# SAMPLING THE TOP-K REPRESENTATIVE DATA FOR CLASSIFICATION

by

Ping Wang

B.Sc., Tianjin University, China, 1995

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

In the School
of
Computing Science



SIMON FRASER UNIVERSITY

Spring 2005

# APPROVAL

**Name:** **Ping Wang**

**Degree:** **Master of Science**

**Title of Thesis:** **Sampling The Top-k Representative Data For Classification**

**Examining Committee:**

**Chair:** **Dr. Binay Bhattacharya**

---

**Dr. Martin Ester**
Senior Supervisor

---

**Dr. Ke Wang**
Supervisor

---

**Dr. Jian Pei**
Examiner
Assistant Professor

**Date Defended/Approved:** _____

# SIMON FRASER UNIVERSITY

# PARTIAL COPYRIGHT LICENCE

# ABSTRACT

Building classification models based on databases is an exciting area in data mining research. In many classification tasks, only a small set of labelled training data are given. These data are not sufficient for a good classification. We need to sample and label more data as training data for better performance. However, labelling data is time-consuming and costly. The challenge is to effectively select the most representative data for labelling.

While most active learning methods for this problem follow the incremental query learning paradigm in which the classifier is retained upon each newly labelled query, we present a distance-based method which samples the top-k representative data simultaneously and can be applied to any distance-based classifiers. Redundancy reduction makes classifier retraining unnecessary and makes it find more balanced examples with regard to class distribution in database. Experiment results from two data sets and two classifiers demonstrate the advantages of our method.

# DEDICATION

*To Andy*

# ACKNOWLEDGEMENTS

I would like to thank my senior supervisor, Dr. Martin Ester. He has provided me with ideas, guidance and encouragement as a supervisor. He has also given me strength and support in my most difficult time as a friend. I feel exceedingly appreciated to have had his guidance and I owe him a great many heartfelt thanks.

I would also like to thank my supervisor, Dr. Ke Wang. His suggestions and input are very helpful. Thank Dr. Jian Pei for being my examiner of this thesis.

My appreciation also goes to Kersti, Val for the administration procedure.

My deepest gratitude and appreciation is reserved for my parents, my husband and other family members. To My beloved husband Hongyin Cui, thank you for your love and unlimited support both on the study and on the daily life. You are my biggest treasure. To my parents, thank you for your endless love, your help, your support and your patience. Thank you for taking care of Andy in the period of my studying. To my parents in law and brother in law, thank you for giving us both spiritual and financial support. You make us feel that we are not alone here.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER 1

# INTRODUCTION

We are deluged by data - scientific data, medical data, demographic data, financial data, and marketing data. As our capabilities of both generating and collecting data are rapidly increased, the volume and complexity of data are unprecedented. These data are rich with hidden information that can be used for making intelligent decisions. Classification can be used to extract models describing important data classes or to predict categorical labels and future data trends. For example, a classification model can be built to categorize protein sequences as either Outer Membrane Protein or non Outer Membrane Protein.

To construct a precise classification model, a set of high quality training data are the prerequisite. Usually classification tasks are given a small set of labelled training data at the very beginning. This data is not enough to construct an acceptable classification model for a database. More data need to be sampled, labelled and added into training data set to achieve a precise classification. Thus, effectively and efficiently sampling a number of most representative data is an important and interesting research problem.

## 1.1 Background Information and Motivation

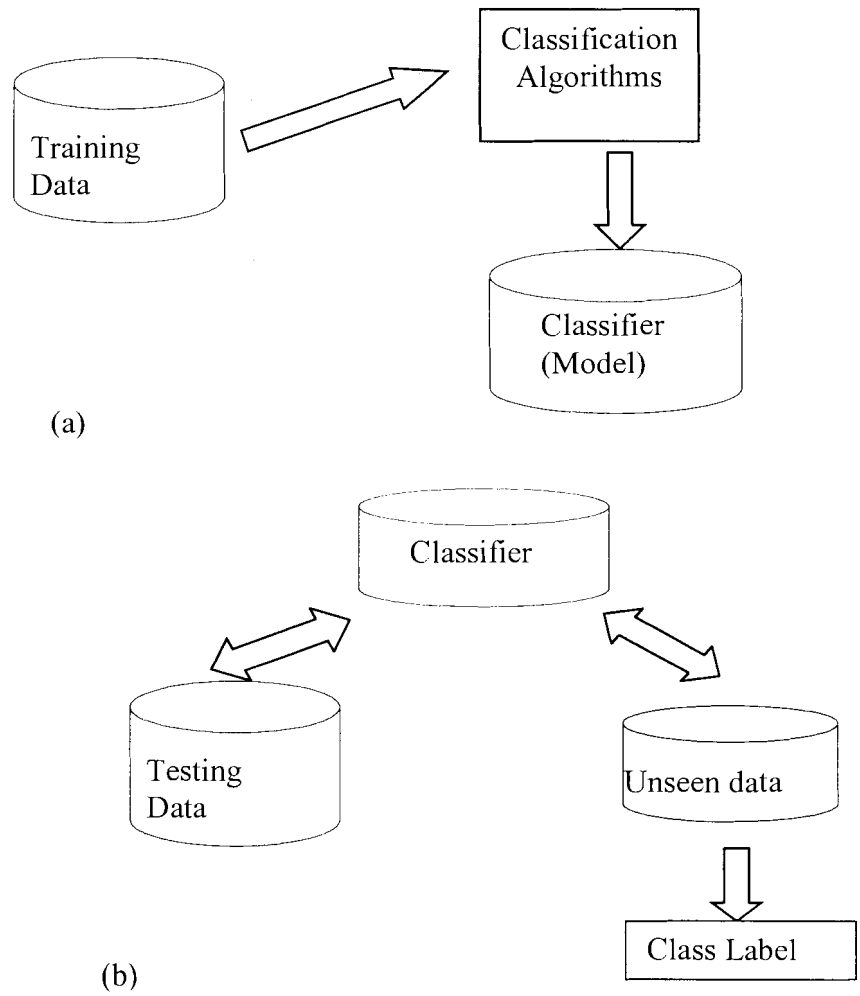
Data classification [1] is a two-step process (Figure 1.1). In the first step, a model is built describing a predetermined set of data classes or concepts. The model is constructed by analyzing database tuples described by attributes. Each tuple is assumed to belong to a predefined class, as determined by one of the attributes, called the *class label attribute*. In the context of classification, data tuples are also referred as *samples, examples*, or *objects*. The data tuples that are analyzed to build the model collectively form the *training data set*. The individual tuples making up the training data set are referred to as *training samples* and are usually randomly selected from the sample population (data bases). If the class label of each training sample is provided, this step is also known as ***supervised learning*** (i.e. the learning of the model is "supervised" in that it is told to which class each training sample belongs). It contrasts with ***unsupervised learning***, in which the class label of each training sample is not known.

In many supervised learning tasks, the given training samples are not sufficient for learning the classification model. Sometimes these training samples are biased so that the classification performance is poor. In order to achieve a good classification model, more data need to be sampled and labelled.

However, labelling data may be complex, time-consuming and costly since it requires substantial domain expertise. For instance, to label a DNA or protein data needs a complex sequence analysis and may cost an experienced biologist several days. At the same time, the explosive growth of data makes it impossible to label all the data. For

example, the number of entries in the data bank of protein sequences SWISSPROT [2] has grown by over 30 times since its first establishment in 1986.

**Figure 1.1 The data classification process: (a) Learning: Training data are analysed by a classification algorithm and the classifier model is built. (b) Classification: Test data are used to estimate the performance of the classifier and the classifier can be applied to classify unseen data.**




One way to save the cost of labelling data is to select a limited number (k) of data for labelling. At the same time, we hope these data can improve the classification performance most. It will be beneficial to sample the top-k representative data from the

large database based on the initial set of training samples. In this way, k is specified by users and the classifier aims to reach high performance using the same number of examples selected and labelled.

The traditional approach to this problem is to randomly sample data from the database according to the data probability distribution. However, random sampling will not always be effective. For example, if only 1 in 1000 data is positive class, and only 500 data can be labelled, then a random sample will usually contain 500 negative class examples and no positive class ones. This will not help a classifier to distinguish positive from negative examples. In real world, the positive class and negative class are often not balanced with negative class as the majority. Random sampling has another shortcoming that the average amount of novel information obtained per sample will decrease as more data are sampled. The reason is that with growing size of the training data set, the classifier's knowledge about large regions of the input space becomes more and more confident so that additional samples from these regions are basically redundant so far as they do not contribute too much to improve the classification performance.

Some active learning methods try to solve this problem. In an active learning setting, a learner has access to a pool of unlabeled data and trains a classifier based on current observed labelled data. Then based on the current state of the classifier(s) one selects some of the most informative or representative data so that knowing labels of the selected data can greatly improve the classification accuracy of the classifier(s). In order to select the most representative data, typical active learning methods employ the idea of "uncertainty sampling", in which the unlabeled data whose class labels

are most uncertain based on the current classifier(s) are presented to experts for labelling.

Query by Committee [3] [4] is one of the earliest algorithms of active learning. It uses a prior distribution over hypotheses. The method samples a set of classifiers from this distribution and selects an example with the maximal disagreement among the committee of these classifiers. For a linear classifier, e.g. a linear support vector machine (SVM) [11] [16], the most uncertain data is the one closest to the classification hyper-plane. Tong & Koller's SVM active learning [5] proposed a similar idea of uncertainty sampling using SVM and applies it to text classification. But both of them are restricted to the incremental query learning paradigm in which the classifier is retrained upon each newly labelled query. This paradigm is quite impractical in real world, since human experts can only label one datum at a time and have to wait for next one until the classifier is retrained. This leads to an extremely long time for training and improving a classifier.

*Xu* [7] introduced a method which is able to select more than one example at a time. This algorithm takes the distribution of unlabeled data into account when selecting the unlabeled data close to the current SVM hyperplane. However, it has no an explicit quality criterion combining these two together and can only be applied to Support Vector Machine (SVM) classifier.

