beta = 1000$
Flare-Solar	0.79 $\pm$ 0.05	**0.68** $\pm$ 0.08	0.80 $\pm$ 0.04
Image	0.47 $\pm$ 0.01	0.48 $\pm$ 0.01	**0.45** $\pm$ 0.01
Waveform	**0.60** $\pm$ 0.05	**0.60** $\pm$ 0.05	0.63 $\pm$ 0.05
XOR	0.47 $\pm$ 0.03	**0.39** $\pm$ 0.03	0.67 $\pm$ 0.02

Table 5: COMB's *deficiency* with different values of β, the probability scaling factor parameter (see Figure 4). For each data set the winner appears in boldface.

9. On the CEM Criterion

While formal connections between CEM and generalization are currently unknown, the rather informal discussion in this section provides further insights into CEM and attempts to characterize conditions for its effectiveness.

A sequence of sets $S_1, S_2, \ldots$ is called an *inclusion sequence* if $S_1 \subset S_2 \subset, \cdots$. Consider an inclusion sequence of training sets. The sequence of training sets generated by an active learner is an inclusion sequence in which S_1 is the initial training set given to the learner. Let $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ 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 (that is, 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 $S_C = (x_1, C(x_1)), \ldots, (x_m, C(x_m))$ is *majority-biased* (with respect to S) if the majority class in S is also a majority class in S_C and its proportion in S_C is larger than or equal to r. Let $I = S_1 \subset \cdots \subset S_T$ be an inclusion sequence of labeled samples. We say that a *learning algorithm* ALG is *majority-biased* (with respect to I) if the classification of S_T by each of the classifiers $C_1, \ldots, C_T$ (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 true accuracy, in which case the CEM criterion can be used to compare online the performance of active learners. In other words, by comparing the growth of a learner's pool classification entropy, we can get a useful indication on the growth of the true accuracy. We next consider a couple of examples that demonstrate this behavior and then consider a negative example in which the majority-bias property does not hold and CEM fails. All our examples consider a setting in which the prior of the majority class is significantly larger than that of the other class.

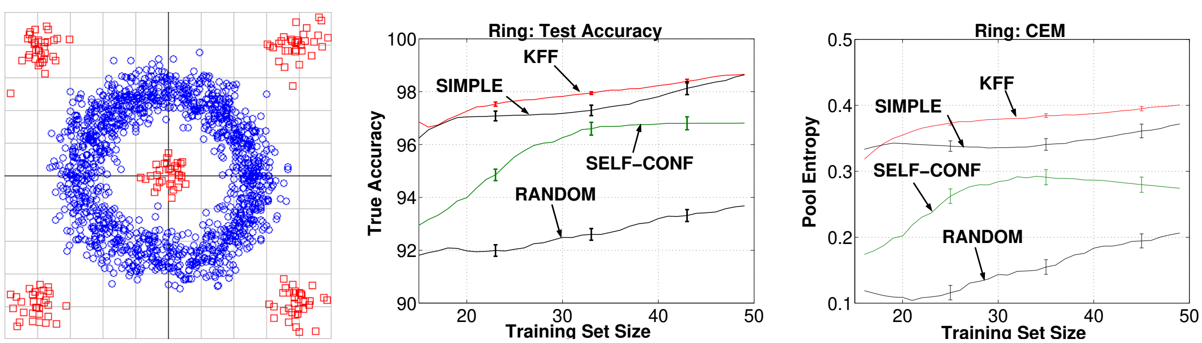


Figure 6: **Left**: A synthetic 'Ring' problem consisting of 3500 points, 90% (3150 circles) in the large 'ring' cluster, and 10% equally divided over the five small clusters (70 squares in each cluster); 1500 points were taken (uniformly at random) to form the pool and the rest form the test set. **Middle**: True accuracy curves of four active learners - KFF, SIMPLE, SELF-CONF and RAND on this 'Ring' data set as estimated on an independent test set. **Right**: Corresponding pool classification entropies (CEM values) of these four learners. All estimates are averages of 100 folds. Error bars (diluted to reduce clutter) represent one standard error of the mean.

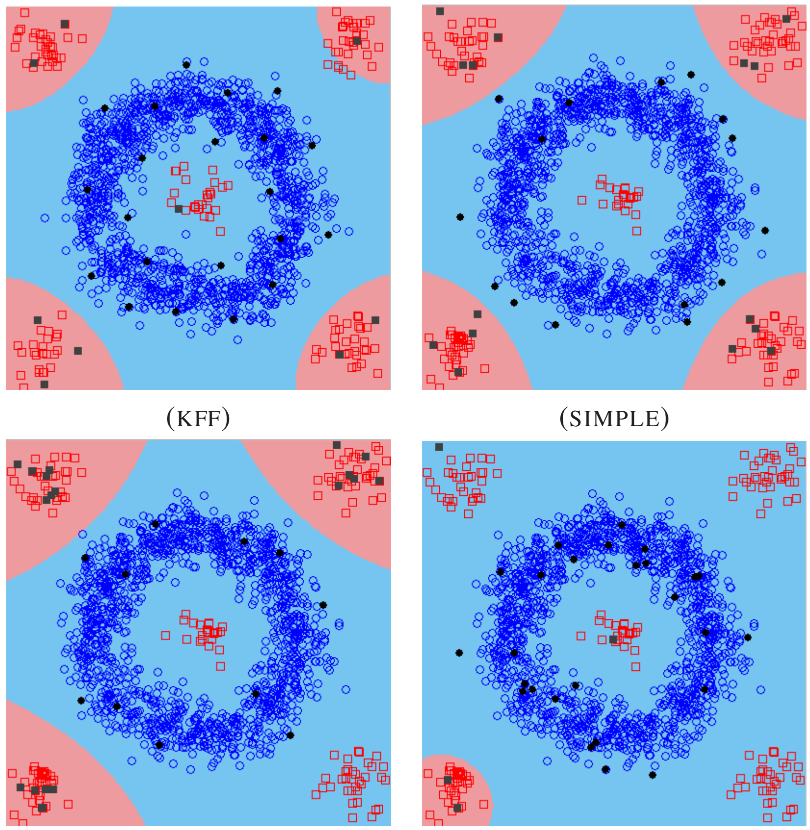


Figure 7: Decision boundaries generated by four active learners using 30 queries on the 'Ring' example of Figure 6 (left). The 30 queries are darker than the pool points.

