SARA SHANIAN

Selective Sampling for Classification

Mémoire présenté
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de maîtrise en informatique
pour l'obtention du grade de Maître ès sciences (M.Sc.)

Faculté des sciences et de génie
UNIVERSITÉ LAVAL
QUÉBEC

2007

Résumé

Une des objectifs poursuivis par la recherche en apprentissage automatique est la construction de bons classificateurs à partir d'un ensemble d'exemples étiquetés. Certains problèmes nécessitent de réunir un grand ensemble d'exemples étiquetés, ce qui peut s'avérer long et coûteux. Afin de réduire ces efforts, il est possible d'utiliser les algorithmes d'apprentissage actif. Ces algorithmes tirent profit de la possibilité de faire quelques demandes d'étiquetage parmi un grand ensemble d'exemples non-étiquetés pour construire un classificateur précis. Il est cependant important de préciser que les algorithmes d'apprentissage actif actuels possèdent eux-mêmes quelques points faibles connus qui peuvent les mener à performer inadéquatement dans certaines situations. Dans cette thèse, nous proposons un nouvel algorithme d'apprentissage actif. Notre algorithme atténue certains points faibles des précédents algorithmes d'apprentissage actif, et il se révèle trés compétitif aux algorithmes d'apprentissage actif bien-connus. De plus, notre algorithme est facile à implémenter.

Abstract

One of the goals of machine learning researches is to build accurate classifiers form an amount of labeled examples. In some problems, it is necessary to gather a large set of labeled examples which can be costly and time-consuming. To reduce these expenses, one can use active learning algorithms. These algorithms benefit from the possibility of performing a small number of label-queries from a large set of unlabeled examples to build an accurate classifier. It should be mentioned that actual active learning algorithms, themselves, have some known weak points which may lead them to perform unsuccessfully in certain situations. In this thesis, we propose a novel active learning algorithm. Our proposed algorithm not only fades the weak points of the previous active learning algorithms, but also performs competitively among the widely known active learning algorithms while it is easy to implement.

Acknowlegements

I wish to acknowledge many people who have supported me throughout the master's program at Laval University.

I would like to express my warmest thanks to my supervisors Dr. François Laviolette and Dr. Mario Marchand for their guidance and support throughout my master studies. They were my reason and inspiration for coming to Quebec. I am thankful to them for their supervision, teaching, patience and understanding.

A special appreciation also goes to Dr. François Laviolette for not only guiding me but also for his generosity, patience, kindness, and availability. He often steered me in the right direction when I was not certain which path to investigate, I acknowledge his significant contribution to my master thesis. I have indeed been fortunate for having him as a supervisor.

I am indebted to my many student colleagues for providing a stimulating and fun environment in which to learn and grow.

I am grateful to the secretaries and staff in the computer science departments of Laval university, for helping the departments to run smoothly and for assisting me in many different ways.

Lastly and most importantly, I sincerely thank my parents for the many sacrifices that they have made for my studies. I would like to thank my sister, Solmaz for her kindness and encouragement. I also acknowledge my fiancé, Heidar Pirzadeh, for his infinite patience, unwavering belief in me, and not only emotional support but also intellectual support as well. I'm very lucky to have him.

To my parents and my sister

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Classification problems are parts of our daily life. Detecting a friend's voice or face, sorting the clothes based on their material or their colors to fill the washing machine, detecting whether milk has gone bad by tasting or smelling, are all classification problems that we face in everyday life. However all classification tasks are not as simple as these tasks. As an example, take the task of categorizing bank clients who are demanding credit cards into eligible or not eligible. In this task, the bank needs to decide by analyzing the credit records of each person such as his income, other bank accounts, age, and other personal information. This can not be simple due to the huge amount of information and the large quantity of people asking for credit cards. To deal with this problem they use computer to make this job faster and more automated. The computer program which determines if the application should be approved by bank or not is called *classifier*.

Building an accurate classifier is one of the problem addressed by machine learning. In machine learning, regularly a learner trains on completely labeled example data (the data that has already been classified) and creates a classifier. The examples that are used for this task are either labeled manually by a human expert or by a non-manual process that can be expensive. Besides the expenses of this task, it has been observed that labels assigned to the examples by human expert may not be accurate due to fatigue.

On one hand, in some problems it is necessary to gather a large training set. On the other hand, in the labeling process, the accuracy of labeling and the number of examples are inversely related. Considering these two facts, the algorithms which use completely labeled example data, can become impractical.

Then again, unlabeled data are available inexpensively and they are simply accessible. For example, in character recognition task, although gathering examples with handwritten characters is easy, manually labeling each character is a tiresome task. Thus, it would be extremely useful if we could find an algorithm that can improve classification accuracy even if the labeled examples are insufficient and we do not have enough time or means to label them. The machine learning approach which deal with the problem of inadequately labeled examples is called *Active Learning*.

The goal of this thesis is not only building accurate classifier, but also to benefit from the large amount of data that were not being properly exploited due to the limited semi manual means. Thus, here we present an active learning algorithm which helps us to achieve our purpose. In the next section, we present the organization of this thesis.

1.1 Thesis Overview

This section provides the structure of this thesis.

Chapter 2 provides an overview of machine learning.

Chapter 3 describes the formal outline of the algorithms that we will be referring to in further chapters including Support Vector Machine, Set covering Machine, Adaboost with decision stumps as weak learners.

Chapter 4 presents the notations of Active Learning and discuss the issue of Active Learning including different active learning approaches such as Query by committee, uncertainty sampling and also it represents an SVM-based active learning algorithm which is one of the well known active learning algorithms.

Chapter 5 presents some basic definitions of the proposed active learning algorithm along with our new active learning algorithm.

Chapter 6 shows a comparison of the proposed algorithm with related algorithms through an empirical study.

Chapter 7 presents some related works done by other researches in related domains.

Chapter 8 concludes this thesis with some suggestions for future work.

Chapter 2

Machine Learning Overview

2.1 What is Machine Learning?

The ability of a device to improve its performance based on its past performance is called Machine Learning. Considering machine learning as a science, building computer programs which can automatically improve their performance through experience would be the subject of this science. Talking about performance improvement, we might be interested in programs that can adapt to changes, make accurate predictions, can behave with good judgment or learn to do things. Learning is always based on observations or data such as examples, direct experiences or instructions. Thus, machine learning is about learning to perform better in future based on the gained experiences in the past without the human intervention or assistance. In other words, the goal is to devise learning algorithms that do learning automatically. Machine learning can be viewed as programing by examples in which we are interested in finding a method by which the computer will come up with its own program based on the provided examples.

Among various machine learning problems, here we focus on *classification problems* in which the purpose is to categorize objects into a fixed set of categories. Some examples of classification problems are:
Optical character recognition: categorizing images of handwritten character by the represented letters.
Face detections: finding faces in images or indicating whether a face is present or not.
Medical diagnosis: diagnosing whether a patient is suffering or not suffering from some diseases.
Spam filtering: identifying email messages as spam or non-spam.
Topic spotting: categorizing news articles (i.e., whether they are about politics, sports,

entertainment, etc.).
Weather prediction: predicting weather, for instance, whether or not it will rain tomorrow (In this case, we most likely would be more interested in estimating the probability of rain tomorrow).

Besides classification, there are also other important learning problems which are named *regression* problems. In classification, we want to classify objects into fixed set of categories while in regression we try to predict a real value assigned to the objects. For instance, in weather prediction, we may be interested in predicting *How much* it would rain tomorrow.

2.2 Goals of Machine Learning Research

The main goal of machine learning research is to develop general purpose algorithms from an amount of data. Obviously, such algorithms should be efficient. Although from the computer scientists point of view, time and space efficiency are the main issues, in the context of learning, the amount of data which is required by the learning algorithm has a great deal of importance. Since we are interested in algorithms that can be simply applied to a wide range of learning algorithms, the learning algorithms should also be as general as possible. Clearly, we want the result of learning, which may be a prediction rule, to be as accurate as possible. Occasionally, we may also be interested in prediction rules which can be simply interpreted. In other words, in some contexts (such as medical diagnosis), we want the computer to find prediction rules that are easily understandable by human experts.

Another reason to do machine learning research is the advantage of machine learning over direct programing. As it has been mentioned in previous section, machine learning can be considered as programing by examples, so machine learning algorithms are data driven and are able to examine large amounts of data, therefore the results of using machine learning are often more accurate than the outcomes of direct programing. To state the matter differently, consider human expert who is working on a set of examples. Normally, he can examine only a fairly small number of examples in order to gain a knowledge. Although, humans often have trouble expressing what they know, they have no difficulty labeling items using the acquired knowledge. For instance, it is simple for all of us to label images of letters by the character represented, but we would have a great deal of trouble explaining how we do it in exact terms. Since direct programing is based on the knowledge of the programmer and the programmer often has trouble expressing this knowledge, it might not be a good approach in comparison to the machine learning which can be modeled as labeling items by the program which is built based on the examples.

Last but not the least reason to study machine learning is to find responses for the questions which are about the general phenomenon of learning, such as *What are the properties of a given learning problem that make it hard or easy to solve?* or *How much do we need to know about what is being learned in order to be able to learn it effectively?*

Since practical machine learning and theoretical machine learning share mostly the same goals, we hope the theoretical study to be helpful in designing practical algorithms. We also anticipate to analyze these algorithms mathematically to understand their efficiency and difficulties. In addition, we try to explain the phenomena observed in actual experiments with learning algorithms.

2.3 Learning Models

To study machine learning and its problems mathematically, we need to present a formal definition of a possible solution for a learning problem which is called a *learning model*. This learning model should be precise enough to cover all the important aspects of real learning problems. Furthermore, considering the great effect of the assumption simplicity on mathematical study of a problem, the learning model should be also simple. Generally, a learning model should be able to answer several questions such as: *what is the goal of the learning model? what is being learned? where does the data come from? How is the data presented to the learner? (does the learner see all the data at once or it sees only one example at a time?)*

2.3.1 Formalization of a Learning Model

To define a learning model, the following basic definitions are used. An *example* or an *instance* is an object which has been classified. For instance in spam filtering, the email messages are the examples that are being classified as spam or not-spam email. An example can be described by a set of *attributes*, also known as *features* or *variables*. For instance, in medical diagnosis, a patient might be described by attributes such as gender, age, weight, blood pressure, body temperature, etc. As mentioned before in classification problems, we try to classify the examples by predicting their category. Each category is called a *label*. For instance in spam filtering, the possible labels are spam and not-spam. In order to avoid complexity, we assume that only two labels are possibles namely -1 and +1. We also make this simplifying assumption that there is a mapping from examples to labels which is called a *concept*. A concept is a function of the form $f : \mathcal{X} \rightarrow \{-1; +1\}$ where $\mathcal{X}$ is the space of all possible examples called the *domain* or *instance space*. A set of concepts (functions) is called a *concept class (classifier class)*.

Thus, a general model of learning can be described by an instance space $\mathcal{X}$ (for which the examples can be drawn as random vectors $\mathbf{x} \in R^n$ according to the fixed but unknown distribution), a label $y \in \mathcal{Y}$ for every vector $\mathbf{x}$, and a learning algorithm A that constructs a function f from some concept class (classifier class) $\mathcal{H}$ over the instance space. Each $f \in \mathcal{H}$ is known as a classifier. The problem of learning is choosing the best classifier from the given set of functions (concepts) that can approximate the labels of the vectors accurately. This classifier is selected based on the *training set*. A training set S consists of m labeled examples. Each labeled example can be represented by $z = (\mathbf{x}, y)$ where $\mathbf{x}$ is a vector form $\mathcal{X}$ and y is its respective label. It is assumed that each labeled example is drawn independently according to the distribution D.

To measure the accuracy of our classifier, we need to measure the *risk* which is the degree of disagreement between the label y of vector $\mathbf{x}$ and the label assigned to the vector $\mathbf{x}$ by the classifier f denoted by $f(\mathbf{x})$. To define a *risk* we need to define a *loss function* which indicates a quantitative measure of loss when the label y of the vector $\mathbf{x}$ is different from the assigned label by the classifier. We denote the loss function by $L(y, f(\mathbf{x}))$ that outputs the loss when y differs from $f(\mathbf{x})$. Now, according to this loss function, we define the *risk* of classifier as:

$$(2.1) \quad R(f) = \int L(f(\mathbf{x}), y) dD(\mathbf{x}, y)$$

Where $D(\mathbf{x}, y)$ is a fixed but unknown probability distribution according to which, the example $(\mathbf{x}, y)$ is drawn. This risk is referred to as the *true risk* of the classifier f. In the case of zero-one loss, i.e., $L(f(\mathbf{x}), y)$=1 when $y \neq f(\mathbf{x})$ and 0 otherwise. We can write the risk as:

$$(2.2) \quad R(f) = Pr_{(\mathbf{x},y)\sim D}(f(\mathbf{x})) \neq y)$$

Given a training set $S = (z_1, \ldots, z_m)$ of m examples, where $z_i = (\mathbf{x}_i, y_i)$ $\forall i = 1 \ldots m$, the task of the learning algorithm is to construct a classifier with minimum risk without any information about D. Since computing the true risk of a classifier as given above can be difficult (the distribution D is unknown in our model), the learner computes the *empirical risk* $R_S(f)$ which is the risk of the classifier with respect to the training data as follows:

$$(2.3)\quad R_S(f) = \frac{1}{m}\sum_{i=1}^{m} L(y_i, f(\mathbf{x}_i))$$

It can be shown that for any fixed classifier f the empirical risk converges exponentially to the true risk respecting to the sample size m [1].

Generally, there are two main steps in the learning process, namely the training step and the testing step. In the training step, the learning algorithm is supplied with *labeled examples* while during testing only *unlabeled examples* are provided to test the performance of learning algorithm. The diagram illustrating this process is shown in Figure 2.1. The machine learning algorithms is learned on the labeled training examples and it generates the concept function or prediction rule for predicting the category (label) of a new example.

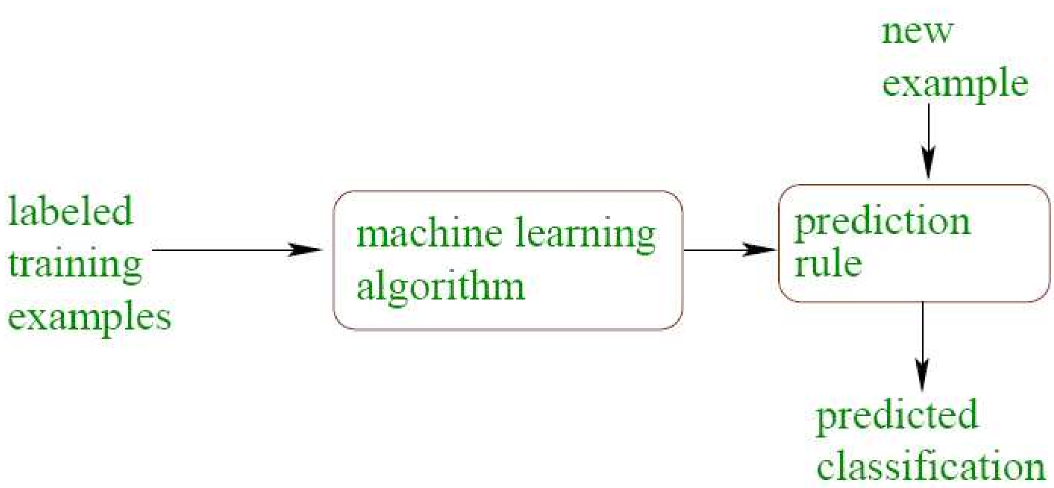


Figure 2.1: The simple diagram illustrating the use of machine learning algorithm.

2.4 Classes of Learning Problems

Machine learning algorithms are divided into three main groups:

- supervised learning: The algorithm is given examples with correct labels and is asked to label new examples.

- unsupervised learning : The algorithm is given only unlabeled examples and is asked to define the labels by grouping the examples.
- semi supervised learning :the algorithm is given both examples with correct labels and unlabeled examples and is asked to label a new example by making use of both labeled and unlabeled data for training.

In the following subsections these three most commonly learning groups are described more precisely.

2.4.1 Supervised Learning

The supervised learning is a machine learning approach for constructing a discrimination function given a set of labeled examples. The labeled examples consist of pairs of input objects and desired outputs (labels). Each labeled example is drawn according to the same but unknown distribution. The output of the function can be a real value (regression) or a prediction of the category label of the input object (classification). In other words, the task of supervised learning is to predict the value of the function for any input having seen a number of labeled examples. To achieve this, the learner should deal with the problem of *generalization* which means after analyzing a set of labeled examples, the learner should output a function that works well on all possible input objects. The supervised learning methods relies on the availability of labeled examples.

2.4.2 Unsupervised Learning

The unsupervised learning is a machine learning technique in which the learning algorithms receives only unlabeled input examples and learns the similarity or the difference of objects. In this way, it finds groups of the examples and the features to distinguish one group from another.

2.4.3 Semi-Supervised Learning

Gathering labeled examples is often difficult and expensive. Furthermore, obtaining labeled examples is time consuming since they require human efforts to obtain them.

Contrarily, unlabeled data may be easy to collect. Considering the stated facts, Semi-supervised learning aims to use unlabeled data among a small amount of labeled data. Many machine learning researchers have found that using unlabeled data with labeled data can improve the accuracy of the learning algorithm. Thus, since semi-supervised learning requires less human effort and in some cases gives us higher accuracy, it is of great interest both in theory and practice.

2.5 Classification

Classification is a special case of supervised learning in which learning algorithms assign labels to examples. Here, we consider binary classification problems where the input space $\mathcal{X}$ consist of an arbitrary subset of $\mathbb{R}^n$and the output space $\mathcal{Y} = \{-1, +1\}$. An example $(\mathbf{x}, y)$ is an input-output pair where $\mathbf{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$. Each example $(\mathbf{x}, y)$ is drawn according to a fixed, but unknown, probability distribution D on $\mathcal{X} \times \mathcal{Y}$. We denote marginal distribution over $\mathcal{X}$ by $D_{\mathcal{X}}$.