Two previous studies [8] [9] on uncertainty sampling proposed a quite general algorithm, which can be used with any type of classifier that both predicts a class and provides an uncertainty measurement. So it can select multiple examples at a time, but it may introduce redundancy in these examples. It means some chosen examples are very

similar. This redundancy will decrease the effectiveness and waste domain expert's time in labelling similar data.

Hasenjager and Ritter [10] proposed an algorithm with local models. It is based on the idea of selecting examples on the current decision boundary. It constructs Voronoi diagram [10] and divide the input space into regions based on labelled data. The intersecting points of region borderlines are chosen for labelling. However, the application of the approach is limited to low dimensions in the range of 1 to 8, because the complexity of Voronoi diagram calculation grows exponentially with the dimensionality.

To address the challenges and limits in above methods, this thesis proposes a method to sample top-k representative data.

## 1.2 Contribution

In this thesis, we study the problem of effectively and efficiently selecting top-k representative data from databases, which will be labelled and added into training data set to improve the classification performance. In particular, we make the following contributions.

(1). We introduce a novel distance-based method including a concept to define the representativeness of data object. In this method, we try to avoid choosing very similar data. Thus the selected data contains more information and improves the classifier quality as much as possible.

(2). No classifiers and retraining of classifiers are needed during the selecting period. The top-k representative data can be selected at a time. So the

domain experts or users can analyze and label these data in a batch mode leading to a much faster training and improvement of the classifier.

(3). This method and concept is not restricted to one particular classifier, but can be used in any distance-based classifier such as nearest neighbour classifier and Support Vector Machine (SVM).

The experiments demonstrate the above advantages.

## 1.3 Thesis Organization

The remainder of the thesis is organized as follows:

- In Chapter 2, we present an overview of related work systematically.
- In Chapter3, we first define the problem and terminologies, and then present our algorithm and optimizations, at last summarize the whole method.
- In Chapter 4, we explain our data set, evaluation methodologies, and experiment design.
- In Chapter 5, we show the experiment results and analyze the results in details.
- The thesis concludes in Chapter 6. Some future works are presented.

# CHAPTER 2

# RELATED WORK

The topic of representative sampling for classification has received significant attention in both data mining and machine learning communities. A lot of researches have been conducted. In this Chapter, we review several previous studies that are related to our work technically.

## 2.1 Query by Committee

Query by Committee was first introduced by Seung, Opper and Sompolinsky in [3]. We refer it as the **QBC** algorithm.

**QBC** devise a paradigm of incremental query learning, in which the training data set is built up one example at a time. An incremental learning procedure consists of two components: a training algorithm and a query algorithm. Given a set of P labelled examples, the *training* algorithm builds a model satisfying the training set. The *query* algorithm is then used to select example P+1, whose expected information gain is high. Then the training algorithm is run again on the newly incremented training set, and so on.

We denote by X an arbitrary unlabelled sample space over which a distribution $D$ is defined. The target concept is a mapping $c$: $X \rightarrow \{+1, -1\}$. We assume that the

learning procedure has access to two functions: **Sample** and **Label**. A call to **Sample** returns an unlabelled example $x \in X$, randomly chosen based on the (unknown) distribution D. A call to **Label** with input x, returns the label of x according to the target concept $c(x)$. If we simply make P calls to the two oracles, a training data set of pairs $\xi_t = (x_t, c(x_t))$ are constructed, $t = 1,\ldots, P$. The training set determines the *version space*

$$H_P = \{h: h(x_t) = c(x_t), t = 1, \ldots, P\},$$

which is the set of all hypotheses consistent with the training set. The learner is assumed to know the distribution Pr(h) over the version space [12] (the subset of the hypothesis space that correctly classifies the labelled examples). The *training* algorithm is the Gibbs algorithm [14], in which the hypothesis *h* is drawn at random from this distribution. This will enable us to use techniques from statistical mechanics [15]. After training 2k models on the same training data set, a committee is constructed with 2k members, where k is a user defined parameter. Then ideally the query algorithm selects a data object that is classified as positive by half of the committee, and negative by the other half. Practically it chooses the one maximizing disagreement among the committee.

The version space is a representation of the information contained in the set of labelled examples observed by the learner. A natural measure of the progress of the learning process is the rate at which the size of the version space decreases. Any input $x_{P+1}$ divides the version space $H_p$ into two parts,

$$H^+ = \{h \in H_p: h(x_{P+1}) = +1\},$$

$$H^- = \{ h \in H_p: h(x_{p+1}) = -1\}.$$

The expected information gain is given by

$$I(x_{P+1}) = -\frac{V^+}{V_p}\log\frac{V^+}{V_p} - \frac{V^-}{V_p}\log\frac{V^-}{V_p},$$

where $V^{\pm}$ are the volumes of $H^{\pm}$ and $V_P$ is the volume of the version space $H_P$. After **Label** answers the query, the class label of $x_{P+1}$ is known with certainty. Before the answer arrives, the label of $x_{P+1}$ is uncertain: according to the Bayesian, it is +1 with the probability $\frac{V^+}{V_p}$, and -1 with the probability $\frac{V^-}{V_p}$. The entropy of this distribution is precisely the information value of the query, and is the expression of the above formula. The expected information gain is maximized by $x_{P+1}$ such that $V^+ = V^-$, i.e. by the query that divides the version space in half. In this optimal case, I = 1 bit exactly.

Unfortunately, for most learning models, the geometry of the version space is complex, and one can not practically calculate the volumes $V^{\pm}$ for any given example, much less find an example for which $V^+ = V^-$. Training algorithms typically yield individual hypotheses in the version space, not global information about the version space. However, a committee of hypotheses can be used to obtain global information. Train a committee of 2k hypotheses using the Gibbs algorithm. Find an example that is classified as a positive example by k members of the committee, and classified as negative by the other k. Query the **Label** about this example. Train the committee again using the enlarged training data set, and repeat. As $k \to \infty$ this algorithm approaches the bisection algorithm.

The Fraud and Seung [4] proved that the **QBC** algorithm is an efficient algorithm for the perceptron concept class with distributions that are close to uniform. It also shows that the information gain approaches a finite value and the prediction error decreases

exponentially fast with the number of queries for some natural learning problems. This is in marked contrast to the case of learning with random inputs, in which the information gain approaches zero as the number of examples increases. In the random input case, the prediction error decreases relatively slowly.

The limits or shortages of QBC algorithm are as follows:

- It can only select one example at a time. This is inefficient, if **Label** function is time consuming or costly so that query in a batch will be more efficient.
- It assumes that the data is noise free, and a perfect deterministic classifier exists. Both of the assumptions are problematic in real world.

## 2.2 An Algorithm for Uncertainty Sampling

An algorithm [8] for sequential sampling during machine learning of statistical classifiers was developed by Lewis and Gale. This method is called uncertainty sampling, which is an iterative process of manual labelling of examples, classifier built from those examples, and use of the classifier to select new examples whose class membership is unclear.

Figure 2.1 presents an algorithm for uncertainty sampling from a set of unlabelled examples using a single classifier. In each iteration it can select one or b examples for labelling. This algorithm can be used with any type of classifier that both predicts a class and provides a measurement of how certain that prediction is. For instance, probabilistic, nearest neighbour, and neural classifiers, along with many others, satisfy this criterion or can be easily modified to do so.

**Figure 2.1 An algorithm for uncertainty sampling with a single classifier**

---

1. Create an initial classifier
2. While teacher is willing to label examples
   (a) Apply the current classifier to each unlabeled example
   (b) Find the b examples for which the classifier is least certain of class membership
   (c) Have the teacher label the b examples selected in (b)
   (d) Train a new classifier on all labelled examples

---

Uncertainty sampling [8] employs a similar strategy of training on misclassified instances which was first introduced in [17] [18]. The difference is in that when data is not labelled we must use the classifier itself to guess which examples are high likely to be misclassified.

However, this uncertainty sampling algorithm may select some very similar examples or examples whose corresponding patterns[1] are similar and even same. This will greatly decrease the effectiveness.

## 2.3 Support Vector Machine Active Learning

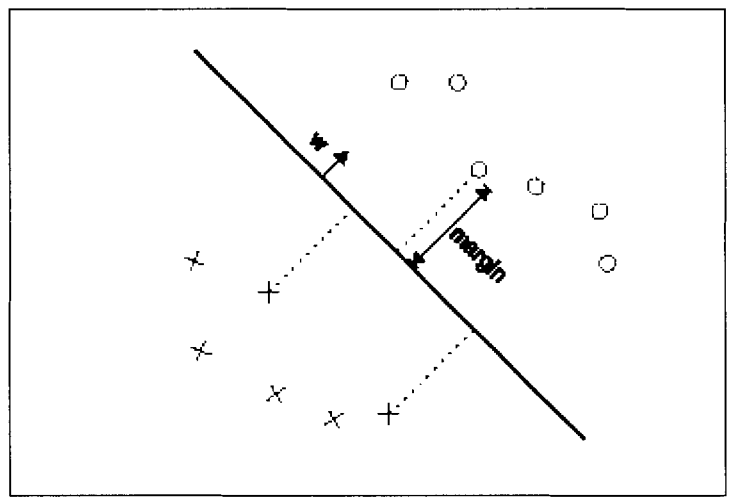
Support vector machines [11] have met with significant success in numerous real-world learning tasks. However, they are generally applied using a randomly selected training set labelled in advance. Instead of using a randomly selected training set, a pool-based active learning algorithm with support vector machine was introduced by Simon Tong and Daphne Koller [5].

[1] We are careful to distinguish an example e from the corresponding pattern w, since different examples may have the same feature vector.

## 2.3.1 Support Vector Machine (SVM)

Support vector machines have strong theoretical foundations and excellent successes. They have been applied to tasks such as handwriting recognition, object recognition, and text classification. In this subsection, we briefly review SVMs in the binary classification settings.