Figure 6 (left) depicts a synthetic 'Ring' learning problem whose majority class is the 'ring' (blue circles) and its other (minority) class consists of the five small clusters (red squares). The class proportion in this data set is $r = 0.9$ for the 'ring' majority class. We ran the four active learners on this data set. Consider Figure 7 showing the pool classification (decision boundaries) and the training sets chosen by these four learners after 30 active-learning iterations. In this example all pool classifications are majority-biased. Intuitively, the progress of an active learner corresponds to how fast the learner discovers the small clusters and their boundaries. The better active learner, KFF, discovered all five red clusters while the second best learner, SIMPLE, has found four. SELF-CONF has found only three clusters, and RAND found two. We can see a clear correspondence between the pool classification entropy (CEM) and the true accuracy of the learners. Fast exploration of the minority class clusters corresponds to fast entropy maximization as well as fast learning (generalization).

The graphs in Figure 6 (middle) plot the true accuracy of the learners as estimated using an independent test set. The graphs in Figure 6 (right) represent the corresponding pool classification entropies (CEM values). The striking similarity between these curves is further exhibited in Table 6 giving the performance of each of the four active learners after 30 queries. Specifically, the table provides for each learner:

(i) The proportion of the majority class in the training subset (consisting of 30 samples). We call this the "training proportion".

(ii) The proportion of the majority class in the pool classification (generated by the classifier trained over the 30 samples). We call this the "pool proportion".

(iii) The CEM value (entropy) of the pool classification.

(iv) The "true" accuracy estimated on an independent test set.

As usual, all these numbers are averages over 100 random folds.

	RAND	SIMPLE	KFF	SELF-CONF
Training Proportion	86.38 ± 0.53	51.88 ± 0.24	66.06 ± 0.37	37.91 ± 0.66
Pool Proportion	97.46 ± 0.21	92.75 ± 0.20	92.05 ± 0.07	93.54 ± 0.24
Pool Classification Entropy (CEM)	0.14 ± 0.01	0.34 ± 0.01	0.38 ± 0.01	0.29 ± 0.01
True Accuracy	92.39 ± 0.23	97.15 ± 0.19	97.80 ± 0.05	96.09 ± 0.25

Table 6: Characterization of the state and progress of four active learners after querying 30 samples from the pool of the 'Ring' data set given in Figure 6 (left). Each entry provides the mean (over 100 random folds) and the standard error of this mean.

Although in this example the true proportion of the majority class is $r = 0.9$, some of the learners exhibited a significantly smaller training proportion (for example, SIMPLE sampled almost equally from both classes, and SELF-CONF even favored the minority class). The SVMs, which were induced using these minority-biased training sets, have still generated a *majority*-biased classification. In particular, all generated pool proportions are larger than 0.9. We note that this behavior is typical of SVM inducers but not of other learning algorithms.[24] Clearly, this example shows that the CEM criterion (as measured by the pool proportion) correctly ranks the true accuracy of these four learners.

Our next example is shown in Figure 8 (left). This example can be viewed as an "inverse" of the 'Ring' data set of Figure 6. This data set, called here the 'Satellites' data set, contains five clusters from the majority class (squares) and one small cluster from the minority class (circles). The class proportion in this data set is also $r = 0.9$.

As in the previous example, Figure 9 shows the decision boundaries obtained by the four active learners. Similarly, the learning curves in Figure 8 show the true accuracy (middle) and CEM values (right) and Table 7 is similar to Table 6 of the 'Ring' data set. As in the 'Ring' data set, all pool classifications are majority-biased. However, in contrast to the 'Ring' example, where good performance corresponds to quickly finding the minority clusters, in this example good performance should correspond to quickly finding the boundaries of the single minority cluster. The better active learner, now SIMPLE, has mapped more rapidly more areas of the minority cluster than the other

24. When running C4.5, 5-Nearest Neighbor and Bayes Point Machine (using a kernel perceptron for sampling) on these same samples, the pool proportions obtained are significantly minority-biased.

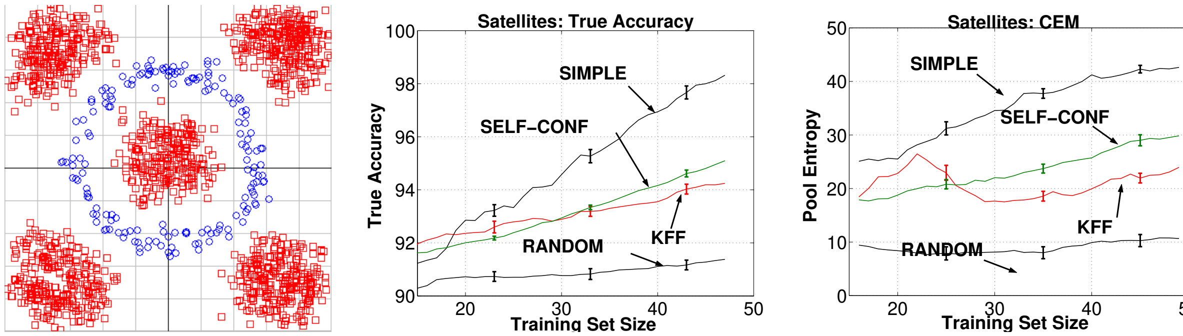


Figure 8: **Left**: The 'Satellites' data set consisting of 3500 points, 10% (350 circles) in the ring "cluster" and 90% equally divided between the five large round clusters (630 squares in each cluster); 1500 points were taken (randomly) to form the pool and the rest form the test set. **Middle**: True accuracy of four active learners - KFF, SIMPLE, SELF-CONF and RAND on this 'Satellites' data set as estimated using 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.