The risk $R(f)$ of any classifier f is defined as the probability that it misclassifies an example drawn according to D. Here, by replacing the loss function $L(.,.)$ in Equation 2.3 by the indicator function, the true error in this case can be represented as:

$$R(f) = Pr_{(\mathbf{x},y)\sim D}(f(\mathbf{x}) \neq y) = \mathbf{E}_{(\mathbf{x},y)\sim D} I(f(\mathbf{x}) \neq y)$$

where $I(a) = 1$ if a is true and 0 otherwise. In the same way, the empirical risk $R_S(f)$ can be shown as:

$$R_S(f) = \frac{1}{m} \sum_{i=1}^{m} I(y_i \neq f(\mathbf{x}_i))$$

In machine learning, there are several classification methods. We will present some of them in the next chapter.

Chapter 3

Machine Learning Algorithms

In this section, we present an overview of some machine learning algorithms which have been used throughout this work.

3.1 Support Vector Machine

Support vector machines were proposed as machine learning classification and regression methods by [6]. Support Vector Machines (SVM) are a set of classification and regression methods which were originally related to supervised learning ones. They are used for creating functions from a set of labeled training data where the function can be either classification (the output is binary) or a general regression function. For classification, the operation would be finding a hyperplane in the space of possible inputs. This hyper surface attempts to separate the positive examples from the negative ones. Although there are many hyperplanes that can separate the data, the separation region will be chosen in such a way to provide the largest distance from the hyper surface to the nearest of the positive and negative examples (see Figure 3.1). The examples that lies closest to the hyperplane are called *Support vectors*.

Here, we consider SVMs in the binary classification setting. The training input of SVMs are the the real valued numbers $\{\mathbf{x}_1 \ldots \mathbf{x}_m\}$ which are in some space $\mathcal{X} \in R^n$, associated with their labels $y_i \in \{-1, 1\}$. An optimal separating hyperplane is defined as:

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

where the vector $\mathbf{w}$ is the normal vector perpendicular to the separating hyperplane.

The offset b allows the hyperplane not to pass through the origin. The set of examples is said to be optimally separated by the hyperplane if it is separated without error and the distance between the closest examples to the hyperplane is maximal. Without loss of generality, it is appropriate to consider the canonical hyperplane [3]. Two parallel hyperplanes closest to the canonical hyperplane that separate the data can be considered in either the side of positive examples or the negative ones. The total margin is the perpendicular distance between these two hyperplanes. It can be shown that these two parallel hyperplane can be described by following equations:

$$\mathbf{w} \cdot \mathbf{x} + b = 1,$$

$$\mathbf{w} \cdot \mathbf{x} + b = -1$$

On both side of the optimal separating hyperplane the instances are at least $\frac{1}{\|\mathbf{w}\|}$ away and the total margin is $\frac{2}{\|\mathbf{w}\|}$. Support vectors are examples located on $\mathbf{w} \cdot \mathbf{x} + b = \pm 1$.

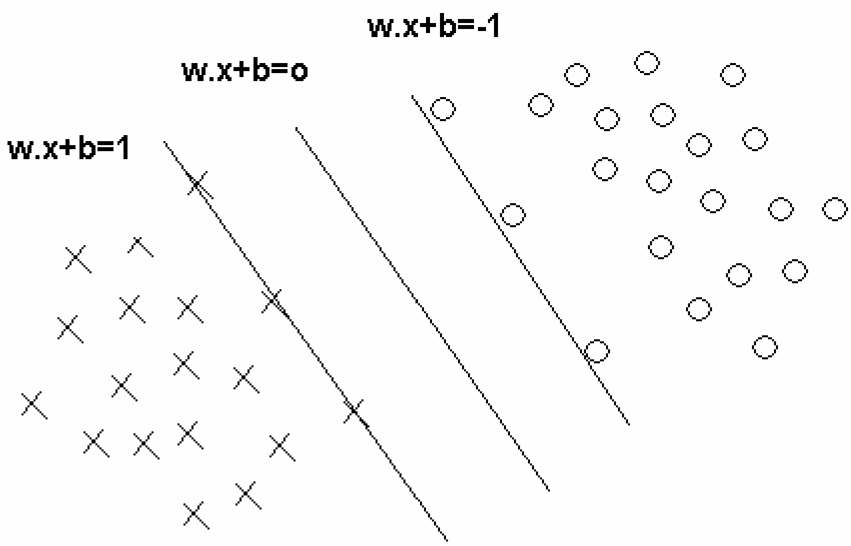


Figure 3.1: The optimal separating hyperplane for SVM trained with samples from two classes. Support vectors are examples located on $\mathbf{w}.\mathbf{x} + b = \pm 1$.

Finding a separating hyperplane in canonical form can be posed as the following problem: $\forall i \in (1, \ldots, m)$ find $\mathbf{w}$ such that $\mathbf{w} \cdot \mathbf{x}_i + b \geq 1$ if $y_i = 1$ or $\mathbf{w} \cdot \mathbf{x}_i + b \leq -1$ if $y_i = -1$. This can be written as:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \; 1 \leq i \leq m$$

The separating hyperplane should be as far away from the data of both classes as possible. On the other hand, we are interested to find a separating hyperplane with maximum margin. By using geometry, we can find out that the distance between two parallel hyperplane is $2/\|\mathbf{w}\|$ so to have a hyperplane with maximum margin we should minimize $\|\mathbf{w}\|$ subject to the constrained $y_i(\mathbf{w}\cdot\mathbf{x}_i+b)\geq 1, 1<i<m$ (for more details, see [4] (section 1.4)).

One idea of SVMs is to map the original training examples into a higher dimensional feature space $\mathcal{F}$ using Mercer Kernel operator K. In other words, the set of classifiers can be considered as:

$$(3.1)\quad f(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{m}\alpha_i K(\mathbf{x}_i,\mathbf{x})\right)$$

If K satisfies Mercer's condition [5] then we can write $K(\mathbf{u},\mathbf{v}) = \boldsymbol{\phi}(\mathbf{u})\cdot\boldsymbol{\phi}(\mathbf{v})$ where $\boldsymbol{\phi}:\mathcal{X}\rightarrow\mathcal{F}$ and $(.)$ indicates an inner product. We can represent f as:

$$(3.2)\quad f(\mathbf{x}) = \operatorname{sign}(\mathbf{w}\cdot\boldsymbol{\phi}(\mathbf{x})),\ where\ \mathbf{w} = \sum_{i=1}^{m}\alpha_i\boldsymbol{\phi}(\mathbf{x}_i)$$

Therefore, by using K, the training data can be projected into a different (higher dimensional) feature space $\mathcal{F}$. The SVMs computes the α_is that corresponds to the maximal margin hyperplane in $\mathcal{F}$. By choosing different kernel function we can project the training data form $\mathcal{X}$ into feature spaces $\mathcal{H}$ for which hyperplanes in $\mathcal{F}$ correspond to more complex decision boundaries in the original space $\mathcal{X}$.

Two commonly used kernels are the polynomial kernel given by $K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v}+1)^p$ which induces polynomial boundaries of degree p in the original space $\mathcal{X}$, and the radial basis function kernel $K(\mathbf{u},\mathbf{v}) = (e^{-\gamma(\mathbf{u}-\mathbf{v}).(\mathbf{u}-\mathbf{v})})$ which induces boundaries by placing weighted Guassians upon key training instances.

The α_i parameters that specify the SVM can be solved in polynomial time by solving a convex optimization problem [6]:

$$minimize\,\alpha\ ,\ \sum_i\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j K(\mathbf{x}_i,\mathbf{x})$$

$$subject\,to:\alpha_i\geq 0,\ \ i=1\ldots m$$

3.2 Set Covering Machine

The Set Covering Machine (SCM) was proposed by [7]. The set Covering Machine algorithm is the generalized form of the two-step algorithm which was proposed by [8, 9]. The set covering Machine extends this algorithm for learning conjunctions and disjunctions of boolean attributes over arbitrary sets of boolean features which are constructed from data (i.e., *Data dependent*). This learning algorithm also provides some learning parameters which controls the trade off between the accuracy and the size of the conjunction or disjunction. In this section, we present the idea of the SCM along with the learning algorithm.

We consider binary classification algorithms. The training input of the SCM are any arbitrary n-dimensional vector $\mathbf{x}$ which are in some subspace $\mathcal{X} \subset R^n$. The training set S consist of two parts, $S = P \cup N$, the set of positive example P and the set of negative example N. A *feature* is an arbitrary boolean valued function that maps $\mathcal{X}$ to $\{0, 1\}$.

Let h_i be a feature $h_i \in \mathcal{F}$. The learning algorithm returns a small subset $R \in \mathcal{F}$ of features from any such set $\mathcal{F}$. The output of SCM is defined to be:

$$f(\mathbf{x}) = \begin{cases} \vee_{i \in R} h_i(\mathbf{x}) & \text{for a disjunction} \\ \wedge_{i \in R} h_i(\mathbf{x}) & \text{for a conjunction} \end{cases}$$

Here, we follow the *consistency* definition given by [7]:

Definition 1 *A function (or a feature) is said to be consistent with an example if it correctly classifies that example. Similarly, a function (or a feature) is said to be consistent with a set of examples if it correctly classifies all the examples in that set.*

Let $\mathcal{P}$ indicates the set P in conjunction case but the set N in the disjunction case. In the same way, let $\mathcal{N}$ denotes the set N in the conjunction set and the set P in the disjunction case. In this case, a classifier f is consistent with $\mathcal{P}$ iff each $h_i \in R$ makes no error on $\mathcal{P}$. Furthermore, let $\mathcal{Q}_i$ be the subset of examples of $\mathcal{N}$ on which feature h_i makes no error. Thus, f makes no error on $\mathcal{N}$ iff $\cup_{i \in R} \mathcal{Q}_i = \mathcal{N}$. Hence, the problem of finding the smallest set R for which f makes no training error can be considered as the problem of finding the smallest collection of $\mathcal{Q}_i$ that covers all $\mathcal{N}$, which is the well known *minimum set cover problem*[10]. Although this problem is NP-hard and it is hard to find the set cover of minimum size, but the greedy algorithm of the minimum set covering problem will always find a cover of size $z\ln(|\mathcal{N}|)$ if the smallest cover is of size z [11].

The set covering greedy algorithm is a simple algorithm : first choose the set $\mathcal{Q}_i$ which covers the largest number of elements in $\mathcal{N}$, remove from $\mathcal{N}$ and each $\mathcal{Q}_j$ the elements that are in $\mathcal{Q}_i$, then repeat this process of finding the set $\mathcal{Q}_k$ of largest cardinality and updating $\mathcal{N}$ and each $\mathcal{Q}_j$ until there are no more elements in $\mathcal{N}$.

The SCM built on the features found by the set covering greedy algorithm will be consistent when there exists a subset of features whose conjunction (or a disjunction) is consistent.

As mentioned before, the Set Covering Machine algorithm provides some learning parameters which control the trade off between the accuracy and the size of the conjunction or disjunction. Indeed, a small SCM which makes a few error on the training set might give better generalization than larger SCM with more feature that makes zero training error. One way to control the trade off between the accuracy and the size of the conjunction or disjunction is to stop the set covering algorithm when there still exists some training examples to be covered. In this case, the SCM has fewer features and also makes errors on those training examples which are not covered. According to the algorithm, the training examples which are not covered by SCM, all belong to $\mathcal{N}$ and, since it is not suitable in general to make all the errors in $\mathcal{N}$, early stopping is not sufficient. Hence, let $\mathcal{Q}_h$ be the set of examples in $\mathcal{N}$ covered by feature h and also let R_h be the set of examples in $\mathcal{P}$ on which h makes an error. Given that, each example in P misclassified by h should decrease by some fixed *penalty* p its importance. The following usefulness function on $\mathcal{H}$ expresses this idea:

$$U_h = |\mathcal{Q}_h| - p.\,|R_h|$$

Thus, the set covering greedy algorithm is modified as follows: Instead of using the feature that covers the largest number of examples in $\mathcal{N}$, the feature $h \in \mathcal{H}$ that has the highest usefulness value U_h is used. We remove from $\mathcal{N}$ and from each $\mathcal{Q}_g$ (for $g \neq h$) the elements that are in $\mathcal{Q}_h$ and we remove from each R_g (for $g \neq h$) the elements that are in R_h. Note that we update each such set R_g because a feature g that makes an error on an example in $\mathcal{P}$ does not increase the error of the machine if another feature h is already making an error on that example. We repeat this process of finding the feature h of largest usefulness U_h and updating $\mathcal{N}$, and each $\mathcal{Q}_g$ andR_g, until only a few elements remain in $\mathcal{N}$ (early stopping the greedy).

3.2.1 Data-dependent Balls

Marchand and Shawe-Taylor[7] gave an implementation of the SCM algorithm with a set of features which is called *data-dependent balls*. In this case, each feature is indicated by a training example, called a *center* $(\mathbf{x}_c, y_c)$, and a radius ρ. The output of $h(\mathbf{x})$ on any input examples of such a feature is given by:

$$h(\mathbf{x}) = \begin{cases} y_c & \text{if } d(\mathbf{x}, \mathbf{x}_c) \leq \rho \\ -y_c & \text{otherwise} \end{cases}$$

where d is the distance between a pair of points of $\mathcal{X}$. It is proposed by [7] to use another training example $\mathbf{x}_b$ called a border point so that $\rho = d(\mathbf{x}_c, \mathbf{x}_b)$

3.3 Adaboost Algorithms

There are several different boosting algorithms; the algorithm that we apply here is AdaBoost[12]. The main idea in this approach is to generate a weighted majority vote of the base rules. This can be done by iteratively generating rules, each next rule being based on the mistakes made by previous rules. The characteristics of AdaBoost are as follows:

1. AdaBoost is a sequential algorithm that minimizes an upper bound of the empirical classification error by selecting weak classifiers h. Each time, a weak classifier is selected to minimize the upper bound of the training error.
2. AdaBoost assigns weights to the training examples. The weights are summed to one. Each time a new weak classifier h is added, these weights should be updated which means that the training examples that are misclassified by h are given more weight so that they receive more "attention" when selecting the next weak classifier.
3. At each step, a new weak classifier is selected to minimize the weighted error, so the algorithm pays more attention to previously misclassified examples.
4. At each step, the weights of the data points are normalized by:

$$Z_t = \sum_{i=1}^{m} \exp(-y_i \alpha_t h_t(\mathbf{x}_i)) D_t(\mathbf{x}_i)$$

The pseudo code on the following subsection gives the outline of the Adaboost Algorithms and we will refer to this algorithm in the coming next chapters.

3.3.1 AdaBoost algorithm

Algorithm 1 AdaBoost algorithm

1: **Initialize:** the training example with uniform weight: $D_1(\mathbf{x}_i) = \frac{1}{m} \, \forall \mathbf{x}_i \in \mathcal{X}$
2: At step t,compute the weighted error for each weak classifier:
$\varepsilon_t(h) = pr_{\mathbf{x}_i \sim D_t}[h_t(\mathbf{x}_i) \neq y_i] \, \forall h \in \mathcal{H}$
3: Choose a new weak classifier which has the least weighted error:
$h_t = argmin \, \varepsilon_t(h) \, \forall h \in \mathcal{H}$
4: Choose a new weak classifier which has the least weighted error:
$h_t = argmin \, \varepsilon_t(h) \, \forall h \in \mathcal{H}$
5: Assign weight for the new classifier:
$\alpha_t = \frac{1}{2} \ln(\frac{1-\varepsilon_t}{\varepsilon_t})$
6: Update the weights of the data points
$D_{t+1}(\mathbf{x}_i) = \frac{1}{Z_t} D_t(\mathbf{x}_i) \exp\{-y_i \alpha_t h_t(\mathbf{x}_i)\}$
7: $t \leftarrow t + 1$
8: repeat 2-7 until maximum step T (t=T)
9: **output:**$H(\mathbf{x}) = sign(\sum_{t=1}^{t} \alpha_t h_t(\mathbf{x}))$;

3.3.2 Decision Stumps

AdaBoost is an algorithm that must be combined with another "weak" learning algorithm, so we need to implement at least one other algorithm (nevertheless the weak learner might work better if the two algorithms are working as a pair). *Decision stumps* can be applied as a weak learning algorithm. Each decision stump is a threshold classifier that depends on a single attribute. Its output is y if the tested attribute exceeds the threshold and $-y$ otherwise, where $y \in \{+1, -1\}$. For a given attribute, the possible values for threshold are determined by the values that the attribute can have. Given data, it is straightforward to search through all possible choices for the attribute to build the decision stump with minimum training error. These make good, truly weak hypotheses for AdaBoost. It is also possible to use disjunctions or conjunctions of the possible thresholds, called respectively, *Decision stump disjunction* and *Decision stump conjunction*.

On can also perform a single test on a single attribute with threshold Θ. In this case the output of $h(\mathbf{x})$ on any input example $\mathbf{x} \in \mathcal{X}$ where $\mathcal{X} \in R^n$ is given by:

$$h(x) = \begin{cases} y & \text{if } \mathbf{x} > \Theta \\ -y & \text{Otherwise} \end{cases}$$

We can also have:

$$h(x) = \begin{cases} y & \text{if } \mathbf{x} < \Theta \\ -y & \text{Otherwise} \end{cases}$$

Chapter 4

Active Learning

4.1 What is Active Learning?

Since gathering experimental data is essential for any learning task, usually we first gather the amount of training data and then we infer a classifier or model from that. This methodology is called passive learning. However, when we want to build an accurate classifier, using a large number of labeled examples for training set is inevitable.

Often, gathering of data is expensive and time consuming so that the learning task can become impractical when it is difficult to gather a large training set. On the other hand unlabeled data are often inexpensively available. Therefore, it would be valuable if we could find a way to come up with the problem of insufficient labeled examples. The common approach in almost all settings is to randomly gather data instances that are independent and identically distributed. However, in some situations, there are some ways for more careful sampling process. For instance, in the document classification task, gathering a large set of unlabeled document is often simple. In this case, instead of randomly selecting the document to be manually labeled for the training set, we have another alternative to choose or query the document more carefully from the set of unlabeled document to be labeled. This guiding sampling process is called active learning.

