2-10-2005

Scaling Up Support Vector Machines with Application to Plankton Recognition

Tong Luo
University of South Florida

Scaling Up Support Vector Machines with Application to Plankton Recognition

by

Tong Luo

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Co-Major Professor: Lawrence O. Hall, Ph.D.
Co-Major Professor: Dmitry B. Goldgof, Ph.D.
Sudeep Sarkar, Ph.D.
Suresh Khator, Ph.D.
Thomas Sanocki, Ph.D.

Date of Approval:
February 10, 2005

Keywords: machine learning, data mining, kernel machines, active learning, bit reduction

ACKNOWLEDGEMENTS

This dissertation can not be done without many people's support. I am indebted to my advisor, Dr. Lawrence O. Hall, who stimulated many of the ideas in my dissertation. His careful review helped to greatly improve the dissertation. I hope I can be as professional as him in future. I am also very grateful to my co-major professor–Dr. Dmitry Goldgof. He introduced the plankton recognition project to me and provided many research guidances. The work in this project finally become this dissertation.

I would like to thank all the committee members and chair for their review and comments to my dissertation: Dr. Sudeep Sarkar, Dr. Suresh Khator, Dr. Thomas Sanocki and Dr. Michael X. Weng. In particular, I learned a lot from the courses taught by Dr. Sarkar. He is a great teacher and researcher.

I would like to thank my colleagues and friends in the vision lab: Kurt Kramer, Yan Qiu and Kevin Shallow. We had a lot of fun together. Also, I enjoyed the co-operation with Kurt.

Most of all, I would like to thank my family. My parents, sister and brother-in-law visited me during the last semester. Their supports keep me going and growing. My wonderful wife, Yang Wang, encouraged and supported me when I was busy and felt frustrated. This dissertation can not be done without her love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SCALING UP SUPPORT VECTOR MACHINES WITH APPLICATION TO PLANKTON RECOGNITION

Tong Luo

ABSTRACT

Learning a predictive model for a large scale real-world problem presents several challenges: the choice of a good feature set and a scalable machine learning algorithm with small generalization error. A support vector machine (SVM), based on statistical learning theory, obtains good generalization by restricting the capacity of its hypothesis space. A SVM outperforms classical learning algorithms on many benchmark data sets. Its excellent performance makes it the ideal choice for pattern recognition problems. However, training a SVM involves constrained quadratic programming, which leads to poor scalability. In this dissertation, we propose several methods to improve a SVM's scalability. The evaluation is done mainly in the context of a plankton recognition problem.

One approach is called active learning, which selectively asks a domain expert to label a subset of examples from a lot of unlabeled data. Active learning minimizes the number of labeled examples needed to build an accurate model and reduces the human effort in manually labeling the data. We propose a new active learning method "Breaking Ties" (BT) for multi-class SVMs. After developing a probability model for multiple class SVMs, "BT" selectively labels examples for which the difference in probabilities between the predicted most likely class and second most likely class is smallest. This simple strategy required several times less labeled plankton images to reach a given recognition accuracy when compared to random sampling in our plankton recognition system.

To speed up a SVM's training and prediction, we show how to apply bit reduction to compress the examples into several bins. Weights are assigned to different bins based on the number of examples in the bin. Treating each bin as a weighted example, a SVM builds a model using the reduced-set of weighted examples. Experimental results indicate bit reduction SVM (BRSVM) runs up to 245 times faster during the training phase and up to 33 times faster in the prediction phases. At a well-chosen compression ratio, it also beats random sampling in accuracy.

CHAPTER 1

INTRODUCTION

Applying machine learning to solve a real-world pattern recognition problem presents several challenges:

1. How do we produce a set of features that best summarizes the targeted patterns and differentiates between those patterns?

2. Can we find a good machine learning algorithm, which learns a model on a set of features extracted from the training data, to predict the targeted patterns correctly from future data?

3. Is this machine learning algorithm scalable to a large number of examples?

This dissertation addresses the above questions mainly in the context of plankton recognition. However, our methods can also be applied on general pattern recognition problems and other data sets. Among the three challenges, scalability is the main focus.

The knowledge of distributions of underwater plankton helps to predict particle flux, fisheries recruitment and biomass production in the ocean. Therefore, the first generation Shadow Image Particle Profiling Evaluation Recorder (SIPPER I) was developed to continuously sample plankton and suspended particles in the ocean [78]. It is capable of producing tens of thousands images an hour. As a result, a plankton recognition system is necessary to identify those plankton automatically to avoid otherwise prohibitive image labeling work for people. The images from SIPPER I are 1-bit, black and white images without clear contours. This situation brings us to the answer for the first question: How do we produce a set of good features that maximize the classification accuracy? Since contour features like Fourier descriptors were not stable for SIPPER I images, we chose several

robust features: the moment invariants and granulometric features which do not depend heavily on contour information. Also, several domain specific features were designed to differentiate between several particular types of plankton. A wrapper approach was used to select the optimal subset of features.

After feature computation, another issue was to find a machine learning algorithm which generalizes well to future unseen data. Many traditional machine learning algorithms such as nearest neighbor [1], decision trees [73], neural networks [33], and bayesian networks [68] etc. often suffer from overfitting or underfitting due to the lack of an elegant way to control the trade-off between empirical risk minimization (ERM) and generalization ability. Advances in statistical learning theory [90][91] introduced the VC dimension to measure the capacity of a hypothesis space, in which a learning algorithm searches for the optimal function for classification. Vapnik and Chervonenkis [91] indicated a better generalization bound can be achieved by restricting a learning algorithm's VC dimension. A support vector machine (SVM) is developed to separate the data in a high-dimensional feature space using a large margin hyperplane. Such a large margin classifier has a low VC dimension, thus leading to good generalization ability. Therefore, we use a support vector machine to classify the feature vector extracted from the SIPPER I images. Our experiments indicate a SVM performs better than a decision tree, neural network and even ensembles of decision trees [11][12][4][26].

The focus of this dissertation is the third challenge: scaling up a support vector machine. As many applications generate massive labeled and unlabeled data sets, it is crucial for a SVM to handle large amount of data. We tackle the scalability issues of a SVM as follows.

First, the high sampling rate of SIPPER allows us to obtain a large number of images without class labels. At the same time, the SVM classifier built on an initial small set of labeled images is expected to be more accurate by being retrained with more labeled images. The time required is prohibitive for a person to manually label all the images for retraining. A smart sampling strategy is needed to selectively choose to label a small set of images, which most helps to improve the accuracy. Such a smart selective sampling method is also

called active learning. Most previous work [89][77] on active learning with SVMs focuses on two-class problems and can not be directly extended to multiclass problems. We develop a simple active learning method–"Breaking Ties" for multi-class SVMs. Interpreting SVM outputs through a probability model, we chose to label the examples whose two largest class probabilities are close to each other. Breaking the potential tie helps improve the model's classification ability. Our experiments show "Break Tie" outperforms random sampling and a least certain active learning method.

Second, as the number of labeled examples accumulate, training a SVM becomes very time consuming and sometimes even prohibitive. A large number of support vectors resulting from a large data set makes the prediction slow too. Many methods [90], [63] [43] [70] [45] [27] [34][97] of speeding up the quadratic programming (QP) problem for a SVM have been well developed. However, training a SVM is still very slow for a very large data set. The reduced set method [15][16][64][82] has been proposed to use a small set of points called pre-images to approximate the support vectors, which in turn reduces the prediciton time. However, solving the pre-image problem itself is time consuming. Believing there is not enough space for improvement in speeding up the QP problem, we turn to the "data squashing" method [29] instead. This method compresses the large data set into several small bins. A model is fit by only using a representative example instead of all examples within a bin. The reduced training set results in significantly less training and prediction time. However, previous work on data squashing + SVM [99][85][9] applied clustering algorithms to partition the data, while a clustering algorithm itself is relatively expensive computationally. Moreover, all but one previous experiment was done using the simplest kernel: a linear kernel. This adds doubt about whether their methods can be generalized to other complicated kernels. In this dissertation, we propose a very fast data squashing method: bit reduction. Bit reduction groups similar examples by reducing their resolution. The mean statistics of examples from each bin are computed. We assign a weight to the mean according to the number of examples within a bin. A SVM trained with weighted examples is developed. We call this method bit reduction SVM (BRSVM). Given a good

choice of compression ratio, BRSVM can achieve significant speedup in training and prediction with a statistically insignificant loss in accuracy. It also outperforms random sampling at a well chosen compression ratio.

The dissertation is organized as follows. Chapter 2 introduces the support vector machine and its generalization bound. Chapter 3 presents our systematic method to recognize underwater SIPPER I images: extracting robust image features, selecting an optimal subset of features, applying a SVM to classify image features and compare with other classifiers and interpreting a SVM's outputs with a new probability model. Chapter 4 describes our active learning method "Breaking Ties" (BT) for multi-class SVMs to reduce human effort in labeling images. "BT" is applied to recognizing a new feature set of 3-bit, graylevel images from the advanced SIPPER (SIPPER II). Chapter 5 introduces bit reduction SVM (BRSVM), which employs a simple bit reduction method to compress the data and speed up the training and prediction. We conclude this dissertation and present future research directions in Chapter 6.

CHAPTER 2

BACKGROUND

2.1 Machine learning problem

Given m pairs of examples: $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_1, y_1)$, ..., $(\mathbf{x}_m, y_m)$; $\mathbf{x}_i \in R^p$. A machine learning algorithm learns a function f mapping the input $\mathbf{x}_i$ to the output y_i, where $y_i = f(\mathbf{x}_i)$. If y_i is a real value, this problem is called a regression problem. Otherwise, if y_i is an unordered discrete value, it is a classification problem. All chapters in this dissertation deal with classification problems. Therefore, we only discuss classification problems in the rest of this chapter.

A function learned by a machine learning algorithm is expected to generalize well to unseen data. For example, given a new data point $\mathbf{x}_t$ which does not belong to the m pairs of examples, we want to predict its response y_t correctly by using f. Before we address the generalization ability of learning algorithms, we give some definitions in the following.

Definition 1. The given m pairs of examples are called training data, which are assumed to be independently, identically distributed (iid) according to an unknown probability distribution $P(x, y)$.

Definition 2. A hypothesis space is a function space where a learning algorithm searches for the optimal function f based on certain criteria.

Definition 3. A loss function defines the loss associated with the prediction $f(\mathbf{x}_i)$ when the true output is y_i. For example, a widely used loss function in classification problems is the

0-1 loss function, in which

$$L(\mathbf{x}_i, y_i, f) = \begin{cases} 0 & : \quad f(\mathbf{x}_i) = y_i \\ 1 & : \quad f(\mathbf{x}_i) \neq y_i \end{cases}$$

Definition 4. Generalization error is also called true risk or expected error. It is defined as follows.

$$L(f) = \int L(x, y, f) dP(x, y)$$

Noting $P(x, y)$ is unknown, we can not measure generalization error precisely.

Definition 5. Empirical loss is also called empirical risk or training error. It is the loss occured on the training data and is usually measured by the average loss on the training data.

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

A good machine learning algorithm should minimize the generalization error. While the generazliation error is not measureable due to a lack of knowledge about the true data distribution $P(x, y)$, a natural choice for learning is to minimize the empirical risk instead. However, a pure empirical risk minimization (ERM) algorithm only gives an accurate prediction on the training data and does not ensure equally good performance on the unseen data. Overfitting is often observed in practice such that a function that fits the training data well performs poorly on the new data. To tackle this problem, statistical learning theory [91][92] indicates that restricting the capacity of a hypothesis space provides a bound between the generalization error and the empirical loss. Vapnik and Chervonekis [92] proposed to use the Vapnik-Chervonekis (VC) dimension to measure the capacity of a hypotheis space. We study the VC dimension by first introducing the concept of shattering.

Definition 6. In binary classification, a hypothesis space (function space) shatters a set of examples if for any class label assignments to these examples, there exists at least one hypothesis (function) within the hypothesis space that can separate them.

The shattering ability shows the classification ability in a hypothesis space. No matter how the class labels are arranged over those examples, a rich hypothesis space provides a hypothesis (function) making no error, that is, shatters them.

Definition 7. In binary classification, the VC dimension is the maximum number of examples such that the hypothesis space (function space) can shatter them.

The VC dimension indicates the capacity of a hypothesis space. In an extreme case, a hypothesis space, whose VC dimension is m, can always provide a hypothesis (function) with no training error on m training data. Intuitively, a very high-capacity hypothesis space has the ability to classify very large number of examples correctly. On the other hand, a learning algorithm searching in such a hypothesis space is prone to overfitting.

Theorem 1. Given a hypothesis space with a VC dimension h and a sample size m, the generalization error bound holds with probability 1-δ:

$$L(f) \leq L_{emp} + \sqrt{\frac{h(ln\frac{2m}{h}) + h - ln\frac{\delta}{4}}{m}} \tag{2.1}$$

A small VC dimension gives a tight bound for generalization error. In this situation, the generalization error can not deviate from the empirical loss very much with a large probability. In order to keep the generalization bound low, a learning algorithm should minimize the empirical loss in a hypothesis space with a small VC dimension.

2.2 Support vector machines

A support vector machine (SVM) minimizes the empirical loss and restricts the capacity of a hypothesis simultaneously. A regularized form of a SVM can be written as:

$$\min_{f \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} L(\mathbf{x}_i, y_i, f) + \lambda ||f||^2_{\mathcal{H}} \tag{2.2}$$

$\mathcal{H}$ is a hypothesis space in which a learning algorithm searches for an optimal function f. $||f||^2_{\mathcal{H}}$ is a L_2 norm defined on the hypothesis space. Minimizing this term amounts to restricting the capacity of a hypothesis space, which in turn results in a tight generalization

bound. λ is the regularization constant that controls the trade-off between the empirical loss and the capacity of a hypothesis space.

2.2.1 Deriving the SVM

A SVM uses the hinge loss as its loss function:

$$L(\mathbf{x}_i, y_i, f) = (1 - y_i f(\mathbf{x}_i))_+ = \begin{cases} 0 & : \quad y_i f(\mathbf{x}_i) > 1 \\ 1 - y_i f(\mathbf{x}_i) & : \quad otherwise \end{cases}$$

where $(k)_+ = \max(k, 0)$. As $y_i \in (-1, 1)$, a SVM tries to make $y_i f(\mathbf{x}_i)$ as large as possible. The value of $y_i f(\mathbf{x}_i)$ is defined as the margin. Intuitively, a large value of $y_i f(\mathbf{x}_i)$ brings a strong confidence of making a correct classification. That is why a SVM is also called a large margin classifier. If $y_i f(\mathbf{x}_i) > 1$, there is no loss. Otherwise, we pay $1 - y_i f(\mathbf{x}_i)$ as the penalty.

In a SVM, the input data $\mathbf{x}_i$ is mapped to a high-dimensional feature space by using a function ϕ. The hypothesis space $\mathcal{H}$ in a SVM is a set of hyperplanes in that feature space. A SVM works in a high-dimensional feature space because many linearly unseparable problems in a low-dimensional space become separable in a high-dimensional feature space. A SVM's hypothesis space is in the form of a linear model:

$$f(\mathbf{x}_i) = \langle w, \phi(\mathbf{x}_i) \rangle + b \tag{2.3}$$

where (w, b) are the coefficients for the optimal hyperplane. Consequently, the L_2 norm of the SVM hypothesis space is $\langle w, w \rangle$.

Substituting the loss function and L_2 norm of a SVM into Eq. (2.2), we get

$$\min_{w,b} \frac{1}{m} \sum_{i=1}^{m} (1 - y_i(\langle w, \phi(\mathbf{x}_i) \rangle + b))_+ + \lambda \langle w, w \rangle \tag{2.4}$$

Eq. (2.4) can be further simplified by introducing slack variables $\xi_i = 1 - y_i(\langle w, \phi(\mathbf{x}_i) \rangle + b)$, $i = 1,...,m$. It leads to

$$\min_{w,b} \frac{1}{m} \sum_{i=1}^{m} \xi_i + \lambda \langle w, w \rangle \tag{2.5}$$

$$\text{subject to: } y_i(\langle w, \phi(\mathbf{x}_i) \rangle + b) \geq 1 - \xi_i \tag{2.6}$$

$$\lambda, \xi_i > 0 \tag{2.7}$$

As a result, a SVM is equivalent to searching for the optimal (w, b) in the above quadratic programming problem.

2.2.2 Kernel

According to the representer theorem [46], the function f with optimal (w, b) can be written as

$$f(x) = \sum_{i=1}^{m} \beta_i \langle \phi(\mathbf{x}_i), \phi(x) \rangle \tag{2.8}$$

Therefore, the optimal function f only depends on the inner product between two examples in the feature space. In other words, if we can define an inner product in the feature space $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$, we only need to work with $k(\mathbf{x}_i, \mathbf{x}_j)$ without explictly calculating $\phi(\mathbf{x}_i)$, i = 1,...,m. The function k which defines an inner product computation is called a kernel. The "kernel trick" saves many computations, which are otherwise involved in a high-dimensional space. In an extreme case, we can work with a kernel k without knowing what the corresponding feature space is.

Definition 8. A kernel matrix (Gram matrix) K is positive semi-definite if

$$\sum_i \sum_j c_i c_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

for any given $c_i \in \mathbf{R}$.

If a kernel is positive semi-definite, there exists a feature mapping ϕ such that

$$\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle = k(\mathbf{x}_i, \mathbf{x}_j)$$

Therefore, any positive semi-definite kernel is a valid kernel. It implicitly determines a feature space in which a inner product is well defined. Such a feature space is called a Reproducing Kernel Hilbert Space (RKHS). More details about kernels and RKHS space can be found in [6][92][37].

There are three widely used kernels: the linear kernel, polynomial kernel and Gaussian RBF kernel.

$$\text{Linear kernel:} \qquad k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle \tag{2.9}$$

$$\text{Polynomial kernel:} \qquad k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle^d \tag{2.10}$$

$$\text{RBF kernel:} \qquad k(\mathbf{x}_i, \mathbf{x}_j) = exp(-g||x_i - x_j||^2) \tag{2.11}$$

Paper [36] is a good reference for many other kernels and their applications.

2.2.3 Margin, VC dimension of large margin hyperplane and structure risk minimization

The margin of a hyperplane classifier is defined in [83] as follows.

Definition 9. For a hyperplane $\{\langle w, x \rangle + b = 0\}$, the margin of a point $(\mathbf{x}_i, y_i)$ is $\rho(\mathbf{x}_i, y_i)$, where

$$\rho(\mathbf{x}_i, y_i) = \frac{y_i(\langle w, \mathbf{x}_i \rangle + b)}{||w||}$$

The margin of $(\mathbf{x}_1, y_1)$,..., $(\mathbf{x}_m, y_m)$ is the minimum value of the individual margin.

$$\rho = \min_{i=1}^{m} \rho(\mathbf{x}_i, y_i)$$

Geometrically, the margin of $(\mathbf{x}_i, y_i)$ is just the distance from $\mathbf{x}_i$ to the hyperplane. An example being correctly classified has a positive margin, otherwise it has a negative margin. Intuitively, a large margin hyperplane has good generalization ability. Since training data are sampled from the unknown true data distrubtion $P(x, y)$, the future data can be taken as the training data with a certain noise. As long as the amplitude of the noise is less than the margin, a large margin hyperplane will correctly classify it.

If we assume $|\langle w, x\rangle + b| = 1$, the margin of the m examples is $\frac{1}{||w||}$. Therefore, Eq. (2.6) minimizes the empirical loss ξ_i and maximizes the margin $\frac{1}{||w||}$ by minimizing $\langle w, w\rangle$.

The good generalization ability of a large margin classifier is explained in [90] using the VC dimension theory.

Theorem 2. Given a set of hyperplanes which satisfy $\langle w, \mathbf{x}_i\rangle + b = 1$ and $||w|| < \Lambda$. It has a VC dimension (h) satisfying

$$h \leq R^2\Lambda^2$$

where R is the radius of the smallest sphere centered at the origin and containing all the examples.

Therefore, h is related to the margin. We can restrict the VC dimension of a large margin classifier by minimizing $||w||$ as shown in Eq. (2.6). This principle is called structure risk minimization (SRM). The idea of SRM is to search through a set of hypothesis space with reduced capacity or VC dimension. In Eq. (2.6), a small value of $||w||$ corresponds to a hypothesis space with a small capacity. In this way, the generalization ability is improved.

Please refer to [3][84] for more details about the VC dimension of a large margin hyperplane and SRM.

2.2.4 Primal and dual form of SVM

To be consistent with the conventional expression of a SVM, we introduce a scalar variable C and rewrite Eq. (2.6) as

$$\min \frac{1}{2}\langle w, w\rangle + \frac{C}{m}\sum_{i=1}^{m}\xi_i \tag{2.12}$$

$$\text{subject to: } y_i(\langle w, \phi(\mathbf{x}_i)\rangle + b) \geq 1 - \xi_i \tag{2.13}$$

$$C, \xi_i > 0, i = 1, ...m \tag{2.14}$$

It is not hard to see that Eq. (2.12) is equivalent to Eq. (2.6). Eq. (2.12) is called the primal form of a SVM.

Introducing the Lagrangian multiplier α_i, Eq. (2.12) leads to

$$\begin{aligned} L(\alpha, w, b) &= \frac{1}{2}\langle w, w\rangle + \frac{C}{m}\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i(y_i(\langle w, \phi(\mathbf{x}_i)\rangle + b) - 1 + \xi_i) \\ \alpha_i &> 0,\ i = 1, ..., m \end{aligned} \tag{2.15}$$

where α is the vector of $(\alpha_1, \alpha_2, ..., \alpha_m)$.

From the knowledge of convex optimization [10], the optimal solution to Eq. (2.15) is at a saddle point: the minimum with respect to (w, b) and the maximum with respect to α.

Take the partial derivatives of $L(\alpha, w, b)$.

$$\frac{\partial L(\alpha, w, b)}{\partial w} = 0 \text{ and } \frac{\partial L(\alpha, w, b)}{\partial b} = 0 \tag{2.16}$$

It leads to

$$w = \sum_{i=1}^{m}\alpha_i y_i \phi(\mathbf{x}_i) \tag{2.17}$$

$$\sum_{i=1}^{m}\alpha_i y_i = 0 \tag{2.18}$$

Substitute them into Eq. (2.15), the dual form of a SVM is as follows.

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} \quad & 0 \leq \alpha_i \leq \tfrac{C}{m}, i = 1, ..., m \\ & \textstyle\sum_{i=1}^{m} \alpha_i y_i = 0 \end{aligned} \tag{2.19}$$

The dual problem of a SVM is simpler than its primal form. A quadratic programming (QP) solver can be used to solve it. SVM's decision function is

$$f(x) = \sum_i \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b \tag{2.20}$$

The Karush-Kuhn-Tucker condition of the optimal solution is

$$\alpha_i [y_i (\langle w, \phi(\mathbf{x}_i) \rangle + b) - 1 + \xi_i] = 0 \tag{2.21}$$

The variable α_i is nonzero only when Eq. (2.22) is satisfied. In this case $\mathbf{x}_i$ contributes to the decision function and is called a support vector (SV).

$$y_i (\langle w, \phi(\mathbf{x}_i) \rangle + b) = 1 - \xi_i \tag{2.22}$$

Therefore, we get a sparse solution of the decision function, where only SVs contribute.

2.3 Multi-class SVM

There are two main approaches to extending SVMs to multi-class classification: one-vs-all and one-vs-one.

1. One-vs-all: A set of binary SVMs are trained to separate one class from the rest. The drawback is that we are handling unbalanced data when building binary SVMs. Moreover, each binary SVM is built on a totally different training set. There might

be cases in which some binary SVMs conflict with each other for some examples. It is difficult to assign the class by just the real-valued outputs from every binary SVM.

2. One-vs-one: All possible groups of 2 classes are used to build binary SVMs. In the N class case, we will build $\frac{N(N-1)}{2}$ binary SVMs. When a new example is tested, all the binary SVMs vote to classify it.