**Figure 2.2 A simple linear support vector machine**




We are given a training set $\{x_1, \ldots, x_n\}$, where training data $x_i$ is a vector in some space $X \subseteq R^d$. We are also given their labels $\{y_1, \ldots, y_n\}$ where $y_i \in \{-1, 1\}$. Basically SVMs are finding hyperplanes that separate the training data by a maximal margin (see Figure 2.2). All instances lying on one side of the hyperplane are labelled as -1, and all instances lying on the other side are labelled as 1. The training instances that are closest to the hyperplane are called *support vectors*. However, sometimes the training data can not be linearly separated by a hyperplane. To address this problem, SVMs allow one to project the original training data in space X to a higher dimensional feature space F via a Mercer kernel function K. In other words, we consider the set of classifiers of the form:

$$f(x) = (\sum_{i=1}^{n} \alpha_i K(x_i, x)) \ .$$

When K satisfies Mercer's condition [16], we can write: $K(u, v) = \phi(u) \bullet \phi(v)$ where $\phi$: $X \rightarrow F$ and "$\bullet$" denotes an inner product. We can then rewrite f as:

$$f(x) = w \cdot \phi(x), \text{ where } w = \sum_{i=1}^{n} \alpha_i \phi(x_i) \text{, and } w \in W$$

Thus, by using kernel function the training data are transferred into a different feature space F. In addition, the parameter space W is simply equal to F.

Given labelled training data and a kernel function K, there exists a set of hyperplane that separate the data in the induced feature space F. We call this set of consistent hypotheses the *version space*. In other words, hypotheses f is in version space if for every training data $x_i$ with label $y_i$ we have that $f(x_i) > 0$ if $y_i = 1$ and $f(x_i) < 0$ if $y_i = -1$. Then the version space, V is defined as:

$$V = \{w \in W | \ ||w|| = 1, \ y_i(w \bullet \phi(x_i)) > 0, \ i = 1, \ldots, n\}.$$

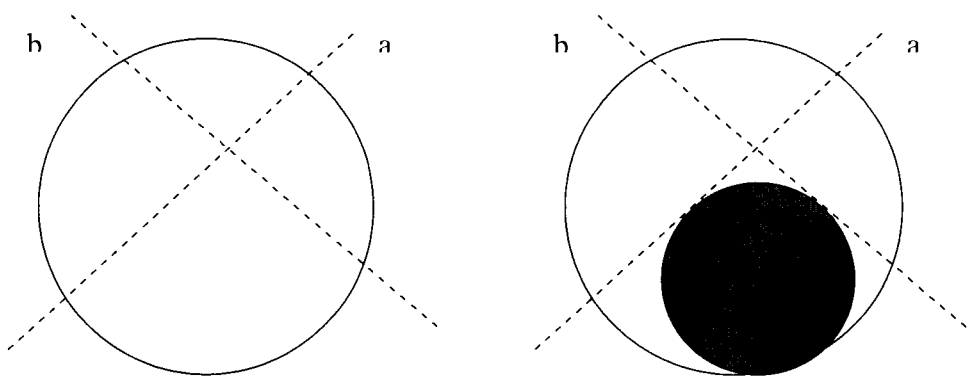
Because of the duality between the feature space F and parameter space W, we have: points in F correspond to hyperplanes in W and *vice versa*. Then we can show that the set of consistent hypotheses are the points (w) in W, which are restricted to lie on one side of a hyperplane in W. Notice that the version space is a connected region on the surface of a hypersphere in parameter space W. Since SVMs find the hyperplane that maximize the margin in the feature space F, it can be achieved by:

$$\text{maximize}_{w \in F} \ \ \min_i \{ \ y_i(w \bullet \phi(x_i)) \ \}$$

$$\text{subject to: } ||w|| = 1 \text{ and } \ y_i(w \bullet \phi(x_i)) > 0, \ i = 1, \ldots, n$$

It is shown in Figure 2.3.

**Figure 2.3** **(a) version space duality. The surface of the hypersphere represents unit weight vectors. Each of the two hyperplanes corresponds to a labelled 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) A SVM classifier in a version space. The dark embedded sphere is the largest radius sphere 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, its radius is proportional to margin of the SVM in F, and the training points corresponding to the hyperplanes that it touches are the support vectors.**




(a) (b)

## 2.3.2 Active Learning with SVM

Before we can proceed to introduce the algorithms, we need two definitions:

- **Area(V)** is the surface that the version space V occupies on the hypersphere ||w|| = 1.
- Let $V_i$ denote the version space after i queries have been made. Now, given the $(i+1)^{th}$ query $x_{i+1}$, define:

$$V_i^- = V_i \cap \{w \in W \mid -(w \cdot \phi(x_{i+1})) > 0\},$$
$$V_i^+ = V_i \cap \{w \in W \mid +(w \cdot \phi(x_{i+1})) > 0\}.$$

So $V_i^-$ and $V_i^+$ denote the resulting version spaces when the next query $x_{i+1}$ is labelled as -1 and 1 respectively.

Basically it tries to reduce the version space as fast as possible. One good way is to choose a query that halves the version space. However it is not practical to explicitly compute the sizes of the new version space $V^-$ and $V^+$. There are three ways of approximating this procedure.

- **Simple Margin.** Recall from previous subsection, given some data $\{x_1,\ldots,x_n\}$ and their labels $\{y_1,\ldots,y_n\}$, the SVM unit vector $w_i$ based on this training set is the center of the largest hypersphere that can fit inside the current version space $V_i$. The center of the sphere is often approximately in the center of the version space. Now, we can test each of unlabeled instances x to see how close their corresponding hyperplanes in W come to the centrally placed $w_i$. This is simply the distance between the feature vector $\phi(x)$ and the hyperplane $w_i$ in F — which is easily computed by $| w_i \cdot \phi(x) |$. The smaller the distance is, the more it bisects the version space. It results in the natural rule: learn an SVM on the existing labelled data and choose as the next instance to query the instance that is closest to the hyperplane in F.

- **MaxMin Margin.** Since we want an equal split of the version space, we wish *Area(V⁻)* and *Area(V⁺)* to be similar. Now consider min(*Area(V⁻), Area(V⁺)*). It will be small if *Area(V⁻)and Area(V⁺)* are very different. Since the radius m of the hypersphere can be used to indicate the size of the version space. Thus

we will take $\min(m^-, m^+)$ as an approximation and we will choose x for which this value is largest. Hence the algorithm is as follows: for each unlabeled instance x compute the margins $m^-$ and $m^+$ of the SVMs obtained when it is labelled as -1 and +1 respectively; then choose to query the unlabelled instance for which the quantity $\min(m^-, m^+)$ is greatest.

- **Ratio Margin.** This method is similar to MaxMin Margin method. Instead of using $\min(m^-, m^+)$, it take $\min(\frac{m^-}{m^+}, \frac{m^+}{m^-})$ as the criteria to choose the next instance.

The experiments show that the performance of three methods is similar. But because it can only select one example at a time, it has the same inefficient drawback as QBC algorithm. Especially for the MaxMin Margin and Ratio Margin methods, they require retrain the SVM twice for each selection.

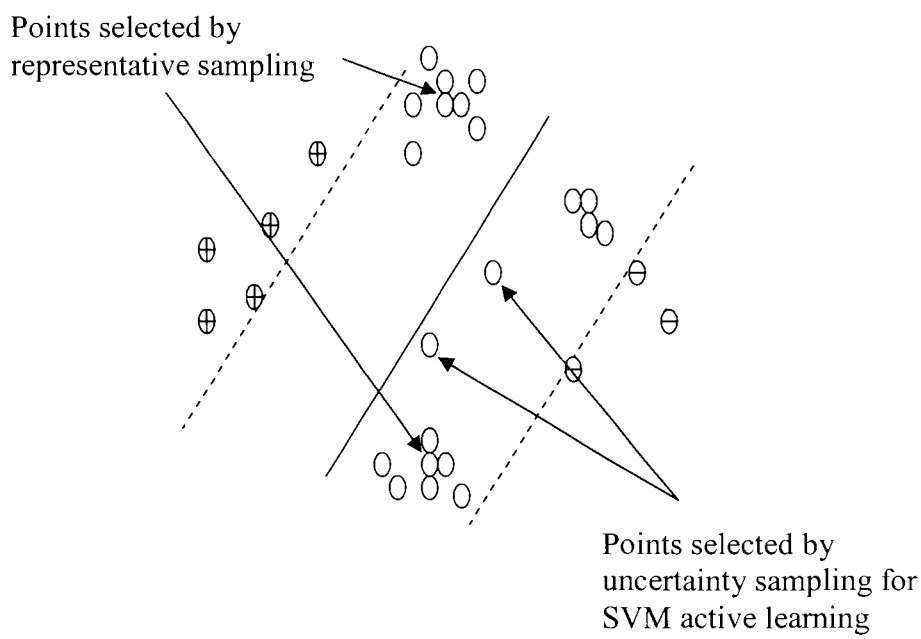
## 2.4 Representative Sampling Using SVM

To address the problem of selecting more than one unlabeled examples, Xu [7] devised a representative sampling algorithm using SVM. This algorithm takes the distribution of the input domain X into account. It follows the heuristic that the learner should select the *important informative* vectors $x_i$ whose labels are yet unknown and quite uncertain according to the current SVM. The pseudo code of this algorithm is given as follows:

**Figure 2.4 Representative sampling algorithm using SVMs**

1. Train a linear SVM model based on all the labelled instances gathered so far.
2. Let U be the set of the unlabeled instances that lie in the margin of newly trained SVM.
3. Cluster U into k groups by k-means clustering and identify the k medoid instances.
4. Present the k selected documents to human experts for labelling.
5. Return to the first step and repeat until some stopping criterion is satisfied

**Figure 2.5 Representative sampling with SVM vs. SVM active learning**




This algorithm differs from the SVM active learning algorithm (subsection 2.3) in that it analyzes the distribution of the unlabeled instances within the margin where the classification is with low confidence; in comparison, SVM active learning only simply

picks up the unlabeled data closest to the current SVM hyperplane. An example is shown in Figure 2.5.

Xu also proposed how to define the value of data for active learning. Two important issues should be considered:

- First, data points are relevant which define the class boundary best, i.e. data points are close to the separating hyperplane.
- Second, the distribution of unlabeled data.