In other words, active learning includes any form of learning in which the learning program has some control over the inputs it trains on. In active learning, the learning algorithm selects the examples to be labeled and includes them in its training set. Instead of randomly selecting the examples to be labeled, the learning algorithm can more carefully choose or query them from a finite set of examples (*pool based model*) or chooses them from a sequence of examples (*stream-based model*). Hence, it is expected that the amount of training data can be considerably reduced. The common example in this domain is the World-Wide Web, which provides a profusion of training pages for text categorization problems.

4.1.1 Active learner vs Passive learner

A passive learner (see Figure 4.1) receives a random data set from the world and then outputs a classifier while an *active learner* (see Figure 4.2) gathers information form the world by asking queries and receiving responses and then outputs a classifier based on that.

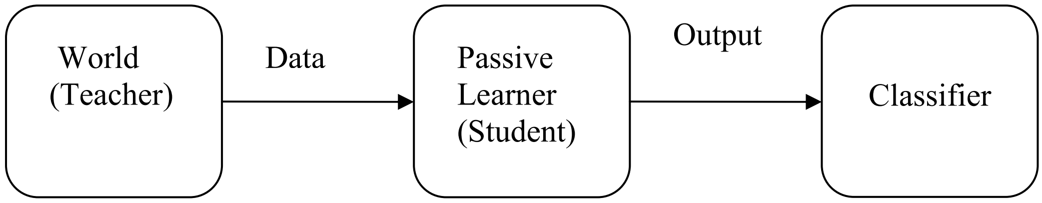


Figure 4.1: General diagram for a passive learner.

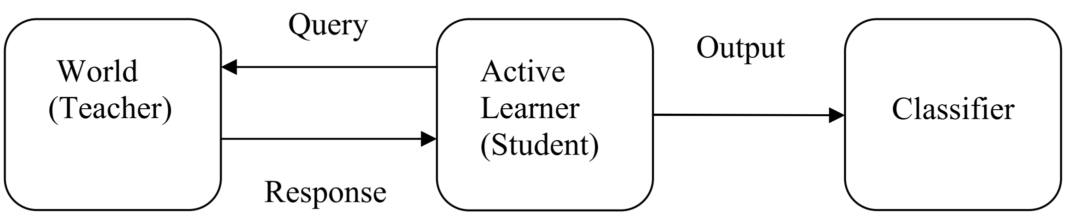


Figure 4.2: General diagram for an active learner.

In other words, a passive learner is a student that gather information by sitting and listening to the teacher whereas an active learner is a student that asks questions to the teacher, listens to teachers and asks more question based on the previous answers. In many situations, the extra ability of asking query of the active learner yields better performance in comparison to the passive learner.

4.2 Active Learning Approaches

Active Learning algorithms aim at reducing the labeling effort in machine learning settings. The major gaol of any active learning algorithms is to obtain a good classifier within a reasonable number of queries (calling for labeling). In order to make a query, the active learner goes through the entire pool or stream of unlabeled examples and selects the example to be labeled next (*selective sampling*).

Active learning methods fall under two main categories based on the criterion used to select the next queries. In the following subsections, the two most commonly used active learning methods are described, namely *Uncertainty Sampling* and *Query by Committee* algorithm.

4.2.1 Uncertainty Sampling

Uncertainty sampling is adjusted on the basis of misclassified examples [16]. This method can be used with any kind of classifier that both predicts the class and provides a measure of how certain that prediction is [17]. In uncertainty sampling approaches, the learner select the examples to be labeled for which there is great uncertainty for classification it makes and then insert them into its training set. Therefore it would benefit from having such examples in its training set.

4.2.2 Query by Committee

Considering that many learning algorithms are not able to add uncertainty estimates to their predictions and also using uncertainty estimates from a single learning algorithm may cause problem in some cases [15], it was suggested by [15] to use a committee of classifiers and measure the uncertainty in the predictions by the degree of disagreement among the classifiers. Another active learning approach is the query by committee algorithm [18], which uses a probability distribution over hypotheses, to randomly select two consistent hypotheses for classifying a new example. If their predictions differ on an example, the algorithm asks for the true label of the example and adds it to its training set [19].

The intuition behind the query by committee idea is that, in a noiseless setting, the fastest way to find the correct concept is to always split the *version space*[1] into two equal parts. Often the version space is so complex that it is not easy to determine

[1]The *Version space* is the set of all hypothesis which are consistent with the data that have been seen so far.

how to bisect it. The Query by Committee approach faces this problem by considering that a query will halve the version space if for any answer,half of the hypothesis will be removed form the version space i.e., the number of hypothesis that classify example as negative should be equal to the number of the hypothesis that classify example as positive (*version space reduction* approach). If a finite number of hypothesis is sampled form the version space, each example from the pool or stream of unlabeled example can be simply verify for this property. The algorithms works well even if a small finite number of hypothesis is randomly sampled. Indeed this approach requires only a logarithmic number of queries in comparison to *random sampling* (i.e., selecting the queries randomly and then label them for adding to the training set) in a noiseless settings[20].

In general, committee based sampling tends to be associated with the version space reduction approach. However, for base learners such as support vector machines, one can use a single hypothesis to make queries that remove half of the version space (*SVM-based algorithm for active learning*) [21]. In the following section, we present briefly this algorithm.

4.3 SVM-based Algorithm for Active Learning

THe SVM-based algorithm for active learning was presented by [21] which is similar to Query by committee approach based on halving the version space. We follow the definitions given by [21] to represent the SVM-based Algorithm for Active Learning.

Given a set of labeled training data and a Mercer kernel K, there is a set of hyperplane in feature space $\mathcal{F}$ that separate the data and is called version space [22]. Thus, hypothesis f is in the version space if for every training examples $\mathbf{x}_i$ with label y_i we have that $f(\mathbf{x}_i) > 0$ if $y_i = 1$ and $f(\mathbf{x}_i) < 0$ if $y_i = -1$. Now, let us consider the set of hypothesis f in feature space that separate the data. More formally:

$$\mathcal{H} = \left\{ f | \ f(\mathbf{x}) = \frac{\mathbf{w}.\phi(\mathbf{x})}{\|\mathbf{w}\|}, \text{ where } \mathbf{w} \in \mathcal{W} \right\}$$

where $\mathcal{W}$ is the parameter space and is equal to $\mathcal{F}$.

The version space, $\mathcal{V}$ then can be defined as:

$$\mathcal{V} = \{ f \in \mathcal{H} | \ \forall i \in \{i \ldots n\} : \ y_i f(\mathbf{x}_i) > 0 \}$$

Since there is a bijection (an exact correspondence) between $\mathbf{w}$ and $f \in \mathcal{H}$, we can redefine $\mathcal{V}$ as:

$$\mathcal{V} = \{\mathbf{w} \in \mathcal{W}| \ \|\mathbf{w}\| = 1, \ y_i(\mathbf{w}.\boldsymbol{\phi}(\mathbf{x})) > 0, \ i = 1 \ldots n\}$$

Thus, by definition, points in $\mathcal{W}$ correspond to hyperplanes in $\mathcal{H}$.

Definition 2 *Area*($\mathcal{V}$) *is the "surface area" that the version space* $\mathcal{V}$ *occupies on the subspace of* $\mathcal{W}$ *defined by* $\|\mathbf{w}\| = 1$.

Suppose we have a new training instance $\mathbf{x}_i$ with label y_i then any separating hyperplane that correctly classify $\mathbf{x}_i$ must satisfy $y_i(\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i)) > 0$. In this case, rather than viewing $\mathbf{w}$ as the normal vector of a hyperplane in $\mathcal{F}$, we think of $y_i\boldsymbol{\phi}(\mathbf{x}_i)$ as being the normal vector of a hyperplane in $\mathcal{W}$ therefore $y_i(\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i)) = \mathbf{w} \cdot y_i\boldsymbol{\phi}(\mathbf{x}) > 0$ defines a half space in $\mathcal{W}$. Moreover $\mathbf{w} \cdot y_i\boldsymbol{\phi}(\mathbf{x}_i) = 0$ defines a hyperplane in $\mathcal{W}$ which acts as one of the boundaries to version space $\mathcal{V}$ (see Figure 4.3 a). Here by SVMs we find the hyperplane that maximize the margin in $\mathcal{W}$. which can be posed as the following problem:

$$Maximize_{\mathbf{w} \in \mathcal{W}} \ \ min_i \{y_i(\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i))\}$$

$$subject\,to : \|\mathbf{w}\| = 1$$

$$y_i(\mathbf{w}.\boldsymbol{\phi}(\mathbf{x}_i)) > 0 \ i = 1 \ldots n$$

These conditions restrict the solution to lie in the version space. Given the duality between $\mathcal{F}$ and $\mathcal{W}$ and since here we assume that the feature vectors are normalized, i.e., $\|\boldsymbol{\phi}(\mathbf{x}_i)\| = 1$, then each $y_i\boldsymbol{\phi}(\mathbf{x}_i)$ is a unit normal vector of a hyperplane in $\mathcal{W}$ that delimits the version space. Thus, we want to find a point ($\mathbf{w}$) that maximize the minimum distance to any of the hyperplane. Here by SVMs we find the center of the largest radius hypersphere whose center is in the version space and whose surface does not intersect with the hyperplanes corresponding to the labeled instances. The touched hyperplanes by this maximal radius hypersphere correspond to the support vectors and the radius of the hypersphere is the margin of the SVM (see Figure 4.3 b).

Considering the definition and concept of the version space according to SVMs, we present the SVM-based active learning algorithm. Here, we consider a pool-based active learning algorithm in which there is a pool of unlabeled instances $\mathcal{U}$. We suppose that the instance are independently and identically distributed according to some underlying distribution. The active learner has three components (f, q, S) where f is a classifier which is trained on the set of labeled instances S (and may be on some instances of $\mathcal{U}$), and where q is a querying function.. As mentioned previous, the main difference between active learner and passive learner is the querying component. It still remains this question that how to choose the next unlabeled instance in the SVMs based algorithms?

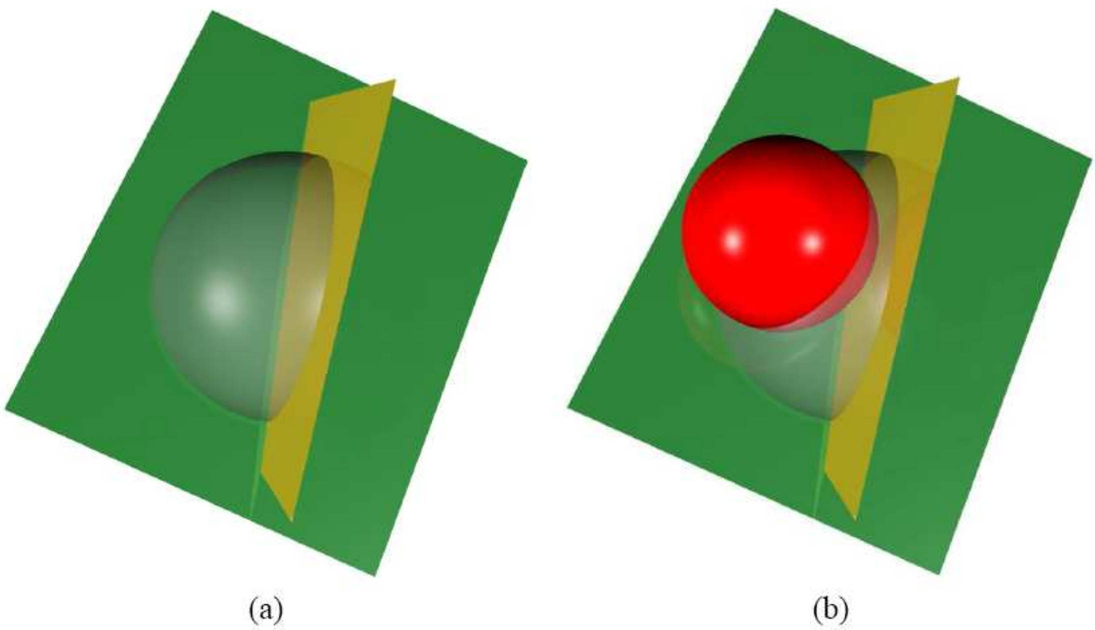


Figure 4.3: (a) Version space duality. The surface of the hypersphere represents unit weight vectors. Each of the two hyperplanes corresponds to a labeled training instance. Each hyperplane restricts the area on the hypersphere in which consistent hypotheses can lie. Here the version space is the surface segment of the hypersphere closest to the camera. (b) An SVM classifier in a version space. The red (dark in black and white printing) embedded sphere has the largest radius value among all the spheres whose center lies in the version space and whose surface does not intersect with the hyperplanes. The center of the embedded sphere corresponds to the SVM classifier, its radius is proportional to the margin of the SVM in $\mathcal{H}$ and the training points corresponding to the hyperplanes.(The picture has been reproduced from [21])

As mentioned before, this algorithm like query by committee algorithms aims to reduce the size of the version space. Therefore, in order to reduce the version space as fast as possible, we can choose a query that halves the version space. The following lemma makes this idea clear.

Lemma 3 *[21] Suppose we have an instance space $\mathcal{X}$, a finite dimensional feature space $\mathcal{F}$ (include via a kernel K), and a parameter space $\mathcal{W}$. Suppose active learner l^* always queries instances whose corresponding hyperplanes in parameter space $\mathcal{W}$ halves the area of the current version space. Let l be any active learner. Denote the version spaces of l^* and l after i queries as $\mathcal{V}_i^*$ and $\mathcal{V}_i$ respectively. Let $\mathbb{P}$ denote the set of all conditional distributions of y given $\boldsymbol{x}$. Then,*

$$\forall i \in N^+ \quad \sup_{p\in\mathbb{P}} \mathbb{E}_p \left[Area(\mathcal{V}_i^*)\right] \leq \sup_{p\in\mathbb{P}} \mathbb{E}_p \left[Area(\mathcal{V}_i)\right]$$

with strict inequality whenever there exists a query $i \in \{1 \dots n\}$ by l that does not halve version space.

The proof of this lemma has appeared in the long version of [21]. Thus, according to this lemma, we wish to find an example form the set of unlabeled data that split the current version space as much as possible. Since there is no practical solution for this problem, three different approximations are suggested in [21]. We represent these approaches in the following subsections.

4.3.1 Simple Margin

In this method, the example closest to the current hyperplane is chosen for query (see Figure 4.4). As mentioned in the previous, section given some data and labels, the SVM unit vector $\mathbf{w}_i$ is obtained form this data. This unit vector, viewed as a point of the space, is the center of the largest hypersphere which can fit inside the current version space $\mathcal{V}^i$. The position of $\mathbf{w}_i$ depends on the shape of version space. Since it is not simple to find out the shape of version, it is often approximated that the position of $\mathbf{w}_i$ is in the center of the version space.

Thus, in this method, we can query the example whose corresponding hyperplane in $\mathcal{W}$ comes closest to the centrally placed vector $\mathbf{w}_i$. The closer the hyperplane in $\mathcal{W}$ is to the point $\mathbf{w}_i$, the more centrally it is placed in the version space and the more it can bisect the version space.

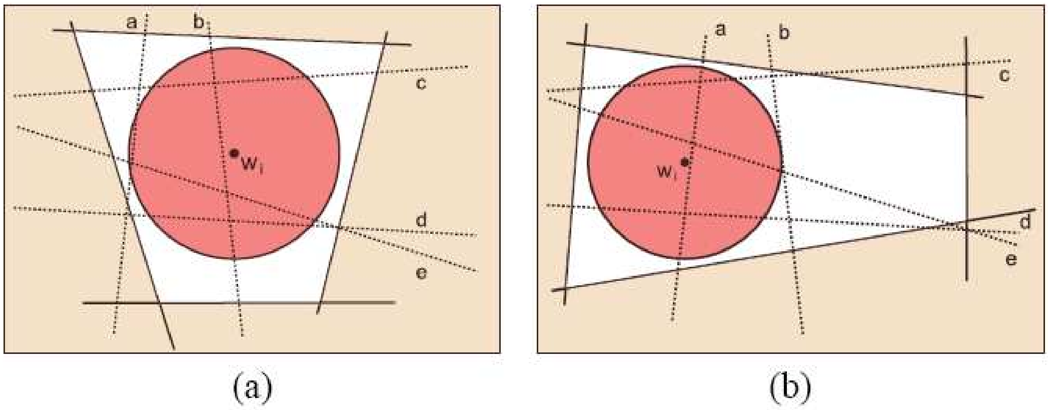


Figure 4.4: (a) Simple Margin will query b. (b) Simple Margin will query a (The picture has been reproduced from [21]).

For each unlabeled example $\mathbf{x}$, the shortest instance between its hyperplane in $\mathcal{W}$ and the vector $\mathbf{w}_i$ is equal to the distance between the feature vector $\boldsymbol{\phi}(\mathbf{x})$ and the hyperplane $\mathbf{w}_i$ in $\mathcal{W}$ which can be computed by $\frac{|\mathbf{w}_i \cdot \boldsymbol{\phi}(\mathbf{x})|}{\|\boldsymbol{\phi}(\mathbf{x})\|}$. Here, we have $\|\boldsymbol{\phi}(\mathbf{x})\| = 1$, thus, $\frac{|\mathbf{w}_i \cdot \boldsymbol{\phi}(\mathbf{x})|}{\|\boldsymbol{\phi}(\mathbf{x})\|} = |\mathbf{w}_i \cdot \boldsymbol{\phi}(\mathbf{x})|$.