The one-vs-one approach needs to build more binary SVMs than the one-vs-all approach, however, each of its binary SVMs only learns on a fraction of the data, thus it can be time efficient in a large data set. Hsu [40] compared the one-vs-all and one-vs-one approach to handle multiple class problems in SVMs. They found the one-vs-one approach was much faster and more accurate than the one-vs-all approach on most data sets. We also compared the two approaches on our data sets in Chapter 3 and the one-vs-one approach was superior to the other in our experiments.

CHAPTER 3

RECOGNIZING PLANKTON IMAGES FROM THE SHADOW IMAGE PARTICLE PROFILING EVALUATION RECORDER

We present a system to recognize underwater plankton images from the Shadow Image Particle Profiling Evaluation Recorder (SIPPER). The challenge of the SIPPER image set is that many images do not have clear contours. To address that, shape features that do not heavily depend on contour information were developed. A soft margin support vector machine (SVM) was used as the classifier. We developed a way to assign probability after multi-class SVM classification. Our approach achieved approximately 90% accuracy on a collection of plankton images. On another larger image set containing manually unidentifiable particles, it also provided 75.6% overall accuracy. The proposed approach was statistically significantly more accurate on the two data sets than a C4.5 decision tree and a cascade correlation neural network. The single SVM significantly outperformed ensembles of decision trees created by bagging and random forests on the smaller data set and was slightly better on the other data set. The 15-feature subset produced by our feature selection approach provided slightly better accuracy than using all 29 features. Our probability model gave us a reasonable rejection curve on the larger data set.

3.1 Introduction

Recently, the Shadow Image Particle Profiling Evaluation Recorder (SIPPER) was developed to continuously sample plankton and suspended particles in the ocean [78]. The SIPPER uses high-speed digital line-scan cameras to record images of plankton and other particles, thus avoiding the extensive post-processing necessary with analog video particle images. The large sampling aperture of the sensor combined with its high imaging resolu-

tion (50 μm per pixel), means that it is capable of collecting tens of thousands of plankton images an hour. This soon overwhelms a scientist attempting to manually classify the images into recognizable plankton groups. Therefore, an automated plankton recognition system is necessary to solve the problem or at the very least to help with the classification.

Tang [88] developed a plankton recognition system to classify plankton images from video cameras. The moment invariants and Fourier descriptor features from contour images were extracted. Also, granulometric features from the gray-level images were computed. Finally, a learning vector quantization neural network was used to classify examples. Tang [88] achieved 92% classification accuracy on a medium-size data set.

The project ADIAC (Automatic Diatom Identification and Classification) has been ongoing in Europe since 1998. Different feature sets and classifiers have been experimented with to recognize separate species of diatoms taken from photo-microscopes. Loke [51] and Ciobanu [20] studied some new contour features. Santos [79] extended the contour features to multi-scale Gabor features together with texture features. Wilkinson [96] applied morphological operators to help extract both contour and texture information. Fischer [35] summarized these features and used ensembles of decision trees to classify the combined feature set. Greater than 90% overall accuracy was achieved on the diatom images.

However, images from previous work are of relatively good quality or at least with clear contours. Therefore, complicated contour features and texture information can be extracted easily. The SIPPER images, on the other hand, present several difficulties:

1. Many SIPPER images do not have clear contours. Some are partially occluded. Therefore, we cannot primarily depend on contour information to recognize the plankton.

2. The SIPPER image gallery includes many unidentifiable particles as well as many different types of plankton.

3. The SIPPER images in our experiments are binary, thus lacking texture information.

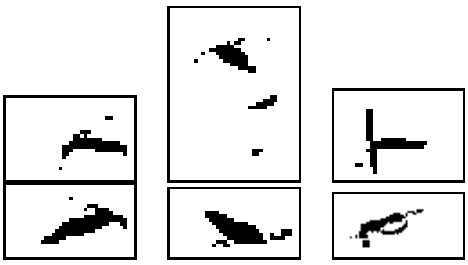

Figure 3.1. Copepod in SIPPER I images.

Tang [87] proposed several new features for SIPPER images and applied multilevel dominant eigenvector methods to select a best feature subset. A Gaussian classifier was employed to recognize the image features and validate the feature selection methods on selected identifiable plankton.

This chapter is organized as follows. Section 3.2 introduces the binary SIPPER images used in the experiments. In Section 3.3, we discuss the preprocessing of the images and the extraction of the features. Section 3.4 describes the probability assignment in a multi-class support vector machine. We applied wrappers with backward elimination to select the best feature subset in Section 3.5 and experimental results for the system are detailed in Section 3.6. Finally we summarize our work in Section 3.7.

3.2 Image gallery

The image gallery includes 7285 binary SIPPER images: 1285 images from five types of plankton were initially selected by marine scientists as our starting point. The other 6000 images were samples from a deployment of SIPPER in the Gulf of Mexico. The 6000 images were from the five most abundant types of plankton and manually unrecognizable particles. All the images were manually classified by marine scientists. Figures 3.1 to 3.7 are typical examples of plankton and unidentifiable particles from the SIPPER image set.

3.3 Feature computation

In the field of shape recognition, some general features like invariant moments, Fourier descriptors and granulometric features etc. are widely used [23]. However, those general

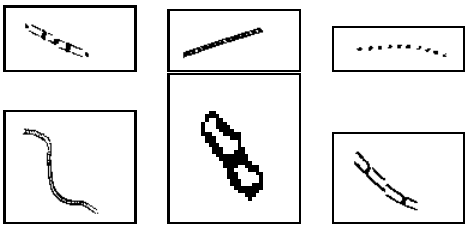

Figure 3.2. Diatom in SIPPER I images.

Figure 3.3. Doliolid in SIPPER I images.

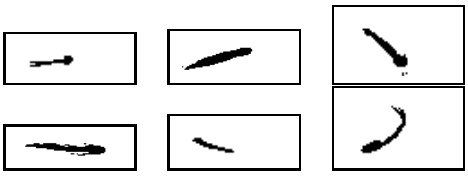

Figure 3.4. Larvacean in SIPPER I images.

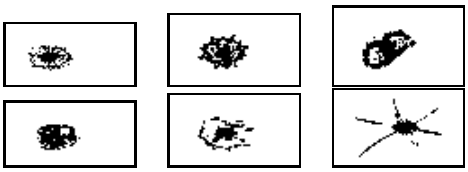

Figure 3.5. Protoctista in SIPPER I images.

Figure 3.6. Trichodesmium in SIPPER I images.

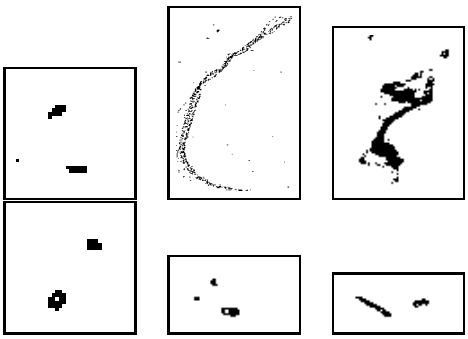

Figure 3.7. Unidentifiable particles in SIPPER I images.

Table 3.1. Description of 29 features.

Features	Number of features
Moment invariants of the original image	7
Moment invariants of the contour image after closing	7
Granulometric features	7
Domain specific features	8

features are insufficient to capture the information contained in SIPPER images sampled from the Gulf of Mexico. Moreover, the SIPPER images have a lot of noise around or on the plankton and many images do not have clear contours, thus making a direct implementation of the contour features (Fourier descriptor [100] etc.) not stable. To solve this problem, we first preprocessed the images to suppress noise. We only extracted invariant moments and granulometric features, which are relatively stable with respect to noise and do not depend heavily on the contour image. To capture the specific information from our SIPPER image set, domain knowledge was used to extract some specific features such as size, convex ratio, transparency ratio, etc. There are 29 features in total as shown in Table 3.1. There rest of this section will provide more details about these features.

3.3.1 Object detection and noise suppression

Marine scientists used specialized software to detect objects: A series of morphological dilations were performed to connect the nearest image pixels. If the bounding box of the

connected image pixels after dilation was bigger than 15 × 15, the original image was stored as an object. Otherwise, the image pixels were considered irrelevant and deleted.

There are many noise suppression methods [86]. In this work, we applied a simple method–connected component analysis to eliminate the noise pixels far from the object bodies. Under the eight-connectivity condition (that is, all eight neighbor pixels of a pixel are considered connected to it), if a pixel's connected path to the image body is more than 4, it will be regarded as noise and eliminated. In addition, a morphological closing operation with a 3 × 3 square window as the structure element was used to get a roughly smooth image shape and separate the holes inside the plankton body from the background [69]. This operation also helps to compute several domain specific features described in Section 3.3.4 and to get a rough contour of the image.

3.3.2 Moment invariants

Moment features are widely used as general features in shape recognition. The standard central moments are computed as follows:

$(\bar{x}, \bar{y})$ is the center of the foreground pixels in the image. The $(p + q)$-order central moments are computed with every foreground pixel at (x, y):

$$\mu(p, q) = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q \tag{3.1}$$

Then central moments are normalized by size as shown in Eq. (3.2).

$$\eta(p, q) = \frac{\mu(p, q)}{\mu(0, 0)^{(\frac{p+q}{2}+1)}} \tag{3.2}$$

Hu [41] introduced a way to compute the seven lower order moment invariants based on several nonlinear combinations of the central moments. Using the normalized central moments, we got scale, rotation and translation invariant features. We computed the same 7 moment invariants on the whole object and the contour image after a morphological closing operation, respectively. The computation of seven moment features are as follows.

Table 3.2. Hu's moment invariants.

Feature number	Moment
1	$\mu(2,0)+\mu(0,2)$
2	$(\mu(2,0)-\mu(0,2))^2+4\mu(1,1)^2$
3	$(\mu(3,0)-3\mu(1,2))^2+(3\mu(2,1)-\mu(0,3))^2$
4	$(\mu(3,0)+\mu(1,2))^2+(\mu(2,1)+\mu(0,3))^2$
5	$(\mu(3,0)-3\mu(1,2))(\mu(3,0)+\mu(1,2))[(\mu(3,0)+\mu(1,2))^2-3(\mu(2,1)+\mu(0,3))^2]+(3\mu(2,1)-\mu(0,3))(\mu(2,1)+\mu(0,3))[3(\mu(3,0)+\mu(1,2))^2-(\mu(2,1)+\mu(0,3))^2]$
6	$(\mu(2,0)-\mu(0,2))[(\mu(3,0)+\mu(1,2))^2-(\mu(2,1)+\mu(0,3))^2]+4\mu(1,1)(\mu(3,0)+\mu(1,2))(\mu(2,1)+\mu(0,3))$
7	$(3\mu(2,1)-\mu(0,3))(\mu(3,0))+\mu(1,2))[(\mu(3,0)+\mu(1,2))^2-3(\mu(2,1)+\mu(0,3))^2]-(\mu(3,0)-3\mu(1,2))(\mu(2,1)+\mu(0,3))[3(\mu(3,0)+\mu(1,2))^2-(\mu(2,1)+\mu(0,3))^2]$

3.3.3 Granulometric features

Since the Hu moments only contain low order information from the image, we also extracted granulometric features [55], which are robust measurements of the high order information. Granulometric features were computed by doing a series of morphological openings with different sizes of structure elements. Then we recorded the differences in size between the plankton before and after openings. Granulometric features are relatively robust to noise and contain inherent information on shape distribution. Tang [88] found that granulometric features were the most important features in his experiments.

We applied 3×3, 5×5, 7×7 and 9×9 square windows as structure elements and did a series of morphological openings. Then differences in size were normalized by the original plankton size to obtain the granulometric features. Also, we applied 3×3, 5×5 and 7×7 square windows as structure elements, and did a series of morphological closings. The differences in size were normalized in the same way. We did not apply a 9×9 square window to the closing because the SIPPER images are so small that most of them are diminished after the closing with a 7×7 square window as the structure element. The granulometric features are computed from Eq. (3.3).

$$g_i = \frac{\#\text{ pixels changed after } i^{th} \text{ morphological operations}}{\#\text{ pixels in the original image}} \tag{3.3}$$

3.3.4 Domain specific features

Moment invariants and granulometries only capture some global information, which is insufficient to classify SIPPER images. Given advice from domain experts, we developed some domain specific features to help classification. The domain specific features include size, convex ratio, transparency ratio, eigenvalue ratio, and the ratio between the plankton's head and tail.

1. Size: It is the area of the plankton body, that is, the number of foreground pixels in the plankton image. The size features were extracted for both the original image and the contour images.

2. Convex ratio: We implemented a fast algorithm [7] to get the convex hull of the plankton image. The convex ratio is the ratio between the plankton image size and the area of the convex hull. This feature contains information about the plankton boundary irregularity. We computed convex ratios with Eq. (3.4) for the original image and the image after a morphological opening. The morphological opening was to eliminate the noise around the plankton, which may cause an incorrect convex hull.

$$cr = \frac{\#\text{ pixels in the original image}}{\#\text{ pixels in the convex hull}} \tag{3.4}$$

3. Transparency ratio: This is the ratio between the area of the plankton image and the area of the plankton after filling all inside holes. The transparency ratio helps in recognizing the transparent plankton. We computed the transparency ratios with Eq. (3.5) for both the original image and the image after a morphological opening. The morphological closing was used to get rid of the noise inside the plankton body.

$$tr = \frac{\#\text{ pixels in the original image}}{\#\text{ pixels within the contour}} \tag{3.5}$$

4. Eigenvalue ratio: We first computed the covariance matrix between the x and y coordinates of all pixels on the plankton bodies. Then the ratio between the two eigenvalues from the covariance matrix was calculated in Eq. (3.6). This ratio helps classify elongated plankton.

$$er = \frac{min(f_1, f_2)}{max(f_1, f_2)} \tag{3.6}$$

where f_1, f_2 are eigenvalues of cov(X,Y).

5. Ratio between the head and the tail: Some plankton such as larvaceans have a large head relative to their tail. We computed the ratio between the head and tail to differentiate them. To do this we first rotated the image to make the axis with the bigger eigenvalue parallel to the x-axis. Assuming the smallest and largest x values are 0 and T respectively, we accumulated the number of foreground pixels along the x-axis from 0 to $\frac{1}{4}T$ and from $\frac{3}{4}T$ to T respectively. Then we computed the ratio between them as the ratio between the head and the tail.

3.4 Assigning probability values in support vector machines

A probability associated with a classifier is often very useful and it provides an indication of how much to believe the classification result. For example, the classifier could reject the example and leave it to an expert to classify it when the probability is very low. The classification probability will be used to develop an active learning strategy for a multi-class SVM in Chapter 4.

Platt [71] introduced the sigmoid function as the probability model to fit $P(y = 1|f)$ directly where f is the decision function of the binary SVM. The parametric model is shown

in Eq. (3.7).

$$P(y = 1|f) = \frac{1}{1 + exp(Af + B)} \tag{3.7}$$

where A and B are scalar values, which are fit with maximum likelihood estimation. Platt tested the model with 3 data sets including the UCI Adult data set and two other web classification data sets. The sigmoid-model SVM had good classification accuracy and probability quality in his experiments.

Hastie et al. [39] proposed a method to estimate classification probability for a series of pairwise classifiers. Given the estimated probability for each binary classifier (P_{pq}), the probability of being class p in a binary classifier (class p vs. class q), they minimize the average Kullback-Leibler distance between P_{pq} and $\frac{P(p)}{P(p)+P(q)}$, where $P(p)$ and $P(q)$ are the probabilities of a given example belong to class p and q, respectively. An iterated algorithm is given to search for $P(p)$. Following this line of the work, Wu [98] et al. develop two new criteria for the goodness of the estimated probabilities and applied their method to multi-class SVMs. Their approach has three steps to get the probability estimation. First, a grid-search is used to determine the best SVM parameters (C, g) based on k-fold cross validation accuracy. Second, with the optimal (C, g) found in the first step, A and B are fit individually for each binary SVM. Third, a constrained quadratic programming method is used to optimize the criteria they proposed.

However, this approach is time consuming. The second step involves estimating $N(N-1)$ parameters for SVMs using one-vs-one approach. The third step needs quadratic programming to solve N variables for each example. On a data set with m examples, this step needs to run m times. Another issue is that the SVM parameters (C, g) are estimated based on accuracy and thus might not be good for probability estimation in the following two steps.

In real-time plankton recognition, the probability computation needs to be fast since retraining the probability model is frequently needed as more plankton images are acquired on a cruise.

We developed a fast approximation method to compute the probability value while avoiding expensive parameter fitting. By normalizing the real valued output $f(x)$ from each binary SVM, our probability model assumes the same A for all binary SVMs. Also, our approach can optimize SVM parameters (C, g) together with probability parameter A simultaneously using a log-likelihood criterion.

1. We assume $P(y = 1|f = 0) = P(y = -1|f = 0) = 0.5$. It means that a point right on the decision boundary will have a 0.5 probability of belonging to each class. We eliminate B in this way.

2. Since each binary SVM has a different margin, a crucial criterion in assigning the probability, it is not fair to assign a probability without considering the margin. Therefore, the decision function $f(x)$ is normalized by its margin in each binary SVM. The probability model of SVMs is shown in (3.8) and (3.9). P_{pq} represents the probability output for the binary SVM on class p vs. class q, class p is +1 and class q is -1. We add a negative sign before A to ensure that A is positive.

$$P_{pq}(y = 1|f) = \frac{1}{1 + exp(\frac{-Af}{\|w\|})} \tag{3.8}$$

$$P_{pq}(y = -1|f) = 1 - P_{pq}(y = 1|f) = P_{qp}(y = 1|f) \tag{3.9}$$

3. Assuming $P_{pq}, q = 1, 2, ...$ are independent, the final probability for class p is computed as follows:

$$P(p) = \prod_{q}^{q \neq p} P_{pq}(y = 1|f) \tag{3.10}$$

   Normalize $P(p)$ to make $\sum_p P(p) = 1$.

4. Output $k = arg\ max_p P(p)$ as the prediction.

Although it is arguable whether P_{pq} and P_{pk} are really independent since P_{pq} and P_{pk} are both estimated using data from class p, the one-vs-one approach does not suffer from dependence very much because as a discriminative classifier, a binary SVM depends on the data from both positive examples and negative examples. For example, the support vectors from both sides determine a SVM' s decision boundary, which in turn affects the probability estimation. In the one-vs-one approach, there are no overlaps both in positive examples and negative examples at the same time between any pair of binary SVMs. Knowing there is only a weak dependence between P_{pq} and P_{pk}, Eq. (3.10) provides a reasonable approximation.

(A, C, g) are determined based on the cost function L from (3.11), where t_i is the true class label of x_i.

$$L = -\sum_i logP(t_i) \tag{3.11}$$

There are two ways of searching for the three parameters. The simple one is to search for (C, g) first based on k-fold cross validation, then search for A based on L in Eq. (3.11) using the (C, g) determined in the first step. This method can provide good classification accuracy since (C, g) are selected based on accuracy in the first step. We compare Platt's method and ours using this method.

Both Platt's approach for two-class problems (search for A and B) and our approach for multiple class problems (search for A) try to minimize the log-likelihood loss function as in Eq. (3.11).

Since the loss function is not convex, we used line search for a single parameter A to avoid local minima. We also compared it with gradient descent search for A and B as Platt proposed. The comparison will be detailed in Section 3.6.4.

After learning a SVM model and setting a rejection threshold p, we reject an example and leave it to be classified by a person if $P(k) < p$.

The other method is to employ a grid-search to find the optimal (C, g, A) simultaneously using the log-likelihood criterion in Eq. (3.11). This method will be explained in active learning in Chapter 4 where probability quality is very important.

Both methods can be run in parallel to achieve a big speedup. If we want to update the probability model after adding more labeled images, we can fix C and g, and only search for A. As a result, it is very fast to update the probability model. Moreover, normalizing f by its margin and assuming the same A for each binary SVM trades off some flexibility to gain a regularization effect and speedup since it restricts the otherwise big ($N(N+1)$) parameter space.

3.5 Feature selection

Feature selection helps reduce the feature computation time and increase the accuracy. There are two basic ways to do feature selection [24]. The filtering approach attempts to select a subset of features without applying learning algorithms. It is fast, but seems unlikely to result in the best accuracy. The wrapper approach [47] selects a feature subset by applying the learning algorithm. It has the potential to result in very good accuracy but is computationally expensive. A feature selection method specifically for SVMs was proposed recently. Weston [95] tried to minimize the generalization bound by minimizing the radius of the sphere including all the training examples. The drawback of this approach is that the generalization bound is loose, and minimizing the loose bound may not provide a feature subset with good accuracy.

In our system, we applied the wrapper approach with backward elimination. Backward elimination means one starts with all the features and systematically eliminates features. The average accuracy from a five-fold cross validation was used as an evaluation function. In our case, we start with 29 features and remove 1 feature from the feature set and get 29 different feature subsets with 28 features. We evaluate the 29 feature subsets by running 5-fold cross validation and choose the feature subset with best average accuracy to explore. For instance, if the feature subset with best average accuracy is M, we remove

1 more feature from M and get 28 feature subsets with 27 features to add to the remaining candidate feature subsets. In this way, we can use certain search strategies to explore those feature subsets and keep removing features.

The algorithm halts if there is no improvement in accuracy for p successive feature subsets explored. Best first search (BFS), which is embedded in the wrapper approach, is used to explore the feature subset space. However, it tends to stop with many features because BFS selects the most accurate nodes to explore and those nodes tend to have many features. In order to explore feature subsets with small numbers of features, greedy beam search (GBS) [38] was employed on the final feature subsets selected by BFS. GBS operates by only expanding the best q (beam width) leaf-nodes without any backtracking. It can quickly reduce the number of features to 1.

To reduce the effect of overfitting, we randomly chose 20 percent of the data as a held-out data set, and did the feature selection on the remaining data while testing the selected feature subsets on the held-out data. The feature selection procedure is described in the following.

Feature selection algorithm

1: N=$\{a_1,a_2,...,a_k\}$, S=$\{$N $\}$, T=$\emptyset$, q = constant, $max = 0$ where a_i is the ith feature, k is the number of features, S is a sorted list, max is the maximum accuracy, and q is the beam width.

2: Compute the average accuracy $f(N)$ from a 5-fold cross validation on the training data using all features. $max = f(N)$.

3: M=pop the set of features with best average accuracy off S, however randomly choose a feature subset candidate from S every 5 node expansions.

4: *if* $f(M) > max$ *then*

5: $max = f(M)$.

6: *else if max has not been changed in p expansions then*

7: *Go to 15.*

8: *endif*

9: *forall* $a_i \in M$ *do*

10: $M_i = M - a_i$.

11: *Run a 5-fold cross validation using the* M_i *subset of features and record the average accuracy* $f(M_i)$.

12: *Add* M_i *onto S, which is sorted in ascending order by* $f(M_i)$.

13: *end for*

14: *Go to 3.*

15: *Remove all the elements from S except for the five most accurate feature subsets.*

16: *if* $S = \emptyset$ *then*

17: *Stop.*

18: *endif*

19: *forall elements in S do*

20: *M=pop the next element off S.*

21: *forall* $a_i \in M$ *do*

22: $M_i = M - a_i$.

23: *Run a 5-fold cross validation using the* M_i *subset of features and record the average accuracy* $f(M_i)$.

24: *Add* M_i *and* $f(M_i)$ *onto T, which is a queue.*

25: *end for*

26: *end for*

27: *Pick the q most accurate* M_i *from T and add them to S.*

28: *Go to 16.*

After the selection algorithm, we acquired every B_t $(t = 1, 2, ..., 29)$, the best average accuracy in 5 fold cross validation with t features combination. Then we tested the B_t (the best combination of t features) on the held-out data set and selected the feature subset with the least number of features and good accuracy.

