Labeling Large Scale Social Media Data using Budget-driven One-class SVM classification

by

© Hao Yuan

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the requirements for the degree of

Master of Science

Department of Computer Science

Memorial University of Newfoundland

May 2016

St. John's Newfoundland

Abstract

The social media classification problems draw more and more attention in the past few years. With the rapid development of Internet and the popularity of computers, there is astronomical amount of information in the social network (social media platforms). The datasets are generally large scale and are often corrupted by noise. The presence of noise in training set has strong impact on the performance of supervised learning (classification) techniques. A budget-driven One-class SVM approach is presented in this thesis that is suitable for large scale social media data classification.

Our approach is based on an existing online One-class SVM learning algorithm, referred as STOCS (Self-Tuning One-Class SVM) algorithm. To justify our choice, we first analyze the noise-resilient ability of STOCS using synthetic data. The experiments suggest that STOCS is more robust against label noise than several other existing approaches. Next, to handle big data classification problem for social media data, we introduce several budget driven features, which allow the algorithm to be trained within limited time and under limited memory requirement. Besides, the resulting algorithm can be easily adapted to changes in dynamic data with minimal computational cost. Compared with two state-of-the-art approaches, Lib-Linear and kNN, our approach is shown to be competitive with lower requirements of memory and time.

Acknowledgements

First, I want to thank my supervisors, Dr.Minglun Gong and Dr.Jian Tang for their help in our study and research in the last two years. Without their help, it would be impossible for me to finsih my thesis successfully. They helped me to learn how to do research, how to face the difficulties and how to be confident, which will influence my whole life. I really appreciate their care, help and tolerant. Then I want to thank my friends Yiming Qian, Shibai Yin, Wenbin Zhang, Yinghan Zhang for their help and encouragement in my life. I would like to thank the faculty and staff members in our department as well who are friendly and helpful. At last, I thank my parents for their care and support, they making my dream come true to study abroad.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Data mining and machine learning are two research areas in computer science and statistics, and have been widely studied in recent years (e.g. [30], [46]). Both of them deal with the construction and study of algorithms that can learn from data. Machine learning tasks can be of several forms, such as supervised learning, unsupervised learning, and semi-supervised learning. All of them are learning from data examples. Supervised learning learns from labeled training data, then analyzes the internal relation between examples and builds classifiers with feedbacks, whereas unsupervised learning tries to find the hidden structure from unlabeled data and does not have any error or reward signal to evaluate a potential solution. The semi-supervised learning is a combination of supervised learning and unsupervised learning, which makes use of both labeled and unlabeled data examples, generally, a small amount of labeled data with much more unlabeled data.

Supervised learning which is also known as data classification in machine learning, plays an important role in data mining and machine learning. There are many well-known classification algorithms, such as Naïve Bayes Classifier, Decision Tree, k-Nearest Neighbor Algorithm, and Adaptive Boosting Algorithm.

These classification methods are used to support many applications in different areas, e.g. economics, geography, finance, medical science, etc. Most of the existing approaches only address small or median scale problems; however, we are in the era of social media and big data. Social media is defined as a group of Internet-based applications that build on the ideological and technological foundations of Web 2.0, and that allow the creation and exchange of user-generated content [29]. Big data is high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation. Even social media and big data are two different concepts, social media problems are usually big data problems.

Social media problems draw a lot of attention in recent years because of the rapid development of Internet and information technology. The classification of social media problem introduces new challenges, because the data sets of social media problems are often different from other problems due to their dynamic, noisy, and large scale features.

The focus of this thesis is to solve the big data classification problem for social media data under limited budget environment, where there are three main challenges [3]: (1) *volume*, which corresponds to the ever increasing amount of data. The rapid expansion of social network services (SNSs) results in billions of users, astronomical information. Not only the size of classification data sets becomes larger and larger, but also the number of features in each example. For example, Facebook reports about 6 billion new photos every month and 72 hours of video are uploaded to YouTube every minute [37]. It leads to the problem that the excessive data volume cannot fit in computer memory, especially for commodity machines, whereas most of existing methods assume data can be stored in memory. (2) *velocity*, which means data is streaming in at unprecedented speed. Users all around the world are using the social media platforms all the time. To cope with that, online learning model is proposed to provide immediate response to the newly generated data, e.g. [49] [13] [32]. (3) *variety*, which refers to data diversity. Information in social media is provided by the users and different providers have different judgments and criteria. As a result, real data is often heterogeneous, coming in different types of formats and from different sources. Some users may even provide fake information intentionally, which leads to large amount of noise. However, many classification methods are sensitive to noise such that their performance decreases significantly with the presence of noise.