Data Set 'Satellites'	RAND	SIMPLE	KFF	SELF-CONF
Training Proportion	87.23 ± 0.52	50.97 ± 0.29	74.97 ± 0.30	45.19 ± 0.06
Pool Proportion	98.33 ± 0.33	92.42 ± 0.28	96.67 ± 0.20	95.28 ± 0.17
Pool Classification Entropy (CEM)	0.08 ± 0.01	0.35 ± 0.01	0.18 ± 0.01	0.22 ± 0.01
True Accuracy	90.81 ± 0.19	94.18 ± 0.28	92.80 ± 0.19	92.83 ± 0.10

Table 7: Characterization of the state and progress of four active learners after querying 30 samples from the pool of the 'Satellites' data set of Figure 8 (left). Each entry provides the mean (over 100 random folds) and the standard error of the mean.

learners. Note that RAND treats the entire minority class points as noise. Once again we can see a clear correspondence between the pool classification entropy (CEM) and the true accuracy of the learners where now fast exploration of the minority cluster boundaries corresponds to fast entropy maximization as well as fast learning (generalization).

Here again, although the true proportion is 0.9, some of the learners exhibited a significantly smaller training proportion and still the SVMs, which were induced using these training sets, have generated a majority-biased classification.[25] This example also shows that the CEM criterion correctly identifies the true accuracy of these four learners (where the best learner is SIMPLE, followed by SELF-CONF, KFF and RAND).

Our last example is shown on the left-hand side of Figure 10, is referred to as 'Concentric-Rings'. This data set consists of one very large and one very small ring clusters with a small ring

25. In this example as well, the C4.5, 5-Nearest Neighbor and Bayes Point Machine learning algorithms were not majority-biased.

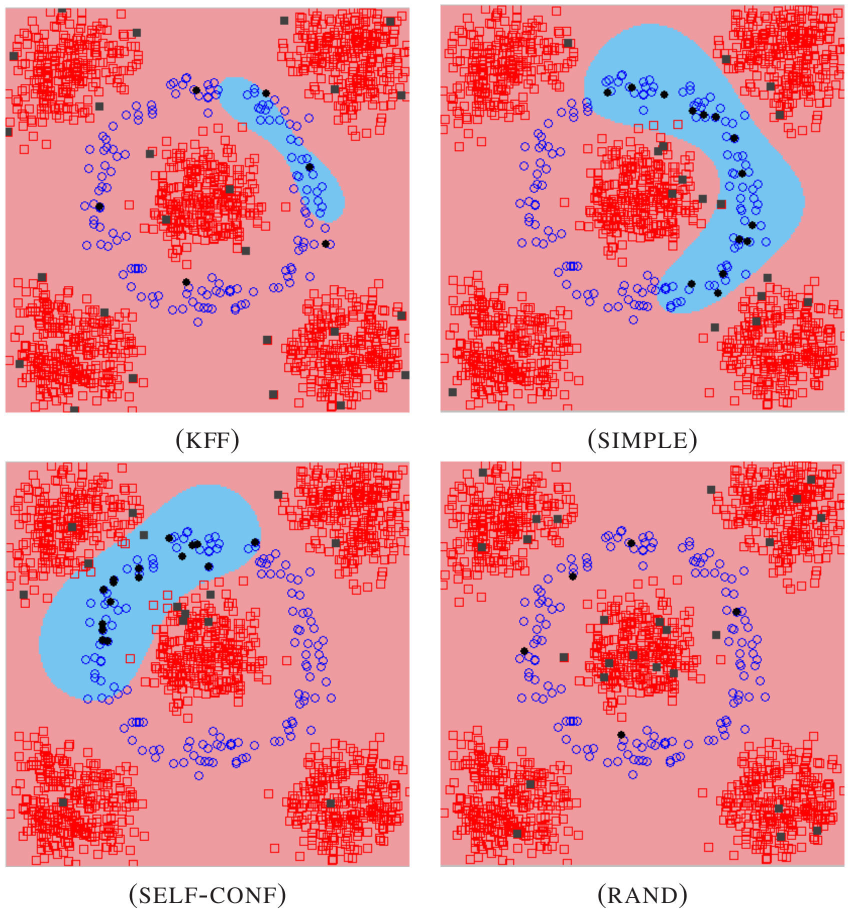


Figure 9: Decision boundaries generated by four active learners using 30 queries on the 'Satellites' example of Figure 8 (left). The 30 queries are darker than the pool points.

cluster between them. This example considers a case where the learning algorithm (SVM) is not majority-biased with respect to a particular inclusion sequence (generated by the SIMPLE querying function). In this example, the increase in the CEM criterion does not match the increase in true accuracy. This happens when the estimated pool entropy at some point is larger than the true entropy $H(r)$ so that the estimated entropy must eventually decrease to its true level while the generalization accuracy can simultaneously improve.

In Figure 10 (right) we observe SIMPLE's decision boundary after 20 queries. This boundary misclassifies the small central cluster and therefore favors the minority class. The pool classification entropy in this case is larger than its final entropy $H(0.9)$. Clearly, when the central cluster is discovered, the entropy will decrease but the true accuracy will increase. Note that the size of the central (isolated) cluster is related to its discovery rate; that is, larger clusters should be discovered faster by a good active learner. In the above example, even one point from this cluster will immediately change the pool classification to be majority-biased. In general, as motivated by this example,