3.6 Experiments

Several experiments have been done to test our system. The Libsvm [19] support vector machine software was modified and used in our experiments. Libsvm applies sequential minimal optimization [70] in its optimization and a one-vs-one approach to do multi-class classification. We modified libsvm to produce a probabilistic output. For comparison, we also implemented a one-vs-all approach. In all experiments the gaussian radial basis function ($k(x, y) = exp(-g\|x - y\|^2)$) was used as the kernel. The parameters C and g were chosen by 5-fold cross validation using all the examples from each data set.

To evaluate the accuracy of SVMs, we compared with a cascade correlation neural network [33], a C4.5 decision tree with the default pruning settings [73], and two ensembles of decision trees: bagging unpruned decision trees [11] and random forests [12]. There were 100 trees built for each ensemble of decision trees.

3.6.1 Initial experiments

The first training set has a total of 1285 SIPPER images (50μm resolution), which were selected by marine scientists. It contains images of 64 diatoms, 100 protoctista, 321 doliolids, 366 larvaceans, and 434 Trichodesmium. We used C-SVM with parameters $C = 200$ and $g = 0.03$ for one-vs-one and $C = 64$ and $g = 0.08$ for one-vs-all. Table 3.3 shows the average accuracy of different learning algorithms from a 10-fold cross validation. A paired-t test was used to compare the results at the 95% confidence interval. The SVM one-vs-one approach is significantly more accurate than the other learning algorithms at the 95% confidence level. Also, the running time for one-vs-all and one-vs-one are 9 seconds and 2 seconds respectively on a Pentium 4 PC at 2.6 GHZ. Therefore, the SVM one-vs-one approach outperforms the one-vs-all approach both in accuracy and running time on this data set.

Table 3.4 shows the confusion matrix of the SVM one-vs-one approach from a 10-fold cross validation experiment. The overall average accuracy is 90.0%. While we have greater than 84% accuracy on most plankton, we only achieve 79% accuracy on the diatom class.

Table 3.3. 10-fold cross validation accuracy on the initial 1285 image set.

Classifiers	10-fold cross validation accuracy
C4.5 Decision tree	82.2%
Neural network	86.1%
Bagging	87.4%
Random forests	88.2%
SVM (one-vs-all)	86.5%
SVM (one-vs-one)	90.0%

Table 3.4. Confusion matrix of SVM (one-vs-one) from a 10-fold cross validation on 1285 SIPPER images with all 29 features. P, Di, Do, L and T represent Protoctista, Diatom, Doliolid, Larvacean and Trichodesmium respectively.

	as P	as Di	as Do	as L	as T
P	84.4%	1.6%	9.4%	4.7%	0.0%
Di	2.0%	79.0%	11.0%	6.0%	2.0%
Do	0.8%	0.3%	92.8%	3.1%	0.0%
L	0.8%	0.3%	4.4%	88.0%	6.6%
T	0.0%	0.5%	0.2%	6.2%	93.1%

The reason is that we only have 64 diatom samples in our training set and the SVM favors classes with more samples. For instance, assume there is an overlap in the feature space between two classes: one with many examples and one with few examples. It is likely that most examples within that overlap come from the class with more examples. To minimize the overall hinge loss described in 2.2.1, the decision boundary is pushed away from the class with more examples and thus will favor that class.

3.6.2 Experiments with unidentifiable particles

The second image set was collected from a deployment of SIPPER in the Gulf of Mexico. A set of 6000 images was selected from the five most abundant types of plankton, which account for 95% of the plankton samples in that run, and manually unrecognizable particles. The five types of plankton are copepods, doliolids, larvaceans, protoctista and Trichodesmium. The image quality in this training set is not as good as in the initial experiment. Apart from the shape of image objects, some prior knowledge was used by

Table 3.5. 10-fold cross validation accuracy on the 6000 image set.

Classifiers	10-fold cross validation accuracy
C4.5 Decision tree	64.1%
Neural network	70.4%
Bagging	74.2%
Random forest	74.5%
SVM (one-vs-all)	68.7%
SVM (one-vs-one)	75.1%

marine scientists to label the images. Also, we have to classify unidentifiable particles in this experiment.

There are a total of 6000 images: 1000 images of each plankton class and 1000 unidentifiable particles. We used C-SVM with $C = 200$ and $g = 0.032$ for one-vs-one and $C = 216$ and $g = 0.114$ for one-vs-all. Table 3.5 shows the average accuracy of different classifiers from 10-fold cross validation. A paired-t test was used to compare the results at the 95% confidence interval. The SVM one-vs-one approach is significantly more accurate than all other learning algorithms except the two ensembles of decision trees at the 95% confidence level. Also, the running time for one-vs-all and one-vs-one are 160 seconds and 610 seconds respectively on a Pentium 4 PC at 2.6 GHZ. Therefore, the SVM one-vs-one approach outperforms the one-vs-all approach both in accuracy and running time on this data set.

Table 3.6 shows the confusion matrix of the SVM one-vs-one approach from a 10-fold cross validation. The overall average accuracy is 75.12%. The average accuracy from the five types of plankton is 78.56%.

There are a significant number of larvaceans confused with Trichodesmium. This observation disagrees with the first experiment where we had high classification accuracy for both types of plankton. The reason is that some larvacean and Trichodesmium are linear objects. Domain experts have prior knowledge of the abundance of larvacean and Trichodesmium in some ocean areas. They labeled the linear objects as larvacean or Trichodesmium when they know the other plankton were less commonly found in the particular ocean areas examined. Therefore, there are many linear particles without significant fea-

Table 3.6. Confusion matrix of SVM (one-vs-one) from a 10-fold cross validation on 6000 SIPPER images with all 29 features. C, D, L, P, T, and U represent Copoped, Doliolid, Larvacean, Protoctista, Trichodesmium and Unidentifiable particles respectively.

	As C	As D	As L	As P	As T	As U
C	84.2%	0.6%	3.1%	1.0%	5.5%	5.6%
D	0.2%	82.9%	2.4%	8.7%	0.4%	5.4%
L	3.2%	1.9%	68.8%	1.4%	11.1%	13.6%
P	1.7%	5.3%	1.1%	84.4%	3.1%	4.4%
T	3.3%	0.6%	9.4%	1.8%	72.5%	12.4%
U	4.3%	3.1%	15.8%	5.4%	13.5%	57.9%

tures to differentiate between the two types of plankton in this training set, which result in lower classification accuracy on larvaceans and Trichodesmium.

It is clear that the one-vs-one approach is superior to the one-vs-all on the two data sets. Therefore, we choose to use one-vs-one approach in our system. We use the term SVMs to represent SVMs created with the one-vs-one approach by default in the rest of this section.

3.6.3 Feature selection

Feature selection was tested on the larger training set as described in Section 3.6.2. Although the single SVM seems superior to the other two single classifiers, there is no guarantee that it is still true after feature reduction. Therefore, we experimented with feature selection (wrapper approach) on the SVM and its direct competitor: the cascade correlation neural net. We did not use the decision tree in the comparison because it is far less accurate than the SVM on this data set, thus unlikely to be the best. We did not choose random forest because the ensembles of classifiers increase the complexity of the classifier while not resulting in a better accuracy.

The data set was randomly divided into two parts: 80% as training and 20% as validation. In this way, we have 1200 data as validation which makes the test result relatively stable and 80% data in training, which is likely to provide a similar feature subset to using

Table 3.7. Description of 15 selected feature subset.

Features	Number of original features	Number of selected features
Moment invariants of the original image	7	4
Moment invariants of the contour image	7	1
Granulometric features	7	4
Domain specific features	8	6

all the data. We set the stopping criterion p to be 150 and the beam width q as 5 in our experiment.

Figures 3.8 and 3.9 show the experimental results of the average accuracy from the 5-fold cross validation on the training data and the test accuracy on the validation data respectively. The SVM provided better accuracy than the neural net on both the training set and the validation set when the number of features was greater than 4. To choose the least number of features for the SVM, McNemar's test [25] was applied on the validation set to compute the 95% confidence interval. When the number of features was less than 15, the accuracy would be outside the confidence interval. Therefore, we chose the 15-feature subset as the optimal feature subset and it provided slightly better accuracy than using all the features on the validation data set.

Table 3.7 briefly describes the selected feature subset. A detailed description of the 15 selected features is is as follows.

1. Moment invariants of the original images: The first 4 Hu moments were selected.

2. Moment invariants of the coutour images: The first one of Hu moments was selected.

3. Granulometric features: Morphological openings with 3×3, 5×5, 7×7 square windows were selected. Also, a morphological closing with a 5×5 square window was selected. There are four granulometric features selected.

Table 3.8. Confusion matrix of SVM (one-vs-one) from a 10-fold cross validation on 6000 SIPPER images with the best 15-feature subset. C, D, L, P, T, and U represent Copoped, Doliolid, Larvacean, Protoctista, Trichodesmium and Unidentifiable particles respectively.

	As C	As D	As L	As P	As T	As U
C	84.5%	0.9%	3.1%	0.5%	5.6%	5.4%
D	0.7%	85.2%	1.1%	9.3%	0.4%	3.3%
L	4.3%	2.1%	67.2%	1.1%	12.5%	12.8%
P	1.8%	5.0%	0.7%	85.8%	3.0%	3.7%
T	4.5%	0.4%	10.0%	1.5%	72.5%	11.0%
U	5.1%	2.3%	15.6%	5.4%	13.4%	58.2%

4. Domain specific features: Among the domain specific features, the convex ratio and transparency ratio for images after morphological opening were eliminated. There are 6 domain specific features selected.

Only 1 moment invariant for contour images was selected. This was reasonable because the contours of the plankton images were not stable and hence the moment invariants for contour images were not very helpful in classification. Among the domain specific features, the convex ratio and transparency ratio for images after morphological opening were eliminated. They seem to be redundant for the same features computed on the original images. Therefore, our feature selection approach seems to eliminate irrelevant and redundant features on this image set.

To test the overall effect of feature selection, we applied 10-fold cross validation on the whole 6000 image set. The confusion matrix is shown as Table 3.8. The overall average accuracy is 75.57%. The average accuracy from the five types of plankton is 79.04%. Both indicate that the best 15-feature subset performs slightly better than all 29 features. It is certainly faster to compute the 15 features.

3.6.4 Probability assignment experiments

In this experiment, we compared our approach (line search for A) and Platt's approach extended to multiple classes (gradient descent search for A and B). We used the same training set as in the last experiment with the 15-feature subset. To reduce the overfitting

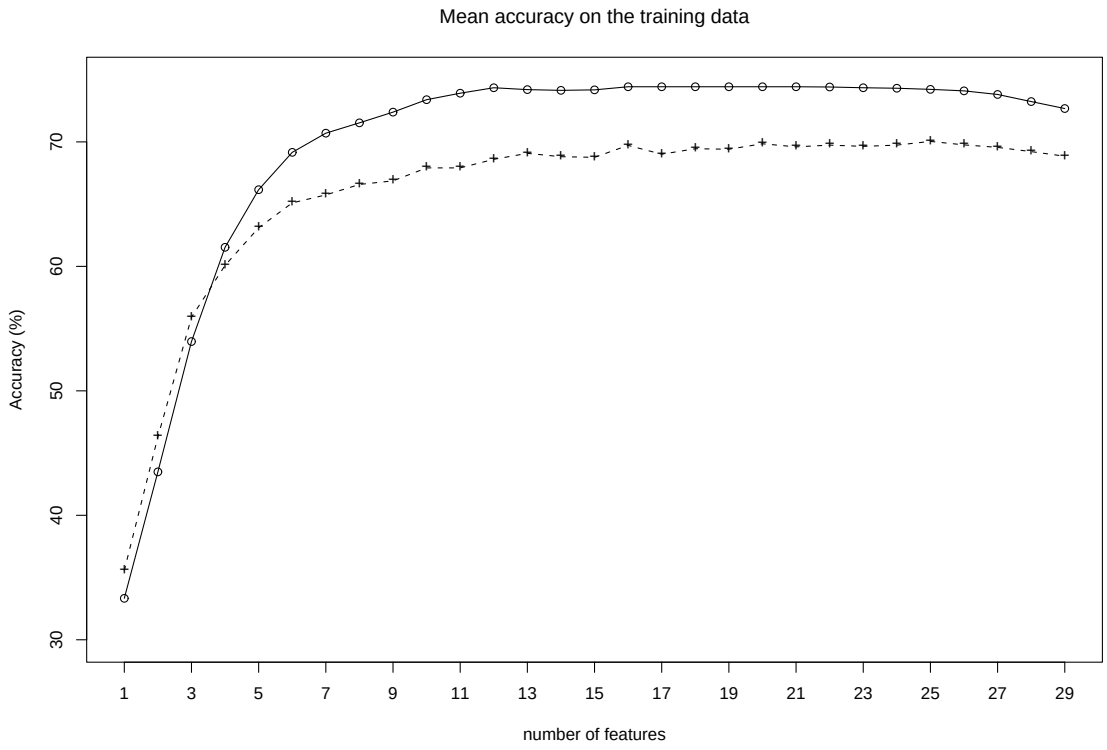

Figure 3.8. Feature selection on the training set: The solid line represents accuracy of the SVM and the dashed line represents the accuracy of the neural net.

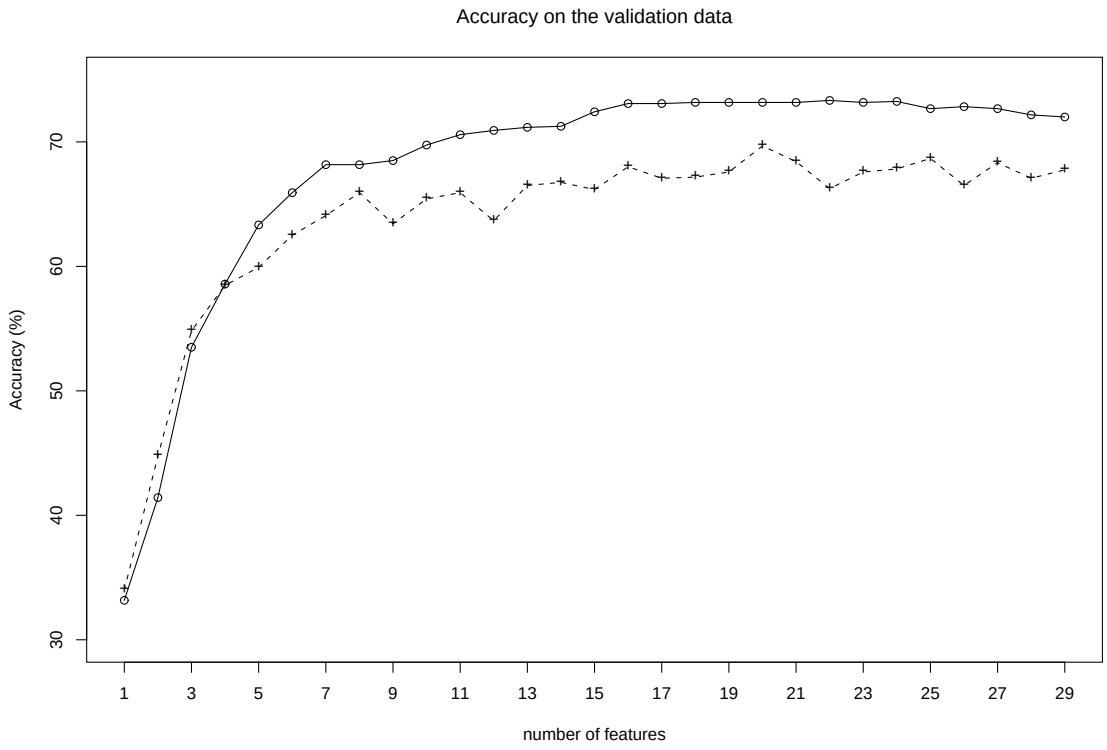

Figure 3.9. Selected feature subsets on the validation set: The solid line represents accuracy of the SVM and the dashed line represents the accuracy of the neural net.

Table 3.9. Best parameters for log-likelihood loss function.

	A	B	-log(L)
Line search for A	87.0	–	4516.5
Gradient descent search for A and B	71.0	0.412	4496.4

effect from parameter fitting, a 3-fold cross validation was applied to search for the best parameters in Platt's paper [71]. We used 3-fold cross validation for both approaches. Since gradient descent search for A and B is easily stuck in local minima, we varied the initialization several times to obtain the minimal loss. Table 3.9 describes the optimal parameters for both approaches. The gradient descent search provided parameters with smaller loss. The line search for a single parameter A is definitely faster than gradient descent search for A and B with different initializations.

To compare the different parameter sets, we drew a rejection curve from 10-fold cross validation using the best parameters for both approaches. The points on the rejection curve were sampled by varying the rejection threshold p, whose range is between 0 and 1. Figure 3.10 shows our approach is at least as good as the MLE of A and B. It indicates that $B = 0$ is a reasonable assumption, at least for our data set.

3.7 Conclusions

This chapter presents a plankton recognition system for binary SIPPER images. General features as well as domain specific features were extracted and a support vector machine was used to classify examples. We also developed a way to assign a probability value after the multi-class SVM classification. We tested our system on two different data sets. The recognition rate exceeded 90% in one experiment and was over 75% on the more challenging data set with unidentifiable particles. A SVM was more accurate than a C4.5 decision tree [73] and a cascade correlation neural network [33] at the 95% confidence level on the two data sets. The single SVM was significantly more accurate than bagging [11] applied to decision trees and random forests [12] on the smaller data set and was insignificantly more accurate on the larger data set. The wrapper approach with backward elimination

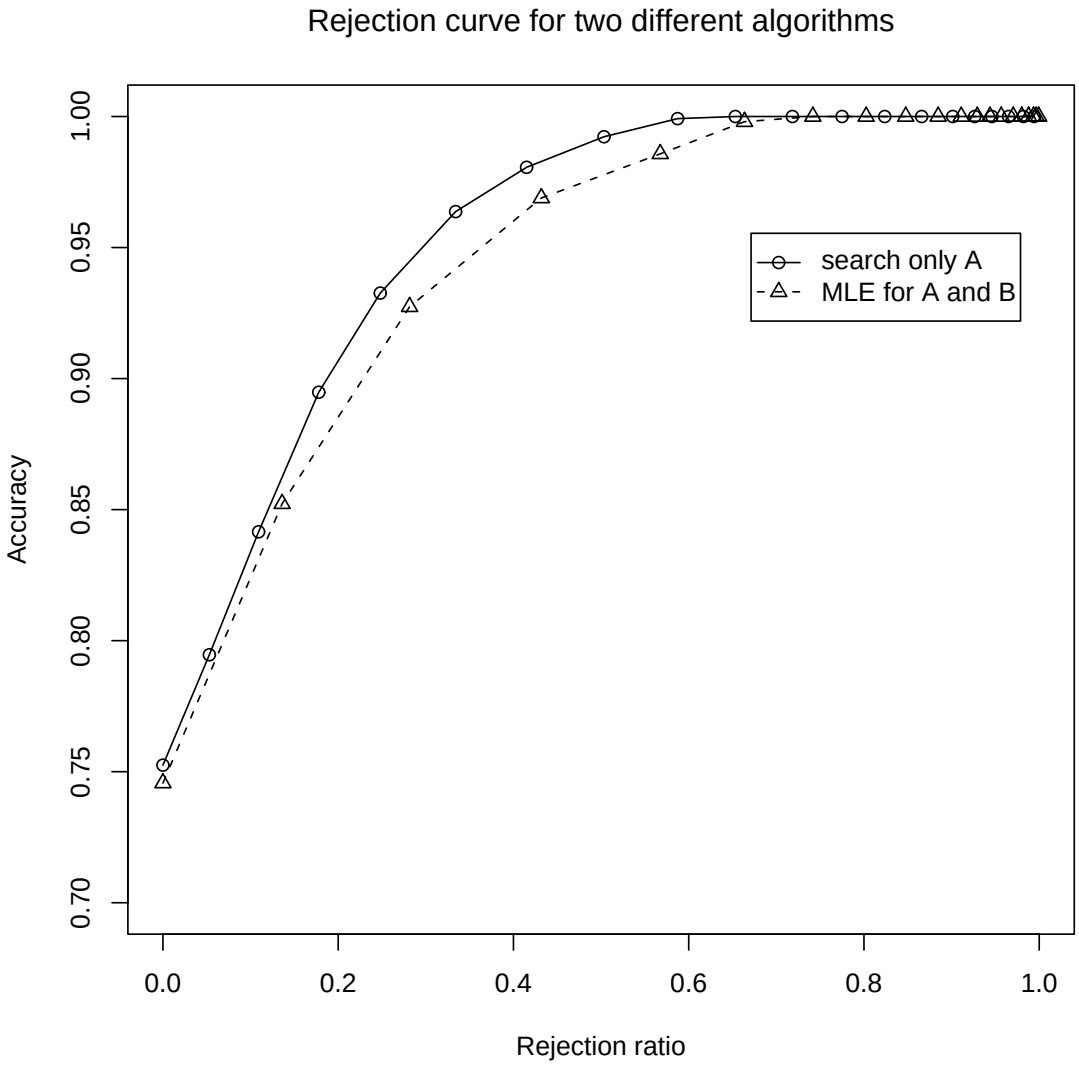

Figure 3.10. Rejection curve for both approaches-Overall accuracy vs. rejection rate.

successfully reduced the number of features from 29 to 15 and allowed a classifier to be built with slightly better accuracy than using all the features. Our probability model for multiple class SVMs provided a reasonable rejection curve.

CHAPTER 4

ACTIVE LEARNING TO RECOGNIZE MULTIPLE TYPES OF PLANKTON

This chapter presents an active learning method to reduce domain experts' labeling efforts in applying support vector machines to recognize underwater zooplankton from higher-resolution, new generation SIPPER II images. Most of the previous work on active learning with support vector machines only deals with two class problems. In this chapter, we propose an active learning approach "Breaking Ties" [48] for multi-class support vector machines using the one-vs-one approach with a probability approximation. Experimental results indicate that our approach often requires significantly less labeled images to reach a given accuracy than the least certainty active learning method and random sampling. It can also run in batch mode with an accuracy comparable to labeling one image at a time and retraining.

4.1 Introduction

Recently, an advanced shadow image particle profiling evaluation recorder (SIPPER II) has been developed to produce 3-bit grayscale images at 25 μm resolution. SIPPER II uses high-speed digital line-scan cameras to continuously sample plankton and suspended particles in the ocean. The high sampling rate of SIPPER II makes it necessary to develop an automated plankton recognition system. For example, a previous study using approximately 150,000 SIPPER images from a two hour sampling deployment took over one month to manually classify [75]. Also, this automated system is expected to continuously evolve from a previous model to a more accurate model created by training after adding some

new labeled images into the training set. Since it is impossible to manually label all the new images during the time they are acquired on the ship, active learning seems attractive.

Recently, active learning with SVMs has been developed and applied to a variety of applications [89][81][17][80][94][14] [93][62][2][52][61][66][58][59]. We review the most representative and relevant work as follows.

Tong and Koller [89], Schohn and Cohn [81], and Campbell et al. [17] independently developed a similar active learning approach for support vector machines (SVMs) in two class problems. Their approach, which we call "simple", labeled the new examples closest to the decision boundary. Tong and Koller [89] used version spaces to analyze the hypotheses space of SVMs. They indicated that "simple" approximately found the examples which most dramatically reduced the version space. Compared to random sampling, "SIMPLE" reduced the number of labeled images in their experiments on text classification. Mitra et al. [58] argued the greedy search method employed in "simple" is not robust and proposed a confidence factor to measure the closeness of the current SVM to the optimal SVM. A random sampling factor was introduced when the confidence factor was low. Their proposed method performed better than "simple" in their experiments.

Roy and McCallum [77] used a different strategy to select a candidate example to label. Based on the probability model, they labeled examples which could maximize the posterior entropy on the unlabeled data set. This approach is called "CONF". "CONF" amounts to improving the current classifier' s classification confidence on the unlabeled data set. Although it initially was applied with naive bayes classifiers, it could be easily extended to any classifier with probability outputs. For example, the probability outputs of SVMs can be roughly approximated by a sigmoid function [71].

