

Stochastic Scheduling of Active Support Vector Learning Algorithms

Gaurav Pandey
Computer Science
University of Minnesota
gaurav@cs.umn.edu

Himanshu Gupta
Computer Science
IIT Kanpur
himg@cse.iitk.ac.in

Pabitra Mitra
Computer Science
IIT Kanpur
pmitra@cse.iitk.ac.in

ABSTRACT

Active learning is a generic approach to accelerate training of classifiers in order to achieve a higher accuracy with a small number of training examples. In the past, simple active learning algorithms like random learning and query learning have been proposed for the design of support vector machine (SVM) classifiers. In random learning, examples are chosen randomly, while in query learning examples closer to the current separating hyperplane are chosen at each learning step. However, it is observed that a better scheme would be to use random learning in the initial stages (more exploration) and query learning in the final stages (more exploitation) of learning. Here we present two novel active SV learning algorithms which use adaptive mixtures of random and query learning. One of the proposed algorithms is inspired by online decision problems, and involves a hard choice among the pure strategies at each step. The other extends this to soft choices using a mixture of instances recommended by the individual pure strategies. Both strategies handle the exploration-exploitation trade-off in an efficient manner. The efficacy of the algorithms is demonstrated by experiments on benchmark datasets.

Keywords

SVM, Pool Based Active Learning, Multi-Arm Bandit Problem, Stochastic scheduling

1. INTRODUCTION

Active learning is a popular paradigm for reducing the sample complexity of a learning algorithm [6] where the learner selects its own training data. Here, instead of learning from instances selected randomly, the learner can select its own training data. This is done iteratively, and the output of a learning step is used to select the examples of the next step. A particular setting of this framework is referred to as pool based active learning [1], where the learner is presented with a fixed pool of unlabeled instances and on each

trial it chooses a set of instances from the pool to be labeled. The learner is then provided the true label of the instances and it induces a new classifier based on all the labeled samples seen so far. Several strategies for choosing the set of instances from the pool exist in practice, e.g., random selection, error driven techniques, uncertainty sampling, version space reduction and adaptive resampling. Pool based active learning is highly suited for applications where unlabeled data is abundant but labeling instances is costly and time consuming. Such a scenario is often encountered in text categorization and information retrieval, where the labeling is obtained by a human user [12]. In the case of remote sensing and biomedical image analysis, labeling is even more costly as it can be done only by experts. Labelling in drug discovery and molecule screening problems also involve costly and time consuming chemical experiments.

The support vector machine (SVM) [4, 13] has been successful as a high performance classifier in several domains including pattern recognition, data mining and bioinformatics. It has strong theoretical foundations and demonstrates good generalization capability. A limitation of the SVM design algorithm, particularly for large data sets, is the need to solve a quadratic programming (QP) problem involving a dense $n \times n$ matrix, where n is the number of points in the data set. Since most QP routines have cubic complexity, SVM design requires huge memory and computational time for large data applications. Reducing the sample complexity (n) of SVM design using pool based active learning thus makes SVMs suitable for large data applications.

In the context of SVM, several pool based active learning algorithms have been proposed. The earliest ones selected the new set of instances randomly [9] (referred to as Random SVM in the rest of the paper). In [5], a query learning strategy for large margin classifiers is presented, which iteratively requests the label of the data point closest to the current separating hyperplane (referred to as Query SVM in the remainder of the paper). Similarly, [3] chooses the next batch of training points w.r.t. the angles made by the examples with the current hyperplane. A strategy based on version space splitting is presented in [12]. The points which split the current version space into two halves having equal volumes are selected at each step, as they are likely to be the actual support vectors. A greedy optimal strategy is described in [11]. Here logistic regression is used to compute the class probabilities, which are further used to estimate the expected error after adding an example. The example that minimizes this error is the candidate SV. This approach, though optimal, is not practical because the selection of each

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'05 March 13-17, 2005, Santa Fe, New Mexico, USA
Copyright 2005 ACM 1-58113-964-0/05/0003 ...\$5.00.

candidate point requires solving two QP problems and can be a bottleneck for large datasets.

In spite of the wide variety of techniques used, a limitation these strategies is that they are essentially pure strategies in the sense that throughout the training process they either query for points close to the current separating hyperplane or select random samples consisting mostly of interior points of a class. Both these strategies represent extreme cases; the former one is fast but unstable, while the latter one is robust but slowly converging. The former strategy is useful in the final phase of learning, while the latter one is more suitable in the initial phase. An adaptive mixture of the above two strategies is expected to yield better performance.