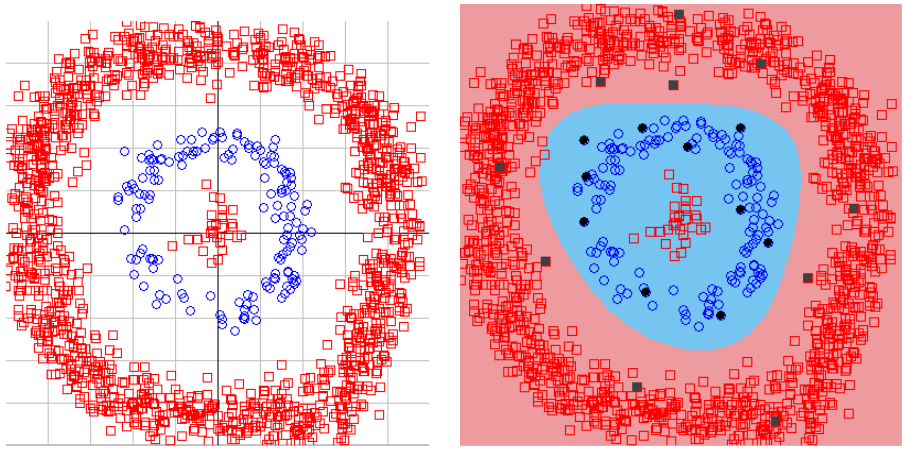


Figure 10: **Left**: The 'Concentric-Rings' problem consists of 3500 points, 10% (350 points) in the inner ring "cluster", 2% (70 points) in the small round cluster in the middle, and the rest in the big outer ring "cluster"; one class consists of all the points in the inner ring (consisting of circles) and the other class is the union of the small round middle cluster and the outer ring (both consisting of squares); 1500 points form the pool and the rest form the test set. **Right**: Decision boundary generated by SIMPLE using 20 queries on this problem. The 20 queries are darker than the pool points.

whenever a learning algorithm is *minority*-biased (with respect to some inclusion sequence) the CEM criterion will fail in the sense that the entropy will not correspond to the true accuracy.

Summarizing the above discussion and considering the experimental results of Section 8, we observe that the success of the CEM criterion depends on the properties of both the data set and the learning algorithm. It appears that data sets with isolated "clusters," whose total proportion is small, will more easily allow for the generation of inclusion sequences of training sets for which the learning algorithm will be *minority*-biased. The dependency of CEM on the learning algorithm can be (intuitively) tied to *algorithmic stability*. For example, if a learning algorithm is "stable" and it is majority-biased over a prefix of some inclusion sequence of training sets, then it is likely that the stability of the algorithm will prevent the generation of classifiers which are minority-biased. Thus, we speculate that the success of CEM in our context is tied to the algorithmic stability of SVMs (see, for example, Bousquet and Elisseeff, 2002, Kutin and Niyogi, 2002). On the other hand, our limited experiments with other, not so stable learning algorithms, such as C4.5, suggests that their CEM estimates are not accurate.

Let us now go back to the 'Concentric-Rings' data set where the SVM was not majority-biased on the inclusion sequence of training sets produced by SIMPLE. The reason for SIMPLE's minority-biased classification is that the training set does not include points from the very small isolated cluster belonging to the majority class, but this training set does include points from the minority class "surrounding" the small cluster. The consequence of this configuration is that the classifier will misclassify the majority class small cluster. This problem can be circumvented by including in the training set at least one point from the majority class small cluster. Formulating this setup we can bound the probability that this configuration will occur due to a random choice of an initial training set of size k (instead of our "standard" choice of a training set of size two; see Section 2). Let n be the total number of points in the pool. Let n_{maj} be the number of points in a small cluster from the majority class. Let n_{min} be the number of points in a subset of points from the minority class surrounding that small cluster. The classifier (SVM) will misclassify the small cluster if the resulting training set does not include any point from that cluster and at the same time will include at least two points from the minority class subset (to induce two support vectors that will guarantee the wrong classification of the small cluster). Thus, using $(1-x)^z \le e^{-zx}$, the probability ρ of choosing such a random training set satisfies

$$\rho \le (\frac{n_{min}}{n})^2(1-\frac{n_{maj}}{n})^{k-2} \le (\frac{n_{min}}{n})^2 \exp\{-(k-2)(\frac{n_{maj}}{n})\}. \tag{3}$$

Assuming that the bound (3) is smaller than δ we solve for k:

$$\begin{aligned} & (\frac{n_{min}}{n})^2 \exp\left\{-(k-2)\frac{n_{maj}}{n}\right\} \le \delta \\ \Leftrightarrow\quad & 2\ln\frac{n_{min}}{n} - (k-2)\frac{n_{maj}}{n} \le \ln\delta \\ \Leftrightarrow\quad & n_{maj}(k-2) \ge 2n\ln\frac{n_{min}}{n} - n\ln\delta \\ \Leftrightarrow\quad & k \ge 2 + 2\frac{n}{n_{maj}}\ln\left(\frac{n_{min}}{n}\right) - \frac{n}{n_{maj}}\ln\delta \\ \Leftrightarrow\quad & k \ge 2 + \frac{n}{n_{maj}}\ln\left(\left(\frac{n_{min}}{n}\right)^2/\delta\right). \end{aligned} \tag{4}$$

Thus, with probability at least $1-\delta$, if we choose a random initial training set of size k, the above "bad" configuration will not occur. In particular, if the argument of the 'ln' in (4) is smaller than 1, it is sufficient to take $k=2$. For example, taking $\delta = 0.01$, we have $(\frac{n_{min}}{n})^2/\delta = 1$ when n_{min} is 10% of the data.

In summary, the above discussion indicates that when sampling more points to be in the initial training set provided to an active learner (based on SVMs), the CEM criterion will not fail with high confidence.