Baram et al. [2] observed there was no single winner from different active learning strategies on several data sets. They proposed to dynamically select from four learning algorithms: "SIMPLE", "CONF", random sampling and sampling examples furthest from the current labeled data set. The automatic selection was done by solving a multi-armed bandit problem through online learning.

Brinker [14] and Park [66] independently proposed a similar selection method "combined" named by Brinker [14] to label several examples at a time for two-class problems. Based on "SIMPLE", it chose to label examples which are close to the decision boundary and whose feature vectors have large angles to the previous selected candidates. A parameter λ was introduced to control the trade-off between the two criteria. Although Brinker did not give a way to set the optimal value of λ, "combined" performed better than "SIMPLE" in batch mode, namely labeling several images at a time, on several data sets.

Two things in our work make it different from previous approaches. The images sampled from first generation SIPPER (SIPPER I) did not have clear contours. The low image quality resulted in many unidentifiable particles, which made it important to create robust image features and handle unidentifiable particles [53]. Higher resolution SIPPER (SIPPER II) images provide relatively better quality images with clear contours. Also, 3-bit graylevel images have more texture information than binary images. As a result, there was no longer an issue in handling many unidentifiable particles. Therefore, new contour features and texture features are needed to help recognition. Also, little previous work in active learning has been done with multiple class SVMs, which is required in plankton recognition. For instance, SVMs solve multiple class problems by building several two-class SVMs. A new example usually has different distances to the decision boundaries in each of the two-class SVMs. It is hard to apply the "SIMPLE" approach because we do not know which distance to choose. A very recent paper [59] simply applied "simple" to each binary SVM in a multi-class SVM. For a multi-class problem with N binary SVMs, N examples were labeled at a time. However, this method is far from elegant. They did not provide a way to judge which example is best for all binary SVMs. It is not unusual that an "informative" example for one binary SVM is useless for other binary SVMs. The "combined" method suffers from the same problem. It does not know which distance to minimize and which angle to maximize. "CONF" seems to be a natural solution for multi-class problems as long as we have a probability estimation for the output from a multi-class SVM.

However, applying the "CONF" approach involves estimating the decision boundary after adding each unlabeled example into the training data in each round. Suppose m is the number of unlabeled examples and c is the number of classes, "CONF" needs to train a SVM cm times to decide the next example to label. Although there are several heuristics to speedup such a procedure, it is still extremely computationally expensive.

In this chapter, we develop a new image feature set [49], which adds some contour features and texture features into the previous feature set in [53]. We also propose a new active learning strategy for one-versus-one multi-class SVMs and compare with a least certainty method in [52]. After developing a probability model for multiple class SVMs as described in [53] and Chapter 3, we label the example for which the difference in probabilities between its most likely class and second most likely class is smallest. We compare our approach with other methods like random sampling and least certainty for the plankton recognition problem. To obtain the same classification accuracy, our approach required many fewer labeled examples than random sampling. It also outperformed the least certainty approach in terms of needed examples to reach a given accuracy level. Our proposed method can run in batch mode, labeling up to 20 images at a time, with an accuracy comparable to labeling one image at a time and retraining.

This chapter is organized as follows. Section 4.2 describes the feature computation of grayscale SIPPER images. Section 4.3 introduces our active learning approach for multi-class support vector machines using the probability model developed in Chapter 3. Experimental results for the system are presented in Section 4.4. Finally we summarize our work and propose some ideas for future work in Section 4.5.

4.2 Feature computation

The advanced SIPPER (SIPPER II) made improvements in both resolution and grayscale values over the last generation SIPPER (SIPPER I). The resolution went from 2048 to 4096 pixels per scan line and grayscale values from 1 bit to 3 bits. The higher resolution resulted in perceptually superior defined images with clear contours. The 3 bit grayscale value al-

lows for 8 levels of grayscale which gives the images texture that was lacking before. As a result of these improvements the images are far easier to identify for marine scientists. Given the new and improved data, 20 new features were created and added into the feature set for SIPPER I. There were four new groups of features created, 8 weighted moments, 5 contour, and 5 texture features. There were also two other features created: weighted size and weighted size divided by convex area. The 28 features for SIPPER I included invariant moments, granulometric features, size, convex ratio, transparency ratio, and eigen ratio. They were described in detail in Chapter 3. In this chapter we only present the 20 new image features for SIPPER II.

For purposes of displaying images, the 3-bit grayscale images are re-scaled to 8 bits. Figures 4.1 to 4.5 are typical examples of the images produced by SIPPER II.

Figure 4.1. Calanoid copepod in SIPPER II images.

Figure 4.2. Larvacean in SIPPER II images.

Figure 4.3. MarineSnow in SIPPER II images.

Figure 4.4. Oithona in SIPPER II images.

Figure 4.5. Trichodesmium in SIPPER II images.

4.2.1 Weighted moment features

The weighted moments are the same as those originally developed for the binary data supplied by SIPPER I with the exception that the calculations are weighted by the grayscale intensity value.

$(\bar{x}, \bar{y})$ is the weighted center of the foreground pixels in the image. The $(p+q)$-order weighted central moments $\mu(p,q)$ are computed with every foreground pixel at (x,y):

$$\bar{x} = \frac{\sum_{x=1}^{H}\sum_{y=1}^{W}\frac{I(x,y)}{255}x}{\sum_{x=1}^{H}\sum_{y=1}^{W}\frac{I(x,y)}{255}} \tag{4.1}$$

$$\bar{y} = \frac{\sum_{x=1}^{H}\sum_{y=1}^{W}\frac{I(x,y)}{255}y}{\sum_{x=1}^{H}\sum_{y=1}^{W}\frac{I(x,y)}{255}} \tag{4.2}$$

$$\mu(p,q) = \sum_{x=1}^{H}\sum_{y=1}^{W}(x-\bar{x})^p(y-\bar{y})^q\frac{I(x,y)}{255} \tag{4.3}$$

where $I(x,y)$ is the intensity value at (x,y), W and H are the width and the height of a image respectively.

Hu [41] introduced a way to compute the seven lower order moment invariants based on several nonlinear combinations of the central moments. Using the normalized central moments, we get scale, rotation and translation invariant features. We computed the weighted moments in the same way as described in Chapter 3.3.2.

The original moments alone would give us 56.98% accuracy with a 10 fold cross validation on SIPPER II images. Utilizing weighted moments we were able to get 59.16% accuracy.

4.2.2 Contour features

There are five contour features produced. They are derived from a 1-d Fourier transform of the contour points plotted as complex numbers. The array that results is then divided into five frequency ranges where the average magnitude value of each range is calculated. The frequency range for each pixel in the resultant 1D array is determined by computing its distance from the center of the array. Table 4.1 and Figure 4.6 show the range for each feature.

Table 4.1. The upper and lower boundary regions as a fraction of one half edge length.

Region #	Lower bound(LB)	Upper bound(UB)
1	0	1/2
2	1/2	3/4
3	3/4	7/8
4	7/8	15/16
5	15/16	1

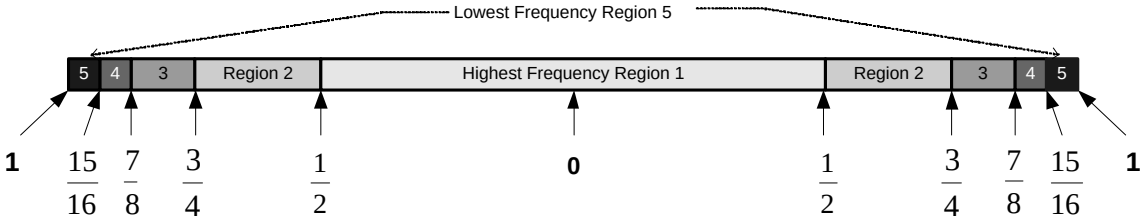

Figure 4.6. Contour frequency domains.

The Fourier descriptor $CVF[r]$, $r = 1, ..., 5$ is computed in Eq. (4.4).

$$CVF[r] = \frac{\sum_{x=1}^{L} F(x)R(x,r)}{PC(r)} \tag{4.4}$$

where L is the length of the contour in pixels, $F(x)$ is magnitude of the complex numbers at position x, $R(x, r)$ is a indicator function which specifies whether edge pixel x is in region r, and $PC(r)$ is the number of pixels in region r.

4.2.3 Texture

With the grayscale values that SIPPER II produces, features that reflect the texture of the image can be computed. A 2D Fourier Transform is performed on the original image. By using the result of this transform the energy of different frequency ranges is captured by computing the average magnitude for each frequency range.

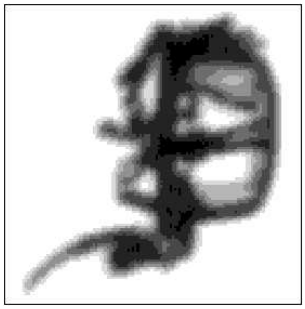

Figure 4.7. Source image.

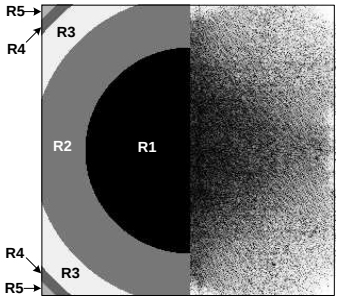

Figure 4.8. The five frequency regions (r1 through r5) of the image.

Figures 4.7 and 4.8 show a typical plankton image and its Fourier transform. In Figure 4.8, the right half of the image represents the amplitude of the Fourier transform; the arcs in the left half indicate the boundaries of the regions. We only process half the Fourier domain since both halves of the Fourier magnitude are mirror images of each other. These five regions result in five Fourier features. The value of each feature is the average value of the Fourier amplitude of their respective region. The Fourier texture features ($TVF[r]$, $r = 1, ..., 5$) are calculated by Eq. (4.5).

$$TVF[r] = \frac{\sum_{x=1}^{H} \sum_{y=1}^{W} J(x, y) R(x, y, r)}{PC(r)} \tag{4.5}$$

Table 4.2. Inner and outer region boundaries.

Region #	Lower bound(LB)	Upper bound(UB)
1	0	1/2
2	1/2	3/4
3	3/4	9/10
4	9/10	19/20
5	19/20	1

where $PC(r)$ is the number of pixels in region r, $R(x, y, r)$ is the indicator function of whether pixel at (x, y) is in region r, J is the 2-D Fourier transformation of the image.

These five features alone enable a 58.34% accuracy to be obtained on a 10 fold cross validation on SIPPER images.

4.2.4 Other features

There were two other features developed: weighted size and weighted convex ratio. The weighted size is meant to reflect not just the size of the image in pixels but also the density of the image as indicated by each pixel's intensity value. Each pixel in the image will be assigned a value in the range of 0.0 to 1.0 (background to foreground).

$$\text{Weighted Size} = \sum_{x=1}^{H}(\sum_{y=1}^{W}(\frac{I(x,y)}{255})) \tag{4.6}$$

The Weighted convex ratio is computed as follows:

$$WCR = \frac{\text{weighted Size}}{\text{convex hull area}} \tag{4.7}$$

4.3 Active learning approach with multi-class support vector machines

In [52], the least certainty active learning approach, which makes use of the estimated probability described in the last subsection, provides good performance in multi-class SVM classification. The idea can be traced back to [50], which uses "uncertainty sampling"

to label the examples with the least classification certainty. We call the least certainty approach in [52] "LC". In this chapter, we propose another active learning approach– "Breaking Ties" (BT). The idea of "BT" is to improve the confidence of the multi-class classification. Recall in a multi-class SVM with probability outputs, we assign the class label of x to $arg\ max_p P(p)$. Suppose $P(a)$ is the largest and $P(b)$ is the second largest probability for example x, where a, b are class labels. "BT" tries to improve the $P(a)-P(b)$. Intuitively, improving the value of $P(a) - P(b)$ amounts to breaking the tie between $P(a)$ and $P(b)$, thus improving the classification confidence. The difference between "LC" and "BT" is that "LC" tries to improve the value of $P(a)$ instead of $P(a) - P(b)$.

The two algorithms work as follows:

1. Start with an initial training set and an unclassified set of images.
2. A multi-class support vector machine is built using the current training set.
3. Compute the probabilistic outputs of the classification results for each image on the unclassified set. Suppose the class with highest probability is a and the class with second highest probability is b. Record the value of $P(a)$ and $P(b)$ for each unclassified image.
4. If LC: Remove the image(s) from the unclassified set that have the smallest classification confidence, obtain the label from human experts and add them to the current training set.
5. If BT: Remove the image(s) from the unclassified set that have the smallest value of $P(a) - P(b)$, obtain the label from human experts and add them to the current training set.
6. Go to 2.

4.4 Experiments

There were 8440 plankton images selected from the five most abundant types of plankton: 1688 images from each type of plankton. 1000 images (200 each type of plankton) were randomly selected as the validation set used in the active learning experiments.

The Libsvm [19] support vector machine software was modified to produce probabilistic outputs. In [76] it was argued the one-vs-all approach was essentially as good as other voting algorithms, however, without postprocessing binary SVMs, we observed the one-vs-one approach provided better accuracy and less training time than the one-vs-all approach in our previous experiments. Also, given N classes, updating models with several more labeled examples, the one-vs-one approach only needs to update N binary SVMs built with a portion of the data, while the one-vs-all approach needs to update N binary SVMs built with all the labeled data. Therefore, the one-vs-one approach was used in our experiments. In all experiments the Gaussian radial basis function (RBF) was used as the kernel: $k(x, y) = exp(-g\|x - y\|^2)$ where g is a scalar value.

The optimal feature subset was determined beforehand by the Wrapper approach with backward elimination. This feature selection method has been described in [53] and Chapter 3. With the best (g, C) parameters found by 5-fold cross validation, we applied the wrapper approach to feature selection. 80% of the images were used as training data and 20% of the image were held out as validation. 5-fold cross validation was used to select the best feature subset for each number of features. Then the best feature subsets were tested on the validation set. As a result, 17 out of 49 features were selected with slightly better 10-fold cross validation accuracy than using all 49 features. In all the active learning experiments, we used the best 17 feature subset instead of the 49 feature set. See [49] for more details about the selected features.

When active learning was applied in our system, we only had some initial training data available. Therefore, the best parameter set for the probability model should be estimated from a small data set. The parameters (g, C, A) were optimized by performing a grid-search across a random selected 1000 images consisting of 200 images per class. We believe

such optimal parameters were built from a relatively small set of data and reasonably stable. A five-fold cross validation was used to evaluate each combination of parameters based on the loss function L from (3.11). The parameters (g, C, A) were varied with a certain interval in the grid space. Since the parameters are independent, the grid-search ran very fast in parallel. The values of $g = 0.04096$, $C = 16$, and A =100 were found to produce the best results.

We did a series of retrainings for the two active learning methods and random sampling on the training data with N randomly selected images per class as the initial training set. Each experiment was performed 30 times and the average statistics were recorded. Instead of exhausting all of the unlabeled data set, we only labeled 750 more images for each experiment because exhausting all unlabeled data was not a fair criterion for comparing between different sample selection algorithms. For example, active learning labeled the most "informative" new examples, which were available in the beginning of the experiment. As more "informative" examples were labeled, only "garbage" examples were left unlabeled in the late stages of the experiment. The term "garbage" examples here means the examples correctly classified by the current classifier and far from the decision boundary. Therefore, "garbage" examples have no contribution to improving the current classifier. In contrast to active learning, random sampling labeled average "informative" examples throughout the whole experiment. It surely would catch up with active learning in the later stages when active learning only had "garbage" examples to label. Moreover, when the plankton recognition system is employed on a cruise, the unlabeled images come like a stream. The nature of such application prevents one from exhausting all the unlabeled images because of the prohibitive labeling work. Therefore, it makes more sense to compare different algorithms in the early stage of the experiment when the unlabeled data set is not exhausted. To see the upper limit of classification accuracy, we built a SVM using all 7400 training images. Its prediction accuracy was 88.3% on the 1000 held-out data set. Figures 4.9(a)–4.9(e) are several misclassified images.

(a) Calanoid copepod misclassified as oithona

(b) Larvacean misclassified as trichodesmium

(c) Marine snow misclassified as larvacean

(d) Oithona misclassified as calanoid copepod

(e) Trichodesmium misclassified as larvacean

Figure 4.9. Some misclassified images.

Several variations of the procedure described above were performed. We varied the number of initial labeled images per class (IIPC) to result in initial classifiers with different accuracy. In this way, we could test active learning when the accuracy of the initial model varied. We also changed the number of images selected for labeling at each retraining step (IPR) to test how well active learning works in batch mode.

4.4.1 Experiments with IPR=1, IIPC varied

Figures 4.10–4.14 show the experimental results of active learning methods on different IIPC values. A paired-t test was used to determine if there was a statistically significant difference between approaches. We used standard error as the error bar because the denominator of t test is in the form of standard error.

As shown in Figure 4.10, with only 10 images per class in the initial training sets we started off with rather poor accuracy (64.6%). At p=0.05, "BT" is statistically significantly more accurate than "LC" and both active learning methods are statistically significantly more accurate than random sampling. At 81% accuracy, random selection required approximately 1.7 times the number of newly labeled images in "BT". Figure 4.11 contains the first five images labeled by "BT" for one of the 30 runs. They seem to be relatively "hard" images.

Active learning is designed to label the most "informative" new images, thus helping improve the classifier. In SVMs, the decision boundary is represented by support vectors (SVs). Therefore, an effective active learning method is likely to find more SVs than random sampling. Figure 4.10 also shows the average number of SVs versus the number of images added into initial training set from the 30 runs. Active learning resulted in many more SVs than random sampling. Also, the slope of both active learning curves are about 0.9, which means that 90% of the labeled images turn out to be SVs. Our active learning approach efficiently captured support vectors. We note that a high slope of the support vector curve is not a sufficient condition for effective active learning because there are many SVs to be added into the current model and different SVs lead to different improvements.

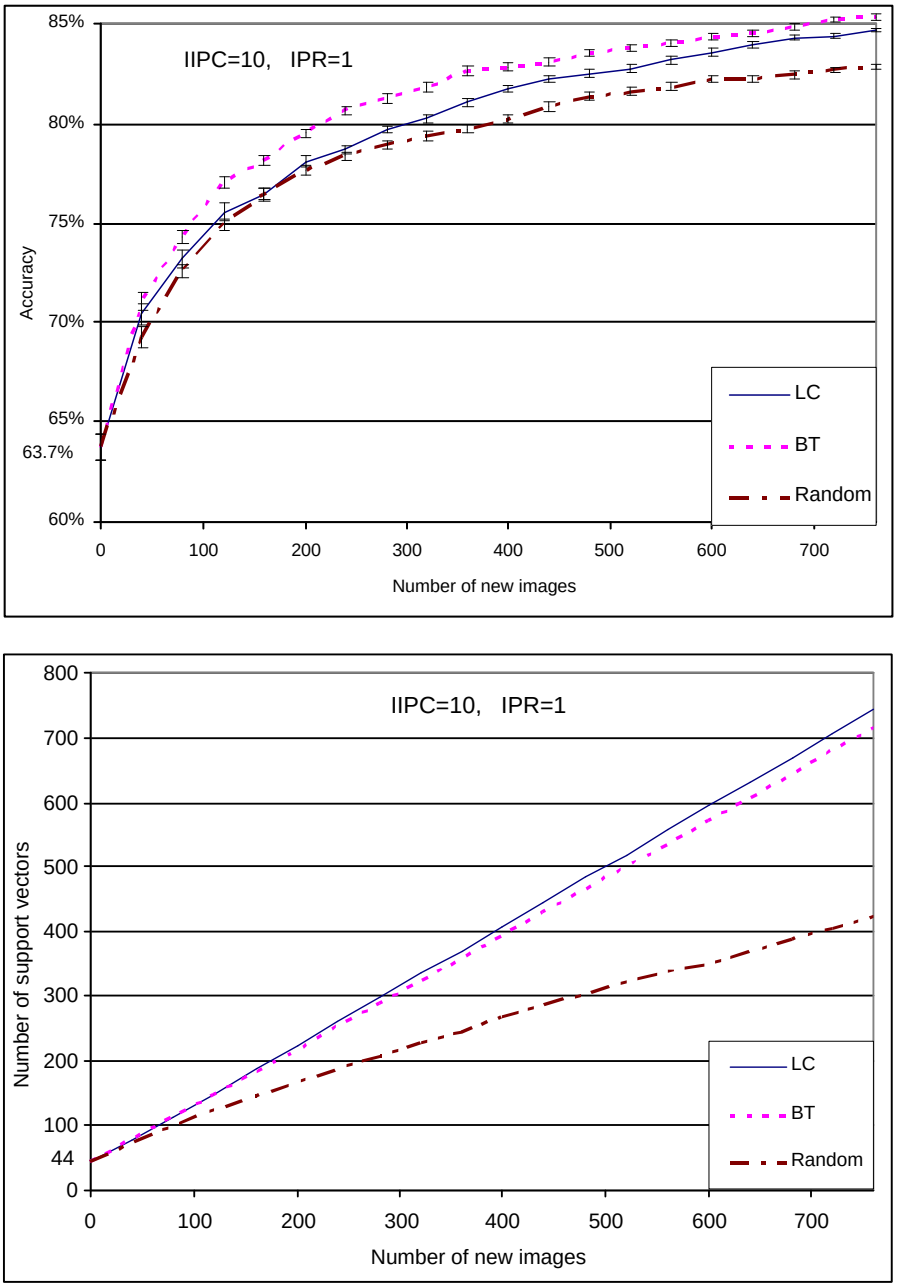

Figure 4.10. Comparison of active learning and random sampling in terms of accuracy and number of support vectors: initial training images per class are 10, one new labeled image added at a time. The error bars represent the standard errors.

Figure 4.11. The first five images labeled by BT in one run. The true class labels of the images from left to right are oithona, calanoid copepod, oithona, larvacean and larvacean

Ideally, a very effective active learning method should find the SVs which can improve the current model the most. In contrast, an active learning method, which always finds the SVs misclassified by the current classifier and far from its decision boundary, may perform very poorly because such SVs are very likely to be noise. Therefore, we cannot compare active learning methods only based on slight differences in the support vector curve.

With 50 IIPC in the initial training set shown in Figure 4.12, we started with 77% accuracy. As compared with 10 IIPC, the accuracy for both active learning approaches improved faster than random sampling. At the 81% accuracy level, random sampling required about 2.5 times and 1.7 times the number of images compared with using "BT" and "LC", respectively. The slopes of support vector curves for active learning are higher than those of random sampling. Also, "BT" outperformed "LC", however, it is not as obvious as with IIPC=10.

In Figures 4.13 and 4.14, we started with more than 80% accuracy using 100 and 200 initial images from each class, and active learning was very effective. Random sampling required more than 3 times the number of images to reach the same level of accuracy as both active learning approaches. The two active learning methods effectively capture many more SVs than random sampling. Also, our newly proposed active learning approach, "BT", requires less images to reach a given accuracy than "LC" after adding 450 labeled images. Before adding 450 labeled images, however, "BT" performs similarly to "LC".