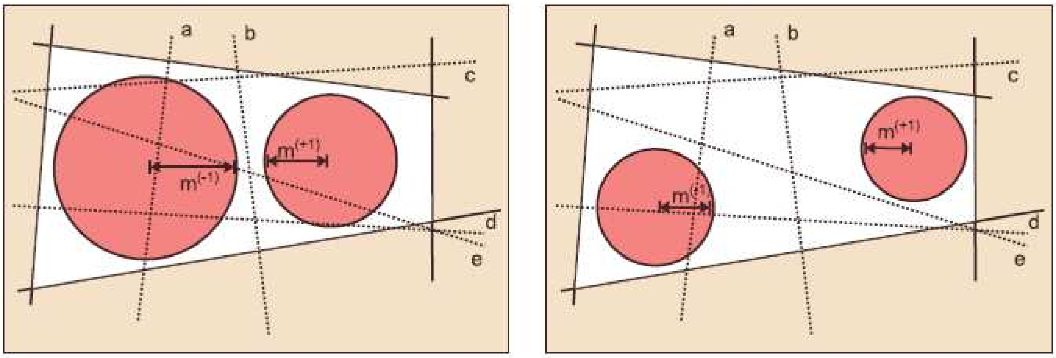
4.3.2 Max Min Margin

In this approach, the examples which maximally reduces the version space of the SVM classifier are selected from the pool (see Figure 4.5 a). In order to achieve this, for each unlabeled example $\mathbf{x}$, the margin m^- and m^+ of the SVM obtained when we label x as negative class and positive class respectively is computed. Then unlabeled instance for which the quantity $\min(m^-, m^+)$ is the greatest will be chosen in order to get a maximum reduction of the version space.

4.3.3 Max Ratio Margin

In this approach instead of maximizing $\min(m^-, m^+)$, for each unlabeled example $\mathbf{x}$ the relative sizes of m^- and m^+ is computed and choose to query the examples for which $\min(\frac{m^-}{m^+}, \frac{m^+}{m^-})$ is the largest in order to get and equal split of version space (Figure 4.5 b).

Among all these three methods, the simple margin is more applicable due to its low computational complexity compared with the other two since it does not have to



(a) (b)

Figure 4.5: (a) Max Min Margin will query b. The two SVMs with margins m^- and m^+ for b are shown. (b) Ratio Margin will query e. The two SVMs with margins m^- and m^+ for e are shown.(The picture has been reproduced from [21])

recalculate the model for every considered example.

Figure 4.4 shows that, in all these algorithms, the version space should not be null ($\mathcal{V} \neq \emptyset$). However, it is possible to use these algorithms when the version space is null ($\mathcal{V} = \emptyset$). Thus, in our experiments, in the following chapters, we disregarded the stated condition.

Chapter 5

Query selection strategy

There are a wide variety of approaches to active learning algorithms. Some active learning algorithms perform better than others depending on the data set. Since there is no consistent winner among these active learning algorithms, it is still unclear how to choose the best active learning algorithm for the problem at hand.

In this chapter, we present our new active learning classification algorithm. Our strategy is the combination of Query by committee and uncertainty sampling approaches while at the same time it tries to reduce the future error of the active learner. Our starting point is the generalization error bounds using unlabeled data proposed by [23]. For classification, the generalization error of a classifier can be bounded by the error of another classifier plus the probability of disagreement between both classifiers. In the following section, we describe the general approach of our active learning algorithm.

5.1 General Approach Of Query Selection Strategy

We now present the outline of our general approach to active learning. Our active learning approach consists of a student as an active learner which learns iteratively on the training set and queries the set of unlabeled examples. In this model, we consider the pool-based model [26] in which the learner is given a small labeled data set as its training set and a set (or a pool) of unlabeled instances. The student is allowed to query an oracle for the labels of a small number of unlabeled instances which has been chosen according to the *teacher*. The teacher is a classifier which help the student learner to find the appropriate examples (the example which the student is uncertain about (uncertainty sampling)) from the pool of unlabeled instances with the help of a *disagreement function*. The disagreement function is the degree of disagreement between the teacher and the student learner on classification of any example (Query by committee). The key step in our approach is to define a disagreement function and the teacher learner in such a way that the result of the student learner at the end of the iteration be as accurate as possible and at the same time the generalization error of the student learner reduces.

To measure the accuracy of our student learner, we have a (testing) set of labeled examples which is separate from its training set. We call it the *test set* and we use it to test the accuracy of the student learner.

To summarize, our general approach for active learning is as follows. We first choose a teacher classifier and disagreement function appropriate for our task. For each query, (to show the performance of our active learning approach), we compare the results of our query selection method with the results of random query selection and the SVM-based algorithm (simple margin). We selected the Simple margin algorithm because it is well motivated and achieves high performance on real world data. We did not use the two other methods (Max Min Margin and Ratio Margin) because they are computationally too expensive. They required SVM to be trained twice, (using positive and negative label), for each query.

5.2 A new query selection Strategy

In this section we present some notions and we propose our classification active learning algorithm based on them.

Here, we present an active learning approach for classification problems, considering a pool-based setting in which the learner is given a small labeled data set, $Z_l = \{z_i\}_{i=1}^m$, with $z_i = (\mathbf{x}_i, y_i)$ where each x_i comes from the input space $\mathcal{X} = \mathbb{R}^n$ and each label y_i is from the output space $Y = \{-1, +1\}$, and also the set of unlabeled data X_U, which is also denoted as a pool of unlabeled examples.

We suppose that each pair $(\mathbf{x}, y) \in Z_l$ is drawn i.i.d. with respect to a fixed but unknown distribution D and we denote the marginal distribution over $\mathcal{X}$ by $D_{\mathcal{X}}$. The learner is allowed to query an oracle for the labels of a small number of unlabeled examples of his choice to add to the labeled set. In this model, the active learning strategy can define the appropriate region for the selection of the unlabeled examples.

As mentioned before, our approach consist two types of classifiers called respectively the teacher and the student classifiers which can be defined as:

Definition 4 ***(Teacher classifier)***: *The teacher classifier is a stochastic classifier, called the* Gibbs classifier, *which is denoted by* H_{Teach}. *Given an input example* $\mathbf{x}$, *the label assigned to* $\mathbf{x}$ *by the teacher classifier (Gibbs classifier) is defined by the following process. We first choose randomly a classifier* h *according to the posterior distribution* Q *and then use* h *to assign the label to* $\mathbf{x}$.

Definition 5 ***(Student classifier)*** *Let* $\mathcal{X}$ *the instance space,* Z_l *be the subset of* $\mathcal{X}\times\mathcal{Y}$, *then the student classifier* h_{Stud} *is an active learner which learns on the training set* Z_l *and queries the pool set* $X_{\mathcal{U}} \subseteq \mathcal{X}$.

Definition 6 ***(The disagreement function)*** *Given a student and teacher classifiers, and a unlabeled example* $\mathbf{x} \in \mathcal{X}$, *the disagreement function on* $\mathbf{x}$ *is the probability that they disagree on* $\mathbf{x}$, *more precisely:*

$$\boldsymbol{d}_{\mathbf{x}}(h_{Stud}, H_{Teach}) \stackrel{\text{def}}{=} \boldsymbol{E}_{h\sim Q} I(h_{Stud}(\mathbf{x}) \neq h(\mathbf{x}))$$

Clearly, $\boldsymbol{d}_{\mathbf{x}}$ *is upper bounded by* 1 *(i.e.,* $1 \geq \boldsymbol{d}_{\mathbf{x}}(h_{Stud}, H_{Teach}) \geq 0$ *).*

Definition 7 *The risks of the student and teacher classifiers are defined as follows:*

$$R(h_{Stud}) \stackrel{\text{def}}{=} Pr_{(\mathbf{x},y)\sim D}(h_{Stud}(\mathbf{x}) \neq y) \stackrel{\text{def}}{=} \boldsymbol{E}_{(\mathbf{x},y)\sim D} I(h_{Stud}(\mathbf{x}) \neq y)$$

$$R(H_{Teach}) \stackrel{\text{def}}{=} \boldsymbol{E}_{h\sim Q} R(H_{Teach}) = \boldsymbol{E}_{h\sim Q} \boldsymbol{E}_{(\mathbf{x},y)\sim D} I(h(\mathbf{x}) \neq y)$$

Similarly, the empirical risks of the student and teacher classifiers on the training set Z_l *are defined as follows:*

$$R_{Z_l}(h_{Stud}) \stackrel{\text{def}}{=} \boldsymbol{E}_{(\mathbf{x},y)\sim Z_l} I(h_{Stud}(\mathbf{x}) \neq y) = \frac{1}{m}\sum_{i=1}^{m} I(h_{Stud}(\mathbf{x}_i) \neq y_i)$$

$$R_{Z_l}(H_{Teach}) \stackrel{\text{def}}{=} \boldsymbol{E}_{h\sim Q} R_{Z_l}(H_{Teach}) = \boldsymbol{E}_{h\sim Q} \sum_{i=1}^{m} I(h_{Stud}(\mathbf{x}_i) \neq y_i)$$

5.3 Generalization Error Bound for the active learner

To obtain a bound for the active learner, we present Lemma 9 based on the presented disagreement function and lemma 10 and from them we get the Theorem 11 which gives us an upper bound for an active learner.

Lemma 8 *For any* $\mathbf{x} \in \mathcal{X}$ *we have :*

$$\boldsymbol{d}_{\mathbf{x}}(h_{Stud}, H_{Teach}) = |\boldsymbol{E}_{h\sim Q} I(h(\mathbf{x}) = 1) - I(h_{Stud}(\mathbf{x}) = 1)|$$

Proof: *Using Definition 6, we get:*

If $h_{Stud}(\mathbf{x}) = 1$ *then:*

$$\begin{aligned}
\boldsymbol{d}_{\mathbf{x}}(h_{Stud}, H_{Teach}) &= \boldsymbol{E}_{h\sim Q}(I(h(\mathbf{x}) = -1) \\
&= \boldsymbol{E}_{h\sim Q}(1 - I(h(\mathbf{x}) = 1) \\
&= 1 - \boldsymbol{E}_{h\sim Q}(I(h(\mathbf{x}) = 1) \\
&= I(h_{Stud}(\mathbf{x}) = 1) - \boldsymbol{E}_{h\sim Q}(I(h(\mathbf{x}) = 1) \\
&= |\boldsymbol{E}_{h\sim Q}(I(h(\mathbf{x}) = 1) - I(h_{Stud}(\mathbf{x}) = 1)|
\end{aligned}$$

If $h_{Stud}(\mathbf{x}) = -1$ *then:*

$$\begin{aligned}
\boldsymbol{d}_{\mathbf{x}}(h_{Stud}, H_{Teach}) &= \boldsymbol{E}_{h\sim Q}(I(h(\mathbf{x}) = 1) \\
&= \boldsymbol{E}_{h\sim Q}(I(h(\mathbf{x}) = 1) - I(h_{Stud}(\mathbf{x}) = 1) \\
&= |\boldsymbol{E}_{h\sim Q}(I(h(\mathbf{x}) = 1) - I(h_{Stud}(\mathbf{x}) = 1)|
\end{aligned}$$

Lemma 9 *For all* $h_1, h_2 \in \mathcal{H}$ *we have :*

$$I(h_1(\mathbf{x}) \neq y) \leq I(h_2(\mathbf{x}) \neq y) + I(h_1(\mathbf{x}) \neq h_2(\mathbf{x}))$$

Proof: *Clearly the lemma is valid if* $h_1(\mathbf{x}) = y$*. In the case of* $h_1(\mathbf{x}) \neq y$ *we have: either* $h_2(\mathbf{x}) \neq y$ *which implies the result, or* $h_2(\mathbf{x}) = y$ *which also implies the result since then* $h_1(\mathbf{x}) \neq h_2(\mathbf{x})$*.*

Lemma 10 *For any posterior distribution Q, and any student h_{Stud} classifier we have:*

$$I(h_{Stud}(\mathbf{x}) \neq y) \leq \boldsymbol{E}_{h \sim Q}(I(h(\mathbf{x}) \neq y) + \boldsymbol{E}_{h \sim Q}\left[(I(h(\mathbf{x}) \neq h_{Stud}(\mathbf{x})))\right]$$

Proof: *The proof is straightforward from Lemma 9.*

Theorem 11 *For any teacher H_{Teach} and student h_{Stud} classifier we have:*

$$R(h_{Stud}) \leq R(H_{Teach}) + \boldsymbol{E}_{\mathbf{x} \sim D_{\mathcal{X}}} \boldsymbol{d}_{\mathbf{x}}(h_{Stud}, H_{Teach})$$

Proof: *using the Definition 7 and Lemma 10, we get:*

$$R(h_{Stud}) = \boldsymbol{E}_{(\mathbf{x},y) \sim D} I(h_{Stud}(\mathbf{x}) \neq y) \leq \boldsymbol{E}_{(\mathbf{x},y) \sim D} \boldsymbol{E}_{h \sim Q} I(h(\mathbf{x}) \neq y) + \boldsymbol{d}_{\mathbf{x}}(h_{Stud}, H_{Teach})$$

$$= R(H_{Teach}) + \boldsymbol{E}_{(\mathbf{x},y) \sim D} \boldsymbol{d}_{\mathbf{x}}(h_{Stud}, H_{Teach})$$

Since according to Lemma 8, the value of $\boldsymbol{d}_{\mathbf{x}}(h_{stud}, H_{Teach})$ is the same for both $(\mathbf{x}, -1)$ and $(\mathbf{x}, +1)$, we can replace $\boldsymbol{E}_{(\mathbf{x},y) \sim D}$ by $\boldsymbol{E}_{\mathbf{x} \sim D_{\mathcal{X}}}$ as:

$$= R(H_{Teach}) + \boldsymbol{E}_{\mathbf{x} \sim D_{\mathcal{X}}} \boldsymbol{d}_{\mathbf{x}}(h_{Stud}, H_{Teach})$$

Therefore, by using unlabeled data, we can deduce an upper bound on the risk $R(h_{Stud})$ from an upper bound on the risk H_{Teach}. In the following subsection, we show that there is an upper bound on the risk of H_{Teach}.

5.3.1 A PAC-Bayes Bound for Teacher classifier

Recall that the risk of the the Gibbs classifier G_Q is defined as the expected risk of classifiers drawn according to Q. Hence,

$$R(G_Q) = \mathbf{E}_{h \sim Q} R(h)$$

Similarly, the empirical risk $R_S(G_Q)$ of G_Q, on training sequence S of examples, is given by :

$$R_S(G_Q) = \mathbf{E}_{h \sim Q} R_S(h)$$

Theorem 12 *[24, 25] For any Set $\mathcal{H}$ of binary classifiers, any prior distribution P on H, and any $\delta \in (0, 1]$, we have:*

$$\Pr_{S \sim D^m}(\forall Q : kl(R_S(G_Q) || R(G_Q) \leq \frac{1}{m}\left[KL(Q||P) + \ln \frac{m+1}{\delta}\right]) \geq 1 - \delta$$

where $KL(Q||P)$ denotes the Kullback-Leibler divergence between the distribution Q and P:

$$KL(Q||P) = \boldsymbol{E}_{h\sim Q} \ln \frac{Q(h)}{P(h)}$$

and where $kl(p||q)$ denotes the Kullback-Leibler divergence between Bernouli distribution with probability of failure q and probability of success p:

$$kl(p||q) = q \ln \frac{q}{p} + (1-q) \ln \frac{1-q}{1-p}.$$

5.4 A Risk Bound for Active Learning

We are interested to find a bound which is uniformly valid for all the classifiers and all possible sequence of queries. This can be done in the same way as for the sample compression in supervised learning via the union bound.

In the sample compression scheme, given a (labeled) training set S of an *a priori* defined size m, any classifier returned by a learning algorithm is described by a compression set. A *compression set* is a subset of the training set S and therefore, when S is given, it can be described as a vector of indices $\vec{i} = (i_1, i_2, \ldots, i_\eta)$ with $i_j \in \{1, \ldots, m\}\ \forall j$ and $i_1 < i_2 < \ldots < i_\eta$. This implies that there exists a deterministic *reconstruction function*, associated with the algorithm, that outputs a classifier when given a training set and a vector of indices. Given an *a priori* defined vector $\vec{i}$ of indices, one can use the examples of the training set that do not correspond to any index of $\vec{i}$ to bound the risk of the classifier defined by $\vec{i}$ (and the training set S). Moreover, provided a prior distribution is given on the set of all possible vector of indices, one can extend a bound (for the risk of the classifier) which is valid simultaneously for all classifiers that can be reconstructed. Since any active learning classification algorithm $\mathcal{R}$ considered here is deterministic, the set of all possible classifiers that can be output by $\mathcal{R}$ depends only on the set of all examples of $X_{\mathcal{U}}$ that have been queried during the execution together with all the corresponding labels that have been given in response to the queries. Moreover, if we make the following assumption:

Assumption 13 *There exists a deterministic function $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{Y}$ such that for all $(\mathbf{x}, y)$ drawn according to D, we have $y = \boldsymbol{\phi}(\mathbf{x})$.*

The set of all possible classifiers that can be output by $\mathcal{R}$ will then depend only on the labeled set Z_l together with the final set of all the activated examples. Thus, as for the sample compression scheme, we have a reconstruction function associated with $\mathcal{R}$. We can therefore apply the same techniques as for the sample compression scheme to deduce risk bounds that will be valid for all classifiers that can be reconstructed. The next results formalize this idea.

Starting from the whole set $X_{\mathcal{U}}$ of unlabeled examples, minimizing the generalization error of h_{Stud} can then be done by considering a subset of $X_{\mathcal{U}}^{(\eta)}$ of η elements of $X_{\mathcal{U}}$ for which the value of $\mathbf{d}_{\mathbf{x}}(h_{Stud}, H_{Teach})$ are maximal. Then, we can ask for the labels of $\mathbf{x} \in X_{\mathcal{U}}^{(\eta)}$ and learn h_{Stud} on $Z_l \cup Z_{\mathcal{U}}^{(\eta)}$, where $Z_{\mathcal{U}}^{(\eta)}$ denotes the labeled dataset, together with examples $X_{\mathcal{U}}^{(\eta)}$ that have been activated.