Remark 2 *The CEM estimator can be derived using the Information Bottleneck framework (Tishby et al., 1999) as follows.*[26] *If X is a random variable representing the data and Y is another* target *variable, the information bottleneck method computes a partition T of X, while attempting to conserve as much as possible from the information X contains on Y. Formally, one seeks T such that the mutual information $I(T;Y)$ is maximized under some constraint on the magnitude of $I(X,T)$.*[27] *The CEM estimator can be derived by applying the information bottleneck principle with the target variable Y being the data X itself. In our context, T is always a binary classification (that is, a binary partition of X into two non-intersecting subsets), $T = (t^+, t^-)$ (with $X = t^+ \cup t^-$), which must be* consistent *with the current training set $\mathcal{L}$. Therefore, we would like to give higher "scores" for (consistent) partitions T that have higher information content $I(T,X)$ on the data. Consider a data set of size n. Assuming a uniform prior over the samples (that is, $p(x) = 1/n$) and noting that $p(t) = |t|/n$*

26. In fact, we discovered CEM using this framework.
27. In particular, in standard applications of the information bottleneck method, $I(X,T)$ is forced to be sufficiently small so as to achieve *compression*; see Tishby et al. (1999) for details.

and that $p(x|t) = 1/|t|$ if $x \in t$ (and $p(x|t) = 0$ otherwise), we have $p(x,t) = p(x|t)p(t) = 1/n$ if $x \in t$ and 0 otherwise. Thus, for any partition $T = (t^+, t^-)$,

$$\begin{aligned} I(T;X) &= \sum_{x \in X, t \in T} p(x,t) \log \frac{p(x,t)}{p(x)p(t)} \\ &= \sum_{x \in t^+} \frac{1}{n} \log \frac{n}{|t^+|} + \sum_{x \in t^-} \frac{1}{n} \log \frac{n}{|t^-|} \\ &= -\frac{|t^+|}{n} \log \frac{|t^+|}{n} - \frac{|t^-|}{n} \log \frac{|t^-|}{n} = H(T). \end{aligned}$$

10. Concluding Remarks

We presented an online algorithm that effectively combines an ensemble of active learners. The algorithm 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. Practitioners can significantly benefit from our new algorithm in situations where not much is known about the classification problem at hand.

Some questions require further investigation. In our experience, the 'classification entropy maximization (CEM)' semi-supervised criterion for tracking active-learning progress outperforms 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 to our master algorithm may 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 data sets. Finally, it would be interesting to extend our techniques to multi-valued classification problems (rather than binary) and to other learning tasks such as regression.

Acknowledgments

We thank Ron Meir and Ran Bachrach for useful discussions. We also thank Simon Tong for providing essential information for reproducing the results of Tong and Koller (2001). The work of R. El-Yaniv was partially supported by the Technion V.P.R. Fund for the Promotion of Sponsored Research.

Appendix A. Some Implementation Details

In this appendix we provide some particular implementation details which are essential for replication.

We operated all our SVMs using a kernel correction method discussed by Shaw-Taylor and Christianini (2002), which guarantees that the training set is linearly separable in kernel space as required by algorithms like SIMPLE. Specifically, this is done by modifying the kernel K so that for each training point x_i, the modified kernel K' is $K'(x_i, x_i) = K(x_i, x_i) + \lambda$ where λ is a positive constant, and for all other arguments the kernel remains the same. In all the experiments described in this paper we took a fixed $\lambda = 2$.[28]

The RBF kernel has one parameter. In the SVM implementation we used (Chang and Lin, 2002) this parameter is denoted by γ. To obtain high performance it is crucial to use appropriate values of γ. This issue of model (and parameter) selection is not the main concern of our work. We therefore assume that all of our SVM based active learners have reasonably good parameters and all learners have the same parameters.[29]

We have normalized each feature to lie in [-1,1] as recommended by Chang and Lin (2002). We have also used a standard method to "eliminate" the bias term by increasing the input dimension by one with a fixed component.

In noisy settings, active learners such as SIMPLE tend to be very unstable at the early stages of the learning process. A description of this phenomenon and a proposed solution are described in Appendix C. The proposed solution is based on "buffering" labeled examples obtained from the learner and postponing the inclusion of buffered points in the training set. The buffer size we used in all our experiments is 3.

Appendix B. Semi-Supervised SVM Model Selection Using CEM

Here we briefly describe some numerical examples of using the CEM criterion of Section 6 as a model selection criterion for choosing the kernel parameters of an SVM (using the RBF kernel). Consider the following semi-supervised binary classification setting. We are given a small training set (for example, containing 20 instances) and a larger pool of unlabeled samples. Our goal is to to train an SVM classifier for the classification problem at hand. Clearly, bad parameter assignment for the SVM kernel will result in poor performance. The kernel parameters can of course be chosen using standard methods such as cross-validation or leave-one-out (LOO). Here we show that the CEM criterion can do slightly better than leave-one-out (and cross-validation) without computation time compromises. We emphasize that this appendix only concerns the CEM criterion and does not discuss active learning.

We applied CEM and LOO on the entire benchmark data set collection of Rätsch et al. (2001) as described in Table 1. Our experimental design is as follows. In all data sets we used an SVM inducer with an RBF kernel. The only relevant parameter for this kernel is γ, which determines the RBF kernel resolution.[30] We fixed a crude feasible set of γ values for all the data sets. This set is $\Gamma = \{0.01, 0.05, 0.1, 0.5, 1.0, 5.0\}$. Let F be a training fold partition (that is, one of the fixed 100 train/test folds in the original benchmark set) consisting of two parts: F_{train} and F_{test}. For each fold we performed the following procedure:

1. We randomly selected 20 instances from F_{train} and designated them as the (labeled) training part S, denoting the rest of the instances in F_{train} by U. The labels of instances in U were kept hidden from both LOO and CEM (and clearly, in all cases neither LOO nor CEM see any instance from F_{test}).

28. This value was crudely chosen to guarantee zero training error (without test error optimization).
29. For each learning problem different γ values were selected for each fold by randomly splitting the training set in half. One half was used as the pool for the active learner. The other half was used for choosing the value of γ out of a fixed grid of γ values using 10-fold cross validation.
30. Since we use the kernel correction "trick" mentioned above, our training sets are guaranteed to be linearly separable in feature space so there is no need to consider a soft margin cost parameter.

2. For each $\gamma \in \Gamma$ we applied LOO and CEM. For LOO this means training 20 classifiers, such that each classifier is trained on different 19 examples from S, tested on the other example and we count success percentage of these 20 classifiers. For CEM this means training one classifier over S and then computing the entropy of the resulting partition with respect to U.