In this paper, mixed active learning strategies, which are combinations of two pure strategies, namely, Random SVM and Query SVM, are proposed to achieve fast and robust convergence. Here, the active support vector machine learning problem is treated as a stochastic scheduling problem, more specifically as a multi-armed bandit problem. This is then transformed to an equivalent online decision problem of selecting a mixture of the two pure strategies. Two algorithms for combining/mixing the pure strategies are proposed. The first algorithm involves a hard choice between one of the two pure strategies at each learning step. This is done by a dynamic allocation algorithm called *Follow the Perturbed Leader* [8]. On the other hand, the second one performs a soft choice between the individual pure strategies to determine the composition of the selected set of instances. Experimental results on benchmark data sets demonstrate the effectiveness of the strategies.

It may be noted that a multi-armed bandit formulation of active learning problem has been proposed in [10] and [1]. However, there training instances are treated as slots of the gambling machine. A probability distribution is maintained over the training instances to determine which instance to select next (i.e., which slot to play next). However, for large data sets maintaining such a distribution would be infeasible in terms of storage. Hence, in our algorithm we reformulate the multi-armed bandit problem as an online decision problem involving choice of one of the two experts (i.e., pure strategies). Here, one needs to only store a (dynamic allocation) index per expert to determine which example to choose next and is thus suitable for large data sets.

The organization of the article is as follows. In the next section we describe Random and Query SVM. Section 3 reformulates the active SVM learning problem as a multi-armed bandit problem. We also mention the equivalent online decision problem. Two online algorithms for active SVM learning are presented in Section 4. Experimental results appear in Section 5. We conclude the paper in Section 6.

2. SUPPORT VECTOR MACHINES

SVMs are a general class of learning architectures inspired by statistical learning theory that performs *structural risk minimization* on a nested set structure of separating hyperplanes [4, 13]. Given training data, SVMs obtain the optimal separating hyperplane in terms of the generalization error.

2.1 SVM Design Using Active Learning

As discussed in the previous section, active learning is a method of incremental learning which enables a learning algorithm to approach the maximum achievable classification accuracy very closely with relatively smaller num-

ber of training examples as compared to the batch learning method. Active learning is equally applicable to Support Vector Machines (SVMs) and various approaches have been suggested for using it in SVM training and classification. We present here two of the most common active learning strategies for SVMs, namely random and query SVM. They differ in the way examples are queried from the data set.

Random SVM: This is a universal and perhaps the simplest active learning strategy for any learning algorithm. It assumes that the probability of any example being chosen at any stage of the training process is independent of that of the others, i.e. all examples are given a uniform probability of being chosen for training. The training progresses with the choice of k random examples at each stage.

Query SVM: Random active learning, as discussed, is an unguided strategy and hence it is expected that it will not perform very well on many problems. An alternative and more intuitive approach is to improve the confidence in dimensions about which we already have information. This can be achieved by continually narrowing the existing margin. Thus, at each stage of the training process, k training examples closest to the current dividing hyperplane are selected and added to the intermediate training set [5].

3. MULTI-ARMED BANDIT, ACTIVE LEARNING AND STOCHASTIC SCHEDULING

The Multi-armed bandit (MAB) problem is a well studied problem in stochastic scheduling and dynamic allocation. In MAB [7], a gambler, visiting a casino, must choose which of K machines to play. At each time step, he pulls the arm of one of the machines and receives a reward or payoff (possibly zero or negative). The gambler's purpose is to maximize his total reward over a sequence of trials. Since each arm is assumed to have a different distribution of rewards, the goal is to find the arm with the best expected return as early as possible, and then to keep gambling using that arm.

The problem is a classical example of the trade-off between exploration and exploitation. On the one hand, if the gambler plays exclusively on one machine that he thinks is best (*exploitation*), he may fail to discover that one of the other arms actually has a higher average return. On the other hand, if he spends too much time trying out all the machines and gathering statistics (*exploration*), he may fail to play the best arm often enough to get a high total return.

In the early years, the bandit problem was studied with the aid of statistical assumptions on the process generating the rewards for each arm. However, it is likely that the cost associated with each arm is determined at each time step by an adversary rather than by some benign stochastic process. We can only assume that the rewards are chosen from a bounded range. Here, there is no single machine which gives the highest payoff at all the time steps. Hence one should find a sequence of machines which gives the highest total reward. The performance of any player is measured in terms of 'regret', i.e., the expected difference between the total reward scored by the player and the total reward scored by the best sequence of arms.