It makes sense that the accuracy of the initial classifier affects the performance of active learning and random sampling. Active learning greedily chooses the most "informative" examples based on the previous model. So a bad model may mislead the active learning approach to choose non-informative examples, which do not help to improve the classifier. While random sampling provides the classifier with average "informative" examples what-

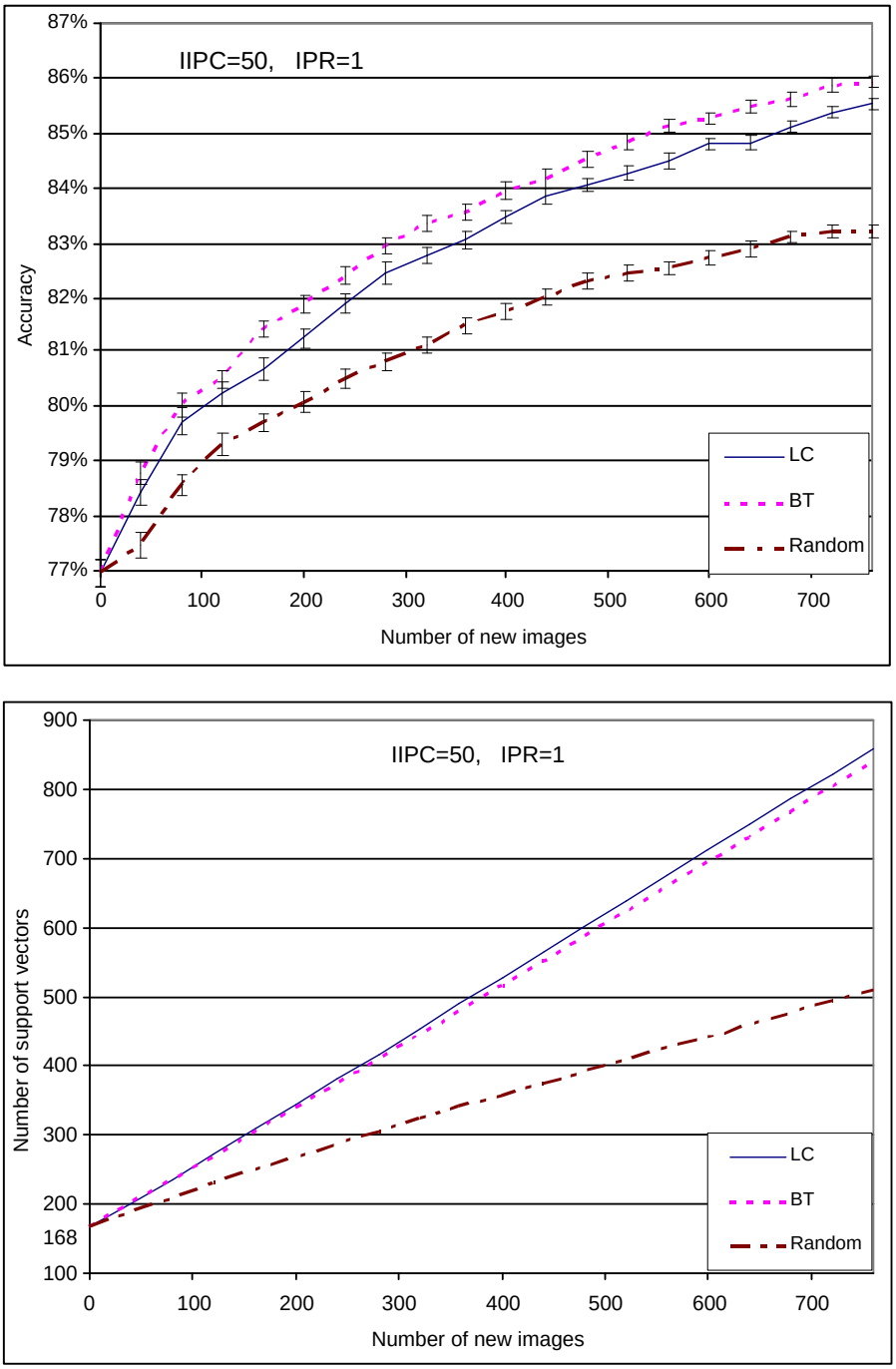

Figure 4.12. Comparison of active learning and random sampling in terms of accuracy and number of support vectors: initial training images per class are 50, one new labeled image added at a time. The error bars represent the standard errors.

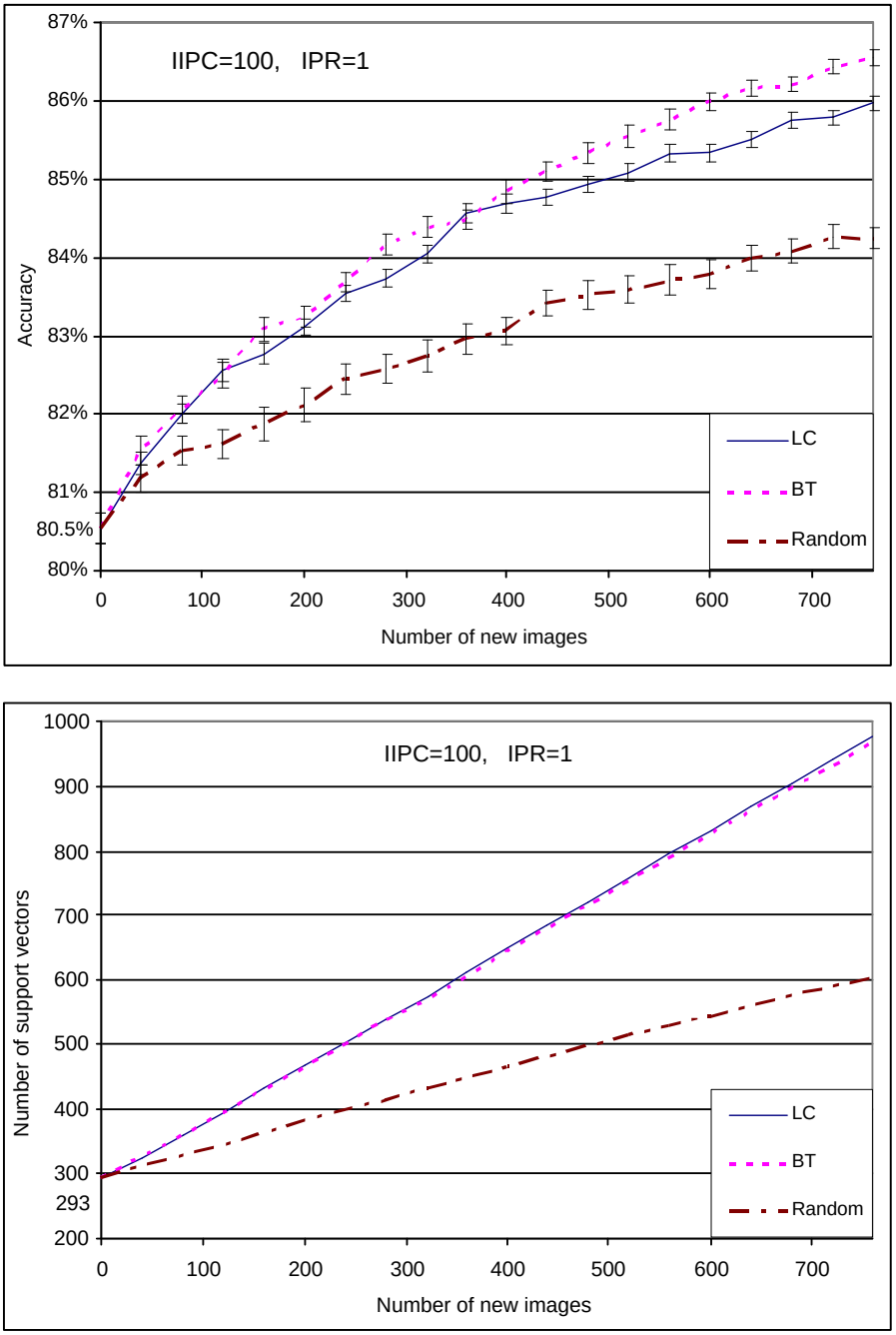

Figure 4.13. Comparison of active learning and random sampling in terms of accuracy and number of support vectors: initial training images per class are 100, one new labeled image added at a time. The error bars represent the standard errors.

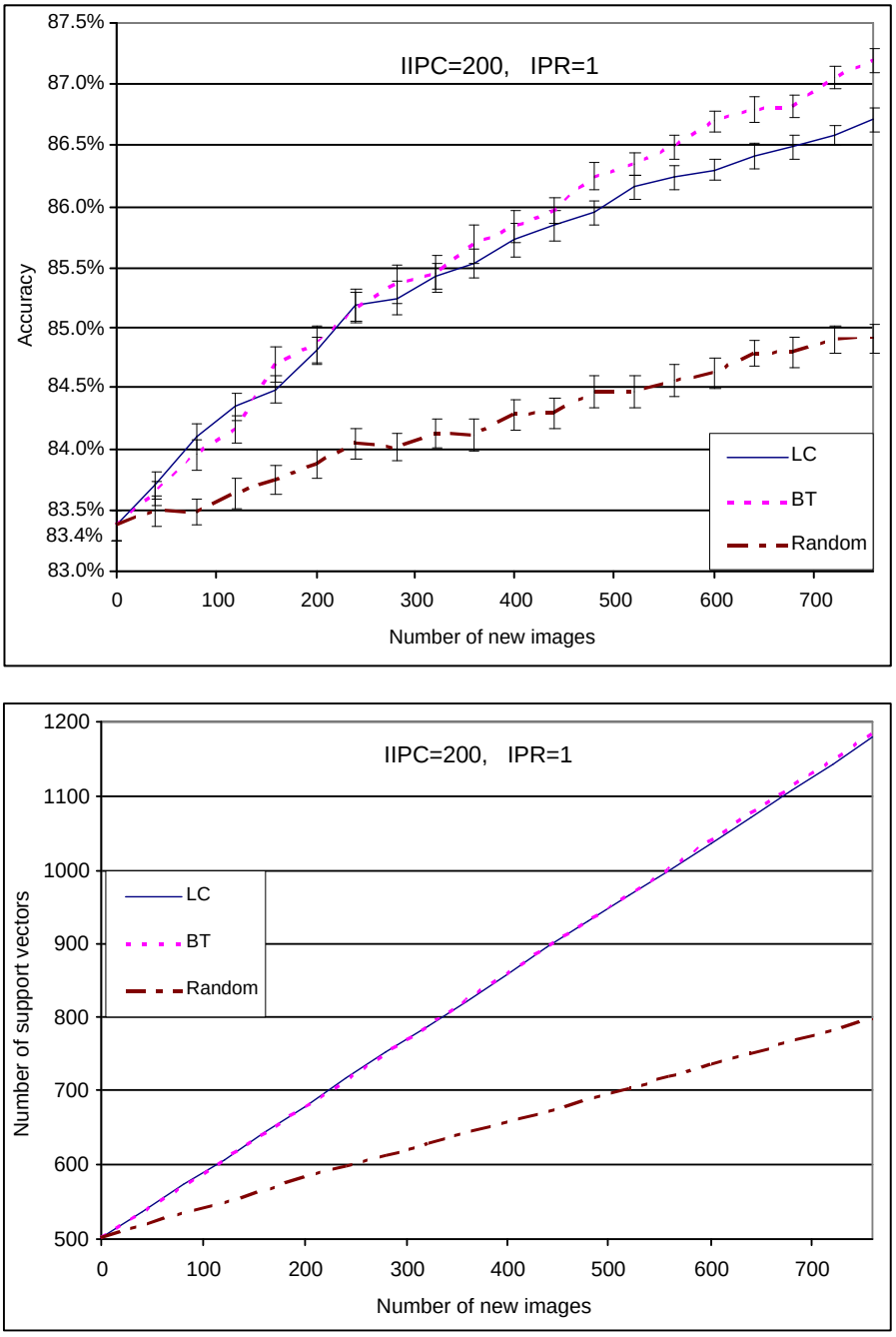

Figure 4.14. Comparison of active learning and random sampling in terms of accuracy and number of support vectors: initial training images per class are 200, one new labeled image added at a time. The error bars represent the standard errors.

ever the initial classifier is. Therefore, if the initial classifier helps active learning to choose examples more informative than average (random sampling), active learning will result in a more accurate classifier with fewer labeled examples. The better the initial classifier, the more labeling effort is saved.

When comparing the two active learning methods, "BT" outperformed "LC" under all four starting conditions. However, the difference in accuracy between them was insignificant as the initial classifier became more accurate. The justification is an accurate initial classifier has less space for improvement using active learning. "BT" improved the accuracy by more than 20% when IIPC=10 while it only boosted the accuracy by less than 4% when IIPC=200. Therefore, as the scale of the accuracy improvement was small, the difference in accuracy between the two active learning methods became insignificant.

4.4.2 Varying the IPR

It is usually expected that more than one image is labeled and added into the training set for retraining. For instance, it is convenient for an expert to label several images instead of one at a time. Also, given the total number of newly labeled images is U, it is approximately k times faster if we label k images at a time because it requires only $\frac{U}{k}$ times model updating. Although an incremental SVM training algorithm was proposed in [18] to reduce the retraining time, it is still very slow for model updating to label one image at a time especially when many images are to be labeled. Therefore, we expect active learning to be effective even when adding several labeled images at a time.

The active learning method "BT" is good for adding only one "informative" example at a time, there is no guarantee that adding several examples at a time will still favor "BT". The reason is that adding one "informative" example will update the model, which in turn changes the criterion for the next "informative" example. Therefore, the most "informative" example set is different from simply grouping several most "informative" examples together. However, such an optimal example set is very hard to compute. Therefore, we ex-

pect grouping several most "informative" examples together is a reasonable approximation of the optimal example set, or at least is superior to randomly sampling several examples.

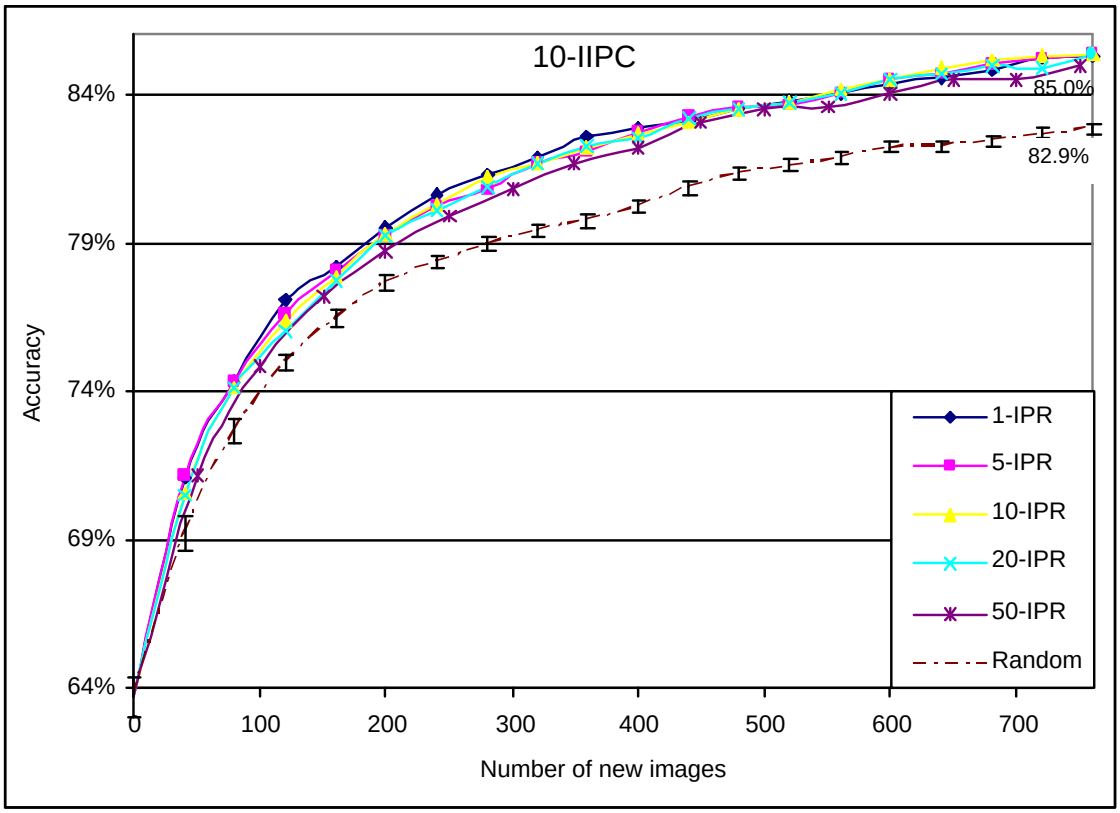

Figure 4.15. Comparison of active learning and random sampling in terms of accuracy with different IPR: initial training images per class are 10. Standard error bars are on the random sampling curve.

Figures 4.15 to 4.18 present the experimental results on "BT" by varying IPR for each IIPC. In all the experiments, the IPR was varied from 1 to 50. We only drew the error bars for random sampling because adding error bars to "BT" will make the graph too busy. We still used a paired-t test to compare "BT" with random sampling. To our surprise, classification accuracy with large IPRs is almost as good as with small IPRs although a very large IPR (IPR=50) results in slightly less accurate classifiers than a small IPR in many cases. In all situations, a large IPR up to 50 is statistically significantly more accurate than random sampling at p=0.05. These results indicate that our active learning

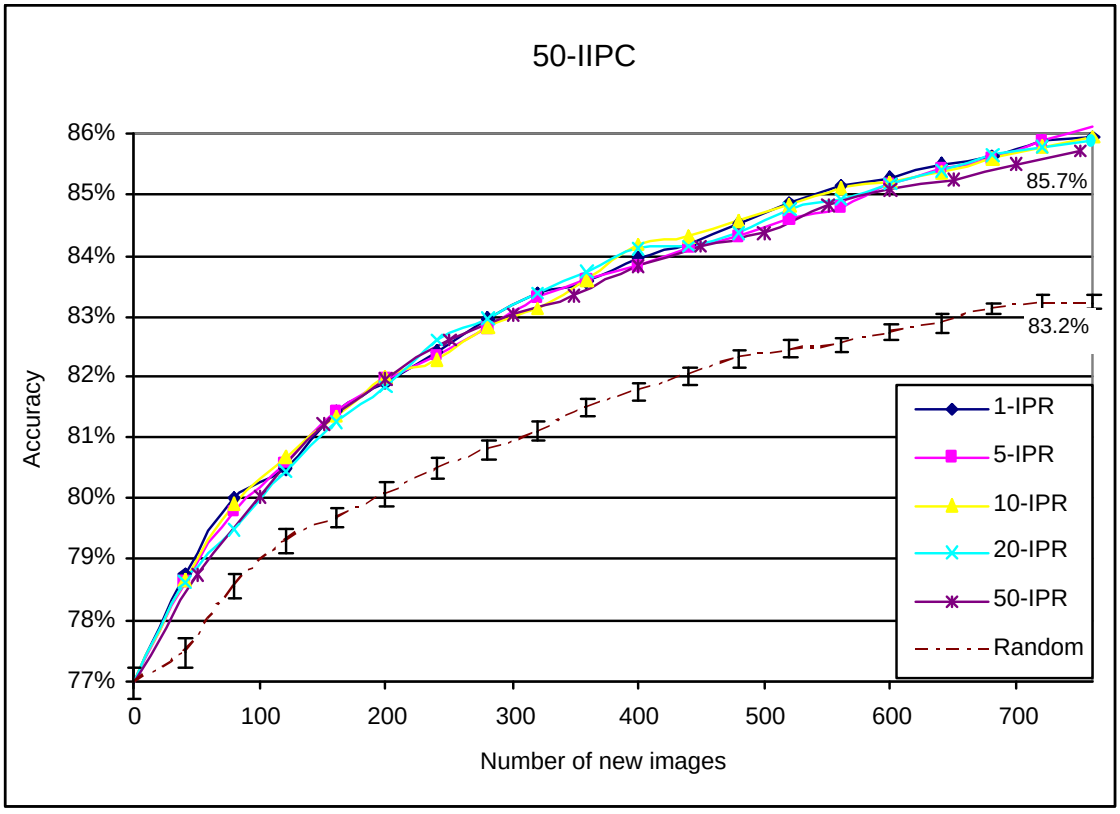

Figure 4.16. Comparison of active learning and random sampling in terms of accuracy with different IPR: initial training images per class are 50. Standard error bars are on the random sampling curve.

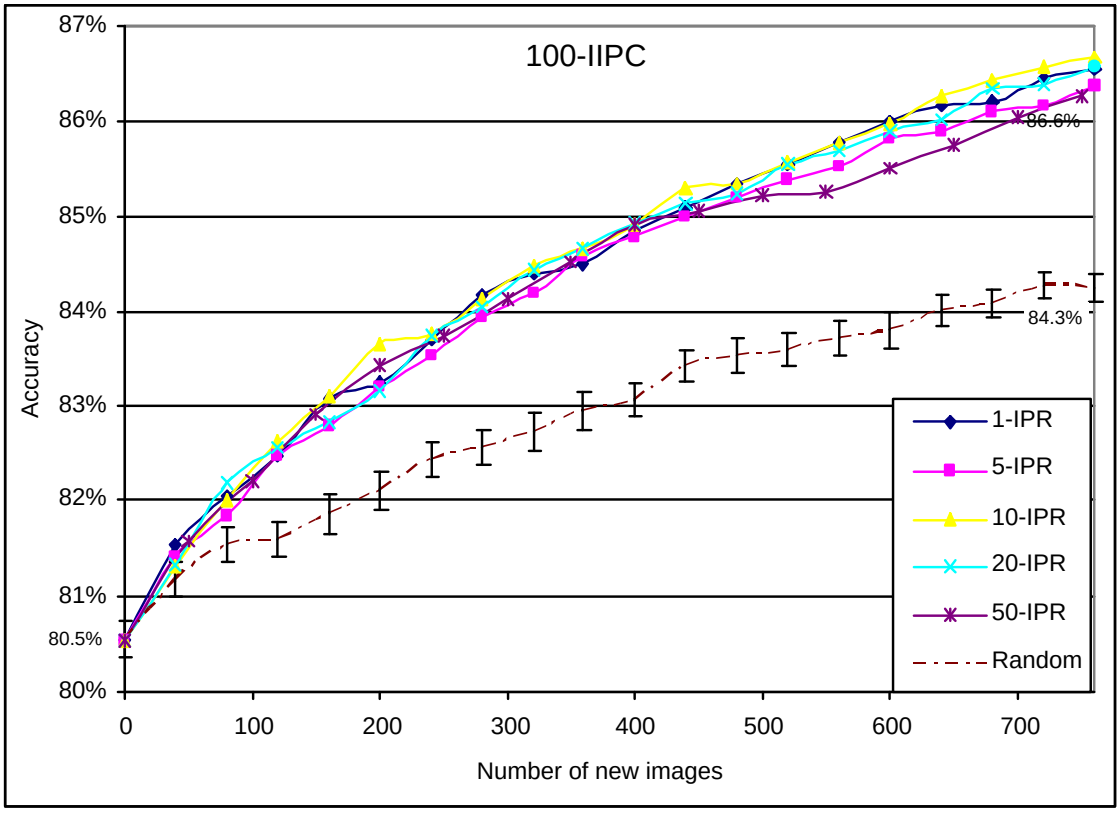

Figure 4.17. Comparison of active learning and random sampling in terms of accuracy with different IPR: initial training images per class are 100. Standard error bars are on the random sampling curve.