3. LOO and CEM then choose their best candidate $\gamma \in \Gamma$. For LOO this means taking the parameter corresponding to the highest average precision, and for CEM, taking the parameter corresponding to the maximal entropy.

4. Two SVMs with the winning parameters (one for LOO and one for CEM) are then trained over S and the corresponding classifiers are tested on F_{test}.

In order to get a correct perspective on the performance of LOO and CEM we also computed the performance of the best and worst models in hindsight. Specifically, for each $\gamma \in \Gamma$ we computed the resulting accuracy over F_{test} of the corresponding SVM (which was trained with γ on S). The results of this procedure are given in Table 8. Each row of the table corresponds to one data set (among the 13) and the accuracy results for each of the methods is specified; that is, the SVM employing the best parameter value obtained for LOO and CEM. Note that each number in the table is an average over 100 folds and standard errors of the means are specified as well.

The first striking observation is that both LOO and CEM perform quite well. In particular, based on a tiny training set both estimators achieve performance quite close to the best possible. Second, it is evident that CEM outperforms LOO. Experiments we performed with other training set sizes show that the relative advantage of CEM increases as the training set size decreases. When the training set size increases CEM's advantage over LOO eventually disappears. For instance, when the sample size is 40, LOO becomes more reliable and slightly outperforms CEM. Similar results were obtained when instead of LOO we used k-fold cross-validation with "standard" values of k (for example, $k = 3,4,5$).

Appendix C. Stabling Active Learner Performance

As mentioned above (and following Tong and Koller, 2001) we use the kernel correction method of Shaw-Taylor and Christianini (2002). This correction guarantees linear separability in feature space, which guarantees the existence of the version space. However, the use of this correction, can introduce severe classification instability in early stages of the an active-learning process (for active learners such as SIMPLE) when learning noisy data sets. In this appendix we briefly present this problem and propose a simple solution based on "buffering".

This kernel correction of Shaw-Taylor and Christianini (2002) works by adding a constant to the diagonal of the kernel matrix, thus providing extra (additive) bias to the classification of the training points. One main effect of this correction is that all training points are correctly classified. This correction introduces discontinuities in the decision boundary.

Consider the example in Figure 11 (left) showing a 2D projection of the 'Twonorm' data set (see the details of this data set in Table 1). Figure 12 shows the pool classification and the training sets generated by the SIMPLE active learner after 4-8 iterations. The decision boundary changes drastically from one iteration to another. Note that the above kernel "correction" enables a correct classification of the training set using all the decision boundaries presented in the figure. A learning curve of SIMPLE (showing the true error) is given in Figure 11 (right). Clearly the true accuracy is extremely unstable.

Data set	Best Possible Accuracy	Worst Possible Accuracy	LOO Accuracy	CEM Accuracy
Banana	71.66±0.66	50.26+-0.51	**66.03±1.13**	65.98±0.97
Breast-Cancer	72.05±0.51	67.16+-0.85	**69.83±0.58**	69.69±0.60
Diabetis	68.44±0.36	63.26+-0.67	66.06±0.55	**67.39±0.34**
Flare-Solar	63.96±0.42	59.00+-0.60	62.04±0.52	**62.94±0.45**
German	70.43±0.38	68.23+-0.67	69.35±0.49	**69.48±0.40**
Heart	75.11±0.82	53.12+-0.62	71.59±0.11	**74.2 ±0.8**
Image	69.69±1.33	55.27+-1.04	64.51±1.71	**68.40±1.40**
Ringnorm	90.35±1.1	53.66+-0.76	88.97±1.21	**89.66±1.1**
Splice	55.56±1.48	49.40+-0.49	52.86±1.3	**55.46±1.44**
Thyroid	89.76±0.73	70.21+-0.61	85.73±0.94	**89.56±0.75**
Titanic	71.32±0.82	63.02+-1.08	**67.41±1.07**	64.65±1.1
Twonorm	88.09±1.09	54.11+-0.86	86.45±1.35	**88.01±1.08**
Waveform	76.73±0.73	65.56+-0.86	73.69±0.87	**76.58±0.72**
Averages	74.08	59.40	71.11	**72.46**

Table 8: Average accuracy (and its standard error) achieved by Classification Entropy Maximization (CEM) and Leave-one-out (LOO) estimators on the 13 UCI problems of Rätsch et al. (2001). For each data set, the winner (among CEM and LOO) appears in boldface. The accuracy of the best and worst possible models (in hindsight) are also given.