3.1 Active Learning as an MAB Problem

In active learning, the learner has the freedom to choose the examples to be used for learning. In other words, the training instances are equivalent to the slots of a gambling

machine that the learner chooses from. Using these queried examples the learner updates itself and queries a new set of examples. Thus the learner has to make a sequence of choices of examples to query, without knowledge of the future. The reward at each step here is the gain in classification accuracy achieved by the learner by learning from the set of examples queried. The learner should try to maximize this reward over all time steps such that the final accuracy is maximum. This is done by maintaining a selection probability distribution $\{p_1^t, \dots, p_l^t\}$, where t is the learning step, l is the total number of training instances, and choosing the set of instances having high probabilities for the $(t + 1)^{th}$ step. In boosting [10], this distribution is maintained by repetitive multiplicative updates on an initial set of uniform probabilities.

3.2 Active Learning & Stochastic Scheduling

An alternate way of formulating the above scheduling problem is to consider a set of experts which advises the learner on which set of examples to choose next. The experts may represent some pure choice strategies themselves. The learner uses a combination or mixture of the experts to select the next set of examples. In our algorithms we have considered a set of two experts, namely, the Random SVM and Query SVM. These are discussed in Section 2. The former expert represents high exploration in learning while the later represents high exploitation. The set of examples to be queried by the learner consists of examples recommended by either or both of the experts. One may note that in the formulation of active learning as a MAB problem the decision consists of choosing the sequence of examples to be queried for learning. However, selecting a particular mixture of experts also leads to querying a particular set of examples and active learning in the MAB setting can also be considered as a sequence of decisions which choose certain mixture of individual experts. In fact, there exists a correspondence between the two decision processes in active learning, namely, the selection of a set of examples and the selection of a mixture of experts. However, in the latter process, one needs to maintain a probability distribution over the set of experts rather than the set of examples, and thus requires considerably less storage for large data sets.

4. PROPOSED ALGORITHMS FOR ACTIVE SUPPORT VECTOR LEARNING

In this section we present two algorithms for online scheduling of a mixture of experts. The experts here are the pure strategies for active learning, namely Random SVM and Query SVM. They are described in Section 2.

The first mixture strategy is a hard strategy which chooses a single expert in each step, although the choice of experts may vary over different steps, while the second one is a soft strategy which chooses a composition of examples recommended by both the experts. The ratio of examples recommended by each expert varies adaptively over the steps. The first strategy uses an efficient online decision algorithm called ‘follow the perturbed leader’ (FPL) [8] and the second uses classification errors in a manner similar to boosting [10]. The algorithms are described in the next two sections.

4.1 Hard Choice: Follow the Perturbed Leader

The correspondence between the problem of active learning for support vector machines and online decision prob-

```

Data : Unlabeled and Labeled examples
          $T_{unlabeled}, T_{labeled}$ 
Result: Final classifier  $C_{final}$ 
 $Q \leftarrow ChooseRandom(T_{unlabeled}, k)$ ;
Label( $Q$ );
 $C \leftarrow SVMTrain(Q)$ ;
 $cost_{0,random} \leftarrow GetError(T_{labeled}, C)$ ;
 $cost_{0,query} \leftarrow 0$ ;
 $t \leftarrow 1$ ;
while termination condition is not met do
   $Q \leftarrow SV(C)$ ;
   $p_{t,query} \leftarrow ExpDistribution(p_{t-1,query})$ /*Choose
  the perturbation factor according to the
  exponential distribution  $e^{-|x|}$ ,  $x = rs * p_{t-1,query}$ ,
   $rs$  is a random number taking on value in
   $\{+1, -1\}$ */;
   $p_{t,random} \leftarrow ExpDistribution(p_{t-1,random})$ ;
  if ( $cost_{t-1,query} + p_{t,query}$ ) <
  ( $cost_{t-1,random} + p_{t,random}$ ) then
     $Q \leftarrow Q \cup ChooseNearest(T_{unlabeled}, k, C)$ ;
    Label( $Q$ )/*Only newly added examples are
    labeled*/;
     $C \leftarrow SVMTrain(Q)$ ;
     $cost_{t,query} \leftarrow$ 
     $cost_{t-1,query} + GetError(T_{labeled}, C)$ ;
  end
  else
     $Q \leftarrow Q \cup ChooseRandom(T_{unlabeled}, k)$ ;
    Label( $Q$ )/*Only newly added examples are
    labeled*/;
     $C \leftarrow SVMTrain(Q)$ ;
     $cost_{t,random} \leftarrow$ 
     $cost_{t-1,random} + GetError(T_{labeled}, C)$ ;
  end
end
return  $C$ ;