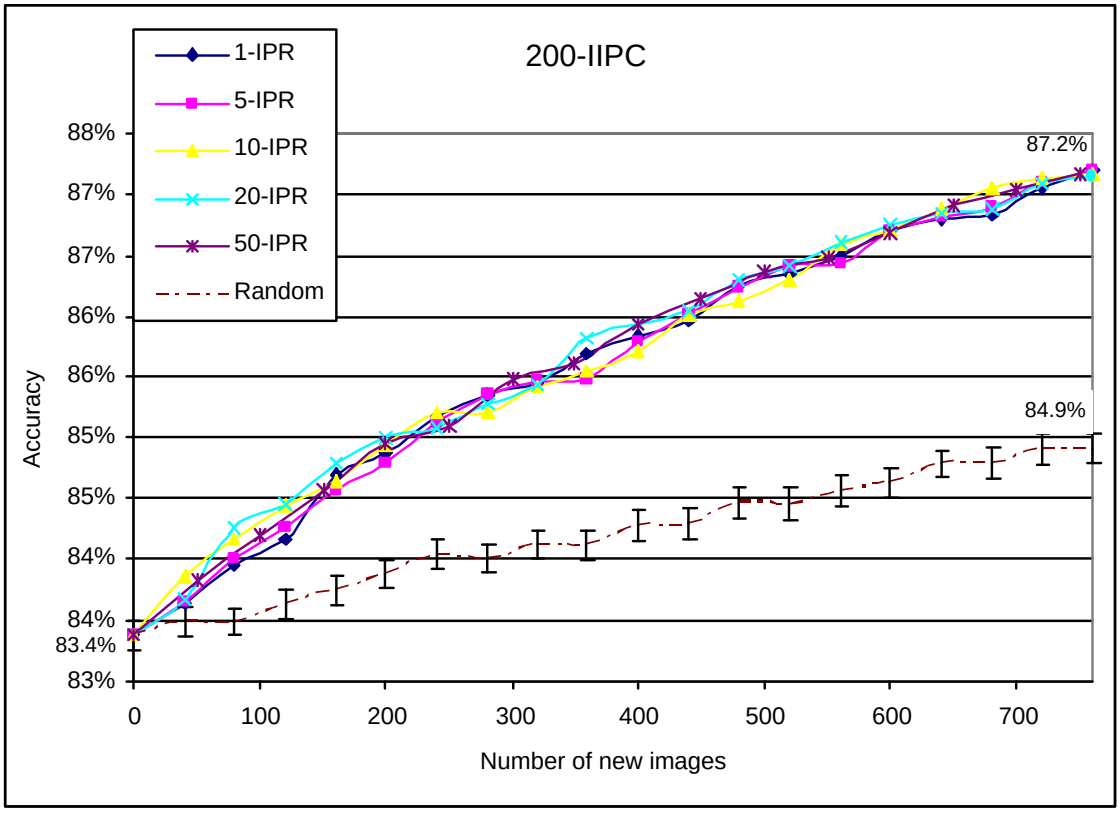

Figure 4.18. Comparison of active learning and random sampling in terms of accuracy with different IPR: initial training images per class are 200. Standard error bars are on the random sampling curve.

approach "BT" can run in batch mode, where tens of examples are labeled at a time, to achieve speedup with at most a little compromise in accuracy.

4.5 Conclusion and discussion

This chapter presents an active learning approach to reduce domain experts' labeling efforts in recognizing plankton from higher-resolution, new generation SIPPER II images. It can be applied to any data set where the examples will be labeled over time and one wants to use the system as early as possible. The "Breaking Ties" active learning method is proposed and applied to a multi-class SVM using the one-vs-one approach on newly developed, image features extracted from gray-scale SIPPER images. The experimental results indicate that our proposed active learning approach successfully reduces the number of labeled images required to reach a given accuracy level when compared with random sampling. It also outperforms the least certainty approach proposed by us earlier in [52]. Our new approach can be run in batch mode, labeling up to 50 images at a time to achieve significant speedup with similar classification accuracy compared with labeling one image at a time. In the following, we will address and discuss several issues of active learning in SVMs for further explorations.

One critique of active learning is the overhead related to searching for the next candidate to label. While random sampling just selects an example to label at random, active learning needs to evaluate every unlabeled example. This overhead becomes significant when the unlabeled data set is very large. A simple solution to that is to use random subset evaluation: Each time searching for the next candidate example to label, instead of evaluating the entire unlabeled data set, one can only evaluate a subset of data randomly drawn from the entire set. We indicate without proof here that for IPR=1, we need to sample 59 data, which provides 95% probability confidence that the best candidate from the 59 data subset is superior to 95% data from the total unlabeled set. See [83, chap 6.5] for more detail.

Another important issue is the change of optimal kernel parameters. We can find the optimal kernel parameters for the initial labeled data set. As more labeled data are added, however, such kernel parameters are no longer optimal. Unless we can afford a held-out, labeled data set, it is hard to tune the kernel parameters online. The key reason is we do not have a good method to evaluate different kernel parameters while active learning proceeds. The standard methods like cross-validation and leave-one-out tend to fail because active learning brings in biased data samples. Such failures were observed and discussed in [2]. An important future direction is to find a good online performance evaluation method for active learning. Otherwise, one could take it as one of the biggest bottlenecks to use SVMs as the base learner in active learning because SVMs depend heavily on good kernel parameters. An effort toward solving this problem is reported in [2], where the classification entropy maximization (CEM) criterion was used to evaluate the performances of different active learners. Their work shows CEM can help select the best active learner on several two-class data sets.

An important thing omitted in most active leaning+SVMs literature is to run active learning in batch mode. Unless labeling an example is extremely expensive, it is always convenient and practical to run active learning in batch mode, namely labeling several examples at a time. As indicated in this chapter, the best candidate set to label might be found in a different way from a single best candidate point. "Combined" [14] only works for two-class problems. A criterion for the best set of data to label in multi-class SVMs needs to be addressed in future active learning work. At the very least, existing active learning methods need to show they work well in batch mode. Fortunately, our proposed active learning method works well in batch mode without employing a new criterion for selecting a set of data to label.

CHAPTER 5

BIT REDUCTION SUPPORT VECTOR MACHINE

Support vector machines are very accurate classifiers and have been widely used in many applications. However, the training and to a lesser extent prediction time of support vector machines on very large data sets can be very long. This chapter presents a fast compression method to scale up support vector machines to large data sets. A simple bit reduction method is applied to reduce the cardinality of the data by weighting representative examples. We then develop support vector machines trained on the weighted data. Experiments indicate that the bit reduction support vector machine produces a significant reduction of the time required for both training and prediction with minimum loss in accuracy. It is also shown to be more accurate than random sampling when the data is not over-compressed.

5.1 Motivation

Support vector machines (SVMs) achieve high accuracy in many application domains including this work in recognizing underwater zooplankton. However, scaling up SVMs to a very large data set is still an open problem. Training a SVM requires solving a constrained quadratic programming problem, which usually takes $O(m^3)$ computations where m is the number of examples. Predicting a new example involves $O(sv)$ computations where sv is the number of support vectors and is usually proportional to m. As a consequence, SVMs' training time and prediction time to a lesser extent on a very large data set can be quite long, thus making it impractical for some real-world applications. In plankton recognition, a SVM needs to make a real-time or near real-time prediction on underwater plankton sampled by in-situ imaging sensors in order to obtain the composition, abundance and

distribution of plankton in a timely fashion. Also, fast retraining is often required as new plankton images are labeled by marine scientists and added to the training library on the ship. As we acquire a large number of plankton images, training a SVM with all labeled images becomes extremely slow. In this chapter, we propose a simple strategy to speedup the training and prediction procedures for a SVM: bit reduction. Bit reduction reduces the resolution of the input data and groups similar data into one bin. A weight is assigned to each bin according to the number of examples in it. This data reduction and aggregation step is very fast and scales linearly with respect to the number of examples. Then a SVM is built on a set of weighted examples which are the exemplars of their respective bins. Our experiments indicate that bit reduction SVM (BRSVM) significantly reduces the training time and prediction time with a minimal loss in accuracy. It outperforms random sampling on most data sets when the data are not over-compressed. We also find that on one high dimensional data set, that bit reduction does not perform as well as random sampling, thus providing a limit on the performance of BRSVM for high dimensional data sets. The rest of this chapter is organized as follows. Section 5.2 reviews previous work in speeding up SVMs. Section 5.3 describes the bit reduction support vector machine (BRSVM). In Section 5.4, we describe experiments with BRSVM on nine data sets and analyze the results. Section 5.5 summarizes this approach.

5.2 Previous work

There are two main approaches to speed up training of SVMs. One approach is to find a fast algorithm to solve the quadratic programming (QP) problem for a SVM. "Chunking", introduced in [90], solves a QP problem on a subset of data. Chunking only keeps the support vectors on the subset and replaces others with data that violate the Karush-Kuhn-Tucker (KKT) conditions. Using an idea similar to chunking, decomposition [63] [43] puts a subset of data into a "working set", and solves the QP problem by optimizing the coefficients of the data in the working set while keeping the other coefficients unchanged. In this way, a large QP problem is decomposed into a series of small QP problems, thus making

it possible to train a SVM on large scale problems. Sequential minimum optimization (SMO) [70] and its enhanced versions [45] [27] take decomposition to the extreme: Each working set only has two examples and their optimal coefficients can be solved analytically. SMO is easy to implement and does not need any third-party QP solvers. SMO is widely used to train SVMs. Another way of solving large scale QP problems [34][97] is to use a low-rank matrix to approximate the Gram matrix of a SVM. As a consequence, the QP optimization on the small matrix requires significantly less time than on the whole Gram matrix.

The other main approach of speeding up SVM training comes from the idea of "data squashing", which was proposed in [29] as a general method to scale up data mining algorithms. Data squashing divides massive data into a limited number of bins. The statistics of the examples from each bin are computed. A model is fit by only using the statistics instead of all examples within a bin. The reduced training set results in significantly less training time. Researchers have applied data squashing to SVMs. Several clustering algorithms [99][85][9] were used to partition data and build a SVM based on the statistics from each cluster. In [9], the SVM model built on the reduced set was used to predict on the whole training data. Examples falling in the margin or being misclassified were taken out from their original clusters and added back into the training data for retraining. However, both [99] and [85] assumed a linear kernel and it might not generalize well to other kernels. In [9], two experiments were done with a linear kernel and only one experiment used a third-order polynomial kernel. Moreover, it is not unusual that many examples fall into the margin of a SVM model especially for a RBF kernel. In such cases, retraining with all examples within the margin is computationally expensive. Following the idea of the likelihood-based squashing [54] [65], a likelihood squashing method was developed for a SVM by Pavlov and Chudova [67]. The likelihood squashing method assumes a probability model as the classifier. Examples with similar probability $p(x_i, y_i|\theta)$ are grouped together and taken as a weighted exemplar. Pavlov and Chudova used a

probabilistic interpretation of SVMs to perform the likelihood squashing. Still, only a linear kernel was used in their experiments.

Most work [15][16][64][82] on enabling fast prediction with SVMs focused on the problem of reducing the number of SVs obtained. Since the prediction time of a SVM depends on the number of support vectors, they searched for a reduced set of vectors which can approximate the decision boundary. The prediction using the reduced set was faster than using all support vectors. However, reduced set methods involve searching for a set of pre-images [82][83], which is a set of constructed examples used to approximate the solution of a SVM. It should be noted that the searching procedure is computationally expensive.

Data squashing approaches seem promising and can be combined with fast QP like SMO etc. for fast training and prediction. However, most work [85][9][99] in data squashing+SVM requires clustering the data and/or linear kernels [85][99][67]. Clustering usually needs $O(m^2)$ computations and high-order kernels, like the RBF kernel, are widely used and essential to many successful applications. Therefore, a fast squashing method and experiments on high-order kernels is necessary to apply data squashing+SVMs to real-world applications. In this chapter, we propose a simple and fast method data compression method: bit-reduction SVM (BRSVM). It does not require any computationally expensive clustering algorithms and works well with RBF kernels as shown in our experiments.

5.3 Bit reduction SVM

Bit reduction SVM (BRSVM) works by reducing the resolution of examples and representing similar examples as a single weighted example. In this way, the data size is reduced and training time is saved. It is simple and much faster than clustering. Another even simpler data reduction method is random sampling. Random sampling subsamples data without replacement. Compared to weighted examples, random sampling suffers from high variance of estimation in theory. Please refer to [21][72] for details about sampling theory. In spite of its high variance, random sampling has been shown to work very well in

experiments [65][85]: It was as accurate as or slightly less accurate than complicated data squashing methods.

5.3.1 Bit reduction

Bit reduction is a technique to reduce the data resolution. One example use is a bit reduction fuzzy c-means (BRFCM) method [44][32], which applied bit reduction to speed up the fuzzy c-means (FCM) [30][8] clustering algorithm. However, bit reduction in clustering does not consider the class label of examples in the same bin. In classification only examples from the same class should be aggregated together. Also, comparison to random sampling was omitted in [44][32].

There are three steps involved in bit reduction for a SVM: normalization, bit reduction and aggregation.

1. Normalization is used to ensure equal resolution for each feature. Different features may have very different scales, thus reducing precision equally along each feature may not be fair. For instance, features with small scales become zero and irrelevant to classification after bit reduction. Therefore, a normalization is needed to make each feature have zero mean and unit variance. To avoid losing too much information during quantization, an integer is used to represent each normalized feature value. The integer $I(v)$ for a floating point value v is constructed as follows:

$$I(v) = int(Z * v)$$

   where Z is an arbitrary number used to scale v and function $int(k)$ returns the integer part of k. In this way, the true value of v is kept and only $I(v)$ is used in bit reduction. In our experiments, we used $Z = 1000$.

2. Bit reduction is performed on the integer $I(v)$. Given b, the number of bits to be reduced, $I(v)$ is right-shifted and its precision is reduced. We slightly abuse notation here by letting the $I(v)$ in the right hand side of Eq. (5.1) be the $I(v)$ before bit

reduction and $I(v)$ in the left hand side be the $I(v)$ after bit reduction.

$$I(v) \leftarrow I(v) \gg b \tag{5.1}$$

where $k \gg b$ shifts the integer k to the right by b bits. Given an r-dimensional example x_i=(x_{i1},x_{i2},...,x_{ir}), its integer expression after bit reduction is ($I(x_{i1})$,$I(x_{i2})$,...,$I(x_{ir})$).

3. The aggregation step groups the examples from the same class whose integer expressions fall into the same bin. For each class, the mean of examples within the same bin is computed as their representative. The weight of the representative equals the number of examples from that class. During the mean computation, the real values $(x_{i1}, x_{i2}, ..., x_{ir})$ are used.

Note the bit reduction procedure loses data precision. A very large b results in too many examples falling in the same bin. The mean statistic is not enough to capture the location information of many examples. A small b does not provide enough data reduction, thus leaving training still slow. The best number of bits reduced (b) varies for different data sets. It can be found by trial-and-error. For instance, one can try different b to see how many training examples exist before building a classifier. The data miner then can choose b based on the size of the data set a SVM can handle within a given time span. Alternatively, if retraining on the same type of data is needed frequently (e.g. new data acquired), cross validation can be used to find the best b, which provides desired speedup and minimum accuracy loss. The optimal number b for bit reduction will be used for retraining on the same type of data.

During bit reduction, it is very likely that a bin has examples from many different classes. Therefore, in the aggregation step, the mean statistic of examples in the same bin was computed individually for each class. This can at least alleviate the side effect of grouping examples from different classes into the same bin. As a result, one bin may contain weighted examples for multiple classes.

Table 5.1 describes the bit reduction procedure for four 1-d examples with class label y_i.

Table 5.1. An 1-d example of bit reduction in BRSVM.

i	Example (x_i, y_i)	I(x_i) and its bit expression $Z = 1000$	I(x_i) after 2-bit reduction
1	(0.008, 1)	8 (1000)	2 (10)
2	(0.009, 1)	9 (1001)	2 (10)
3	(0.010, 2)	10 (1010)	2 (10)
4	(0.011, 2)	11 (1011)	2 (10)

The four examples from two classes are first scaled to integer values by using $Z = 1000$. Then 2-bit reduction is performed by right shifting its integer expression by 2 bits. All four examples end up having the same value, which means all four examples fall into one bin after a 2-bit reduction. Table 5.2 shows the weighted examples after the aggregation step.

Table 5.2. Weighted examples after the aggregation step.

i	New examples (x_i, y_i)	Weight
1	(0.0085, 1)	2
2	(0.0105, 2)	2

Since all four examples are in the same bin, we aggregate them by class and compute their mean for each class using the original values x_i. The weight is computed by simply counting the number of examples from the same class.

Although bit reduction is fast, a sloppy implementation of aggregation may easily cost $O(m^2)$ computations where m is the number of examples. We implemented a hash table for the aggregation step as done in [32]. Universal hashing [22] was used as the hash function. Collisions were resolved by chaining. When inserting the bit-reduced integer values into the hash table, we used a list to record the places that were filled in the hash table. The mean statistics were computed by re-visiting all the filled places in the hash table. The average computational complexity for our implementation is $2m$. Please see [22] for more detail about universal hashing function.

5.3.2 Weighted SVM

Pavlov et al. [67] proposed a method to train a weighted SVM, although its description in [67] is concise and lacks significant details. Following their work, we describe how to train a weighted SVM in more detail in this subsection.

Given examples $x_1, x_2, ..., x_m$ with class label $y_i \in$ {-1,1}, a SVM solves the following problem

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\langle w, w\rangle + \frac{C}{m}\sum_{i=1}^{m}\xi_i \\ \text{subject to:} \quad & y_i(\langle w, \phi(x_i)\rangle + b) \geq 1 - \xi_i \\ & C, \xi_i > 0 \end{aligned} \tag{5.2}$$

where w is normal to the decision boundary (a hyperplane), C is the regularization constant that controls the trade-off between the empirical loss and the margin width, the slack variable ξ_i represents the empirical loss associated with x_i. In the case of weighted examples, the empirical loss of x_i with a weight β_i is simply $\beta_i\xi_i$. Intuitively, it could be interpreted as β_i identical examples x_i. Accumulating the loss of the β_i examples results in a loss of $\beta_i\xi_i$. Substitute ξ_i with $\beta_i\xi_i$ in Eq. (5.2), and we derive the primal problem of a weighted SVM:

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}\langle w, w\rangle + \tfrac{C}{m}\textstyle\sum_{i=1}^{m}\beta_i\xi_i \\ \text{subject to:} \quad & y_i(\langle w, \phi(x_i)\rangle + b) \geq 1 - \xi_i \\ & C, \xi_i > 0, i = 1, ...m \end{aligned} \tag{5.3}$$

The constraint in Eq. (5.3) remains unchanged because the constraint for each of the β_i examples x_i is identical. The β_i identical constraint formulas can be reduced to one constraint as shown in Eq. (5.3).

Introducing the Lagrangian multiplier α_i, Eq. (5.3) leads to

$$\begin{aligned} L(\alpha, w, b) &= \frac{1}{2}\langle w, w\rangle + \frac{C}{m}\sum_{i=1}^{m}\beta_i\xi_i \\ &\quad -\sum_{i=1}^{m}\alpha_i(y_i(\langle w, \phi(x_i)\rangle + b) - 1 + \xi_i) \\ \alpha_i &> 0,\ i = 1, ..., m \end{aligned} \tag{5.4}$$

where α is the vector $(\alpha_1, \alpha_2, ..., \alpha_m)$. Its saddle point solution can be computed by taking the partial derivatives of $L(\alpha, w, b)$.

$$\frac{\partial L(\alpha, w, b)}{\partial w} = 0 \text{ and } \frac{\partial L(\alpha, w, b)}{\partial b} = 0 \tag{5.5}$$

We get

$$w = \sum_{i=1}^{m}\alpha_i y_i \phi(x_i) \tag{5.6}$$

$$\sum_{i=1}^{m}\alpha_i y_i = 0 \tag{5.7}$$

Substitute them into Eq. (5.4) and the dual form of a weighted SVM is as follows.

$$\begin{aligned} &\text{maximize} && \textstyle\sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j k(x_i, x_j) \\ &\text{subject to} && 0 \le \alpha_i \le \tfrac{C\beta_i}{m}, i = 1, ..., m \\ &&& \textstyle\sum_{i=1}^{m}\alpha_i y_i = 0 \end{aligned} \tag{5.8}$$

The dual form of a weighted SVM is almost identical to a normal SVM except for the boundary condition of $\alpha_i \leq \frac{C\beta_i}{m}$ while in a normal SVM $\alpha_i \leq \frac{C}{m}$. Therefore, efficient solvers for a normal SVM such as the SMO [70] can be used to solve a weighted SVM by modifying the boundary condition slightly.

5.4 Experiments

We experimented with BRSVM on nine data sets: banana, phoneme, shuttle, page, pendigit, letter, SIPPER II plankton images, waveform and satimage. Banana includes two-dimensional, banana-shaped data from two classes. It is a widely used benchmark data set and was originally introduced in [74]. Phoneme is provided in the ELENA repository [31]. Its aim is to differentiate between nasal and oral vowels from 5 harmonic attributes. The shuttle data set in the Statlog collection [57] has 9 numerical attributes and 58,000 examples. There are 7 classes. About 80% of the data belongs to class 1. Therefore, the default accuracy is about 80%. Page, pendigit and letter come from the UCI machine learning repository [56]. The problem of the page data set is to classify the page layout of a document into: horizontal line, picture, vertical line and graphic. The goal of the pendigit data set is to recognize the 10 pen-based digits: 0–9. In the letter data set, we are to identify 26 capital letters from 16 image features computed from black-and-white images. The plankton data set was originally used in [52]. Its objective is to classify the five most abundant types of plankton from 17 selected image features from 3-bit plankton images. Waveform was originally used in [13]. The problem is to predict 3 classes of waves from 21 attributes with added noise. Please see [13, 49–55] for details. Satimage is from the Statlog repository. It is to predict the central pixel in 3×3 neighborhoods from a satellite image. Table 5.3 summarizes the characteristics of the nine data sets.

The Libsvm tool [19] for training support vector machines was modified and used in all experiments. The RBF kernel ($k(x, y) = exp(-g\|x - y\|^2)$) was employed. The kernel parameter g and the regularization constant C were tuned by a 5-fold cross validation on the training data. g and C were searched for from all combinations of the values in

Table 5.3. Description of the nine data sets.

Dataset	# of data	# of attributes	# of classes
banana	5300	2	2
phoneme	5404	5	2
shuttle	58000	9	7
page	5473	10	5
pendigit	10992	16	10
letter	20000	16	26
plankton	8440	17	5
waveform	5000	21	3
satimage	6435	36	6

$(2^{-10}, 2^{-9}, ..., 2^4)$ and $(2^{-5}, 2^{-4}, ..., 2^9)$ respectively. We used the same training and test separation as given by original uses of the data sets. For those data sets which do not have a separate test set, we randomly selected 80% of the examples as the training set and 20% of the examples as the test set. Since all nine data sets have more than 5000 examples, 20% of the total data will have more than 1000 examples. We believe it provided a relatively stable estimation. We built SVMs on the training set with the optimal parameters and reported the accuracy on the test set. All our experiments were run on a Pentium 4 PC at 2.6 GHZ with 1 GB memory under the Redhat 9.0 operation system.

5.4.1 Experiments with pure bit reduction

Tables 5.4–5.12 describe the experimental results from using BRSVM on the nine data sets. The last row of each table records the result of a SVM trained on the uncompressed data set. The other rows present the results from BRSVM. The first column is the number of bits reduced. The second column is the compression ratio, which is defined as $\frac{\text{\# of examples after bit reduction}}{\text{\# of examples}}$. We start off with 0-bit reduction which may not correspond to a 1.0 compression ratio. The reason is that repeated examples are grouped together even when no bit is reduced. This results in compression ratios less than 1.0 at 0-bit reduction in some cases. The third column is the accuracy of BRSVM on the test set. McNemar's test [25] is used to check whether BRSVM accuracy is statistically significantly different from the accuracy of a SVM built on the uncompressed data set. The number

in italics indicates the difference is not statistically significant at the $p = 0.05$ level. The fourth column is the time for bit reduction plus BRSVM training time. The time required to do example aggregation is included in this training time. The fifth column is the prediction time on the test set. All of the timing results were recorded in seconds. The precision of the timing measurement was 0.01 seconds. The training and prediction speedup ratio are defined as $\frac{\text{SVM training time}}{\text{BRSVM training time}}$ and $\frac{\text{SVM prediction time}}{\text{BRSVM prediction time}}$, respectively. In the last column, the average accuracy of random sampling on the test set is listed for comparison. The subsampling ratio is set to equal the compression ratio of BRSVM. Since random sampling has a random factor, we ran it 50 times for each subsampling ratio and recorded the average statistics. This accuracy is listed in the last column of Tables 5.4–5.12 titled subsampling accuracy.