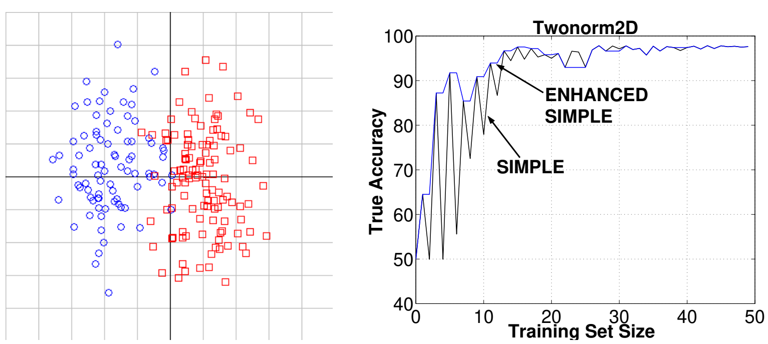
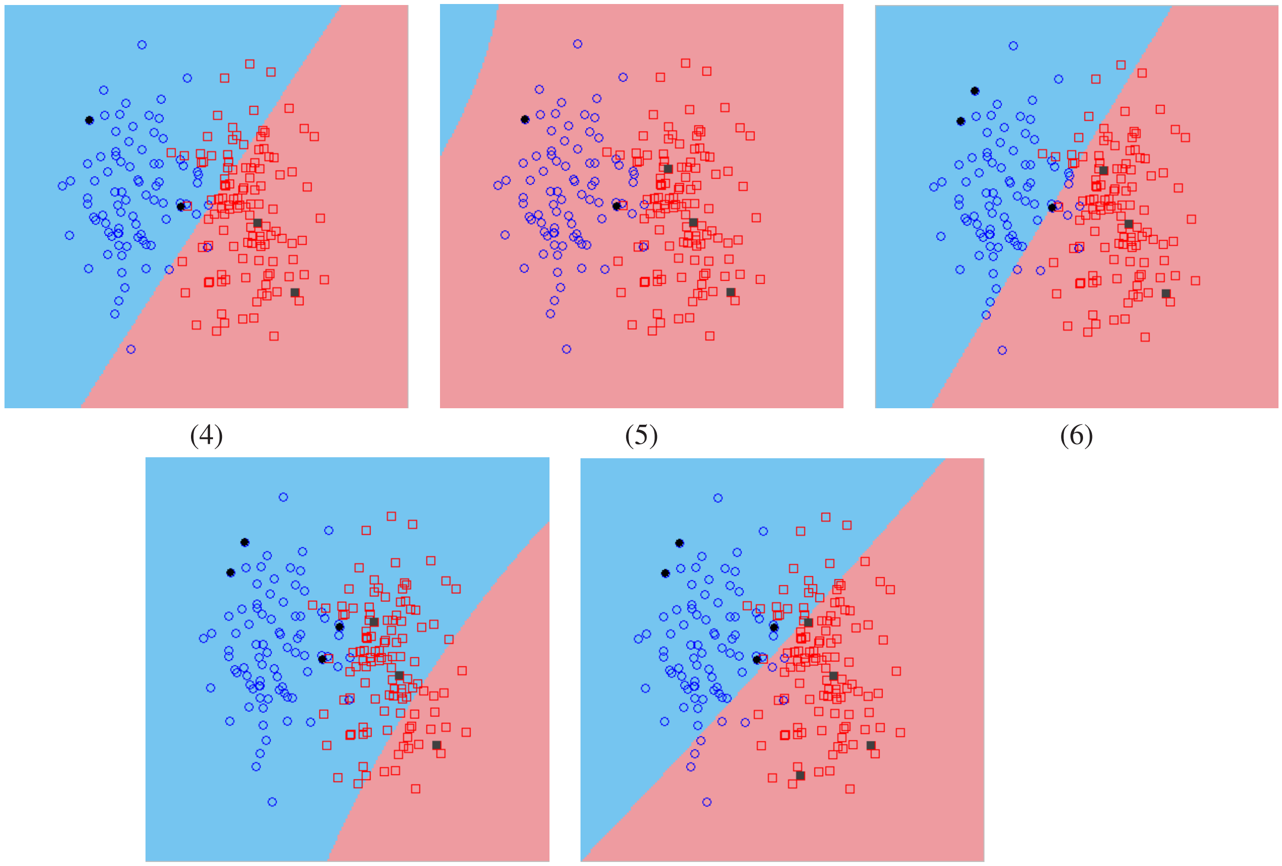


Figure 11: **Left**: Data Set 'Twonorm2D' that was created by projecting the UCI 'Twonorm' data set to two dimensions. This data set demonstrates the unstable general behavior of active learners during early stages of the learning session. **Right**: Learning curves of SIMPLE and enhanced SIMPLE (via buffering) on the 'Twonorm2D' data set.

We propose the following solution using the idea of "buffering", as well as our classification entropy estimator. One of the *symptoms* of the drastic decision boundary instability described above is a (drastic) change in the classification entropy. We observe that when the boundary shifts from the border between the two clusters to the middle of one of the clusters (iterations 5,7) the entropy decreases. Therefore, the solution we propose is to look at the change in entropy of the pool (and training set) classification in the end of each trial. As long as the entropy decreases, new examples are *buffered* and not added to the training set. Once the entropy has not decreased, all buffered



(4) (5) (6) (7) (8)

Figure 12: Pool classification and training sets (blackened points) generated by the SIMPLE active learner after 4-8 iterations on the 'Twonorm2D' data set.

points are added to the training set. If we want to keep a bounded buffer this solution introduces a new parameter; namely, the buffer size. In our experience a small buffer is sufficient and in all our experiments we used a buffer of three points.

This buffering approach provides an effective solution to the above problem. In Figure 11 (right) we see the learning curves of SIMPLE on the 'Twonorm2D' data set with and without this enhancement. It is evident that the learning curve of the original algorithm is very unstable. Adding the buffering enhancement smoothes the learning curve. Figure 13 shows the pool classification and the training sets generated by the enhanced SIMPLE active learner after 4-8 iterations. The decision boundary now does not change drastically from one iteration to another.