Thus the representative sampling guides the learner to concentrate on the most *important uncertain* data instead of the *most uncertain* data.

The weakness of this algorithm is the lack of generality, since it can only be used in SVMs. In addition, it does not have an explicit quality criterion combining the above two important issues.

## 2.5 Active Learning with Local Models

Hasenjager and Ritter [10] proposed a novel active learning algorithm with local learning model. In this algorithm, vector quantization [13] is used, i.e. each labelled training instance is considered as a reference vector. Given N labelled training instances, the input space U of unlabeled instances is divided into N regions. Each of these regions is represented by a reference vector $r_i$, $1 \leq i \leq N$. The set of $R = \{r_i, i = 1, \ldots, N\}$ of reference vectors is called *codebook*. The classification follows the rule: each unlabeled data in a region has the same class membership of the reference vector for that region.

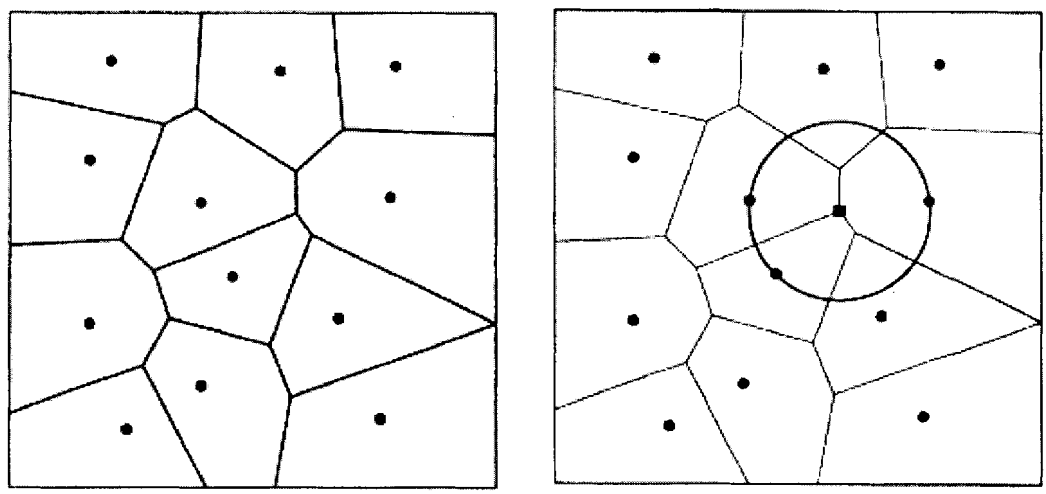
The goal of this algorithm is to construct a suitable codebook incrementally. Next reference vector or example is selected based on the knowledge that the learner has learned so far. Thus it also employs the active learning strategy.

The nearest-neighbour rule is used to divide input space U into regions. Each unlabeled instance in U is assigned to its nearest reference vector in the codebook. Then these regions are called ***Voronoi polyhedra*** defined as follows:

***Voronoi polyhedra V($r_i$)*** = $\{x \in X: d(r_i, x) \leq d(r_j, x); \forall i \neq j\}$, where d is the Euclidean distance.

The set of points that are assigned to more than one region forms borderlines which construct the ***Voronoi diagram*** of the codebook. Figure 2.6 presents an example in the two dimension space.

**Figure 2.6 Voronoi diagram of a set of 12 reference vectors**



**(a)** **(b)**

Figure 2.6 (b) shows a point of intersection of three – or, in a *d*-dimension space – (*d*+1) faces of adjacent regions. These points are called ***Voronoi vertices*** $v_i$. Each of

them have an equivalent distance from at least (d+1) reference vectors, it is the centre of a circle $C(v_i)$ passing through those reference vectors.

The author proposed to make use of these geometrical properties in the following way: the classification of the Voronoi vertices is ambiguous, since they are at the same distance from a maximum number of reference vectors. This makes them promising candidates for the next query about the correct class membership.

To select one candidate from Voronoi vertices, three ways are proposed:

A. The next example is selected randomly from the set of Voronoi vertices. In this case, it assumes that the property of being a Voronoi vertex is a sufficient condition for the selection. All Voronoi vertices are chosen with the same probability.

B. The Voronoi vertex $v_i$ which is the centre of the largest circle $C(v_i)$ is selected as the next example to query.

C. Only the subset of Voronoi vertices whose nearest neighbours among the codebook reference vectors belongs to different classes are considered. These Voronoi vertices lie on the current classification boundary as defined by the nearest neighbour rule. Among these, the one which is the centre of the largest circle is selected. Such a query utilizes not only the geometry information but also the information that has been accumulated about the associated classifications.

This algorithm can easily be generalized to high dimension space. However, the Voronoi diagram needs to be calculated explicitly and the complexity of this calculation

grows exponentially with the dimension, so it limits the application of the approach to dimensions in the range of 1 to 8. Since many learning tasks, such as text classification, have very high dimensionality, it is not a very practical method.

# CHAPTER 3

# SAMPLING TOP-K REPRESENTATIVE DATA

This chapter provides an in-depth discussion of our approach for sampling top-k representative data. In Section 3.1, a precise problem definition is given. A new ***Representativeness*** concept is defined and its design is discussed in detail. Section 3.2 presents the algorithm systematically. Section 3.3 discusses the optimization of the algorithm. Section 3.4 summarizes the whole method.

## 3.1 Definitions and Terminologies

In this subsection, we will define all terminologies and concepts that will be used, and define the problem of sampling top-k representative data. In this thesis, we will only focus on binary classification tasks. In this case, there are only two classes in the given database, positive class and negative class. We also assume that the domain experts are only able to label a limited number of data objects. Thus, it is impossible to find the optimal solution by identifying and labelling all subsets of size k, retraining the classifier and selecting the one which improves the classification performance most. Then the problem is defined as follows.

**Problem Definition** Let D be the entire database, which contains only two classes, positive class (PC) and negative class (NC). But only a small set of data objects are labelled as training data denoted by $L = \{L_1 \cup L_2\} \subseteq D$, where $L_1 \subseteq PC$ and $L_2 \subseteq NC$. The remaining n data objects are unlabelled and denoted by $U \subseteq D$. Our goal is to find the answer set $ANS \subseteq U$ based on L, where $|ANS|=k$, such that the performance of the classifier(s) can be improved as much as possible after the elements of ANS are labelled and the classifier(s) are rebuilt on the extended training data $\{L \cup ANS\}$. The k data objects in ANS are the top-k representative data for classification.

In above problem definition, the size of answer set, k is a user specified parameter. The bigger it is, the more work domain experts need. But if it is too small, then the increase of the performance of classifier(s) may not be satisfying. The performance of the classifier(s) can be evaluated by any of the standard methods [29] such as accuracy, precision, recall or F-measure. We will discuss the evaluation methodologies in Section 4.4.

Since many classification methods are distance-based, we are motivated to propose a distance-based approach. We will introduce two distance function here, one for the distance between two data objects, and the other one for the distance from one data object to a set of data objects.

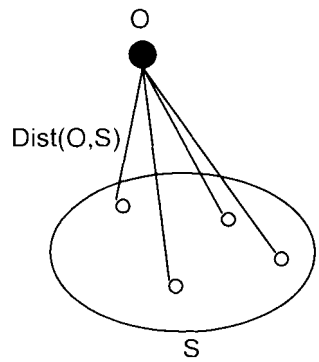
**Definition 1** Given two data objects X and $Y \in D$. Then ***F(X, Y)*** is defined as the distance between X and Y.

**Definition 2** Given a data object $O \in D$ and a set of objects $S = \{Q_1, Q_2, \ldots Q_m\}$, where $S \subseteq D$, the distance between O and S is defined as follows:

***Dist(O, S)*** = $\min\{F(O, Q_i) \mid i = 1, 2, ..., m\}$

From the above definitions, the distance between a data object and a set of data objects is defined as the distance from the object to the closest object in the set. Figure 3.1 shows the definition.

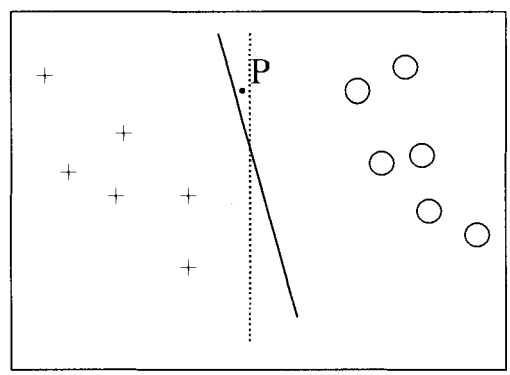
**Figure 3.1 Distance Function**




Like other active learning methods discussed in Chapter 2, we also employ the idea of "uncertainty sampling". We introduce a novel idea to measure the uncertainty of data's classification. The selected data should be those data that are most difficult or uncertain to classify in the current classifier and thus can significantly improve the classifier performance after labeling as training data. The basic idea is simple but elegant: When an unlabelled data has a similar distance to both sets of positive training data ($L_1$) and negative training data ($L_2$), it will be very likely to be misclassified. In another words, the more similar the two distances from the unlabelled object to $L_1$ and $L_2$, the more uncertain the object's classification. So we are going to find the data which can maximize the similarity of the two distances. If this data is sampled and labelled, the data close to it will be easy to classify.

Figure 3.2 shows an example of this basic idea. In this figure, an unlabelled object P has similar distances to both positive and negative training samples. The dashed line is the decision boundary of the current classifier, which misclassifies P as positive object. But P actually belongs to negative class. After labelling P and adding it into the training data set, the classifier is rebuilt and the decision boundary is changed to the solid line. Now P is correctly classified as negative data object.

**Figure 3.2 An example of the basic idea**




Hence, we suggest that the *most uncertain* unlabelled data is the data which has the most similar distances to both positive training samples $L_1$ and negative training samples $L_2$. To measure the similarity of distances, we use a concept, *DistSimilarity*, which is defined as follows.