In the following, we suppose that

$$Z_l = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_m, y_m)\}$$

and $X_{\mathcal{U}} = \{\mathbf{x}_{m+1}, \mathbf{x}_{m+2}, \ldots \mathbf{x}_{m+n}\}$. Moreover, we will also suppose that any single query of the classification active learning algorithm corresponds to an activation of exactly η unlabeled data (for a parameter η fixed *a priori*), the total number of activated examples will then always be of the form $\eta \cdot t_A$ for some $t_A \in \mathbb{N}$. Thus, the compression set of any classifier related to the algorithm $\mathcal{R}$ is the union of the set Z_l and a subset of size $\eta \cdot t_A$ of $X_{\mathcal{U}}$. The set of the labeled data is always in the compression set because the algorithm always consider Z_l. Now, one can define a prior distribution on the set of all outputs of $\mathcal{R}$ by defining a prior $P_{\mathbb{N}}$ on $\mathbb{N}$ together with, for each t_A that has weight in $P_{\mathbb{N}}$, a prior P_{t_A} on the set of all possible vector of indices of the forms $\vec{i} = (1, 2, \ldots, m, i_1, i_2, \ldots, i_{\eta t_A})$ with $i_j \in \{m+1, \ldots, m+n\}$ $\forall j$ and $i_1 < i_2 < \ldots < i_{\eta t_A}$.

Most of the time, the prior $P_{\mathbb{N}}$ will have all its weights on the set $\{1, 2, \ldots T_A\}$ for some parameter T_A defined *a priori*. Moreover, unless n is too big, since the examples of $X_{\mathcal{U}}$ are assumed to be i.i.d., we will choose P_{t_A} as the uniform distribution under the constraint that the m first indices must always be chosen, that is $P_{t_A}(\vec{i}, t_A) = \binom{n}{\eta t_A}^{-1}$ for any $(\vec{i}, t_A)$. We will denote by $\mathcal{R}_{(\vec{i}, t_A)} : \mathcal{X} \to \{0, 1\}$ the corresponding active learner. Under those assumptions and the Hoeffding Bound (Theorem 14) we have Theorem 15.

Theorem 14 *(Hoeffding) Let $X_1, ..., X_n$ be n copies of a $[0, 1]$-valued random variable X, then, for all $\delta > 0$:*

$$\mathbb{P}\Big(\mathbb{E}X \leq \frac{1}{n}\sum_{i=1}^{n} X_i + \sqrt{\frac{\ln(1/\delta)}{2n}}\Big) > 1 - \delta$$

Theorem 15 *Let $\mathcal{R}$ be any classification active learning algorithm whose queries are all of size η. Let $P_{\mathbb{N}}$ and $\{P_{t_A}\}_{t_A \in \mathbb{N}}$ be the priors defined above. Finally, let H_{Teach} be a teacher classifier. Then we have:*

$$\Pr\left(\begin{array}{l}\forall t_A \in \mathbb{N} \text{ and } \forall \vec{i} = (1,2,\ldots,m,i_1,i_2,\ldots,i_{\eta t_A}), \\ \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D}_{\mathcal{X}}} \boldsymbol{d}_{\mathbf{x}}\left(\mathcal{R}_{(\vec{i},t_A)},\, H_{Teach}\right) \leq R_{Z_l \cup X_{\mathcal{U}}^{(\eta t_A)}} + \sqrt{\dfrac{\ln\binom{n}{\eta t_A} - \ln P_{\mathbb{N}}(t_A) + \ln(2/\delta)}{2(n-\eta t_A)}}\end{array}\right) \geq 1-\delta$$

where $R_{Z_l \cup X_{\mathcal{U}}^{(\eta t_A)}} \stackrel{\text{def}}{=} \frac{1}{n-\eta t_A} \sum_{\mathbf{x}\in X_{\mathcal{U}} \setminus X_{\mathcal{U}}^{(\eta t_A)}} \boldsymbol{d}_{\mathbf{x}}\left(\mathcal{R}_{(\vec{i},t_A)},\, H_{Teach}\right)$.

Proof: *For each $(\vec{i}, t_A)$, we use Hoeffding inequality [with $\delta := \frac{\delta . P_{\mathbb{N}}(t_A) \cdot P_{t_A}(\vec{i},t_A)}{2}$] and then apply the union bound.*

To obtain an upper bound on the risk of h_{Stud}, we combine Theorem 11, Theorem 12 and Theorem 15 with δ replaced by $\delta/2$ which bounds the two terms of the right part of Theorem 11. Although this bound is not tight, it at least gives us some convergence guarantees.

5.5 Optimization of the Query Selection Strategy

In this section we present two approaches for optimizing our proposed active learning algorithm on some data sets. These two approaches are called *Soft Max Query Selection* and *Probabilistic Active Learning*[44].

5.5.1 Soft max action selection strategy

Since our new query selection strategy which is based on the disagreement function selects the examples that have the maximum value of disagreement to be labeled, it is a greedy algorithm. Here we try to redefine our query selection strategy based on the soft max selection method in Reinforcement learning. In the following subsections, we first review the idea behind using soft max action selection strategy in reinforcement learning and then we redefine our query selection strategy based on that.

Soft Max Action Selection Strategy in Reinforcement Learning:

Reinforcement learning (RL) [43] is a framework for computational learning agents that use experience from their interaction with an environment to improve performance over time. It receives positive payoff or negative payoff as the result of the selection of an action. The agent must learn to choose actions in order to maximize a long term sum or average of the future payoffs it will receive. There are several simple heuristic exploration methods in this setting. One method can be selection of an action that maximizes its positive payoff called $\epsilon - greedy$ action selection.

Although $\epsilon - greedy$ action selection is an effective means of balancing exploration and exploitation in reinforcement learning its drawback is that when it explores it chooses equally among all actions. This means that it is likely to choose the worst-appearing action as it is to choose the next-to-best action. The most obvious solution is the *soft max* action selection method. It chooses action a on the t^{th} play with probability.

$$\frac{e^{Q_{t(a)}/\tau}}{\sum_{b=1}^{n} e^{Q_{t(b)}/\tau}}$$

where τ is a positive parameter called the *temperature* and $t(a)$ is the action a on the the t^{th} play . This parameter controls the probability of executing actions other than the one with the highest Q-value. If τ is high, or if Q-values are all the same, then a random action will be picked. If τ is low and Q-values are different, it will tend to pick the action with the highest Q-value. In the limit $\tau \to 0$, the soft max selection becomes the greedy action selection. One problem of the soft max action selection mechanism is how to determine the value of τ, which depends on the task.

Soft Max Query Selection Methods:

Our active learning query selection strategy is a greedy algorithm since, in our proposed algorithm the active learner selects the examples which maximize the disagreement function. As mentioned before, the disagreement function is the probability of disagreement between the active learner (student) and the other classifier which is called teacher in our setting. Here, we define a new query selection function, P, based on the disagreement function and the soft max idea in reinforcement learning as:

$$(5.1) \quad P_{\mathbf{x}}^{t_A} = \frac{e^{\mathbf{d}_{\mathbf{x}}^{t_A - 1}(h_{Stud}, H_{Teach})/\tau}}{\sum_{b=1}^{n} e^{\mathbf{d}_{\mathbf{x}_b}^{t_A - 1}(h_{Stud}, H_{Teach})/\tau}}$$

Where the $\mathbf{d}_{\mathbf{x}}^{t_A-1}(h_{Stud}, H_{Teach})$ is the probability of disagreement on example $\mathbf{x}$ in the round $t_A - 1$ and n is the number of unlabeled examples in the pool. Here, we keep τ unchanged on all the rounds of our active learning algorithm.

5.5.2 Probabilistic Active Learning

Since our described active learning algorithm proceeds heuristically, it may not always be advantageous to select the top-ranked query ([44] and this has been confirmed in our experiments as well). In [44] they proposed to use a variant that sorts all the examples in the pool according to the active learning's heuristics, and then choose an item randomly from the top $p\%$ of the pool in place of choosing the top-ranked example[1]. Empirically, it has been shown that the probabilistic active learners outperform the strict algorithms by a factor between 1% and 6% and never result in performance decreamnet[44].

5.6 Implementation of the approach

To implement our active learning algorithm, we used SCM, SVM as active learners. In order to construct the teacher classifier (Gibbs classifiers) with posterior probability we used the Adaboost algorithm and modified its output in the following way. The posterior probability for the generated classifier h_t at each round is defined by:

$$Q_t = \frac{\alpha_t}{\sum_{i=1}^{T} \alpha_i}$$

Where α_t is the weight which is assigned to the new classifier h_t at round t by Adaboost algorithm.

Algorithm 2 described below is our proposed active learning algorithm. The active learning algorithm with soft max idea would be the same as the algorithm 2 except, in line 8 of the algorithm 2 where instead of choosing the examples that have the maximum value $\mathbf{d}_{\mathbf{x}}(h_{Stud}, H_{Teach})$, the algorithm sorts the $P_{\mathbf{x}}^{t_A}$ values and then it produces a random number between zero and one. The next step is to find the interval (between two Ps) to which the random number belongs. In this way the P interval determines

[1] In our experiment $p\% = 10\%$

the example which should be selected. The examples which this P value belongs to, would be queried.

Algorithm 2 Active learning algorithm for classification

Input: A set of labeled Z_l and unlabeled examples $X_{\mathcal{U}}$;
Input: η the number of examples to be activated at each round;
Input: T_A: the maximum number of rounds;
Input: The teacher classifier H_{Teach} (computed from Z_l);
Initialize:$Z_{\mathcal{U}}^{(\eta)} \leftarrow \emptyset$, $t_A \leftarrow 1$;
repeat
    Learn h_{Stud} on $Z_l \cup Z_{\mathcal{U}}^{(\eta)}$
    Select a subset $X_{\mathcal{U}}^{(\eta)}$ from $X_{\mathcal{U}}$ such that for $\mathbf{x} \in X_{\mathcal{U}}$ the value $\mathbf{d_x}(h_{Stud}, H_{Teach})$ is maximal
    Ask for the label y of $\mathbf{x}$ for each $\mathbf{x} \in X_{\mathcal{U}}^{(\eta)}$
    $Z_{\mathcal{U}}^{(\eta)} \leftarrow Z_{\mathcal{U}}^{(\eta)} \cup \left\{(\mathbf{x}, y) \mid \mathbf{x} \in X_{\mathcal{U}}^{(\eta)}, \textit{and } y \textit{ is the queried label}\right\}$
    $t_A \leftarrow t_A + 1$
until $t_A > T_A$
output:h_{Stud} ;

Chapter 6

Empirical Results

6.1 Experiment Setup

In order to test the performance of the proposed active learning algorithm, we need huge data set. We have evaluated our active learning algorithm on five artificial binary numerical data sets (two class data set in which each attribute is a number) (splice, Ringnorm, Waveform, Twonorm, image) from the UCI repository. We have also tested the active learning algorithm on two other huge data sets from the UCI repository namely Mushroom and Pendigit data sets which are respectively non-numerical and non-binary class data sets. Since mushroom is not a numerical data set and Pendigit is not a binary class data set, we transformed them to two equivalent data sets which can fit to our work appropriately, as follows.

The mushroom data set is a two class data set which contains 22 categorical attributes. We transformed each categorical attribute into a series of real attributes. For example, this data set contains an attributes called “cap surface” which can take one of four categories namely “fibrous”,“grooves”, “scaly” or “smooth”. We represent these categories as four real value attributes. Each of them formulated as $\frac{i}{(C-1)}$ where i is the index number of actual category which starts from zero, and C is the total number of categories. Thus the categorical value “fibrous” would be represented by $\frac{0}{3} = 0$ while the value of “scaly” is $\frac{2}{3} = 0.66$. The final data set we used contained 22 features with 8000 points.

The Pendigits data set has 7494 data points. Each represented by 16 attributes. Originally, this dataset is designed to be used for multi-class classification with a total of 10 classes(one for each digit ranging from 0 to 9). Instead, we transformed it into a binary classification task by assigning the negative class to the numbers (6, 8, 9) and the positive class to the remaining digits. (0, 1, 2, 3, 4, 5, 7).

On Mushroom and Pendigit datasets the following specification was performed: We divided them into three parts. The first part had 100 examples as the training set, the second part had 1800 examples as the test set, and the last part contained 6000 examples as the pool set.

Each of the remaining five artificial binary numerical data sets, was split into two parts containing 60 percents and 40 percents of examples, respectively as *Pool set plus training set* and *test set*. Then from the 60 percents data (pool set plus training set) we used 100 examples as a initial training set and the remaining examples as a pool set. Please refer to appendix A for the data sets that were used.

Our proposed active learning algorithm (which may be referred to as *Stud-Teach*) was applied on each data set with $T_A = 200$ (maximum number of rounds) and $\eta = 5$, (the number of activated query in each round). On each round, when the new examples were added the h_{Stud} classifier was retrained and tested on the test set. Please refer to appendix B for algorithms used.

The performance of an active learner can be evaluated using different metrics. These different metrics are usually test set classification error and the accuracy on the test set. The test set classification error is test error vs. number of rounds which is defined as:

$$(6.1)\ \ The\ test\ set\ error = \frac{Number\ of\ test\ examples\ misclassified}{Number\ of\ test\ examples}$$

The accuracy on the test is defined as:

(6.2)

$$The\ accuracy = 1 - \frac{Number\ of\ test\ examples\ misclassified}{Number\ of\ test\ examples} = 1 - The\ test\ set\ error$$

The accuracy on the test set and the test set classification error are corresponding to each other. The higher accuracy means the lower test set classification error and the higher test set classification error corresponds to a lower accuracy. Here, in this thesis, we used test set classification error as the metric except for the section 6.3 in which we gained from this correspondence to match the descriptive notion of "accuracy" with the mathematical definition of accuracy given here.

6.2 Using the SVM as an Active Learner

Our first experiments were on Mushroom and Pendigit data set. In these experiments the active learner (student) and teacher classifier used respectively SVM with linear kernel and Adaboost with Decision stump disjunction (see chapter 3, section 3.3.2) as their learning components. We compared the Stud-Teach algorithm with the random sampling algorithm because active learning by random sampling is the only algorithm for which there exists a proof of convergence to the optimal classifier. This is true for the reason that in random sampling the training set (train+queried) remains i.i.d..

Figure 6.1 shows the comparison between the results of applying our proposed algorithm and the results of random sampling on Pendigit. The same comparison is presented in Figure 6.2 on Mushroom data set. In these figures the solid lines correspond to selective sampling (stud-Teach) and the dotted lines are related to random sampling.

A visual inspection of the Figure 6.1 shows that in mushroom data set at early stages of the learning process, Stud-Teach algorithm tends to be very unstable and noisy, but in general it performs better than random sampling. It is worthy to be mentioned that, except for the small noisy part of the Stud-Teach, we can see a continuation of better performance of the Stud-Teach over random sampling.

As it can be seen in Figure 6.2, at the early stages, the Stud-Teach results in an error rate that is far lower than the error rate of random sampling. This significant out-performance continues up to the middle stages where the error rate of Stud-Teach approximately converges to error rate of random sampling. After this short convergence, the Stud-Teach keeps its relative preference over the random to the final point. Therefore it can be stated that, in the Pendigit data set, the Stud-Teach is better than random sampling.

The results on the Image data set are presented in Figure 6.6. In this figure, at the first parts of the early stages the Stud-Teach algorithms does not perform better than random sampling while after that it improves and, in the middle stages, it out-performs the random sampling and it continues to performs better. We also evaluated the performance of Stud-Teach algorithm on remaining five numerical binary data sets. In these experiments, the active learner (student) and the teacher classifier used respectively the SVM with a RBF kernel and Adaboost with Decision stumps as their learning components. As mentioned before in these figures the solid line corresponds to selective sampling and the dotted line is related to random sampling.

Figure 6.3 shows the results on waveform data set. At early steps, the Stud-Teach algorithm has approximately the same error rate as the random sampling. In the middle stages, the error rate of Stud-Teach more or less converges to random sampling but after this convergence it beats random sampling. Thus it can be stated that, in this case, the out-performance of Stud-Teach over random sampling is negligible.

For the Twonorm data set, the results are shown in Figure 6.4. At early stages, the Stud-Teach algorithm performs much better than random sampling. In middle stages, it continues to be the winner but it is not as good as the early stages and it nearly converges to the random sampling at final stages.

For the results of the splice data set,(see Figure 6.4) although the Stud-Teach performs better than random sampling at the early stages, we can notice some instabilities. Stud-Teach continues to perform better up to the final rounds where it tends to converge to the random sampling up to the last point.

Finally, on the Ringnorm data set (see Figure 6.7), it can be stated that the overall performance of Stud-teach is much better than random sampling by a significant amount.

In conclusion and regarded as a whole, the Stud-Teach algorithm performs better than random sampling.

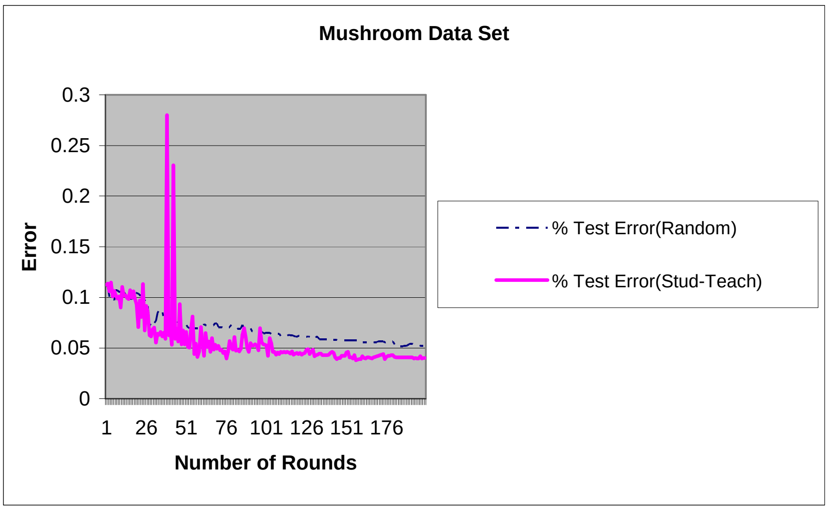


Figure 6.1: Active learning algorithms, Stud-Teach, on the Mushroom data set. The solid lines corresponds to selective sampling (Stud-Teach) and the dotted line is related to random sampling. The Student Learning algorithm is the SVM with a linear kernel and the teacher learning algorithm is Adaboost with decision stumps disjunction.