References

D. Angluin. Queries and concept learning. *Machine Learning*, 2(3):319–342, 1988.

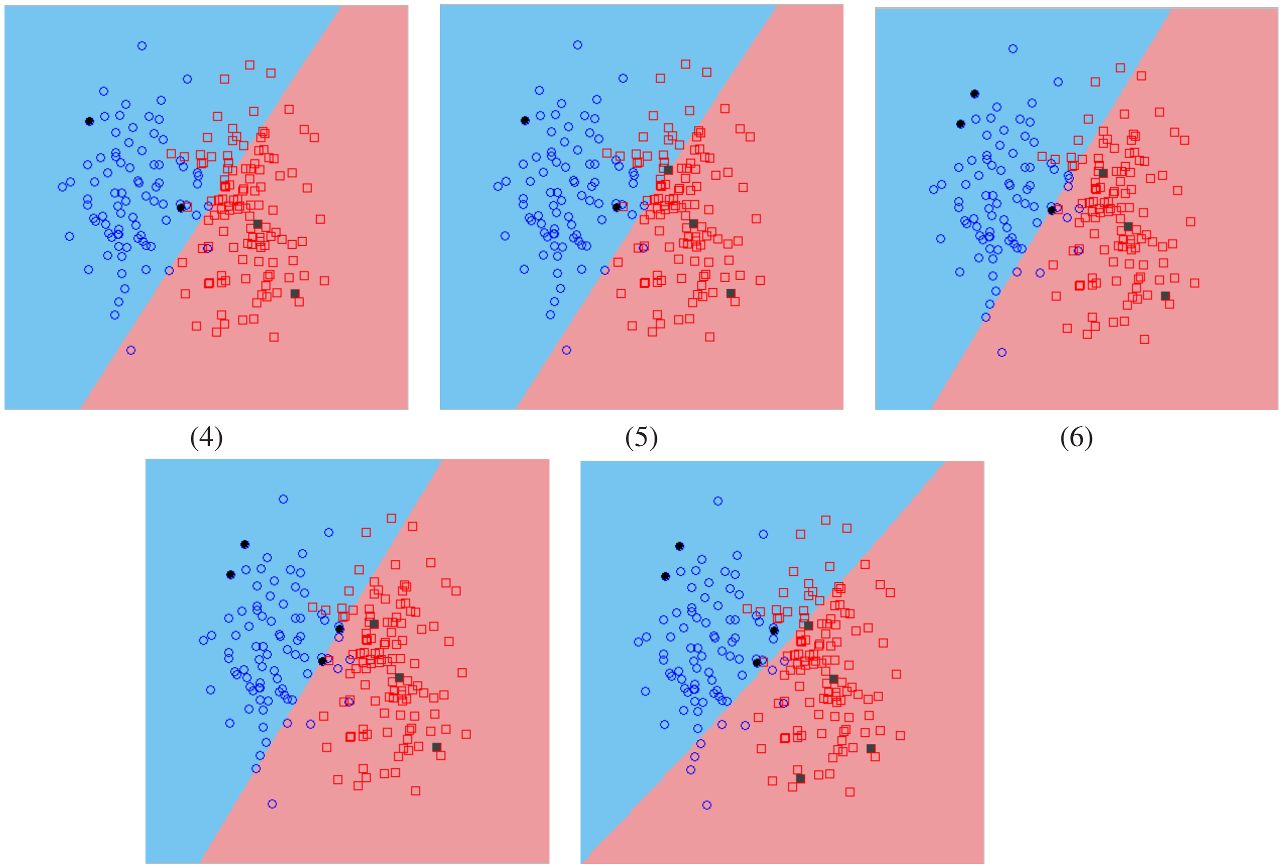


Figure 13: Pool classification and training sets (blackened points) generated by the ***enhanced*** SIMPLE active learner after 4-8 iterations on the 'Twonorm2D' data set.

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

R. Bachrach, S. Fine, and E. Shamir. Query by committee, linear separation and random walks. In *Proceedings of Euro-COLT, 13th European Conference on Computational Learning Theory*, 1999.

O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, pages 499–526, 2002.

C. Campbell, N. Cristianini, and A. Smola. Query learning with large margin classifiers. In *Proceedings of ICML-2000, 17th International Conference on Machine Learning*, pages 111–118, 2000.

G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Neural Information Processing Systems (NIPS)*, pages 409–415, 2000.

N. Ceza-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427–485, May 1997.

C. C. Chang and C. J. Lin. Libsvm: a library for support vector machines, 2002. URL http://www.csie.ntu.edu.tw/ cjlin/papers/libsvm.pdf.

D. A. Cohn, L. Atlas, and R. E. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

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

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

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

I. Guyon, N. Matic, and V. Vapnik. Discovering informative patterns and data cleaning. In *Advances in Knowledge Discovery and Data Mining*, pages 181–203. 1996. URL http://citeseer.nj.nec.com/guyon96discovering.html.

R. Herbrich, T. Graepel, and C. Campbell. Bayes point machines. *Journal of Machine Learning Research*, 1:245–279, 2001. URL http://citeseer.nj.nec.com/herbrich01bayes.html.

D. Hochbaum and D. Shmoys. A best possible heuristic for the K-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 231–238. The MIT Press, 1995. URL http://citeseer.nj.nec.com/krogh95neural.html.

S. Kutin and P. Niyogi. Almost-everywhere algorithmic stability and generalization error. Technical Report TR-2002-03, University of Chicago, 2002.

M. Lindenbaum, S. Markovitch, and D. Rusakov. Selective sampling for nearest neighbor classifiers. *Machine Learning*, 54(2):125–152, 2004.

D. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992. URL http://citeseer.nj.nec.com/47461.html.

A. K. McCallum and K. Nigam. Employing EM in pool-based active learning for text classification. In *Proceedings of ICML-98, 15th International Conference on Machine Learning*. Morgan Kaufmann Publishers.

T. Mitchell. Generalization as search. *Artificial Intelligence*, 28:203–226, 1982.

G. Rätsch, T. Onoda, and K. R. Müller. Soft margins for Adaboost. *Machine Learning*, 42:287–320, 2001.

N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of ICML-2001, 18th International Conference on Machine Learning*, pages 441–448, 2001.

M. Saar-Tsechansky and F. Provost. Active learning for class probability estimation and ranking, 2002. URL `http://citeseer.nj.nec.com/tsechansky01active.html`.

G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *ICMLProceedings of ICML-2000, 17th International Conference on Machine Learning*, pages 839–846, 2000.

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.

M. Seeger. Learning with labeled and unlabeled data, 2002. URL `http://citeseer.nj.nec.com/seeger01learning.html`.

H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Computational Learning Theory*, pages 287–294, 1992. URL `http://citeseer.nj.nec.com/seung92query.html`.

J. Shaw-Taylor and N. Christianini. Further results on the margin distribution. In *Proceedings of the 12th Annual ACM Conference on Computational Learning Theory (COLT)*, pages 278–285, 1999.

J. Shaw-Taylor and N. Christianini. On the generalization of soft margin algorithms. *IEEE Transactions on Information Theory*, 48(10):2721–2735, 2002.

N. Tishby, F. C. Pereira, and W. Bialek. Information bottleneck method. In *37-th Allerton Conference on Communication and Computation*, 1999.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.

V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc., 1998.

T. Zhang and F. Oles. A probability analysis on the value of unlabeled data for classification problems. In *International Joint Conference on Machine Learning*, pages 1191–1198, 2000.