Table 5.4. BRSVM on the banana data set. The accuracy in italics means it is not statistically significantly different from the accuracy of a SVM.

bit reduction	compression ratio	BRSVM accuracy	BRSVM training time	BRSVM prediction time	subsampling accuracy
0-1	1.000	*0.902*	2.59s	0.33s	0.902
2	0.996	*0.902*	2.59s	0.33s	0.902
3	0.987	*0.902*	2.59s	0.33s	0.902
4	0.957	*0.902*	2.45s	0.31s	0.902
5	0.842	*0.902*	1.99s	0.29s	0.902
6	0.572	*0.902*	0.98s	0.23s	0.901
7	0.245	*0.903*	0.21s	0.12s	0.895
8	0.077	*0.900*	0.03s	0.05s	0.890
9	0.024	*0.890*	0.02s	0.01s	0.865
10	0.007	0.740	0.01s	0.01s	0.687
SVM	1.000	0.902	2.58s	0.33s	

The experimental results on the banana data set are shown in Table 5.4. As more bits are reduced, fewer examples are used in training. Thus training time is reduced. Also, less training data results in a classifier with fewer support vectors. The prediction time is proportional to the total number of support vectors. Therefore, the prediction time of BRSVM is reduced accordingly. When 9 bits are reduced, BRSVM runs 129 times faster during training and 33 times faster during prediction than a normal SVM. Its accuracy is not statistically significantly different from a SVM built on all the data at the $p = 0.05$

level. BRSVM is as or more accurate than a SVM with random sampling up to 10-bit reduction.

Table 5.5. BRSVM on the phoneme data set. The accuracy in italics means it is not statistically significantly different from the accuracy of a SVM.

bit reduction	compression ratio	BRSVM accuracy	BRSVM training time	BRSVM prediction time	subsampling accuracy
0	0.992	*0.895*	18.61s	1.03s	0.895
1	0.984	*0.895*	18.59s	1.03s	0.895
2	0.978	*0.895*	17.01s	1.02s	0.894
3	0.973	*0.895*	15.60s	1.02s	0.894
4	0.968	*0.895*	15.59s	1.03s	0.893
5	0.963	*0.895*	15.59s	1.02s	0.892
6	0.950	*0.895*	15.43s	1.02s	0.892
7	0.891	*0.895*	14.21s	0.97s	0.890
8	0.679	*0.893*	9.28s	0.83s	0.873
9	0.303	0.846	2.01s	0.41s	0.824
10	0.059	0.752	0.09s	0.09s	0.730
SVM	1.000	0.895	17.51s	1.03s	

Phoneme is another relatively low-dimensional data set with five attributes. Table 5.5 presents the experimental results of BRSVM on this data set. When 8 bits are reduced, BRSVM runs 1.9 times faster during training and 1.2 times faster during prediction than a normal SVM. Its accuracy is not statistically significantly different from a SVM built on all the data at the $p = 0.05$ level. BRSVM is as or more accurate than random sampling when the compression ratio is larger than 0.059.

Table 5.6 and Table 5.7 present the BRSVM experiments on the shuttle and the page data sets, which have 9 attributes and 10 attributes respectively. After 10-bit reduction, BRSVM achieves 245.2 times speedup in training and 2.4 times speedup in prediction with a loss of 1.2% accuracy on the shuttle data set. BRSVM is as or more accurate than random sampling when the compression ratio is greater than 0.006. On the page data set, the speedup ratios for BRSVM are 7.9 in training and 1.8 in prediction when the compression ratio is 0.187 after 9-bit reduction. It should be noted that BRSVM is 0.5% more accurate than a SVM. Because bit reduction eliminates noisy examples and has the regularization effect, it may improve the classification accuracy.

Table 5.6. BRSVM on the shuttle data set. The accuracy in italics means it is not statistically significantly different from the accuracy of a SVM.

bit reduction	compression ratio	BRSVM accuracy	BRSVM training time	BRSVM prediction time	subsampling accuracy
0-1	1.000	*0.999*	22.57s	1.85s	0.999
2	0.994	*0.999*	22.00s	1.83s	0.999
3	0.904	*0.999*	19.03s	1.73s	0.999
4	0.717	*0.999*	12.43s	1.59s	0.999
5	0.484	*0.999*	8.07s	1.55s	0.999
6	0.258	*0.999*	3.72s	1.37s	0.998
7	0.108	*0.999*	1.40s	1.26s	0.998
8	0.031	*0.999*	0.44s	1.08s	0.996
9	0.014	0.996	0.19s	0.94s	0.994
10	0.006	0.987	0.09s	0.78s	0.986
11	0.003	0.835	0.06s	0.68s	0.982
SVM	1.000	0.999	22.07s	1.85s	

Table 5.7. BRSVM on the page data set. The accuracy in italics means it is not statistically significantly different from the accuracy of a SVM.

bit reduction	compression ratio	BRSVM accuracy	BRSVM training time	BRSVM prediction time	subsampling accuracy
0-3	0.988	*0.970*	2.89s	0.16s	0.970
4	0.987	*0.970*	2.89s	0.16s	0.970
5	0.985	*0.970*	2.89s	0.16s	0.970
6	0.971	*0.970*	2.75s	0.16s	0.970
7	0.833	*0.970*	2.26s	0.14s	0.970
8	0.465	*0.974*	1.37s	0.12s	0.969
9	0.187	*0.975*	0.37s	0.09s	0.964
10	0.073	0.723	0.06s	0.05s	0.952
11	0.028	0.579	0.01s	0.04s	0.930
SVM	1.000	0.970	2.92s	0.16s	

Table 5.8. BRSVM on the pendigit data set. The accuracy in italics means it is not statistically significantly different from the accuracy of a SVM.

bit reduction	compression ratio	BRSVM accuracy	BRSVM training time	BRSVM prediction time	subsampling accuracy
0-7	1.000	*0.981*	3.04s	2.23s	0.981
8	0.999	*0.981*	3.03s	2.23s	0.981
9	0.931	*0.981*	2.81s	2.17s	0.981
10	0.400	0.977	0.99s	1.59s	0.977
11	0.013	0.864	0.02s	0.12s	0.878
SVM	1.000	0.981	3.02s	2.23s	

Table 5.9. BRSVM on the letter data set. The accuracy in italics means it is not statistically significantly different from the accuracy of a SVM.

bit reduction	compression ratio	BRSVM accuracy	BRSVM training time	BRSVM prediction time	subsampling accuracy
0-7	0.941	*0.975*	54.08s	25.81s	0.974
8	0.937	*0.975*	53.98s	25.78s	0.974
9	0.887	*0.975*	50.68s	25.39s	0.973
10	0.490	0.966	21.89s	17.74s	0.956
11	0.059	0.779	0.72s	2.60s	0.807
SVM	1.000	0.975	57.73s	26.25s	

Table 5.10. BRSVM on the plankton data set. The accuracy in italics means it is not statistically significantly different from the accuracy of a SVM.

bit reduction	compression ratio	BRSVM accuracy	BRSVM training time	BRSVM prediction time	subsampling accuracy
0-8	0.995	*0.889*	24.02s	2.42s	0.886
9	0.962	*0.887*	23.14s	2.31s	0.884
10	0.362	0.829	2.79s	0.74s	0.854
11	0.070	0.695	0.09s	0.12s	0.771
SVM	1.000	0.887	24.23s	2.42s	

Table 5.11. BRSVM on the waveform data set. The accuracy in italics means it is not statistically significantly different from the accuracy of a SVM.

bit reduction	compression ratio	BRSVM accuracy	BRSVM training time	BRSVM prediction time	subsampling accuracy
0-9	1.000	*0.859*	3.86s	1.09s	0.859
10	0.995	*0.857*	3.81s	1.09s	0.857
11	0.151	0.845	0.11s	0.14s	0.844
SVM	1.000	0.859	3.84s	1.09s	

Table 5.12. BRSVM on the satimage data set. The accuracy in italics means it is not statistically significantly different from the accuracy of a SVM.

bit reduction	compression ratio	BRSVM accuracy	BRSVM training time	BRSVM prediction time	subsampling accuracy
0-8	1.000	*0.917*	4.40s	2.62s	0.917
9	0.990	*0.917*	4.27s	2.60s	0.916
10	0.727	0.900	2.95s	2.05s	0.911
11	0.126	0.734	0.23s	0.26s	0.871
SVM	1.000	0.917	4.38s	2.62s	

Table 5.8 and Table 5.12 show the experimental results on five relatively high dimensional data sets: pendigit, letter, plankton, waveform and satimage. BRSVM delivers a higher or equal accuracy compared to random sampling when the compression ratio is not very small. On the pendigit data set, BRSVM with 10 bit reduction results in a compression ratio of 0.400, which is 3.1 times faster in training and 1.4 times faster in prediction. It is also as accurate as random sampling. On the letter data set, BRSVM is a more accurate classifier than random sampling when the compression ratio is 0.490 after 10-bit reduction. When 10 bits are reduced, BRSVM has a speedup ratio of 2.6 for training and a speedup ratio of 1.5 for prediction with a loss of 0.9% in accuracy.

Table 5.10 shows the experimental results on a relatively high dimensional data set– plankton. BRSVM is slightly more accurate than random sampling when the number of reduced bits is up to 9. At the 10-bit reduction level, the compression ratio of BRSVM drops sharply from 0.962 to 0.362, resulting in a significant loss in accuracy.

On the plankton data set, BRSVM is slightly more accurate than random sampling when the number of reduced bits is up to 9. At the 10-bit reduction level, the compression ratio of BRSVM drops sharply from 0.962 to 0.362, resulting in a significant loss in accuracy. At the 11-bit reduction level, the accuracy of BRSVM is much lower than random sampling because of the low compression ratio. The same thing happens with the waveform data set. The compression ratio drops from 0.995 to 0.151 with a 11-bit reduction. BRSVM loses 1.4% accuracy accordingly. The reason of this phenomenon is that when the compression ratio is small, it is very likely that many examples from different classes fall into the same bin and the number of examples distribute far from uniformly among different bins. For instance, suppose bit reduction compresses the data into several bins and one bin has 80% of the examples from different classes. BRSVM uses the mean statistic as the representative for each class, which may not be able to capture the information about the decision boundary in this bin. Random sampling, on the other hand, selects the examples more uniformly. If 80% of the examples fall into one bin, random sampling will effectively sample four times more examples that reside in this bin than all others together, and

preserve the local information of the decision boundary much better than BRSVM. As a result, random sampling is likely to be as or more accurate than BRSVM when the compression ratio is very low. This tends to happen on high dimensional data sets. On the other hand, at a higher compression ratio, where examples from the same class fall into the same bin and distributions of the number of examples in bins are not very skewed, BRSVM preserves the statistics of all examples while random sampling suffers from high sampling variance. Therefore, BRSVM is more accurate than random sampling when the compression ratio is relatively high.

On the highest dimensional data satimage, the speedup ratios of BRSVM are only 1.5 and 1.3 with 1.7% accuracy loss at the compression ratio 0.727 after a 10-bit reduction. Also, BRSVM is not as accurate as random sampling at this compression ratio.

It should be noted that the compression ratios on some high-dimensional data sets (plankton and waveform) drop much faster than those on the previous four data sets. This phenomenon is caused by the "Curse of Dimensionality" [5]. The corresponding interpretation in our case is that the data in a high-dimensional space are sparse and far from each other. Bit reduction will either group very few data together or put too many data in the same bin. As a result, BRSVM on the high dimensional data (satimage) does not perform as well as on the relatively lower dimensional data sets.

5.4.2 Experiments with unbalanced bit reduction

We used a simple solution to get a better compression ratio: unbalanced bit reduction (UBR). UBR works by reducing a different number of bits for different attributes. For instance, if reduction of a bits results in very little compression while reduction of $a+1$ bits compresses the data too much, UBR randomly selects several attributes to reduce $a+1$ bits while it applies a-bit reduction to the rest of the attributes. In this way, an intermediate compression ratio can be obtained. Since trying all of attributes to get a desired compression ratio is time consuming especially for high dimensional data sets, we use the following algorithm to choose the optimal number of attributes.

Unbalanced Bit Reduction

1: I^a and C^a are the data set and the compression ratio after reduction of a bits respectively. C^a is too large while C^{a+1} is too small. $A = \{a_1, a_2, ..., a_r\}$ is the set of r attributes.

2: $s = v = \lfloor r/2 \rfloor$.

3: *if v=0 then*

4: *Stop.*

5: *endif*

6: *Randomly select s attributes from A, apply 1 more bit reduction on the s attributes,* I^a *is further compressed to* $I^{a,s}$ *with compression ratio* $C^{a,s}$.

7: *if* $C^{a,s} >$ *desired compression ratio range then*

8: $v = \lfloor v/2 \rfloor$, $s = s + v$, *go to 3.*

9: *endif*

10: *if* $C^{a,s} <$ *desired compression ratio range then*

11: $v = \lfloor v/2 \rfloor$, $s = s - v$, *go to 3.*

12: *endif*

13: *Apply BRSVM on the reduced data set* $I^{a,s}$ *with randomly selected s 50 times and record the mean and the standard deviation of the compression ratio and the test accuracy over the 50 runs.*

In this algorithm, a bits are reduced on all the attributes initially. The desired compression ratio would be a range given by the user. Since one more bit reduction on all the attributes would compress the data too much, steps 2–12 determine the number of attributes s to be reduced by one more bit, which enables a compression ratio falling into a desired range. This algorithm can be also run in an interactive mode by asking the user to judge whether the $C^{a,s}$ is good enough at step 7 and 10. Considering the random factor in selecting the s attributes, we run the UBR 50 times and record the statistics in step 13. This provides more stable results because it experiments with BRSVM on compression ratios resulting from 1 more bit reduction on different combinations of s attributes.

We experimented with UBR on phoneme, pendigit, plankton, waveform and satimage, on which pure bit reduction did not result in ideal incremental compression ratios. Depending upon the application domain, a good compression ratio can be defined in different ways. In this chapter we define a good compression ratio as the minimum compression ratio with an accuracy within 1.2% of that obtained from a SVM trained on the uncompressed data set. In our UBR experiments, the unbalanced bit reduction algorithm was applied in interactive mode. Basically, the program asked one to decide whether $C^{a,s}$ fell into the desired compression ratio range at step 7 and step 10. If the ratio was acceptable, the program proceeded to build SVMs on the reduced data set at step 13. We also ran the random subsampling 50 times at the same compression ratio as UBR for comparison.

We present the experimental results of UBR in Tables 5.13–5.17. The unbalanced bit reduction algorithm was applied to find a s which gave a good compression ratio. In the tables, the first column records the s, the second column is the the mean and the standard deviation (in parentheses) of compression ratios from the 50 runs. The third column and the last column record the mean and the standard deviation of the accuracies over the 50 runs on the test set from BRSVM using UBR and random sampling respectively. Assuming the accuracies of 50 runs follow a normal distribution, we applied the t test to check whether the accuracy is statistically significantly different from the accuracy of a SVM built on the uncompressed data set. The number in italics indicates the difference is not statistically significant at $p = 0.05$ level. The fourth and the fifth column are the average training time and prediction time respectively. We will describe how the unbalanced bit reduction algorithm proceeds on the phoneme data set in detail while only presenting the experimental results from the other data sets.

The pure bit reduction experiments on phoneme were recorded in Table 5.5. After 8 bit reduction, BRSVM gives a 0.679 compression ratio and 1.9 times speedup in the training phase with a loss of 0.3% in accuracy. While after 9 bit reduction, the compression ratio drops to 0.303 and the corresponding 4.9% accuracy loss could not be tolerated. Since we will accept up to 1.2% accuracy loss, we applied UBR to search for a compression ratio

between 0.679 and 0.303. We hoped this could give more speedup than 1.9 times from a 8-bit reduction. We first applied 8 bit reduction to the data and then used the unbalanced bit reduction algorithm to find an s which gives a good compression ratio. See Table 5.13 for the UBR results on the phoneme data set. Initially, $s = v = \lfloor r/2 \rfloor = \lfloor 5/2 \rfloor = 2$ where the number of attributes r is 5 on the Phoneme data set. Since the compression ratio 0.55 from a randomly selected 2 attributes was very different from 0.679 (from a pure 8-bit reduction), we proceeded to step 13 to repeat the random selection 50 times with $s = 2$. We recorded the mean and standard deviation for the compression ratio and the test accuracy over the 50 runs. Using the average compression ratio from UBR as the sampling rate, we applied the random sampling method to build SVMs 50 times for comparison. Observing that the average accuracy 0.888 on the test set from 50 runs was only 0.7% less than 0.895, the accuracy from the uncompressed data, we ran the unbalanced bit reduction algorithm again to get an even lower compression ratio. At $s = 3$, the average accuracy is 1.5% less than 0.895. Therefore, we chose $s = 2$, which results in a speedup ratio of 2.7 and 1.7 for the training and prediction phase, respectively. Also, BRSVM is 2.5% more accurate than random sampling at the given compression ratio.

Table 5.13. BRSVM of UBR after 8-bit reduction on the phoneme data set. The accuracy in italics means it is not statistically significantly different from the accuracy of a SVM. The number in the parentheses is the standard deviation.

# of attributes reduction	compression ratio	BRSVM accuracy	BRSVM training time	BRSVM prediction time	subsampling accuracy
2	0.550	0.888	6.40s	0.67s	0.863
	(0.003)	(0.0027)			(0.0059)
3	0.467	0.880	4.74s	0.59s	0.856
	(0.008)	(0.0049)			(0.0067)
SVM	1.000	0.895	17.51s	1.03s	

On the pendigit data set, Table 5.14 shows that after 10 bit reduction and $s = 8$ (UBR), we get a speedup ratio of 12.1 and 3.0 in the training and prediction phase, respectively. The accuracy drops 1.1% compared to a SVM trained on the uncompressed data set.

Table 5.14. BRSVM of UBR after 10-bit reduction on the pendigit data set. The accuracy in italics means it is not statistically significantly different from the accuracy of a SVM. The number in the parentheses is the standard deviation.

# of attributes reduction	compression ratio	BRSVM accuracy	BRSVM training time	BRSVM prediction time	subsampling accuracy
8	0.146	0.970	0.25s	0.74s	0.970
	(0.016)	(0.0035)			(0.0048)
12	0.055	0.953	0.08s	0.39s	0.954
	(0.008)	(0.0075)			(0.0116)
SVM	1.000	0.981	3.02s	2.23s	

From Table 5.15, we see UBR provides a compression ratio of 0.739 on the plankton data set at $b = 9, s = 10$. The corresponding training and prediction phase were 1.6 and 1.4 times faster respectively with a 0.11 accuracy loss. BRSVM is just slightly more accurate than random sampling.

Table 5.15. BRSVM of UBR after 9-bit reduction on the plankton data set. The accuracy in italics means it is not statistically significantly different from the accuracy of a SVM. The number in the parentheses is the standard deviation.

# of attributes reduction	compression ratio	BRSVM accuracy	BRSVM training time	BRSVM prediction time	subsampling accuracy
8	0.814	0.881	18.03s	1.94s	0.880
	(0.034)	(0.0040)			(0.0030)
10	0.739	0.876	15.03s	1.73s	0.875
	(0.039)	(0.0055)			(0.0039)
12	0.638	0.866	11.22s	1.50s	0.872
	(0.036)	(0.0059)			(0.0045)
SVM	1.000	0.887	24.23s	2.42s	

On the waveform data set, Table 5.16 indicates UBR has 9.8 and 4.2 times speedup at $b = 10, s = 18$ in the training and prediction phase respectively with a 0.11 accuracy loss.

We also experimented with UBR on the highest dimensional data set satimage. When $b = 9, s = 31$, BRSVM speeds up the training and prediction phase by only 1.3 and 1.1 times with a 1.0% accuracy loss. Similar to the experiments on pure bit reduction, random sampling is more accurate than BRSVM with UBR on this data sets.

Table 5.16. BRSVM of UBR after 10-bit reduction on the waveform data set. The accuracy in italics means it is not statistically significantly different from the accuracy of a SVM. The number in the parentheses is the standard deviation.

# of attributes reduction	compression ratio	BRSVM accuracy	BRSVM training time	BRSVM prediction time	subsampling accuracy
15	0.469	0.853	1.04s	0.52s	0.853
	(0.012)	(0.0059)			(0.0053)
18	0.283	0.848	0.39s	0.26s	0.847
	(0.007)	(0.0043)			(0.0058)
SVM	1.000	0.859	3.84s	1.09s	

Table 5.17. BRSVM of UBR after 9-bit reduction on the satimage data set. The accuracy in italics means it is not statistically significantly different from the accuracy of a SVM. The number in the parentheses is the standard deviation.

# of attributes reduction	compression ratio	BRSVM accuracy	BRSVM training time	BRSVM prediction time	subsampling accuracy
27	0.885	0.913	3.80s	2.45s	0.915
	(0.011)	(0.0025)			(0.0021)
31	0.820	0.907	3.41s	2.31s	0.913
	(0.011)	(0.0028)			(0.0028)
33	0.785	0.903	3.20s	2.22s	0.912
	(0.009)	(0.0028)			(0.0029)
SVM	1.000	0.917	4.38s	2.62s	

5.4.3 Summary and discussion

Table 5.18 summarizes the performance of BRSVM on all nine data sets. The second column is the optimal b and s resulting in a "good" compression ratio, at which BRSVM achieves significant speedup with an accuracy loss less than 1.2%. The accuracy loss in the third column is defined as (accuracy of SVM − accuracy of BRSVM). The number in italics means the loss is not statistically significant. The speedups in the fourth and fifth columns are calculated as the speedup ratio in the previous experiments.

Table 5.18. Summary of BRSVM on all nine data sets. The accuracy in italics means it is not statistically significantly different from the accuracy of a SVM.

Data set	Optimal b and s	Accuracy loss (BRSVM)	Speedup in training	Speedup in prediction
banana	b=9,s=0	*1.2%*	129.0	33.0
phoneme	b=8,s=2	0.7%	2.7	1.7
shuttle	b=10,s=0	1.2%	245.2	2.4
page	b=9,s=0	*-0.5%*	7.9	1.8
pendigit	b=10,s=8	1.1%	12.1	3.0
letter	b=10,s=0	0.9%	2.6	1.5
plankton	b=9,s=10	1.1%	1.6	1.4
waveform	b=10,s=18	0.9%	13.0	4.0
satimage	b=9,s=31	1.0%	1.3	1.1

BRSVM works well on the nine data sets. At a small accuracy loss (less than 1.5%), the training and prediction speedup ratios range from 1.3 and 1.1 on the data set with the highest dimension to 245.2 and 33.0 on the lower dimensional data sets. Although accuracy loss exists (e.g. statistically significant) on seven out of nine data sets, it is small (less than 1.2%) and potentially acceptable to save time on large data sets.

Pure bit reduction ($s = 0$) performs very well on the four data sets with up to 10 attributes: banana, phoneme, shuttle and page. It achieves up to 245.2 times speedup in training and up to 33.0 times speedup in prediction without much loss in accuracy on the four data sets. On one relatively high-dimensional data set–letter, BRSVM with pure bit reduction is 2.6 times faster in training and 1.5 times faster in prediction with 0.9% loss in accuracy. BRSVM with pure bit reduction is more accurate than random sampling

on five data sets. On the pendigit, plankton and waveform data sets with relatively high dimensional data, pure bit reduction fails to provide a very good compression ratio, hence making BRSVM not as effective as random sampling. The justification is as follows: a high compression ratio results in minimal speedup while a too low compression ratio makes BRSVM less accurate. The best bit reduction and compression ratio vary across data sets. In our experiments, a high compression ratio is good for low-dimensional data sets while an intermediate compression ratio is desired for high-dimensional data sets. For instance, a 49% compression ratio is very good for BRSVM on the letter data set. As pure bit reduction fails to provide a compression ratio between 0.362 and 0.962 on the plankton data set, BRSVM is not as effective as random sampling. When unbalanced bit reduction was introduced for the data sets, BRSVM obtained intermediate compression ratios, which result in better accuracies than random sampling and significant speedups. On the phoneme data set, UBR was used to search for a smaller compression ratio, which enables more speedup. As a result, at a good compression ratio, BRSVM provides fast training and prediction and is more accurate than random sampling on eight data sets. On the highest dimensional data satimage, BRSVM is not as accurate as random sampling: At the optimal $b = 9$ and $s = 31$, the compression ratio of BRSVM is 0.885 and its corresponding accuracy is 90.7%, which is 0.6% less than that of random sampling.