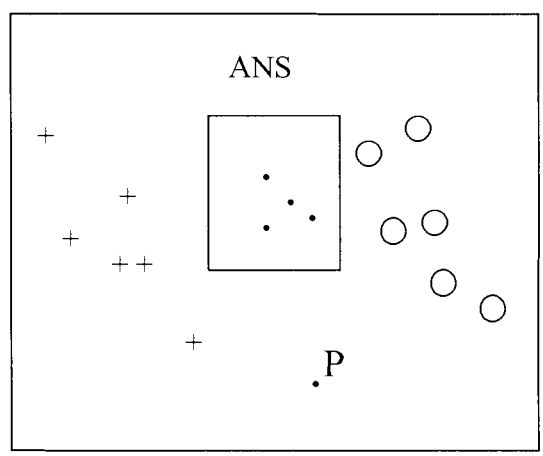
**Definition 3** Given U, $L_1$ and $L_2$, the **DistSimilarity** (with regarding to $L_1$ and $L_2$) of any unlabelled data object $O \in U$ is:

$$\textbf{DistSimilarity (O)} = \frac{\min\{Dist(O, L_1), Dist(O, L_2)\}}{\max\{Dist(O, L_1), Dist(O, L_2)\}} \qquad (1)$$

In above formula (1), the top and bottom are respectively the minimal distance and maximal distance. It is intuitively describes the representativeness of any data within the range of [0, 1]. The greater the *DistSimilarity* is, the more uncertain the classification for the data.

If we use formula (1) as the only criterion to measure the uncertainty, the answer set ANS may contain a lot of data that are very close to each other since they have similar and large *DistSimilarity (O)*. We call them as redundancy. Figure 3.3 show an example of this scenario. In this example, all the objects in the ANS rectangle will be returned as the answer set. But if P can be selected as one of the answers, it will give more help for training the classifier.

**Figure 3.3 An Redundancy Example**




In order to reduce the redundancy in the answer set, we also need consider the distance between the answer set elements. Thus we will select a data object which not only has a high *DistSimilarity* value but also is far to the current answer set. Then a novel concept, *representativeness* is defined as follows.

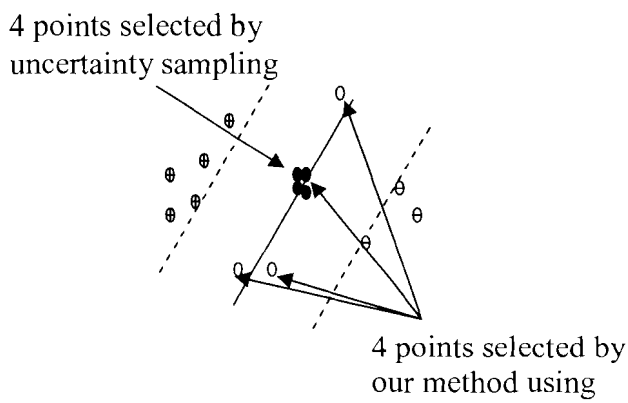
**Definition 4** Given U, $L_1$, $L_2$ and ANS, The **representativeness** (with regard to $L_1$, $L_2$ and ANS which is the current answer set) of any unlabelled object $O \in U$ is:

$$\textbf{Rep(O,ANS)} = DistSimilarity(O) * Dist(O, ANS) \qquad (2),$$

Formula (2) consists of two parts. The first part on the left of "*" represents the similarity between Dist(O,C1) and Dist(O,C2), and its range is in [0,1]. The larger it is, the more similar two distances are. The second part on the right of "*" represents the distance between the unlabeled object O to the current answer set of selected data. In order to make sure that it doesn't dominate the Rep(O,ANS), it is required in the range of [0,1]. Some distance function has this property, such as Cosine Similarity. The larger it is, the farther it is to the current answer set.

Given the definition of *representativeness*, our task is to select k data objects, for each of which the *Rep* value is the highest with regard to L1, L2, and the current ANS at the moment of choosing it. By this way, we aim to sample the top-k representative data. An example is shown in Figure 3.4.

**Figure 3.4 Top-k representative data vs. Uncertainty Samples with SVM**



## 3.2 Algorithm

Our method is to iteratively select top-k representative data in k rounds. In each round, the *representativeness* of each unlabeled data object is computed and the one with the maximal representativeness value is selected and added into answer set. It stops as soon as k data objects are selected. Since our algorithm is based on the problem definition and *representativeness* defined on Section 3.1, we call it as **Rep-Sampling** algorithm. The pseudo code of the basic version is given in Figure 3.5.

**Figure 3.5 The basic Version of Rep-Sampling Algorithm**

---

**Input:** L1, L2, k, U
**Output:** Answer set ANS
**Methods:**
1. Initialize ANS = ∅.
2. **while** (|ANS| < k) **do**
3. **For** each O∈U – ANS
   3.1 Compute Dist(O, $L_1$), and Dist(O, $L_2$) .
   3.2 Compute Dist(O, ANS).
   3.3 Compute Rep(O,ANS).
4. Find the object P whose Rep(P,ANS) is maximal.
5. ANS = ANS ∪ {P}.
6. **end while**
7. Return ANS.

---

The input of the algorithm are: positive training data set $L_1$, negative training data set $L_2$, the number of representative data that users want to select, and the unlabelled data set U. The output will be the answer set, ANS, which includes k data selected by the

algorithm. The methods start with initializing the ANS to empty in Line 1. Then it repeats Line 3 to line 5 until all k representative data are selected from U. So it runs k rounds, each of which selects one data. In each round, for each unlabelled data O in U and not in ANS, it calculates the distance from the object to $L_1$ and $L_2$ in Line 3.1, the distance from the object O to current answer set ANS in Line 3.2, and finally get the *representativeness* value Rep(O,ANS) in Line 3.3. Then it adds the object with the maximal Rep() value into answer set ANS in Line 4, 5. Thus after k rounds, it finds all the k objects and return ANS.

The algorithm is straightforward. However, if we implement it naively, the above algorithm would be very inefficient. Now let us take a look at the run time complexity of the algorithm. Let $|L_1| + |L_2| = m$, then to calculate Dist(O, $L_1$), Dist(O, $L_2$) and Dist (O, ANS) in each round, it needs $n \times m$ pair-wise distance computation according to Definition 2, where n is the size of U. Thus totally its complexity is $O(k \times n \times m)$. Now let's compare it with optimal solution. The optimal solution determines all subsets of size k of U, labels them and selects the one that increases the performance of classification most. Its run time complexity is $O(n^k)$, which is much bigger than the complexity of our greedy algorithm and makes the optimal solution infeasible.

However, the unlabelled data set may be very large, and the complexity of our approach can also be too large to be acceptable. So we need find some efficient ways to optimize the initial version of our algorithm in Figure 3.5. Next subsection will discuss some optimization methods in details.

## 3.3 Optimization

The major factor of complexity is the computations for pair-wise distances. We will try to reduce the number of these distance computations. We have two optimization methods for this aim.

The first method is to compute Dist(O, $L_1$) and Dist(O, $L_2$) only once for each unlabelled object O. Since in Rep-Sampling algorithm, it does not update $L_1$ and $L_2$, it is unnecessary to compute the pair-wise distances in every round. If we can calculate these distances only in the first round and store them in a hashing table or in hard disk, then the subsequent (k-1) rounds can use them directly.

The second method is to reduce the pair-wise distance computation when calculating Dist(O, ANS). Even though the answer set is updated after each round, but only one new member are added. According to Definition 2, we will take the minimal pair-wise distance as the distance between the object to the set. So if we can keep the Dist(O, ANS) and the new member N added into ANS after each round, then in the next round we only need compute F(O, N) and take minimal one between F(O, N) and previous Dist'(O, ANS) as the new value of Dist(O, ANS).

With the above two optimizations, each round only computes one pair-wise distance, F(O, P), where P is the new member added to ANS in previous round. Thus the complexity is now O((k+m)×n), which is reduced.

The pseudo code of Rep-Sampling algorithm with the optimizations is given in Figure 3.6. This optimized algorithm employs three data structures:

- Storage_1 to keep Dist(O, $L_1$) and Dist(O, $L_2$),

- Storage_2 to keep previous Dist(O, ANS),
- New_member to keep the most representative data selected in previous round.

**Figure 3.6 The Optimized Version of Rep-Sampling Algorithm**

---

**Input:** L1, L2, k, U

**Output:** Answer set ANS

**Methods:**

1. Initialize ANS = ∅.
2. **For** each O ∈U
   - 2.1 Compute Dist(O, $L_1$), and Dist(O, $L_2$).
   - 2.2 Storage_1(O) = < Dist(O, $L_1$), Dist(O, $L_2$) >.
   - 2.3 Storage_2(O) = 1, New_member = null.
3. **while** (|ANS| < k) **do**
4. **For** each O∈U – ANS
   - 4.1 < Dist(O, $L_1$), Dist(O, $L_2$) > = Storage_1(O).
   - 4.2 the previous Dist'(O, ANS) = Storage_2(O).
   - 4.3 Compute F(O, New_member).
   - 4.4 Dist(O, ANS) = min {F(O, New_member), Dist'(O, ANS)}.
   - 4.5 Storage_2(O) = Dist(O, ANS).
   - 4.6 Compute Rep(O,ANS).
5. Find the object P whose Rep(P,ANS) is maximal.
6. New_member = P.
7. ANS = ANS ∪ {P}.
8. **end while**
9. Return ANS.

---

# CHAPTER 4

# EXPERIMENT DESIGN AND IMPLEMENTATION

To evaluate the effectiveness of our algorithm, we need conduct extensive experiments. According to the discussion of the related work in Chapter 2, [7] is similar to our approach, but [7] is applicable only for SVM. The general algorithm [8] for uncertainty sampling is also similar to our approach, since it can also both select more than one data object to query and be applied to some distance based classifiers as long as they associate some uncertainty with their predictions. Thus, we will compare our algorithm with this *uncertainty sampling algorithm* [8] and *traditional random sampling*.

In this Chapter, we present experiment design and implementation issues.

## 4.1 Data Sets

To critically evaluate the effectiveness, two data sets from different areas are used to perform the experiments.