```

Algorithm 1: Active learning of SVMs using Follow the Perturbed Leader

lems was shown in a previous section. Based on this analysis, let us consider the two available active learning strategies, namely Query SVM and Random SVM, as experts recommending which machines (instances) to play (chose) next. The cost incurred is equal to the classification error when a strategy is chosen.

Gittin’s indices [7], or dynamic allocation indices, are the proven optimal solution to the above problem, i.e. provides the optimal strategy for choosing the most beneficial machine to play with at each stage. However, calculating these indices is a hard problem and only approximate algorithms have been designed to calculate them. For our solution the online decision algorithm *Follow the Perturbed Leader* (FPL) [8] was used. In this algorithm, for each machine (pure active learning strategy in our case), the total cost incurred whenever the particular machine was chosen, is maintained, and is used in combination with a perturbation factor to choose the optimal machine for the next round.

It should be noted that in the proposed strategy, which is formulated in Algorithm 1, the cost of each active learning strategy (machine) is the classification error committed by

it at each step where it is selected. For the other strategies, the cost incurred for this step is zero. Using this notion, FPL is captured in brief in the following steps:

FPL(α): On each period t ,

1. Choose $p_{t,d}$, the perturbation vector, at random according to the density $d\mu(x) \propto e^{-\alpha|x|_1}$, for each strategy d .

2. Use $M(s_{1:t-1} + p_t) = \operatorname{argmin}_{d \in D} [s_{1:t-1,d} + p_{t,d}]$, where $s_{1:t-1,d}$ is the total cost incurred by strategy d till period $t-1$.

The FPL algorithm has been shown to be optimal within a $(1 + \epsilon)$ bound. Experiments on the hard choice strategy also show its superior performance compared to the random and query learning strategies. It is seen that initially random SVM is the leader and is *followed* by the above algorithm, while after some time the algorithm switches over to query SVM and *follows* it for subsequent steps. Some oscillations between these two occur in the intermittent phases.

4.2 Soft Choice

Various observations show that random SVM performs better classification initially, while query SVM does better in the later part. Since initially the training is done on a very small sized sample, the initial hyperplane found will be very far from the optimal hyperplane. Using query SVM to choose a training point that is closest to this hyperplane will result only in a small drift towards the optimal hyperplane. Choosing a random training point seems to be a better idea as it may lie close to the optimal hyperplane, and its inclusion in the training set will imply significant progress towards the optimal hyperplane. In the later stages, the current hyperplane is close to the optimal hyperplane. Thus, choosing the data point closest to the current hyperplane will be better as it ensures that the point chosen is close to the optimal hyperplane as well. This analysis suggests that an optimal strategy of choosing data points at each stage would be to select some points randomly and some other points close to the current hyperplane, i.e., a soft mixture of the two pure strategies random SVM and query SVM. The ratio of the points recommended by these individual pure strategies varies adaptively over the learning steps.

The strategy at any step consists of two substeps: (i) selecting points randomly, obtaining an updated SVM, and calculating its error on the labeled data set (denote this as ϵ_{random}) and, (ii) selecting data points closest to the current separating hyperplane, obtaining an updated SVM, and determining its error on labeled data set (denote this as ϵ_{query}). A final decision of the number of points to be selected randomly and that of those to be selected on the basis of their distance from the current separating hyperplane is made using ϵ_{random} and ϵ_{query} . An outline of this strategy is given in Algorithm 2. For λ_{query} and λ_{random} , a form similar to AdaBoost [10] has been chosen.