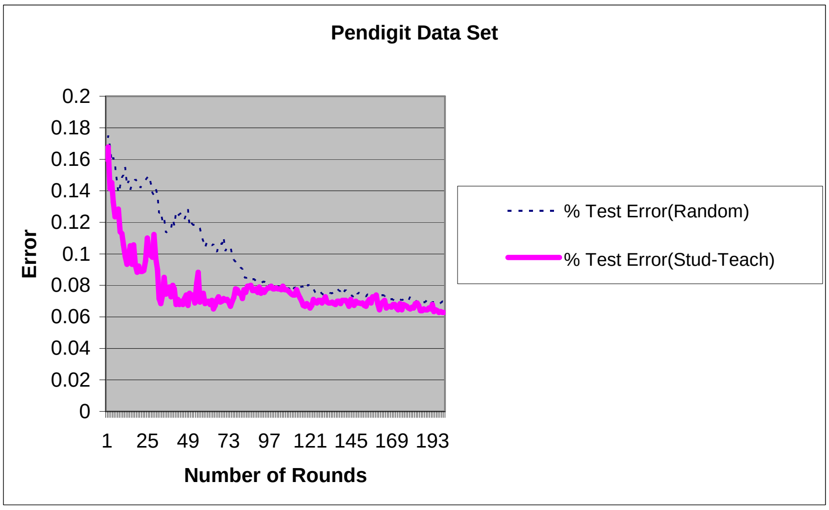


Figure 6.2: Active learning algorithms, Stud-Teach, on the Pendigit data set. The solid lines corresponds to selective sampling (Stud-Teach) and the dotted line is related to random sampling. The Student Learning algorithm is the SVM with a linear kernel and the teacher learning algorithm is Adaboost with decision stumps.

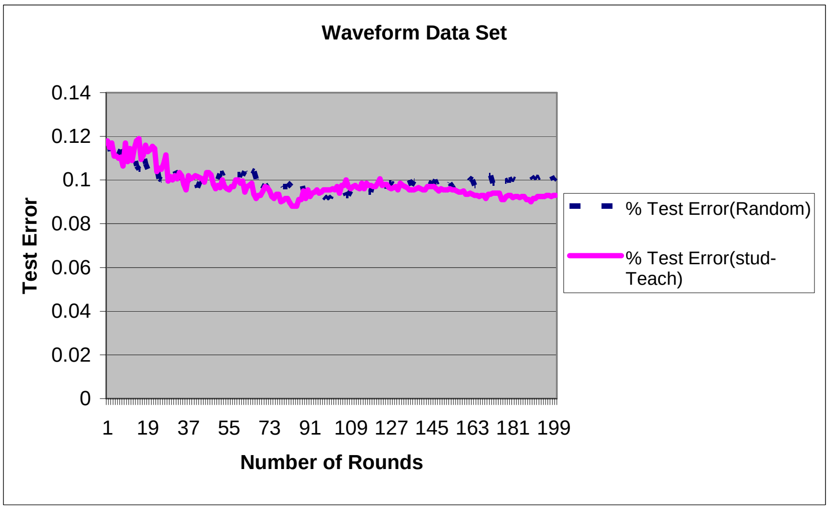


Figure 6.3: Active learning algorithms, Stud-Teach, on the waveform data sets. The solid line corresponds to selective sampling (Stud-Teach) and the dotted line is related to random sampling. The Student Learning algorithm is the SVM with a RBF kernel and the teacher learning algorithm is Adaboost with decision stumps.

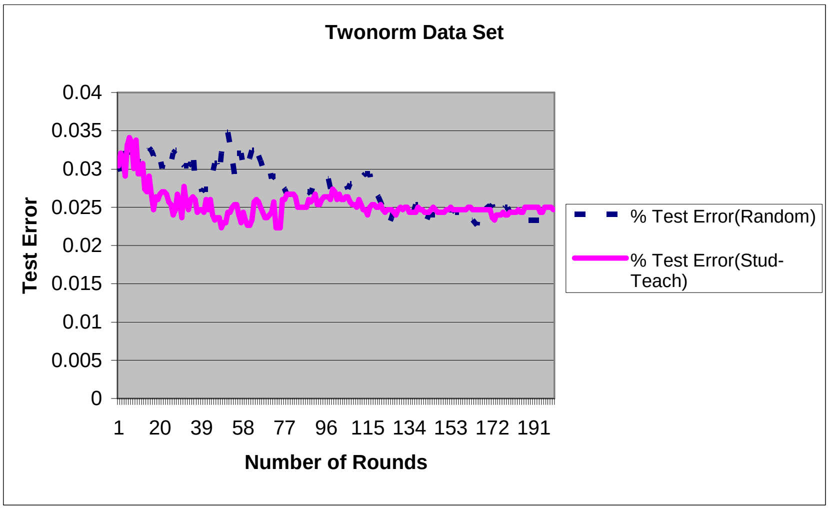


Figure 6.4: Active learning algorithms, Stud-Teach, on the twonorm data sets. The solid line corresponds to selective sampling (Stud-Teach) and the dotted line is related to random sampling. The Student Learning algorithm is the SVM with a RBF kernel and teacher learning algorithm is Adaboost with decision stumps.

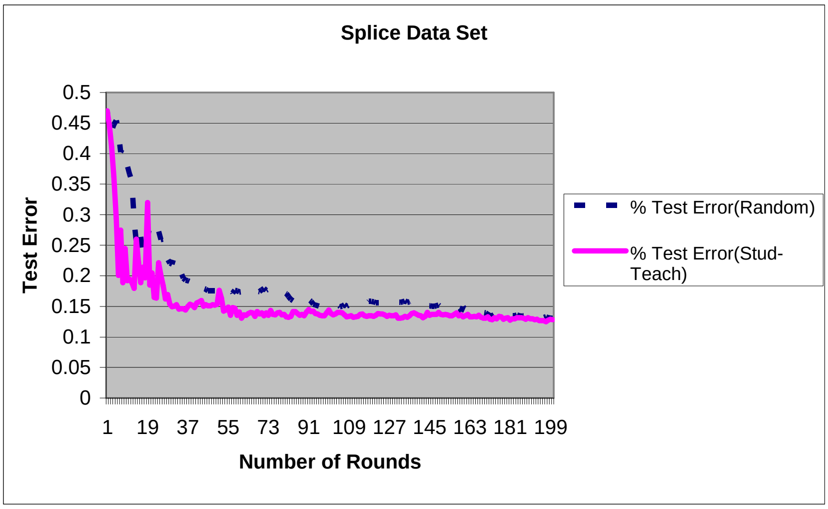


Figure 6.5: Active learning algorithms, Stud-Teach, on the Splice data sets. The solid lines corresponds to selective sampling (Stud-Teach) and the dotted line is related to random sampling. The Student Learning algorithm is the SVM with a RBF kernel and teacher learning algorithm is Adaboost with decision stumps.

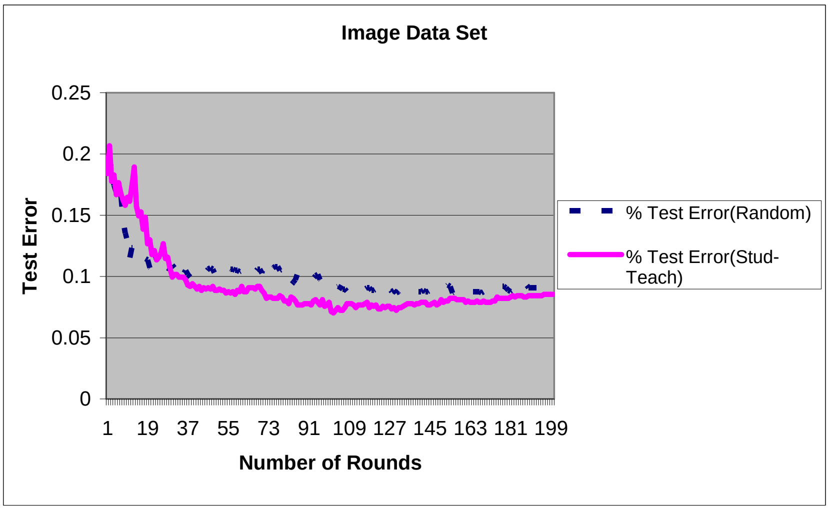


Figure 6.6: Active learning algorithms, Stud-Teach, on the Image data sets. The solid lines corresponds to selective sampling (Stud-Teach) and the dotted line is related to random sampling. The Student Learning algorithm is the SVM with a RBF kernel and teacher learning algorithm is Adaboost with decision stumps.

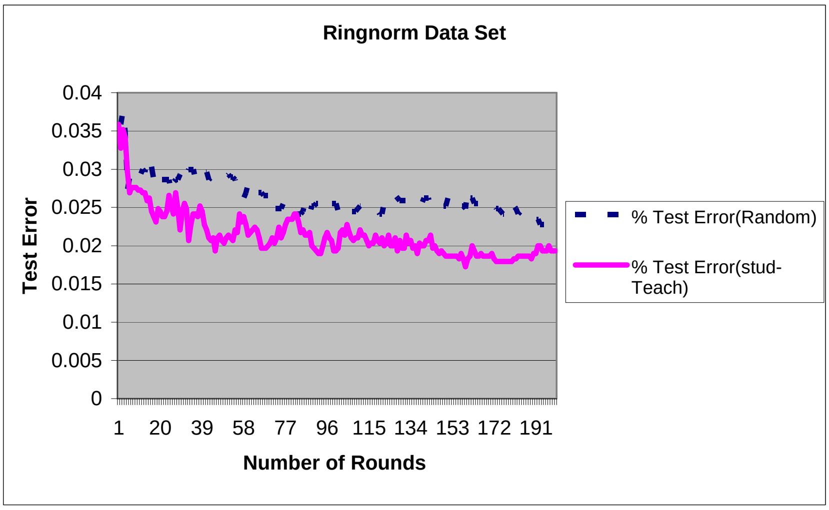


Figure 6.7: Active learning algorithms, Stud-Teach, on the Ringnorm data sets. The solid lines corresponds to selective sampling (Stud-Teach) and the dotted line is related to random sampling. The Student Learning algorithm is the SVM with a RBF kernel and teacher learning algorithm is Adaboost with decision stumps.

6.3 Comparing to Most Common Algorithms

In order to compare the performance of the Stud-Teach algorithm to other active learning algorithms, we implemented the most common ones meaning Simple Margin, and again random sampling. The random was used as the baseline for active learning algorithms performance comparison. We also intended to compare our active learning algorithm (Stud-Teach) with the query by committee algorithm. The assumption of this algorithm is that there should exist a perfect deterministic classifier and also we must have the non-empty version space. In some data sets like Mushroom a perfect deterministic classifier exists. In general, in some of our data sets there is no such classifier. Therefore, it was not possible for us to compare our algorithm with query by committee.

We also implemented another sampling approach which we call *Perfect Sampling*. In perfect sampling, the student learner queries randomly the examples which it can not classify them correctly. Then the learner adds these examples into its training set. It should be noted that in case of perfect sampling, the algorithm may be faced to *Early Stop*. The early stop happens when there is no more example with the stated condition to be added. It should also be mentioned that, in perfect sampling, the pool set is fully labeled, that is, the algorithm has access to the label on any example it needs; this situation does not stand in reality.

We implemented the SVM simple margin algorithm with the SVM with a linear kernel and the SVM with a RBF kernel. The following figures show the accuracy on test set which is obtained on each data set by applying the SVM simple margin algorithm, random sampling, perfect sampling, and Stud-Teach for the active learning algorithms. As in real life, the concept of betterness is represented by the descriptive notion of "accuracy" which itself fits into the definition given in Equation 6.2. It goes without saying that the more accurate a classifier is, the least error it has.

As we can see in Figure 6.8, in the Mushroom data set, although the difference between the accuracy of the four algorithms is not significant, the Stud-Teach algorithm is the overall winner. In spite of the fact that the perfect sampling performs better than the Stud-Teach in some few points, the Stud-Teach algorithms beats the perfect sampling over the stability and overall accuracy.

In Pendigit data set (see Figure 6.9) the random sampling and the perfect sampling are beaten by the two other algorithms. Simple Margin algorithm performs a bit better than Stud-Teach at early stages but in continuation they eventually converge together.

The Stud-Teach algorithm and Simple Margin have roughly the same accuracy.

Figure 6.10 shows the results on the waveform data set. In the first half of the diagram, we have no clear winner, each of the four algorithms may produce better result in some points. In the second half and as the number of rounds grows, it can be stated that random sampling is beaten by Simple Margin and Stud-Teach. Beginning from the middle stages, while Simple Margin results in better accuracy in comparision with the other two algorithms, the Stud-Teach converges to its rival in final stages.

For the Twonorm data set (see Figure 6.11) during the first stages Stud-Teach algorithm approximately overcomes the three other algorithms. This better performance does not last long, as in middle stages and while the perfect sampling has its early stop the Simple Margin algorithm results in better accuracy in comparison with the other two algorithms. All three algorithms tend to converge to each other while the Simple Margin keeps its surmount to the final point.

Figure 6.12 shows the results on the Splice data set. In this figure, the accuracy of Simple Margin and stud-teach are approximately the same. The first half of the diagram shows the convergence of Stud-Teach and Simple Margin. In the same half, random sampling does not have a relative good accuracy and the perfect sampling shows up competitive just in some points. Thus the Stud-Teach algorithm overcomes the random and perfect sampling in this half. In the second half of diagram, Simple Margin would be the overall winner while Stud-Teach shows competitive by beating random and converging to Simple-Margin in final points.

For the Ringnorm data set (see Figure 6.13), there is no distinct winner between Simple Margin and stud-Teach while they perform approximately better than random and perfect sampling.

Finally, for the Image data set (see Figure 6.14), although in early stages and for a short period the random, simple margin and perfect sampling result in better accuracy, in general both Stud-Teach and Simple Margin are better than random sampling. The Stud-Teach algorithm is the overall winner and significantly beats the two others.

As a conclusion, the overall results of Stud-Teach are better than random sampling and perfect sampling and competitive with simple margin. In other words, the Stud-Teach algorithm is more stable and more accurate in comparison to random and perfect sampling. The stated advantages can be due to the fact that the training set (train plus queried examples) of our proposed algorithm remains more similar to the test set in comparision to the perfect sampling.

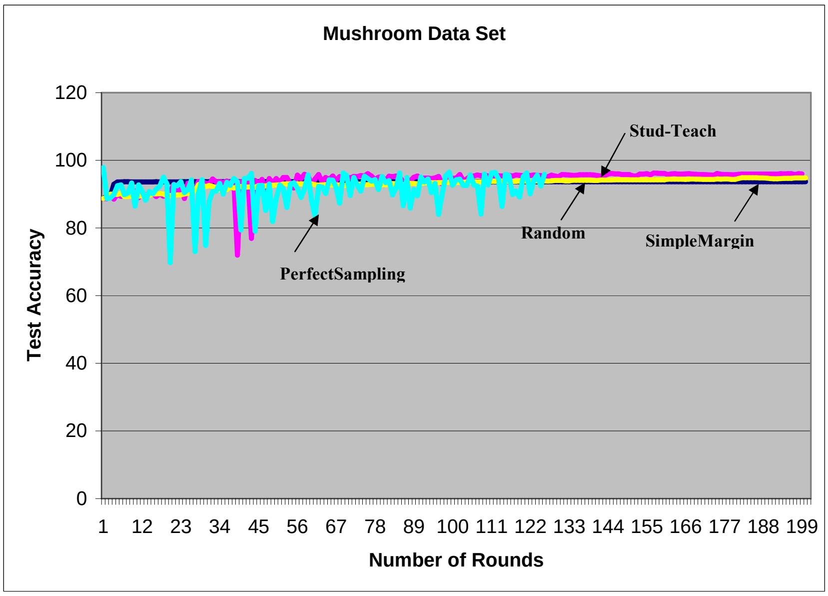


Figure 6.8: The Accuracy of three active learning algorithms (Stud-teach algorithm, random sampling and SVM simple margin) on the Mushroom data set. In Student-Teach algorithm, the student learning algorithm is the SVM with a Linear kernel and the teacher learning algorithm is Adaboost with decision stumps disjunction. In the three other learning algorithms the active learner is the SVM with a linear kernel.

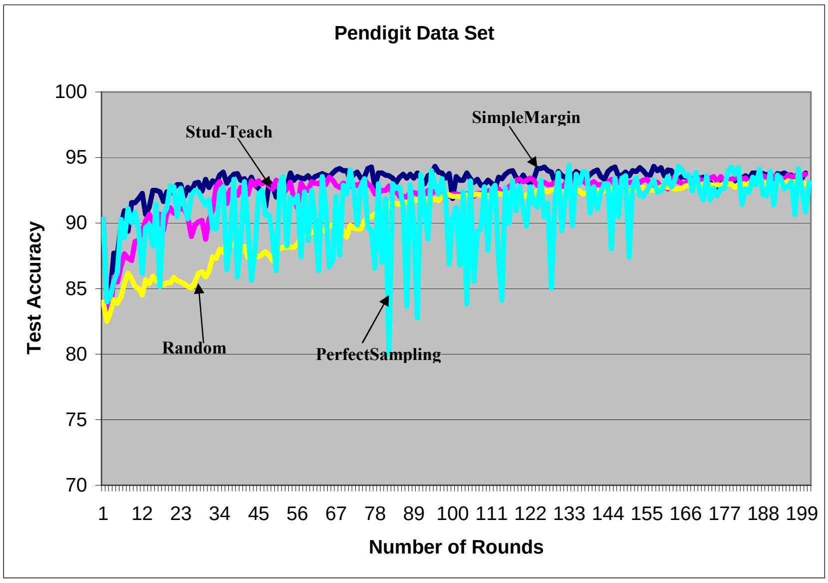


Figure 6.9: The Accuracy of three active learning algorithms (Stud-teach algorithm, random sampling and SVM simple margin) on the Pendigit data set. In Student-Teach algorithm, the student learning algorithm is the SVM with a Linear kernel and the teacher learning algorithm is Adaboost with decision stumps. In the three other learning algorithms the active learner is the SVM with a linear kernel.