### 4.1.1 Protein Sequences

We obtain a dataset of protein sequences from our partners at the department of Molecular Biology and Biochemistry at Simon Fraser University. It is available online at

http://www.psort.org/dataset. The dataset was created by extracting all Gram-negative proteins from the SWISSPROT database [22].

This data set contains protein sequences of variable lengths. Protein sequences are constructed by hundreds, sometimes thousands of amino acids, over an alphabet of 20 amino acids. Each amino acid is represented by a letter: alanine (A), cysteine (C), aspartic acid (D), glutamic acid (E), phenylalanine (F), glycine (G), histidine (H), iosleucine (I), lysine (K), leucine (L), methionine (M), asparagines (N), proline (P), glutamine (Q), arginine (R), seine (S), threonine (T), valine (V), tryptophan (W) and tyrosine (Y). The longest sequence in our dataset consists of 3705 amino acid residues and the shortest sequence has a length of only 50. Two classes are present in this dataset:

- Outer Membrance Protein (OMP)
- non Outer Membrane Protein (non-OMP)

The distribution of these two classes is not balanced, with 27% being "OMP" and 73% being "non-OMP". The details are shown in Table 4.1.

**Table 4.1 Protein Sequences Dataset**

| Data | Number of Sequences | Percentage of Each Class | Minimum Length | Maximum Length | Average Length |
|---|---|---|---|---|---|
| OMP | 427 | 27.4% | 91 | 3705 | 571.1 |
| Non-OMP | 1132 | 72.6% | 50 | 1034 | 256.8 |
| **Total** | **1559** | | | | **342.9** |

Since the biologist is interested in OMP prediction, we take OMP as positive class and non-OMP as negative class in our experiments.

### 4.1.2 Classic Text Document

Text classification has an important role to play, especially with the recent explosion of readily available text data. We use the Classic3 database [21] as one data set, which contains 1400 CRANFIELD documents from aeronautical system papers, 1033 MEDLINE documents from medical journals and 1460 CICI documents from information retrieval papers. All the documents are already labelled.

Based on this text documents database, we take MED as positive class and the other two kinds are negative class. Thus the distribution of these two classes is similar to the class distribution in our protein sequences dataset, with 27% being positive and 73% being negative. The details are shown in Table 4.2.

**Table 4.2 Text Document Data**

| Data | Number of Documents | Percentage of Each Class |
|---|---|---|
| MEDLINE | 1033 | 27% |
| CRANFIELD | 1400 | 73% |
| CICI | 1460 | |
| **Total** | **3893** | |

## 4.2 Classifiers

To compare our algorithm with *uncertainty sampling algorithm* [8] and *random sampling*, we choose two classifiers for our experiments: SVM classifier and KNN classifier [6]. These two classifiers are quite well known and have been used extensively and successfully.

Both of them are distance based classifiers, which satisfy our algorithm's requirement. KNN is obviously distance based. In SVM, the similarity between objects can be measured by dot products (i.e. angles and lengths) [23] in a high-dimensional feature space. Thus SVM can also be considered as distance based.

These two classifiers can both classify an unlabeled data and provide a measurement of the classification uncertainty, which satisfies the requirement of *uncertainty sampling algorithm* [8].

In KNN, we use the distance weighted k-Nearest-Neighbour rule [24] to measure the classification certainty. For an unlabeled data O, we find its k nearest neighbours in the initial labelled training data and order them in the ascending distances, $d_1, \ldots, d_k$ ($d_1 < \ldots < d_k$). A weight $w_i$ is assigned to the *ith* nearest neighbour and it is defined as:

$$w_i = \begin{cases} sign(i) \times \dfrac{d_k - d_i}{d_k - d_1}, & d_k \neq d_1 \\ sign(i) \times 1, & d_k = d_1 \end{cases} \qquad (3)$$

where sign(i) is positive if *ith* nearest neighbour is positive example, otherwise is negative.

Then the classification ***certainty*** of an instance O is defined in formula (4). The smaller the ***KNN-certainty(O)*** is, the more uncertain is the classification for object O.

$$\textbf{\textit{KNN-certainty(O)}} = abs(\sum_{i=1}^{k} w_i) \qquad (4)$$

For SVM, we use the existing software called SVM$^{light}$ [25]. The classification result of this software contains a value of the decision function for each test example. The value of the decision function is actually the distance between the test example to the

decision hyperplane in SVM. The smaller the value is, the closer the test example to the decision hyperplane, and the more uncertain the example's classification. This idea was proposed in [28]. Tong [5] uses the same idea for uncertainty measurement in SVM. In our experiment, we also employ the idea and use the absolute value of the decision function as the certainty of the classification for the corresponding object.

## 4.3 Pre-processing and Distance Function

To apply KNN and SVM classifiers, all the raw data need to be transformed into a vector space representation with a same dimensionality, i.e. each data is represented by a vector. Figure 4.1 presents the algorithm for this transformation.

**Figure 4.1 Data Transformation Algorithm**

---

1. Based on all unlabeled data in U, find frequent patterns[2] with Apriori algorithm, where the minimum support is 1% or 2%[3].
2. Take each frequent pattern as one dimension and form a vector space.
3. Transform every raw data object into a Boolean feature vector by taking 1 for each dimension if the data object contains the corresponding frequent pattern, otherwise 0.

---

For example, suppose there are 10 frequent patterns in the protein sequences data set and a feature vector for a sequence is [1,1,0,0,1,0,0,1,0,0], where 1 means the

[2] Frequent patterns are frequent term sets for text data and frequent subsequences for proteins data.
[3] [27] uses these minimum support value to find the frequent sub-sequences for prediction of OM proteins on the same data set.

sequence contains the corresponding frequent pattern, 0 means the sequence doesn't contain the corresponding frequent pattern. Note that for protein sequence data set the frequent pattern is actually the frequent substring, for text document data set (Classic3) the frequent pattern is the frequent keywords after removing stems and stop words. To find the frequent pattern, we use a public domain implementation of the basic Apriori algorithm [26].

After constructing the feature vector for each data, we can compute the distance between two data X and Y represented by feature vector $\vec{x}$ and $\vec{y}$ using e.g. the Euclidean Distance or Cosine Similarity in vector space model:

**Euclidean Distance:** $\mathbf{F(X, Y)} = |\vec{x} - \vec{y}| = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ (5)

**Cosine Similarity:** $\mathbf{F(X, Y)} = \mathbf{cos}(\vec{x}, \vec{y}) = \frac{(\vec{x} \bullet \vec{y})}{|\vec{x}| \cdot |\vec{y}|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$ (6)

SVM uses the dot product of two data objects to measure the distance, which is actually a similar calculation with the cosine similarity. In addition, the *representativeness* definition in formula (2) requires *Dist (O, ANS)* to be in the range of [0, 1] so that it won't dominate the *representativeness*. The cosine similarity has this property. So in our experiments, we use the cosine similarity in formula (6) as the distance function.

## 4.4 Evaluation Methodologies

Here we introduce the evaluation measure that we will be using in Experiment Studies. The performance of a classifier is usually measured by classification *accuracy*, *precision* and *recall*. They are defined based on a confusion matrix as shown in Table 4.3.

**Table 4.3 Confusion Matrix in Classification**

| | Actual positive | Actual negative |
|---|---|---|
| Classified as Positive | TP (true positive) | FP (false positive) |
| Classified as Negative | FN (false negative) | TN (true negative) |

$$\text{Overall Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \quad (7)$$

$$\text{Precision for positive class prediction} = \frac{TP}{TP + FP} \quad (8)$$

$$\text{Recall for positive class prediction} = \frac{TP}{TP + FN} \quad (9)$$

An often used measure in the information retrieval and data mining communities is the F-measure." According to Yang and Liu [19], this measure was first introduced by

C. J. van Rijsbergen [20]. They state that the F-measure combines recall (r) and precision (p) with an equal weight in the following form:

$$F - measure = \frac{2pr}{p+r} \tag{10}$$

It is understood that accuracy, precision and recall tend to be dominated by the classifier's performance on the majority class. Sometimes it is not reasonable. In addition, it is desirable to have a single measurement to evaluate the effectiveness of a classifier. Hence, we will take F-measure as the evaluation measurement of classifier(s).

Because our method need not retrain classifier(s) and can select multiple unlabeled data at a time, it is obvious that our method is more efficient. In addition, our assumption is that labelling data is very time consuming and costly, in contrast the efficiency of selecting unlabeled data is not a critical problem, but the effectiveness is very important. Thus we will mainly focus on the evaluation of effectiveness in different methods.

# CHAPTER 5

# EXPERIMENT STUDIES

We compare our method with *random sampling* and *general uncertainty sampling algorithm* [8] based on two data sets and two classifiers. Table 5.1 shows the combinations of data set and classifier for our experiments. The evaluation criteria is the average F-measure over 10 initial training sets (10*L) randomly drawn from the unlabeled data set U. To avoid bias in random sampling algorithm, we will do the random sampling 5 times for each of 10 initial training set, and take the average as the F-measure for this training set.

**Table 5.1 Experiments Combinations**

| | **KNN** (k=5) | **SVM** |
|---|---|---|
| Classic3 | KNN & Classic | SVM & Classic |
| Protein data | KNN & Protein | SVM & Protein |

Based on the same initial training set whose size is 10, we perform experiments to find answer sets whose sizes are respectively: 10, 30, 50 and 70. In the subsequent subsections, we present the experiment results and our analysis.