5. EXPERIMENTAL RESULTS

Performance of the proposed active learning strategies has been studied for four benchmark data sets, namely the ionosphere, contraceptive use, Australian credit approval and the credit screening data sets. All data sets are from the UCI Machine Learning Repository [2], and their details are given in Table 1. Missing values were replaced by random values within the range of the corresponding attribute to

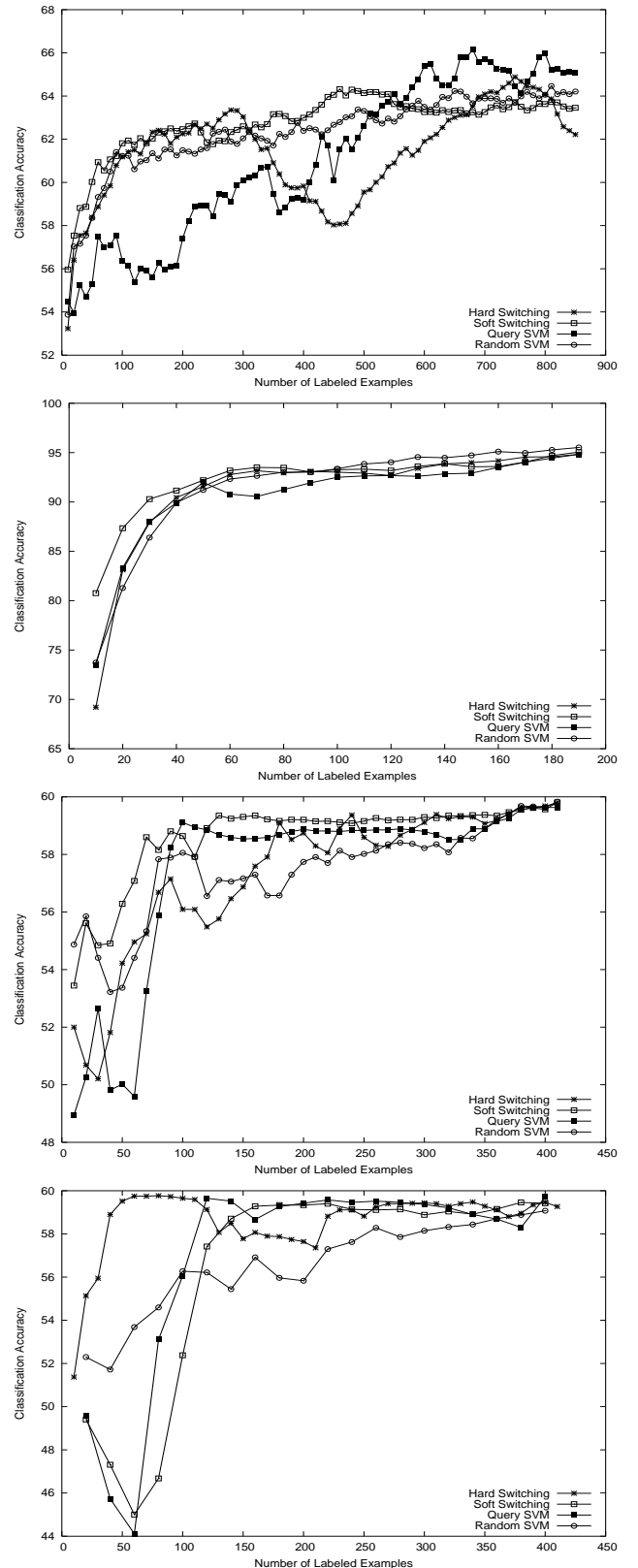


Figure 1: Convergence curves of different active learning algorithms on different data sets. From top to bottom 1. Contraceptive use data set 2. Ionosphere data set 3. Australian credit approval data set 4. Credit screening data set

Data : Unlabeled & Labeled examples
 $T_{unlabeled}, T_{labeled}$

Result: Final classifier C_{final}

$Q_{soft} \leftarrow ChooseRandom(T_{unlabeled}, k)$ /*Initialize the training set with k random examples*/;

$Label(Q_{soft});$

$C_{soft} \leftarrow SVMTrain(Q_{soft});$

$C_{query} \leftarrow C_{soft};$

$C_{random} \leftarrow C_{soft};$

while terminating condition is not met **do**

$Q_{query} \leftarrow Q_{query} \cup$

$ChooseNearest(T_{unlabeled}, k, C_{query})$ /*Add k new datapoints nearest to the hyperplane defined by

C_{query} */;

$Label(Q_{query})$ /*Label new datapoints*/;

$C_{query} \leftarrow SVMTrain(Q_{query});$

$\epsilon_{query} \leftarrow GetError(T_{labeled}, C_{query});$

$Q_{random} \leftarrow$

$Q_{random} \cup ChooseRandom(T_{unlabeled}, k)$ /*Add k new datapoints nearest to the hyperplane defined by C_{random} */;