Although random sampling has higher variances in theory, it works fairly well in our experiments except for banana and phoneme where random sampling is more than 2% less accurate than BRSVM. It performs only slightly worse than BRSVM on six out of nine data sets. This phenomenon was also observed in [65][85], where complicated data squashing strategies brought small or no advantages over random sampling. On satimage–the highest dimensional data set, random subsampling is slightly more accurate than BRSVM. Moreover, when a large compression ratio is needed for very fast training, random sampling outperforms BRSVM especially on high dimensional data sets.

In SVMs, only support vectors determine the decision function. Grouping support vectors together in the aggregation step may result in a less accurate classifier. To improve

the accuracy, one could check all the original data using the BRSVM built on the weighted examples. Then those support vector candidates could be pulled out from the weighted examples. A SVM retrained on the support vector candidates and the rest of the weighted examples is likely to be more accurate than BRSVM. A support vector candidate can be judged by checking if it falls into the margin of a SVM. The following algorithm describes this procedure.

Retraining BRSVM

1: $brSVM$ is the classifier trained by BRSVM; $R = \{(r_1, w_1), (r_2, w_2), ..., (r_n, w_n)\}$ is the reduced, weighted data after bit reduction where r_i is the ith exemplar and w_i is its weight; $X = (x_1, x_2, .., x_m)$ is the full data set without bit reduction; $S = \emptyset$.

2: *for* $i = 1$ *to* m *do*

3: *Use* $brSVM$ *to predict* x_i

4: *if* x_i *falls within the SVM margin and* x_i *is grouped together with other examples in* (r_k, w_k) *then*

5: $S \leftarrow S + (x_i, 1)$, $w_k \leftarrow w_k - 1$, *re-calculate* r_k.

6: *endif*

7: *end for*

8: *Retrain a SVM on* S+R

Retraining may improve the classification accuracy, however, it also slows down the whole training phase because of the additional retraining time and the overhead of prediction and modification of R and S from steps 3–5.

Table 5.19 records the experimental results using BRSVM with retraining on the nine data sets. The second column is the mean accuracy of a SVM using the uncompressed data from cross validation. The third column is the mean accuracy of BRSVM after retraining. The fourth column is the speedup ratio after retraining. It includes the time for BRSVM training, retraining, and the time to select support vector candidates and update S and A in steps 3–5 . The last column is the prediction time of the classifier built with retraining. We also list the BRSVM statistics without retraining in the parentheses for comparison.

Table 5.19. BRSVM after re-training on the nine data set.

Data set	SVM accuracy	BRSVM+retraining accuracy	Speedup in training	Speedup in prediction
banana	0.902	0.890	6.2	2.4
		(0.890)	(129.0)	(33.0)
phoneme	0.895	0.895	0.7	1.1
		(0.888)	(2.7)	(1.7)
shuttle	0.999	0.999	1.0	1.1
		(0.987)	(245.2)	(2.4)
page	0.970	0.972	0.8	1.1
		(0.975)	(7.9)	(1.8)
pendigit	0.981	0.981	0.8	1.0
		(0.970)	(12.1)	(3.0)
letter	0.975	0.975	0.4	1.1
		(0.966)	(2.6)	(1.5)
plankton	0.887	0.887	0.5	1.0
		(0.876)	(1.6)	(1.4)
waveform	0.859	0.858	1.0	1.3
		(0.848)	(13.0)	(4.0)
satimage	0.917	0.917	0.3	1.0
		(0.907)	(1.3)	(1.1)

On seven out of nine data sets, retraining improved the accuracy at the cost of more training time. The total training time increased significantly after retraining due to the second round of training and the overhead. On most data sets, the training speedup ratio is less than 1, which indicates the total retraining time is longer than training a SVM on the uncompressed data set. Therefore, retraining is not very helpful in BRSVM because it reduces the speedup ratio significantly.

5.5 Conclusion

In this chapter, a bit reduction SVM is proposed to speed up SVMs' training and prediction. BRSVM groups similar examples together by reducing their resolution. Such a simple method reduces the training time and the prediction time of a SVM significantly in our experiments when bit reduction can compress the data well. It is more accurate than random sampling when the data set is not over-compressed. BRSVM tends to work better with relatively lower dimensional data sets, on which it is more accurate than random

sampling and also shows more significant speedups. Therefore, feature selection methods might be used to reduce the data dimensionality and potentially help BRSVM to obtain further speedups. It should be noted that no feature reduction has been done on most of the data sets used in our experiments. We can also conclude if a very high speedup is desired in which a high compression ratio is required, random sampling may be a better choice. This tends to happen with high dimensional data.

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH

6.1 Conclusions

We review and conclude this dissertation as follows.

1. A set of robust image features without requiring a precise contour were discussed in Chapter 3. These features including moment invariants and granulometric features are global and do not provide information about any local areas in an image. They have been used to extract information from 1-bit, SIPPER I images, which do not have clear contours and lack texture information. Together with domain specific features, our feature set provides a good recognition rate by using a SVM as the classifier. The success of applying "imprecise", global features to the rather difficult SIPPER I images indicates the potential application of robust features to low-quality images. In our experiments, a single SVM using the one-vs-one approach was more accurate than a decision tree, a cascade correlation neural network, and ensembles of decision trees. The superior performance of a SVM validates its low generalization error bound from statistical learning theory. To select an optimal subset of features for a SVM, a wrapper approach with a best first search and a beam search reduced to half the total number of features needed with a slightly improvement in accuracy. This shows our feature selection method works well with moderate feature sizes.

2. In Chapter 4, the "Breaking Ties" (BT), an active learning method was proposed to reduce human effort in labeling large number of plankton images. Using a probabilistic interpretation of a multi-class SVM's outputs, "BT" selectively labels examples which were closest in probability between the most likely class and second most likely

class. Such a simple strategy required significantly less labeled images to reach a given accuracy than a least certainty active learning method and random sampling. "BT" also worked effectively in batch mode with an accuracy comparable to labeling one image at a time.

3. Chapter 5 presents bit reduction SVM (BRSVM), which speeds up a SVM's training and prediction. Since many methods have been well developed for fast solving the QP problem involved in a SVM, we think there is little room for further improvement in this direction. Data squashing is another potential way of scaling up SVMs by compressing a large data set to a small one. Most previous work on data squashing + SVM applied a clustering algorithm to compress data. However, clustering massive data is also slow. BRSVM uses a very simple strategy to compress data: bit reduction. Bit reduction reduces the resolution of data and can be taken as a special case of a clustering algorithm. However, bit reduction is simple and very fast. Assuming m is the total number of training examples, an efficient implementation of bit reduction only takes $2m$ computations to perform a data partition and aggregation, while most clustering algorithms need $O(m^2)$. Experimental results indicate BRSVM achieved fast training and prediction for a SVM. When the compression ratio is not very small, BRSVM could speed up SVM's training and prediction significantly and was more accurate than random sampling. It has been observed that on relatively high-dimensional data sets, a naive implementation of bit reduction sometimes did not provide a good compression ratio. A simple unbalanced bit reduction method helps relieve this situation.

6.2 Contributions

This dissertation addresses several challenges presented in a real-world plankton recognition problem and focuses on the scalability issue of applying a support vector machine to large-scale data sets. Its contributions are summarized as follows:

1. We designed an automated plankton recognition system to recognize underwater zooplankton from SIPPER using high-speed, line-scanned laser imaging sensors. Robust features were computed for the challenging SIPPER I images. A support vector machine was applied to classify those feature vectors. This system achieved approximately 90% accuracy on a collection of SIPPER I images. On another larger image set containing manually unidentifiable particles, it also provided 75.6% overall accuracy. Our feature selection method reduced the number features from 29 to 15 with slightly higher accuracy than using all features.

2. We proposed "Breaking Ties", an active learning method for multi-class SVMs, to reduce human effort in manually labeling a large number of data. "Breaking Ties" uses a "smart" selective sampling strategy to label data. When applied to recognizing SIPPER II images, it required significantly less labeled images to reach a given accuracy than a least certainty active learning method and random sampling. It ran effectively in batch mode, labeling up to 20 images at a time with an accuracy comparable to labeling one image at a time and retraining.

3. We developed a bit reduction SVM (BRSVM) to speed up a SVM's training and prediction. Compared to previous work in this direction, bit reduction is much simpler and faster. Also, we are among the few, if there any others, who experimented with data squashing on SVMs using a relatively complicated kernel (RBF kernel). In our experiments, BRSVM performed very well on the nine data sets. It achieved up to 245.2 times speedup in training and 33.0 times speedup in prediction without much loss in accuracy on the data sets respectively. The experiments indicate that as long as bit reduction compresses the data at a reasonable ratio, BRSVM outperforms random sampling.

6.3 Future research

Having tackled several scalability issues of SVMs in large scale data sets and plankton recognition, it is important to explore future research directions to improve the current methods. We present some of them in the following.

1. When we dealt with SIPPER I images, there were many manually unidentifiable particles. It is hard to develop a feature set for them because their shapes and appearances vary a lot. The accuracy of a SVM dropped from 90% to 75.6% due to inclusion of those particles. The higher-resolution SIPPER II images alleviated this problem as many blured images became much clearer with a higher resolution. However, this issue still remains because marine scientists only provided plankton images from a limited number of classes and other types of plankton were unclassified. Consequently, a SVM needs to label images which do not belong to any types of plankton in the training data. A SVM is a discriminative learning algorithm, namely differentiating between given classes. Hence it cannot tell us whether a current image belong to a new type of plankton. Generative learning methods such as a Bayesian network specify a probability model for the feature data, and thus are capable of indicating how likely a new example is to belong to a class. However, specifying a probability model is a strong assumption and it often leads to performance inferior to that of a SVM [28][42][60]. An interesting research question would be: Can we build a learning algorithm which provides SVM's classification accuracy and also detects new types of data?

2. In active learning, the kernel parameters of a SVM are usually pre-determined and fixed as more data are labeled. However, such kernel parameters are no longer optimal. Unless we can afford a held-out, labeled data set, it is hard to tune the kernel parameters online. The key reason is we do not have a good method to evaluate different kernel parameters while active learning proceeds. The standard methods like cross-validation and leave-one-out tend to fail because active learning brings in

biased data samples. Such failures were observed and discussed in [2]. An important future direction is to find a good online performance evaluation and kernel tuning method for active learning. Also, as active learning in batch mode is often convenient and fast, active learning for multi-class SVMs in batch mode needs to be further explored. Recent work in batch-mode active learning for two-class problems can be found in [14].

3. In BRSVM, it does not peform as well as random sampling in the highest dimensional data set (satimage) because the distributions of the number of examples in bins tend to be skewed in high dimensional data sets. For those data sets, BRSVM and random sampling have the potential to be used together. Instead of using one weighted exemplar for each bin, one can randomly sample several examples at a ratio proportional to the number of examples in this bin. Then several weighted exemplars would be used to represent the examples in this bin. This combination method can help when the examples distribution is skewed across the bins, and has the potential to improve BRSVM on high dimensional data sets. The combination method is an interesting future research direction.

REFERENCES

[1] D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37 – 66, 1991.

[2] Y. Baram, R. Yaniv, and K. Luz. Online choice of active learning algorithms. *Journal of Machine Learning Research*, pages 255–291, 2004.

[3] P. L. Bartlett and J. Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. In *Advances in Kernel Methods – Support Vector Learning*, pages 43–54. 1999.

[4] E. Bauer and R. Kohavi. An expirical comparison of voting classification algorithms: bagging, boosting and variants. *Machine learning*, 1999.

[5] R. E. Bellman. *Adaptive Control Processes*. Princeton University Press, 1961.

[6] C. Berg, J. P. R. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups*. Springer-Verlag, 1984.

[7] M. D. Berg(Editor), M. V. Kreveld, M. Overmars, O. Schwarzkopf, and M. D. Berg. *Computational geometry: algorithms and applications*. Springer, 2001.

[8] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algoritms*. Plenum Press, New York, 1981.

[9] D. Boley and D. Cao. Training support vector machines using adaptive clustering. In *SIAM International Conference on Data Mining*, 2004.

[10] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2003.

[11] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[12] L. Breiman. Random forest. *Machine Learning*, 45(1):5–32, 2001.

[13] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.

[14] K. Brinker. Incorporating diversity in active learning with support vector machines. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 59–66, 2003.

[15] C. J. C. Burges. Simplified support vector decision rules. In *International Conference on Machine Learning*, pages 71–77, 1996.

[16] C. J. C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector machines. In *Advances in Neural Information Processing Systems*, volume 9, pages 375–381, 1997.

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

[18] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems*, volume 13, pages 409–415, 2000.

[19] C. Chang and C. Lin. LIBSVM: a library for support vector machines (version 2.3), 2001.

[20] A. Ciobanu and H. D. Buf. *Automatic Diatom Identification*, chapter Identification by contour profiling and legendre polynomials, pages 167–186. World Scientific, 2002.

[21] W. G. Cochran. *Sampling Techniques*. John Wiley and Sons, Inc., 3 edition, 1977.

[22] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2 edition, 2001.

[23] L. F. Costa and R. M. C. Jr. *Shape analysis and classification*. CRC press LLC, 2001.

[24] T. G. Dietterich. Machine learning research: four current directions. *AI Magazine*, 18(4):97–136, 1997.

[25] T. G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

[26] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting and randomization. *Machine learning*, 40(2):139–157, 2000.

[27] J. X. Dong and A. Krzyzak. A fast svm training algorithm. *International Journal of Pattern Recognition and Artificial Intelligence*, 17(3):367–384, 2003.

[28] S. Dumais. Using svms for text categorization. *IEEE Intelligent Systems Magazine*, 13(4):21–23, 1998.

[29] W. DuMouchel, C. Volinsky, T. Johnson, C. Cortes, and D. Pregibon. Squashing flat files flatter. *Data Mining and Knowledge Discovery*, pages 6–15, 1999.

[30] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3:32–57, 1973.

[31] ELENA. ftp://ftp.dice.ucl.ac.be/pub/neural-nets/elena/database.

[32] S. Eschrich, J. Ke, L. Hall, and D. Goldgof. Fast accurate fuzzy clustering through data reduction. *IEEE Transactions on Fuzzy Systems*, 11(2):262–270, 2003.

[33] S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems*, volume 2, pages 524–532, 1990.

[34] S. Fine and K. Scheinberg. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001.

[35] S. Fischer and H. Bunke. *Automatic Diatom Identification*, chapter Identification using classical and new features in combination with decision tree ensembles, pages 109–140. World Scientific, 2002.

[36] M. G. Genton. Classes of kernels for machine learning: a statistics perspective. *Journal of Machine Learning Research*, 2:299–312, 2001.

[37] F. Girosi. An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10:1455–1480, 1998.

[38] D. Hand, H. Mannila, and P. Smyth. *Principles of data mining*. the MIT press, 2001.

[39] T. Hastie and R. Tibshirani. Classification by pairwise coupling. In *Advances in Neural Information Processing Systems*, volume 10, 1998.

[40] C. W. Hsu and C. J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

[41] M. K. Hu. Visual pattern recognition by moment invariants. *IRE Trans. Information theory*, IT(8):179–187, 1962.

[42] T. Jebara. *Machine Learning: Discriminative and Generative*. Kluwer Academic Publishers, 2004.

[43] T. Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Machines*, pages 169–184, 1999.

[44] J. Ke. Fast accurate fuzzy clustering through reduced precision. Master's thesis, University of South Florida, 1999.

[45] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to platt's smo algorithm for svm design. *Neural Computation*, 13:637–649, 2001.

[46] G. S. Kimeldorf and G. Wahba. Some results on tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33:82–95, 1971.

[47] J. Kohavi. Wrappers for feature subset selection. *Artificial Intelligence, special issue on relevance*, 97(1-2):273–324, 1997.

[48] K. Kramer. Personal communication, 2004.

[49] K. Kramer. Identifying plankton from grayscale silhouette images. Master's thesis, University of South Florida, 2005.

[50] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 3–12, 1994.

[51] R. E. Loke and H. d. Buf. *Automatic Diatom Identification*, chapter Identification by curvature of convex and concave segment, pages 141–166. World Scientific, 2002.

[52] T. Luo, K. Kramer, D. Goldgof, L. Hall, S. Samson, A. Remsen, and T. Hopkins. Active learning to recognize multiple types of plankton. In *17th conference of the International Association for Pattern Recognition*, volume 3, pages 478–481, 2004.

[53] T. Luo, K. Kramer, D. Goldgof, L. Hall, S. Samson, A. Remsen, and T. Hopkins. Recognizing plankton images from the shadow image particle profiling evaluation recorder. *IEEE Transactions on System, Man, and Cybernetics–Part B: Cybernetics*, 34(4):1753–1762, August 2004.

[54] D. Madigan, N. Raghavan, W. Dumouchel, M. Nason, C. Posse, and G. Ridgeway. Likelihood-based data squashing: a modeling approach to instance construction. *Data Mining and Knowledge Discovery*, 6(2):173–190, 2002.

[55] G. Matheron. *Random sets and integral geometry*. John Wiley and Sons: New York, 1975.

[56] C. J. Merz and P. M. Murphy. Uci repository of machine learning database. http://www.ics.uci.edu/ mlearn/MLRepository.html, 1999.

[57] D. Michie, D. J. Spiegelhalter, and C. C. Taylor. Machine learning, neural and statistical classification, 1994.

[58] P. Mitra, C. A. Murthy, and S. K. Pal. A probabilistic active support vector learning algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):413–418, 2004.

[59] P. Mitra, B. U. Shankar, and S. K. Pal. Segmentation of multispectral remote sensing images using active support vector machines. *Pattern Recognition Letters*, 25(9):1067–1074, 2004.

[60] K. Müller, A. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines. In *Seventh International Conference on Artificial Neural Networks*, pages 999–1004, 1997.

[61] H. T. Nguyen and A. Smeulders. Active learning using pre-clustering. In *Twenty-first International Conference on Machine learning*, 2004.

[62] T. Onoda, H. Murata, and S. Yamada. Relevance feedback with active learning for document retrieval. In *Proceedings of the International Joint Conference on Neural Networks 2003*, volume 3, pages 1757–1762, 2003.

[63] E. Osuna, R. Freund, and F. Girosi. Improved training algorithm for support vector machines. In *Proceedings of IEEE Neural Networks in Signal Processing*, 1997.

[64] E. Osuna and F. Girosi. Reducing the run-time complexity of support vector machines. In *Advances in Kernel Methods: support vector machines*, 1999.

[65] A. Owen. Data squashing by empirical likelihood. *Data Mining and Knowledge Discovery*, pages 101–113, 2003.

[66] J. M. Park. Convergence and application of online active sampling using orthogonal pillar vectors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1197–1207, 2004.

[67] D. Pavlov, D. Chudova, and P. Smyth. Towards scalable support vector machines using squashing. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 295–299, 2000.

[68] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: networks of Plausible Inference*. Morgan Kaufmann, 1997.

[69] I. Pitas. *Digital image processing algorithms and applications*. John Wiley and Sons, Inc., 2000.

[70] J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. The MIT Press, 1999.

[71] J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In *Advances in Large Margin Classiers*, pages 61–74, 2000.

[72] D. N. Politis, J. P. Romano, and M. Wolf. *Subsampling*. Springer, 1999.

[73] J. R. Quinlan. *C4.5: Programs from empirical learning*. Morgan Kaufmann, 1993.

[74] G. Ratsch, T. Onoda, and K. Muller. Soft margins for adaboost. *Machine Learning*, 42(3):287–320, 2001.

[75] A. Remsen, T. L. Hopkins, and S. Samson. What you see is not what you catch: a comparison of concurrently collected net, optical plankton counter, and shadowed image particle profiling evaluation recorder data from the northeast gulf of mexico. *Deep Sea Research Part I: Oceanographic Research Papers*, 51:129–151, 2004.

[76] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

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

[78] S. Samson, T. Hopkins, A. Remsen, L. Langebrake, T. Sutton, and J. Patten. A system for high resolution zooplankton imaging. *IEEE journal of ocean engineering*, pages 671–676, 2001.

[79] L. M. Santos and H. D. Buf. *Automatic Diatom Identification*, chapter Identification by gabor features, pages 187–220. World Scientific, 2002.

[80] M. Sassano. An empirical study of active learning with support vector machines for japanese word segmentation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 505–512, 2002.

[81] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proc. 17th International Conf. on Machine Learning*, pages 839–846, 2000.

[82] B. Schölkopf, S. Mika, C. Burges, P. Knirsch, K. R. Muller, G. Rätsch, and A. Smola. Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, 1999.

[83] B. Schölkopf and A. J. Smola. *Learning with kernels*. The MIT Press, 2002.

[84] J. Shawe-Taylor and P. L. Bartlett. Structural risk minimization over data-dependent hierarchies. *IEEE Trans. on Information Theory*, 44(5):1926–1940, 1998.

[85] Y. C. L. Shih, J. D. M. Rennie, and D. R. Karger. Text bundling: Statistics based data-reduction. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 696–703, 2003.

[86] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis and Machine Vision*. Thomson-Engineering, 2 edition, 1998.

[87] X. Tang, F. Lin, S. Samson, and A. Remsen. Feature extraction for binary plankton image classification. *accepted by IEEE Journal of oceanic engineering*, 2004.

[88] X. Tang, W. K. Stewart, L. Vincent, H. Huang, M. Marra, S. M. Gallager, and C. S. Davis. Automatic plankton image recognition. *Artificial Intelligence Review*, 12(1-3):177–199, 1998.

[89] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *Proceedings of ICML-00, 17th International Conference on Machine Learning*, pages 999–1006, 2000.

[90] V. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer, 2001.

[91] V. Vapnik and A. Chervonenkis. The necessary and sufficient conditions for consistency in the empirical risk minimization method. *Pattern Recognition and Image Analysis*, 1(3):283–305, 1991.

[92] V. N. Vapnik. *The nature of statistical learning theory*. Springer, 2000.

[93] L. Wang, K. L. Chan, and Z. h. Zhang. Bootstrapping svm active learning by incorporating unlabelled images for image retrieval. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 629–634, 2003.

[94] M. K. Warmuth, G. Rätsch, M. Mathieson, J. Liao, and C. Lemmen. Support vector machines for active learning in the drug discovery process. *Journal of Chemical Information Sciences*, 43(2):667–673, 2003.

[95] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for svms. In *Neural information processing systems*, pages 668–674, 2000.

[96] M. H. F. Wilkinson, A. C. Jalba, E. R. Urbach, and J. B. T. M. Roerdink. *Automatic Diatom Identification*, chapter Identification by mathematical morphology, pages 221–244. World Scientific, 2002.

[97] C. K. I. Williams and M. Seeger. Using the nystrom method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, pages 682–688. the MIT Press, 2001.

[98] T. F. Wu, C. J. Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005, 2004.

[99] H. Yu, J. Yang, and J. Han. Classifying large data sets using svm with hierarchical clusters. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 306–315, 2003.

[100] C. Zahn and R. Z. Roskies. Fourier descriptors for plane closed curve. *IEEE transaction on computers*, C(21):269–281, 1972.

ABOUT THE AUTHOR

Luo, Tong received the Bachelor of Engineering degree in Electrical Engineering from Tsinghua Universtiy in 1997, the Master of Engineering degree in Control Theory and System from Institute of Automation, Chinese Academy of Sciences in 2000.

He is currently a Ph.D student in Department of Computer Science and Engineering at University of South Florida. His research interests are in machine learning, data mining, and pattern recognition.