# 5.1 Experiment Results with KNN Classifier

**Figure 5.1 Experiment results with KNN (k=5)**

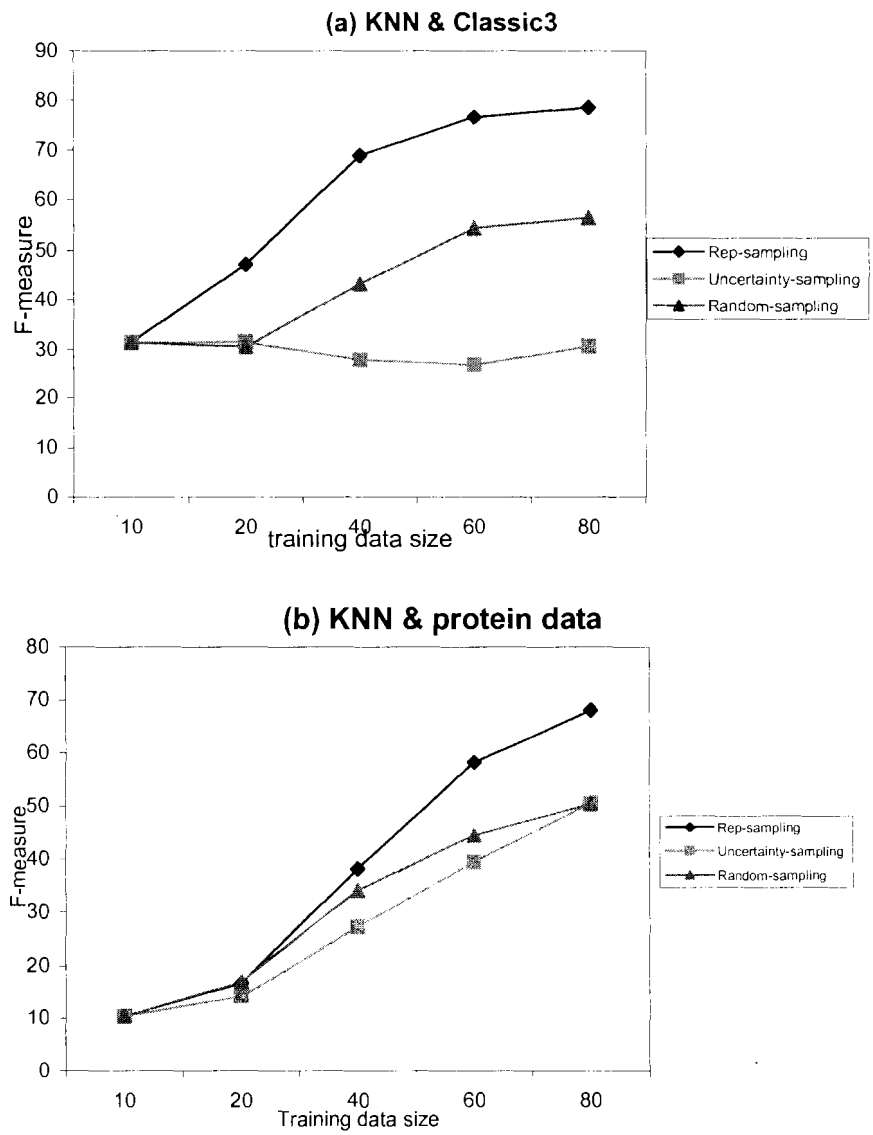


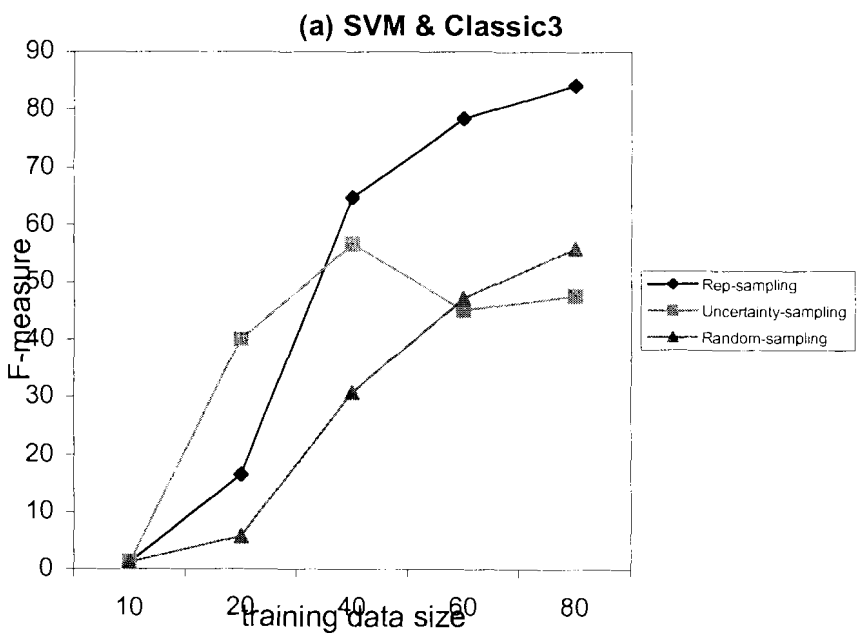
Figure 5.1 shows the experiment results using KNN with K = 5 on two different data sets. The experiments show that the classifier performance increases generally with sampling size under all methods.

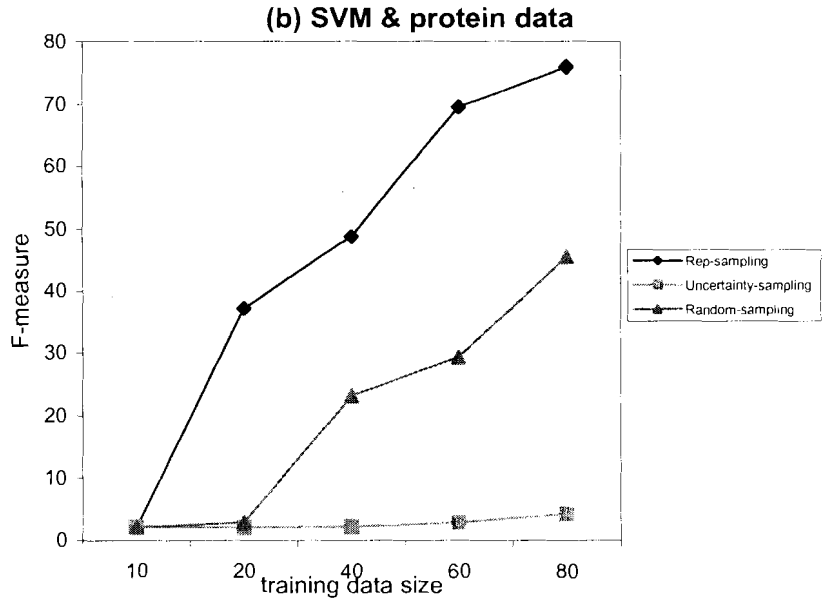
However, our method outperforms uncertainty sampling and random sampling on both datasets with KNN classifier. The superiority of our method is particularly notable with the increase of the sample size.

## 5.2 Experiments Results with SVM Classifier

Figure 3 shows the experiment results using SVM classifier on two different data sets.

**Figure 5.2 Experiment results with SVM**






Our method significantly outperforms both random sampling and uncertainty sampling methods for protein data in Figure 5.2 (b). For Classic3, Figure 5.2 (a) shows that uncertainty sampling outperforms our method a little bit when sampling small number, but our method outperforms uncertainty sampling when sampling more examples and achieves much better classification eventually.

## 5.3 Analysis

From the above experiment results, it is obvious that our method outperforms random sampling significantly. When sampling a small number of examples such as 10, they may have similar performance. But while sampling more examples, the superiority becomes more obvious. The reason is that with more random samples added, they may provides similar information and redundancy may occur, thus they can not contribute significantly to the classifier's performance. However, our method keeps sampling the representative examples and considering the redundancy in the sampled examples.

We also notice that the performance of random sampling varies significantly among different random samples. For example, the F-measure varies from 4% to 43% for a same initial labelled data set L. Hence, the random sampling method may not be reliable and the average F-measure over different examples selected by random sampling methods can't guarantee the performance in real world.

Our method also outperforms uncertainty sampling method, even though uncertainty sampling is a little bit better than our method when sampling a small number (10) of examples on Classic3 using SVM. It can also be explained by the redundancy in answer set. When sampling a small number of examples, it is less likely to have redundancy. But as the number of sampled examples increases, uncertainty sampling algorithm may select very similar examples because it greedily favours those examples with highest uncertainty. Our algorithm won't have this problem by considering redundancy.

Through the experiments, we have another important observation that can also explain why our method is better than uncertainty sampling and random sampling. This observation is:

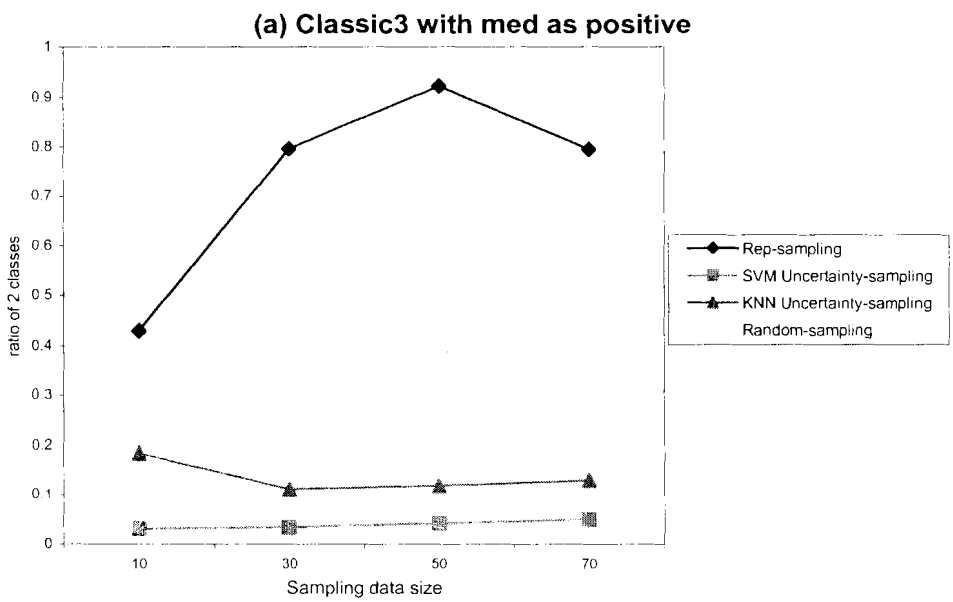
Our method returns a more balanced (with regarding to the distribution of positive and negative class) answer set than uncertainty sample and random sampling, and the ratio of two classes (positive and negative) in our method is becoming closer to 1 with the increase of sampling data size.