$Label(Q_{random})$ /*Label new datapoints*/;

$C_{random} \leftarrow SVMTrain(Q_{random});$

$\epsilon_{random} \leftarrow GetError(T_{labeled}, C_{random});$

$\lambda_{query} \leftarrow \left| \ln \left(\frac{1 - \epsilon_{query}}{\epsilon_{query}} \right) \right|;$

$\lambda_{random} \leftarrow \left| \ln \left(\frac{1 - \epsilon_{random}}{\epsilon_{random}} \right) \right|;$

$\lambda \leftarrow \frac{\lambda_{query}}{\lambda_{query} + \lambda_{random}};$

$Q_{soft} \leftarrow Q_{soft} \cup$

$ChooseNearest(T_{unlabeled}, k\lambda, C_{soft})$ /*Mark these selected points as old*/;

$Q_{soft} \leftarrow$

$Q_{soft} \cup ChooseRandom(T_{unlabeled}, k(1 - \lambda))$ /*Mark these selected points as old*/;

$C_{soft} \leftarrow SVMTrain(Q_{soft});$

$Q_{query} \leftarrow SV(C_{soft});$

$Q_{random} \leftarrow SV(C_{soft});$

end

return $C_{soft};$

Algorithm 2: Soft active learning strategy for SVMs

prevent any bias. These data sets have a balanced mix of continuous, few-value nominal and many-value nominal attributes, and thus are appropriate for the experiments.

The convergence curves of all the active learning strategies considered, namely random SVM, query SVM, active learning using follow the perturbed leader and soft choice algorithm, on the above mentioned datasets are shown in Figure 1. The results presented are for a batch size of 10 (i.e., 10 instances are chosen actively at each step). The average value of the classification accuracies over twenty executions are plotted in Figure 4.2.

From the plots, it can be seen that the proposed algorithms achieve the maximum possible accuracy within a close limit with much fewer exams as compared to query and random SVM. This is clearly observed in the case of soft switching in plots 2 and 3, and in the case of hard switching in plots 1 and 4. This characteristic is precisely the goal of active learning and proves especially useful in applications

Table 1: Characteristics of data sets

Data Set	#Samples	#Attributes
Ionosphere	351	34
Contraceptive use	1473	10
Australian credit approval	690	14
Credit screening	690	15

such as bioinformatics, where labelling data may be costly both in terms of money and time.

These approaches are expected to give better results generally since the most appropriate base strategy (random or query SVM) is chosen (or weighted heavily in the case of soft switching) in the most appropriate stage of learning. This corresponds to choosing random SVM in the early stages and query SVM in the later ones, as justified earlier. In order to illustrate this general efficacy, results have been shown on datasets with varied characteristics.

6. CONCLUSIONS AND DISCUSSION

We have proposed two algorithms for fast and robust active support vector learning. The algorithms are motivated by online decision problems and efficiently handle the exploration-exploitation trade-off in active learning.

The superiority of the algorithms has been demonstrated empirically on benchmark datasets. However, a large body of theoretical results on online decision problems is available. Theoretical bounds on the accuracies of the proposed active learning algorithms will be reported in a subsequent article.

7. REFERENCES

- [1] Y. Baram, R. El-Yaniv, and K. Luz. Online choice of active learning algorithms. *JMLR*, 5:255–291, 2004.
- [2] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
- [3] K. Brinker. Incorporating diversity in active learning with support vector machines. In *ICML*, 2003.
- [4] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:1–47, 1998.
- [5] C. Campbell, N. Cristianini, and A. Smola. Query learning with large margin classifiers. In *ICML*, 2000.
- [6] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15:201–221, 1994.
- [7] J.C. Gittins. *Multi-armed Bandit Allocation Indices*. John Wiley, 1989.
- [8] A. Kalai and S. Vempala. Efficient algorithms for the online decision problem. In *COLT*, 2003.
- [9] N.A. Sayeed, H. Liu, and K.K. Sung. A study of support vectors on model independent example selection. In *SIGKDD*, 1999.
- [10] R. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26:1651–1686, 1998.
- [11] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *ICML*, 2000.
- [12] S. Tong and D. Koller. Support vector machine active learning with application to text classification. *JMLR*, 2:45–66, 2001.
- [13] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.