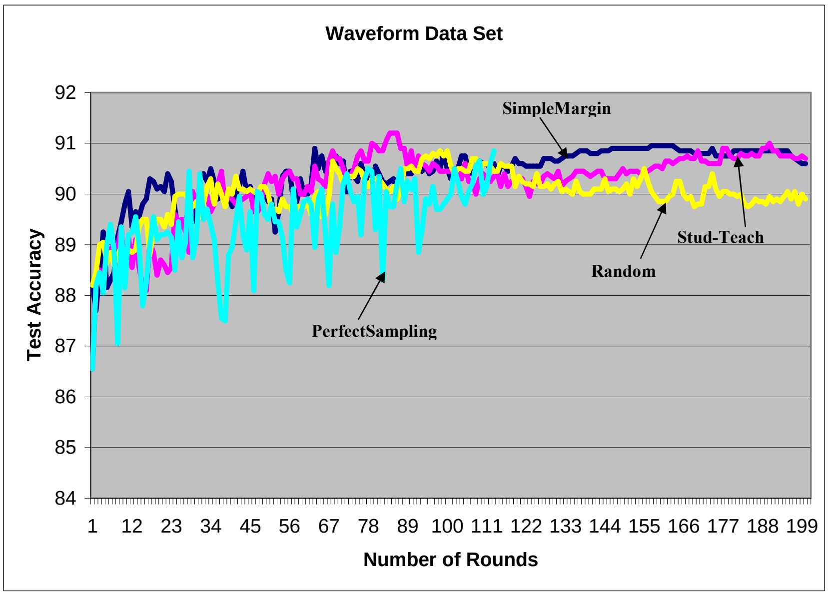


Figure 6.10: The Accuracy of three active learning algorithms (Stud-teach algorithm, random sampling and the SVM simple margin)on the waveform data set. In the Student-Teach algorithm, the student learning algorithm is the SVM with a RBF kernel and the teacher learning algorithm is Adaboost with decision stumps. In the three other learning algorithms the active learner is the SVM with a RBF linear kernel.

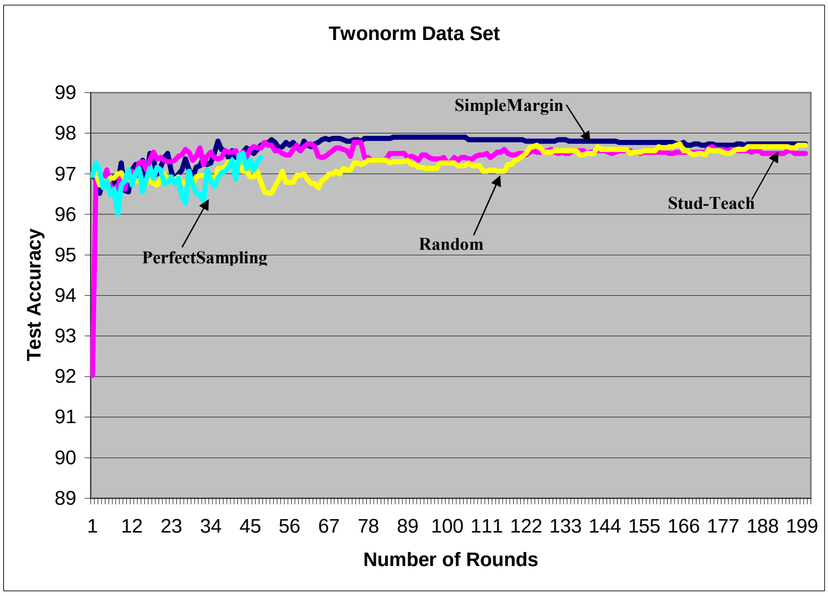


Figure 6.11: The Accuracy of three active learning algorithms (Stud-teach algorithm, random sampling and the SVM simple margin algorithm) on the twonorm data set. In the Student-Teach algorithm, the student learning algorithm is the SVM with a RBF kernel and the teacher learning algorithm is Adaboost with decision stumps. In the three other learning algorithms the active learner is the SVM with a RBF linear kernel.

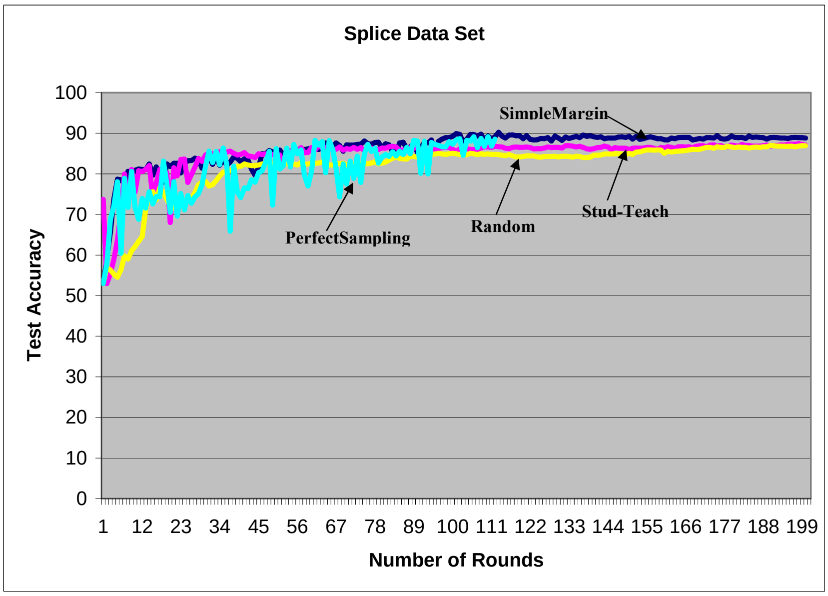


Figure 6.12: The Accuracy of three active learning algorithms (Stud-teach algorithm, random sampling and the SVM simple margin algorithm) on the Splice data set. In the Student-Teach algorithm, the student learning algorithm is the SVM with a RBF kernel and the teacher learning algorithm is Adaboost with decision stumps. In the three other learning algorithms the active learner is the SVM with a RBF linear kernel.

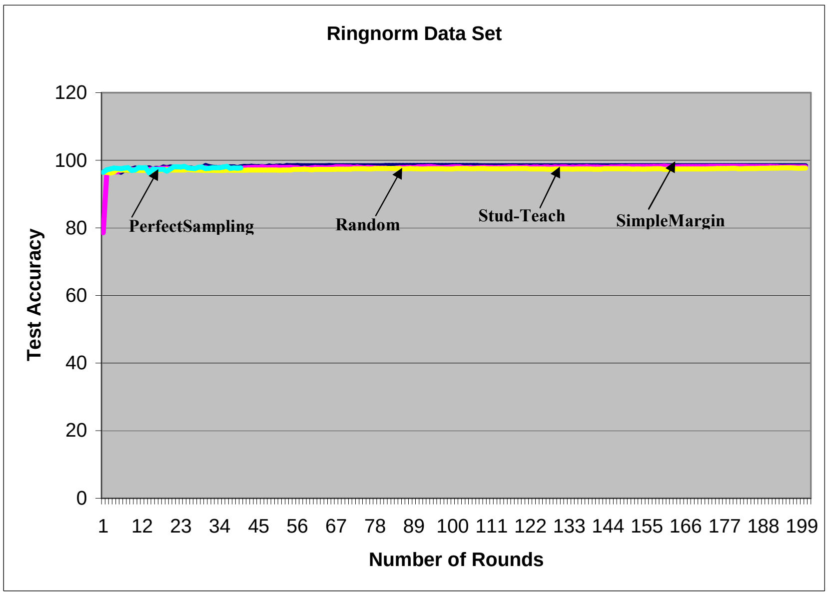


Figure 6.13: The Accuracy of three active learning algorithms (Stud-teach algorithm, random sampling and the SVM simple margin algorithm) on the Ringnorm data set. In the Student-Teach algorithm, the student learning algorithm is the SVM with a RBF kernel and the teacher learning algorithm is Adaboost with decision stumps. In the three other learning algorithms the active learner is SVM with a RBF linear kernel.

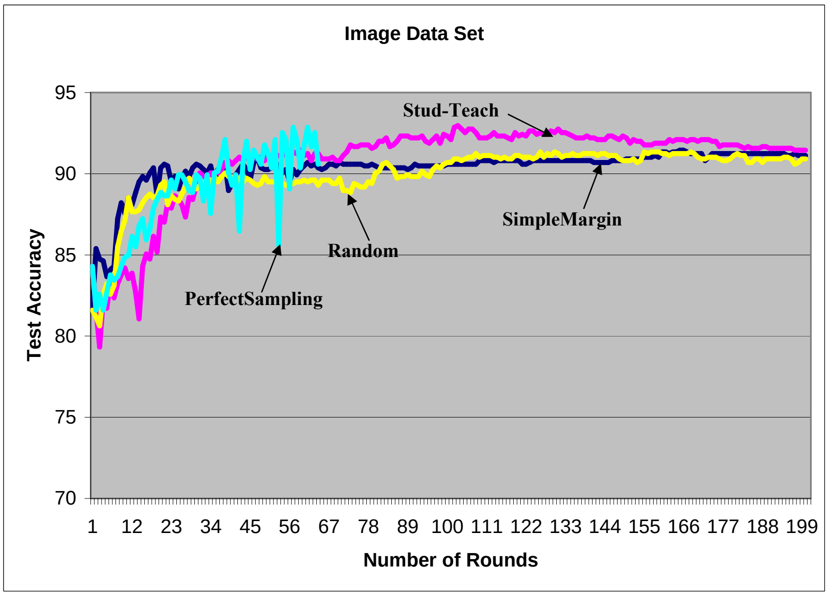


Figure 6.14: The Accuracy of three active learning algorithms (Stud-teach algorithm, random sampling and the SVM simple margin algorithm) on the Image data set. In the Student-Teach algorithm, the student learning algorithm is the SVM with a RBF kernel and the teacher learning algorithm is Adaboost with decision stumps. In the three other learning algorithms the active learner is the SVM with a RBF linear kernel.

6.4 Using the SCM as an active Learner

In previous sections, the student classifier of Stud-Teach was based on the SVM algorithm. In this section we study our algorithm's behavior when the student uses the SCM classification algorithm as its learning component.

Figures 6.15 and 6.16 show the results of this experiment respectively on Pendigit and Mushroom Data Set. As we can see in these figures the Stud-Teach is the winner. In compare with the SVM active learner (see figure 6.1), the test error rate is stable. Also it can be stated that the overall error rate of the active learner using SCM is lower than the case where it uses SVM. It should be mentioned that, the error rate of each of the five data sets when using SCM active learner was higher than the corresponding error rate while using SVM active learner. Thus, we do not present the results here.

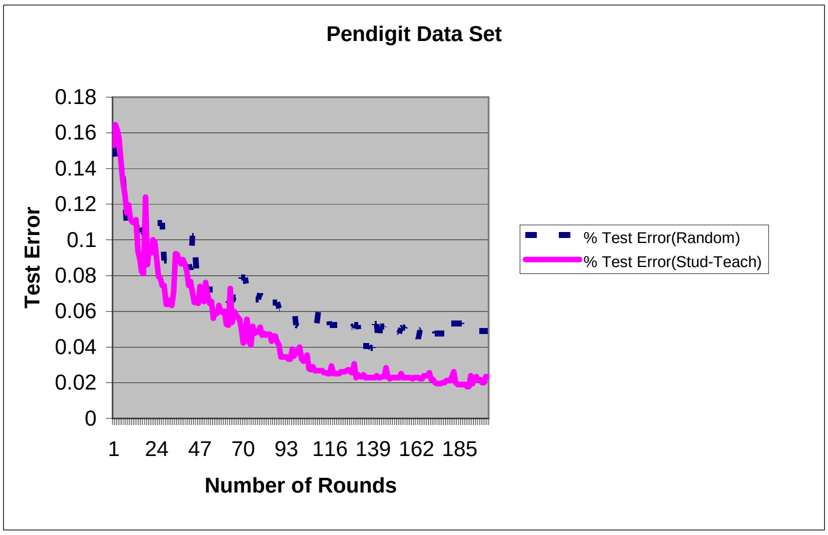


Figure 6.15: Active learning algorithms, Stud-Teach, on the Pendigit data sets. The active learner Stud used the SCM Conjunction as its learning component. The solid lines corresponds to selective sampling (Stud-Teach) while the dotted lines are related to random sampling. The student Learning algorithm is the SCM and the teacher learning algorithm is Adaboost with decision stumps.

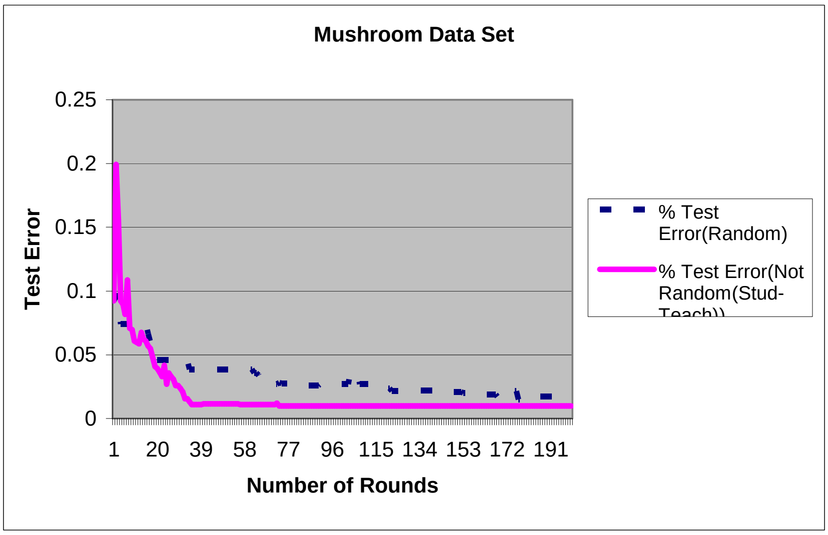


Figure 6.16: Active learning algorithms, Stud-Teach, on the Mushroom data sets. The active learner Stud used the SCM Conjunction as its learning component. The solid lines corresponds to selective sampling (Stud-Teach) while the dotted lines are related to random sampling. The student Learning algorithm is the SCM and the teacher learning algorithm is Adaboost with decision stumps disjunction.

6.5 Soft max Query selection Strategy

Although we replaced the SCM algorithm with the SVM in Stud-Teach algorithm in order to achieve better results, here we intend to optimize the results of Stud-Teach using SVM for the most unstable data set: The Mushroom data set. Therefore we tested the soft max query selection strategy on this data set. Figure 6.17 shows the result of soft max query selection strategy on this data set in comparison with the previously proposed version of the Stud-Teach algorithm.

It should be mentioned that, in the comparisons of this section, we used the test set classification error metric. The better active learning algorithm, the lower its test set classification error.

The result clearly shows that using the soft max for the Mushroom data set leads to much more stable result. It should be mentioned that the same experiment was held for the remaining 6 data sets. Since there was not a significant difference between the results of using Soft max Query selection Strategy and the Stud-Teach algorithm, we do not present them here.

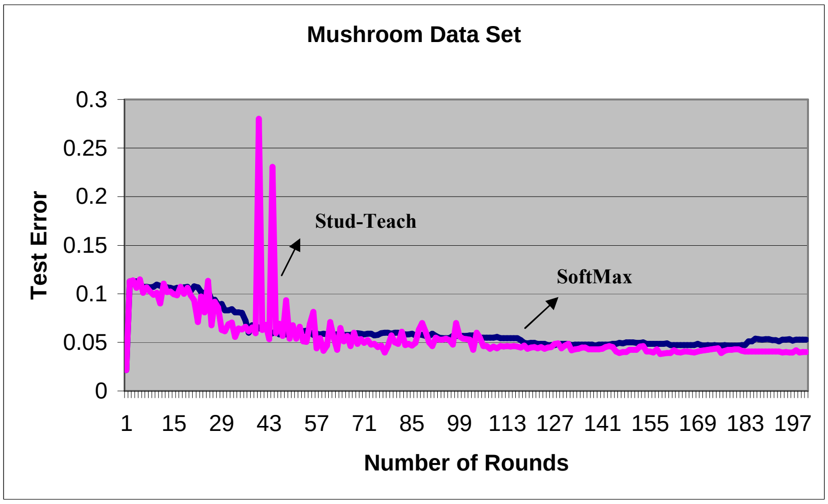


Figure 6.17: Soft Max ($\tau = 20$) query selection strategy on the Mushroom Data Set. The student Learning algorithm is the SVM with a linear kernel and the teacher learning algorithm is Adaboost with decision stumps disjunction.

6.6 Probabilistic Active Learning

To improve the rate of learning, we performed probabilistic active learning on the waveform Data set. We selected this data set since there was a little difference between random sampling and Stud-Teach algorithms. Thus, we wanted to see whether using this approach leads to better results or not.

It should be mentioned that, in the comparisons of this sections we used the test set classification error metrics. The better active learning algorithm, the lower its test set classification error.

The Figure 6.18 shows the result. A visual inspection of this figure shows that the result has been improved in comparision to Figure 6.3.