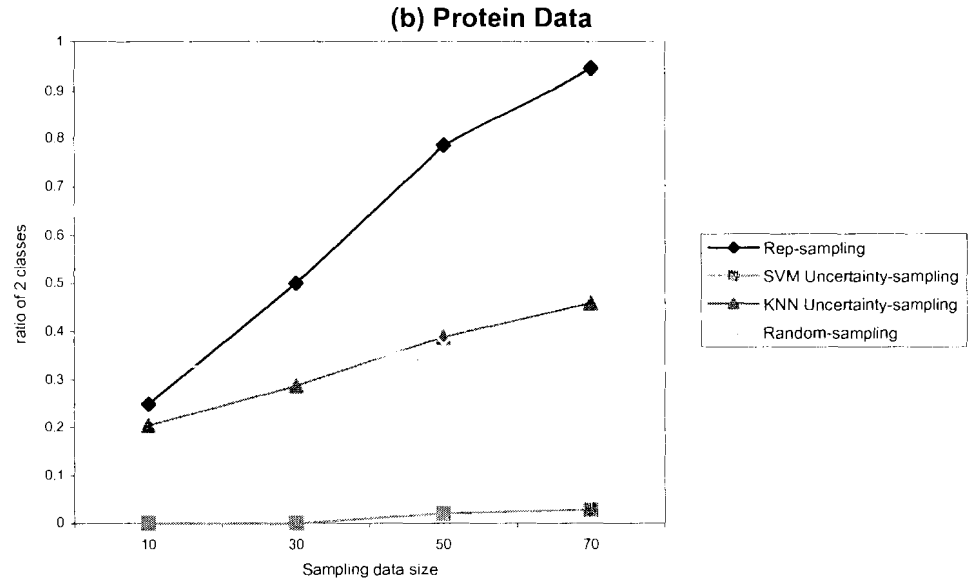
Figure 5.3 presents the ratio of positive and negative classes in the answer set with different methods. In random sampling, the distribution of answer set is close to the distribution of the data set. In uncertainty sampling, the sampled answer set is quite

unbalanced in terms of class distribution, even worse than random sampling in most cases, and the ratio of the two classes does not increase. This can also explain why random sampling performs better than uncertainty sampling in our experiments. It also demonstrates that uncertainty sampling algorithm does not work well in data sets with an unbalanced class distribution.

In our algorithm, the answer set will become close to balanced as number of sampled data increases, even thought the input data sets are unbalanced. In real applications, the unbalanced data set is the common situation, thus our algorithm is a practical method.

**Figure 5.3 The ratio of positive class and negative class in the sampling answer set.**






Even though the answer set returned by our algorithm may contain some noisy data, but through sampling more representative and balanced examples and reducing redundancy, it is impossible that the answer set is dominated by noisy data. Thus, our algorithm can also work well on data set with noise, while uncertainty sampling is noisy sensitive since its answer set may include a lot of noisy data.

All the experiments based on two data sets and two classifiers shows consistent results, so it is time to conclude.

# CHAPTER 6

# CONCLUSION AND FUTURE WORK

In this chapter, we first summarize the thesis, and then discuss some interesting future directions.

## 6.1 Summary of the Thesis

In order to reduce human efforts, there has been increasing interest in applying pool-based active learning for classification. Most previous studies adopt the incremental learning paradigm in which the classifier is retained upon each newly labelled query, thus it is still not very efficient and involves lots of domain experts' work. Most of them also employ an idea of "uncertainty sampling", in which the uncertain data whose class labels are unclear based on current classifier(s) are selected. However, this greedy idea may result in redundancy in the answer set, especially when the unlabeled data pool is unbalanced.

In this thesis, we propose a distance-based method to sample the top-k representative data in a batch, which is applicable to any distance-based classification algorithm, such as SVM, KNN and Naïve Bayesian. The data objects selected can effectively improve the performance of weak classifiers, whose accuracy is above 50%. The following are the contributions of our approach.

- We propose a novel concept to define *representativeness* of each data, which is used as the criteria to select the instances for labelling.
- To find the most representative data, we avoid choosing very similar data to reduce the redundancy in the answer set. Thus the selected data contains more information.
- The selected top-k representative data is more balanced (with regarding to class distribution) than random sampling and uncertainty sampling algorithms.
- No classifiers and retrain are needed during the selecting period. The top-k representative data can be selected at a time. So the domain experts or users can analyze and label these data in a batch mode.
- This method is more general and practical in the real world. Our algorithm can be more tolerant to noisy data input and still works well for unbalanced input data sets. Any distance-based classifier, such as SVM and KNN, can be used in our algorithm.
- The experiments based on two data sets and two classifiers demonstrate the effectiveness of this algorithm.

## 6.2 Future Work

With the success of this method, it is interesting to re-examine and explore many related problems, extensions and applications. Some of them are listed here.

- *Sampling examples that can minimize the future error*. This is another way to solve this problem. Our method actually employs the idea of

"uncertainty sampling", but we use a different criteria to measure the uncertainty and also consider the redundancy during the sampling period. So sampling examples that minimize the future error is different approach and worth some research to compare with our algorithm.

- *Sampling top-k representative data from a large training data set.* It assumes that a very large labelled training data set is given, but we want to select those most representative data for training classifiers. This is an interesting issue in the opposite direction.

# REFERENCE LIST


[1] Jiawei Han and Micheline Kamber. Data Mining: Concepts and Techniques, The Morgan Kaufmann Series in Data Management Systems, Jim Gray, Series Editor Morgan Kaufmann Publishers, pages 279-281, August 2000. ISBN 1-55860-489-8

[2] Yip Y.L., Scheib H., Diemand A.V., Gattiker A., Famiglietti L.M., Gasteiger E., Bairoch A, The Swiss-Prot Variant Page and the ModSNP Database: A Resource for Sequence and Structure information on Human Protein Variants, Hum. Mutat. 23:464-470(2004).

[3] Seung, H.S., Opper, M., and Sompolinsky, H. Query by committee. Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory, pages 287-294, 1992.

[4] Freund, Y., Seung, H., Shamir, E., and Tishby, N. Selective sampling using the Query By Committee algorithm. Machine Learning 28, pages 133-168, 1997.

[5] Tong, S., and Koller, D. Support Vector Machine Active Learning with Applications to Text Classification. Proc. of the Seventeenth International Conference on Machine Learning, 2001.

[6] Therrien, C. W., Decision, estimation and classification, John Wiley & Sons, 1989.

[7] Xu, Yu, Tresp, Xu and Wang. Representative Sampling for Text Classification Using Support Vector Machines. ECIR 2003, LNCS 2633, pp. 393-407, 2003

[8] Lewis, D., & Gale, W. A sequential algorithm for training text classifiers. Proceedings of the International ACM-SIGIR Conference on Research and Development in Information Retrieval, pages3-12, 1994.

[9] Lewis, D., & Catlett, J. Heterogeneous uncertainty sampling for supervised learning. Proceedings of the Eleventh International Conference on Machine Learning, pages148–156, Morgan Kaufmann, 1994.

[10] Hasenjager, M. & Ritter, H. Active Learning with Local Models. Neural Processing Letters 7: pages 107-117, 1998.

[11] Vapnik, V. Estimation of Dependences Based on Empirical Data. Springer Verlag. 1982.

[12] Mitchell, T. Generalization as search. Artificial Intelligence 28, pages 203-226, 1982.

[13] A. Gersho and R. M. Gray, Vector Quantization and Signal Compression, Kluwer Academic Publishers: Boston, MA, 1991.

[14] D. Haussler, M. Kearns and R. Schapire, Bounds on the same complexity of Bayesian learning using information theory and the VC dimension. In M. K. Warmuth and L. G. Valiant, editors, Proceedings of the Fourth Annual Workshop on Computational Learning Theory, pages61-74, San Mateo, CA, 1991.

[15] H. S. Seung, H. Sompolinsky and N. Tishby, Statistical mechanics of learing from examples. Phys. Rev. A45: pages 6056-6091, 1992.

[16] C.J.C. Burges, A tutorial on support vector machines for patter recognition, Data Mining and Knowledge Discovery, pages 121-168, 1998.

[17] P. E. Hart, The condensed nearest neighbor rule. IEEE Transaction on Information Theory, IT-14: pages 515-516, May 1968.

[18] P. E. Utgoff, Improved training via incremental learning. In Sixth International Workshop on Machine Learning, pages 362-365, 1989.

[19] Yiming Yang and Xin Liu, A re-examination of text categorization methods. In Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval, 1999.

[20] C. J. van Rijsbergen. Information Retrieval. Butterworths, London, 1979.

[21] Cornell University, public data sets, ftp://ftp.cs.cornell.edu/pub/smart, Accessed January 20, 2005

[22] ExPASy (Expert Protein Analysis System), Swiss-Prot Protein knowledgebase http://us.expasy.org/sprot/, Accessed January 20, 2005

[23] The Kernel Trick for Distances. Bernhard 2000.

[24] S. Dudani The distance-weighted k-nearest-neighbour rule. IEEE Transactions on Systems, Man and Cybernetics, vol. 6, pp. 325-327, 1976.

[25] T. Joachims, Making large-Scale SVM Learning Practical, Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999 .

[26] Yibin S, An implementation of the Apriori algorithm, 2000. http://www.cs.uregina.ca/~dbd/cs831/notes/itemsets/dic.java, Accessed January 20, 2005

[27] She R., Chen F., Wang K., Ester M., Gardy J.L., Brinkman F.S.L.: "Frequent-Subsequence-Based Prediction of Outer Membrane Proteins", 2003

[28] Gian Paolo Drago and Margo Muselli, Support Vector Machines for uncertainty region detection, 2001

[29] Foster Provost, Tom Fawcett, Analysis and Visualization of Classifier Performance: Comparison under Imprecise and Cost Distribution, KDD, 1997.