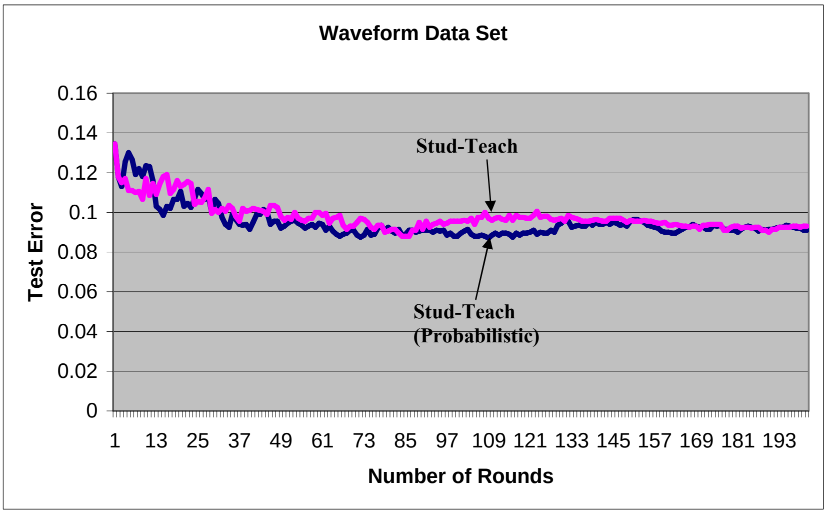


Figure 6.18: probabilistic Stud-Teach on Waveform data set. The solid lines corresponds to probabilistic (Stud-Teach) while the dotted lines are related to random sampling. Student Learning algorithm is SVM RBF kernel and teacher learning algorithm is Adaboost with decision stumps.

Chapter 7

Related work

There are several studies of active learning in the setting of supervised learning. The active learning algorithms in this settings have been developed for classification, regression, and function optimization. The two main active learning algorithms for classification are *Query by committee* and *uncertainty sampling*.

In Query by committee setting, the algorithm uses a probability distribution over hypotheses to randomly select a set of hypothesis as a committee for classifying new examples. If their predictions differ, the algorithm asks for the true label of the examples and adds them to the training set [18, 19]. This general algorithm has been applied by [15] in which the entire version space has been taken as committee members. Committee-based sampling [27] is an adaptation of this idea to probabilistic classifiers. Several ways of combining the predictions of a committee of Winnow-based learners has been compared on a text categorization task by [28]. Query by committee results appear to be very promising, but there are many assumptions: it assumes the data is noise free, a perfect deterministic classifier exists, and it is possible to draw a classifier randomly form the version space. These assumptions are not realistic in practice. Since in our data set there was not a perfect deterministic classifier we could not compare the proposed active learning algorithm with query by committee.

In uncertainty sampling, the instance that the current classifier is most uncertain about is labeled and added to the training set [26, 16, 17]. This approach has been used by [21] in the framework of support vector machine (Margin-Based Method). In Margin-Based Method the uncertainty has been measured by the distance from decision boundaries. In recent work, instead of querying for the label of the predictions that the learner is most uncertain about, the learner might autonomously label the predictions it is most certain about and add them to the training set. It is proposed that to use a committee of learners for co-training [29] or use techniques based on the Expectation Maximization (EM) algorithm [30].

One of the significant problem of active learning problems is the huge retraining cost which is caused by querying only one examples in each iteration of active learning algorithms. To come up this problem, querying a batch of most informative unlabeled examples has been proposed by [31].

In the domain of regression, active learning has been studied by [32]. They use squared error loss of classifiers as the measure of quality and approximate this loss function by choosing queries that reduce the statistical variance of a learner. Also, has been shown that choosing queries that minimize the statistical bias can be an effective approximation to the squared error loss criteria in regression[33]. The effects of different information-based loss functions for active learning in regression setting, including the use of KL-divergence has also been shown by [34].

Active learning has also been used for function optimization. The goal in function optimization is to find a region in the input space for which an unknown function takes on high values [35, 36].

There are less published work on active learning in unsupervised learning settings [37, 38]. There are also some work on active learning in other major area of machine learning such as reinforcement learning [35, 36] and decision theory [35, 36].

Chapter 8

Conclusion and future works

Although each active learning strategy claims to be the best possible, in a realistic view each individual active learning algorithm has a set of weak and strong points. Thus, new active learning strategies should aim to fade the weak points while performing better or at least competitive to the older ones. Traditional active learning strategies have some weaknesses, the major ones being:

- Some active learning algorithms work well just on noise-free data sets.
- In some active learning algorithms, there should exist a prefect deterministic classifier, and also it should be possible to draw classifier randomly from the version space.
- Some of the active learning algorithms assume that the current hypothesis lies in the middle of the version space which can not be always like that.
- There are some active learning algorithms which are computationally expensive.
- Some of the active learning algorithms are instable in their accuracy, meaning that, empirically we observe drastic changes on their test set accuracy from one query round to another.
- There is no regress guarantee for some active learning algorithms.

In this thesis, we proposed a new approach which is called Stud-Teach algorithm. We introduced a disagreement function notion which is a probability of disagreement between two classifiers (student and teacher) on the label of an example. Our proposed active learning algorithm query examples according to this function. The Stud-Teach algorithm tries to overcome the stated weak point while remaining competitive (and in some point better) than the previous ones. In this way, it accepts the data sets without any precondition of being noise free. The Stud-Teach algorithm also does not depend on the existence of a prefect deterministic classifier or on the place of the current hypothesis, that is, there is no necessity for the current hypothesis to lie in the middle of the version space. On the other hand, our proposed algorithm is accurate and easy to implement, while it is not computationally expensive.

The Stud-Teach algorithm not only fades the stated weak points, but also performs competitively among the widely known active learning algorithms. More clearly, the empirical results show that our active learning strategy is competitive to one of the well-known active learning strategies, the simple margin and performs better than the random sampling strategy.

Since the instability of an algorithm degrades the quality of its performance and reduces its trustworthy, the stability of an algorithm can be considered as one of the most important quality factors. Thus, the performance of an algorithm can be represented by its accuracy and its stability. Our proposed algorithm happens to be stable in almost all of our experiments. This has been a surprise, and should be considered as one of the main advantages of our algorithm in compare to the other state-of-the-art algorithms. Indeed, it would have been logical that our algorithm underperforms the perfect sampling one, since the latter "magically" always chooses queries on which the student learner makes mistake. It was not the case and it seems to be a consequence of the fact that our algorithm has a more stable behavior than the "magical" perfect sampling one. Moreover, we empirically showed that we can obtain even more stable results without degrading the accuracy by using soft max query selection strategy.

Beyond all the advantages that has been stated, the Stud-Teach algorithm tries to reduce the future error of the active learner. We obtained an upper bound on the risk of our active learner. One can claim that, our proposed active learning is accurate, simple and easy to implement and it can be used as an alternative for the situations where other active learning algorithms such as query by committee and simple margin can not be used.

The future work would be to obtain a tighter bound on the risk of the student active learner. It woulds also be interesting to study systematically the dynamic of our proposed active learning algorithm to determine how it behaves under different circumstances. Especially, it would be interesting to have some theoretical explanation of the stability of our algorithm that we empirically observed. The stated study might help determine the stop point of the proposed active learning algorithm.

Bibliography

[1] Herbrich R. *Learning Kernel Classifiers*, MIT Press, Cambridge, Massachusetts,subsection A.5.2, 2002.

[2] Vapnik, V. *stimation of dependences based on empirical data.*, Springer Verlag, 1982.

[3] V. Vapnik, S. Golowich, and A. Smola. *Support vector method for function approximation, regression estimation, and signal processing.*, In M. Mozer, M. Jordan, and T. Petsche, editors, Advances in Neural Information Processing Systems 9, pages 281Ű 287, Cambridge, MA, 1997. MIT Press.

[4] Bernhard Scholkopf and Alexandre J.Smola *Learning with kernels*, support vectore machine, Regularization, Optimization, and beyond The MIT Press,Cambridge, Massachusetts, London, England, 2001.

[5] Burges, C. J. *A tutorial on support vector machines for pattern recognition.*, Data Mining and Knowledge Discovery, 2, 121Ű167,1998.

[6] Vapnik, V. *The nature of statistical learning theory.*, Springer, New York, 1995.

[7] Marchand M. and J. Shawe-Taylor *The set covering machine.*, Journal of Machine Learning Research, 3:723-746, 2002.

[8] Valiant L. G. *A theory of the learnable*, Communications of the Association of Computing Machinery, 27:1134-1142, 1984.

[9] Haussler D. *Learning conjunctive concepts in structural domains*, Machine Learning 4:7-40,1989.

[10] Garey M. R. and D. S. Johnson *Computers and intractability, a guide to the theory of np-completeness*, New York, NY: Freeman, 1979.

[11] Chvatal V. *A greedy heuristic for the set covering problem*, Mathematics of Operations Research, 4:233-235, 1979.

[12] Yoav Freund and Robert E. Schapire *A decision-theoretic generalization of on-line learning and an application to boosting.* Journal of Computer and System Sciences, 55(1):119Ű139, August 1997.

[13] Stuart J. Russell and Peter Norvig *Artificial Intelligence: A Modern Approach*, Second edition Prentice Hall, 2003.

[14] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods.*, Cambridge University Press, Cambridge, U.K., 2000.

[15] David A. Cohn, Les Atlas, and Richard E. Ladner. *Improving generalization with active learning.*, Machine Learning, 15(2):201-221, 1994.

[16] David D. Lewis and Jason Catlett. *Heterogeneous uncertainty sampling for supervised learning.*, In Proceedings of the 11th International Conference on Machine Learning (ML-94), pages 148-156, New Brunswick, NJ, 1994. Morgan Kaufmann.

[17] David D. Lewis and William Gale. *Training text classifiers by uncertainty sampling.*, In Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-94), pages 3-12, 1994.

[18] H. Sebastian Seung, Manfred Opper, and Haim Sompolinsky. *Query by committee.*, In Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory (COLT-92), pages 287-294, 1992.

[19] Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. *Selective sampling using the query by committee algorithm.*, Machine Learning, 28:133-168, 1997.

[20] Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. *Information, prediction, and query by committee.* , Advances in Neural Information Processing System 5, pges 483-490, 1993.

[21] Simon Tong and Daphne Koller. *Support vector machine active learning with applications to text classification.*, J. Mach. Learn. Res., 2:45-66, 2002.

[22] Mitchell T. *Generalization as Search.*, Artificial Intelligence 28, 203-226, 1982.

[23] Matti Kaariainen. *Generalization error bounds using unlabeled data.*, In Proceedings of the 18th Annual Conference on Learning Theory, pages 127-142, 2005.

[24] John Langford. *Tutorial on practical prediction theory for classification.*, Journal of machine learning research, 6:273-306, 2005.

[25] Matthias Seeger. *PAC-Bayesian generalization bounds for guassian process.*, Journal of machine learning research, 3:233-269, 2002.

[26] Lewis, David D,Gale, W. A. *A sequential algorithm for training text classifiers.*, Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval (pp. 3–12). Dublin, IE: Springer Verlag, 1994.

[27] Ido Dagan and Sean P.Engelson. *Committee-based sampling for training probabilistic classifiers.*, In A. Prieditis and S. Russell, editors, Proceedings of the 12th International Conference on Machine Learning (ML-95), pages 150-157. Morgan Kaufmann, 1995.

[28] Ray Liere Prasad Tadepalli. *Active Learning with committee for text categorization.*, Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97), pages 591-597, Providence, RI, 1997. AAAI Press.

[29] Avrim L.Blum and Tom Mitchell. *Combining labeled and unlabeled data with co-training.*, In Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT-98), Madison, WI, 1998. ACM Inc.

[30] Andrew Mccallum,Sebastian Thrun and Tom Mitchell. *Learning to classify from labeled and unlabeled document.*, In Proceedings of the 15th National Conference Artificial Intelligence (AAAI-98), Madison, WI, 1998. AAAI Press.

[31] S. C. H. Hoi, R. Jin, J. Zhu and M.R. Lyu. *Batch Mode Active Learning and Its application to Medical Image Classification.*, In Proceedings The 23rd International Conference on Machine Learning (ICML2006), Pittsburgh, Penn, US, June 25-29, 2006.

[32] Cohn, D., Ghahramani, Z., and Jordan, M. *Active learning with statistical models.*, Journal of Artificial Intelligence Research, 1996.

[33] Cohn, D. *Minimizing statistical bias with queries.*, Advances in Neural Information Processing Systems, 1997.

[34] MacKay, D. *Information-based objective functions for active data selection.*, Neural Computation, 4, 590-604, 1992.

[35] Box, G. E. P., and Draper, N. R. *Empirical model-building and response surfaces.*, Wiley, 1987.

[36] Moore, A. W., Schneider, J. G., Boyan, A., and Lee, M. S. *Q2: Memory-based active learning for optimizing noisy continuous functions.*, Proceedings of the Fifteenth Conference on Machine Learning. Morgan Kaufmann, 1998.

[37] Bryant, C. H., Muggleton, S. H., Page, C. D., and Sternberg, M. J. E. *Combining active learning with inductive logic programming to close the loop in machine learning.*, Proceedings of AISB'99 Symposium on AI and Scientific Creativity (pp. 59-64), 1999.

[38] Atkinson, A. C., and Bailey, R. A. *One hundred years of the design of experiments on and off the pages of Biometrika.*, In press, 2001.

[39] Kaebling, L. P., Littman, M. L., and Moore, A. *Reinforcement learning.*, a survey. Journal of AI Research, 4, 237-285, 1996.

[40] Kearns, M., and Singh, S. *Near-optimal reinforcement learning in polynomial time.*, Proceedings of the Fifteenth International Conference on Machine Learning(pp. 260-268). Morgan Kaufmann, San Francisco, CA, 1998.

[41] Kearns, M., and Koller, D. *Efficient reinforcement learning in factored MDPs.*, Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (pp. 740-747), 1999.

[42] Heckerman, D., Breese, J., and Rommelse, K. *Troubleshooting Under Uncertainty.*, (Technical Report MSR-TR-94-07). Microsoft Research, 1994.

[43] R. S. Sutton and A. G. Barto. *Reinforcement learning. an introduction.* Cambridge, MA: The MIT Press, 1998.

[44] Dominic Mazzoni, Kiri L. Wagstaff, and Michael C. Burl *Active Learning with Irrelevant Examples* Pasadena, CA : Jet Propulsion Laboratory, National Aeronautics and Space Administration, 2006.

Appendix A

Data Set Descriptions

To compare and contrast the performance of our proposed active learning algorithm, we did experiments on a variety of data sets. We wanted to show that our new method can work with different kinds of data set and is not constrained to a particular one. These data sets are from UCI repository.

The following tables show some comprehensive details about used data sets.

Description	Number
Number of Attributes	16
Number of Class	2
Number of Training Set	100
Number of Test Set	1800
Number of Pool Set	8000

Table A.1: Pendigit Data Set

Description	Number
Number of Attributes	22
Number of Class	2
Number of Training Set	100
Number of Test Set	1821
Number of Pool Set	8000

Table A.2: Mushroom Data Set

Description	Number
Number of Attributes	20
Number of Class	2
Number of Training Set	100
Number of Test Set	2900
Number of Pool Set	4399

Table A.3: Ringnorm Data Set

Description	Number
Number of Attributes	60
Number of Class	2
Number of Training Set	100
Number of Test Set	1270
Number of Pool Set	1805

Table A.4: Splice Data Set

Description	Number
Number of Attributes	20
Number of Class	2
Number of Training Set	100
Number of Test Set	2960
Number of Pool Set	4340

Table A.5: Twonorm Data Set

Description	Number
Number of Attributes	21
Number of Class	2
Number of Training Set	100
Number of Test Set	2000
Number of Pool Set	2900

Table A.6: Waveform Data Set

Description	Number
Number of Attributes	18
Number of Class	2
Number of Training Set	100
Number of Test Set	924
Number of Pool Set	1286

Table A.7: Image Data Set

Appendix B

Algorithms parameters

The following tables show the parameters of the algorithms which have been applied on data sets.

Data Set	Student Algorithm	Teacher Algorithm	Number of Rounds
Mushroom	SVM Linear Kernel/SCM	Adaboost	200
Pendigit	SVM Linear Kernel/SCM	Adaboost	200
Ringnorm	SVM RBf Kernel	Adaboost	200
Splice	SVM RBf Kernel	Adaboost	200
Twonorm	SVM RBf Kernel	Adaboost	200
Waveform	SVM RBf Kernel	Adaboost	200
Image	SVM RBf Kernel	Adaboost	200

Table B.1: Stud-Teach parameters

Table B.2 shows the parameters used with Adaboost algorithm. We employed Decision Stump Disjunction or Decision Stump as the weak learners.

Table B.3 shows the parameters used for SVM algorithm. In the case of using SVM RBF kernel the σ^2 is the variance for each data set.

The SCM algorithm used the L_2 metric and, as its Learning Method, it used MSformBound (which means that it uses bounds for early stopping and selection of the best parameters). As a testing Method the SCM used Plain (evaluation on Testing Set File). Table B.4 shows the parameters summary of SCM which is used in our setting.

Data Set	Weak Learner	Number of Rounds
Mushroom	Decision Stump Disjunction	23
Pendigit	Decision Stump	161
Ringnorm	Decision Stump	161
Splice	Decision Stump	161
Twonorm	Decision Stump	161
Waveform	Decision Stump	161
Image	Decision Stump	161

Table B.2: Adaboost parameters

Data Set	SVM	Comments
Mushroom	SVM Linear Kernel	
Pendigit	Linear Kernel	
Ringnorm	RBF Kernel	$\sigma^2 = 0.0004$
Splice	RBF Kernel	$\sigma^2 = 0.0004$
Twonorm	RBF Kernel	$\sigma^2 = 0.0004$
Waveform	RBF Kernel	$\sigma^2 = 0.0004$
Image	RBF Kernel	$\sigma^2 = 0.0004$

Table B.3: SVM parameters.

Description	Comment
Learning Method	MSfromBound
TestingMethod	Plain
Metric	L_2

Table B.4: SCM parameters