In social media classification problems, there are two types of noise: feature noise and label noise. Feature noise is represented by errors that are introduced to attribute values, whereas label noise alters the observed labels assigned to instances. To my knowledge, a lot research work focuses on how to handle feature noise, but there is little literature on how label noise shall be dealt with. Label noise is an important issue in classification, with many potential negative consequences. For example, the accuracy of predictions may decrease, whereas the complexity of inferred models and the number of necessary training examples may increase. In the recent survey paper of Benot Frenay [18], a new taxonomy of label noise has been proposed based on whether the noise is related to any class information. There are three types of label noise in this taxonomy: the noise completely at random (NCAR) model , the noise at random (NAR) model and the noise not at random (NNAR) model . There are four random variables depicted to model the label noise: X is the vector of features, Y represents the true class for the an example, whereas $\tilde{Y}$ means the observed label of the example. E is a boolean value that reflects whether the observed label of a example is incorrect ($Y \neq \tilde{Y}$). The NCAR model is defined as that the occurrence of an error E is independent of the other random variables, including the true class itself, whereas the NAR model means that E is still independent of X but depending on the true class Y. The NNAR model reflects that E depends on both variables X and Y.

This thesis focuses on the classification of social media problem, which is always large scale, multi-class and with label noise. The Yahoo! Large-scale Flickr-tag Image data set is selected as the social media problem, which is one of the multimedia grand challenges in ACM Multimedia 2013 [2]. In this challenge, there are 2,000,000 images and 10 image classes: nature, food, people, wedding, music, sky, london, beach, 2012, travel. By using bag of word model, each example has 400 features, which means the problem is high dimensional. Meanwhile, all the images, annotations are provided by Flickr users, and hence they are not accurate and consistent since they are reflections of their tagging behavior. These facts reflect that this dataset exhibits the three challenges mentioned above.

1.1 Contributions

Motivated by the above challenges, this thesis proposes a practical budget-driven One-Class Support Vector Machine (SVM), which is specifically designed for big data classification. We call an approach budget-driven when the approach is required to solve problems with limited budget, which means low time cost and memory requirement. One-Class SVM is a variant of SVM to determine whether new test examples are members of a specific class, when there is only training data of one class. The proposed approach is extended from existing algorithm, called Self-Tuning One-Class SVM (STOCS) [13] [40]. We choose to extend based on STOCS algorithm for two main reasons: (1) STOCS is developed from Support Vector Machine (SVM) and hence inherits the flexibility of using kernels, which helps to address the issue of heterogeneous data; (2) STOCS incorporates online learning framework which could address the issue of *velocity* in big data problem. However, STOCS algorithm requires long training time and expensive memory storage, making it not suitable for big data problems. Hence, the main contributions made in this thesis are the following:

Noise-resilient ability study: In Chapter 4, we study the noise handling ability of STOCS by comparing STOCS with five well-known classification methods: Naïve Bayes, Decision Tree C4.5, Classification and Regression Tree, K-Nearst Neighbors Algorithm and Support Vector Machine. We first give a brief description about heuristics of these methods and analyze their robustness against noise. Second, based on the new proposed taxonomy [18], we conduct the comparisons using synthetic benchmark datasets, under both binary classification cases and multi-class classification cases with different types of noise and different noise percentages. From our experiments, the results show that STOCS performs better under multi-class cases than binary cases. For most benchmark datasets, STOCS could achieve competitive, even better performance compared with other well-known algorithms. It is also concluded that STOCS is the most robust method against label noise among all methods in experiment. These encouraging facts motivate us to extend STOCS and design our approach specifically for social media classification under a limited-budget environment.

Budget-driven big data classification: In Chapter 5, we extend STOCS algorithm and design our approach specifically for the social media big data classification problem. The most important challenge of big data problems is huge computational resource requirements in both memory and time cost. To alleviate these bottlenecks on big data, we propose to extend STOCS in the directions described below.

First, all examples are trained repetitively in STOCS online learner and it requires a large number of iterations to converge, which leads to long execution time. However, redundancies exist in social media data, and it is unnecessary to train on these redundant data. In contrast, we believe that for big data problem, competitive prediction accuracy can be achieved by training on only a portion of the training examples. In our approach, we relax the convergence condition so that the time cost will be reduced.

Second, notice that the algorithm in the paper of STOCS [40] [13] stores all data in the memory since training examples are used in multiple rounds of training. In contrast, our approach randomly reads and trains only one example in each iteration, allowing to only store one training example instead of the whole training set. Hence, the memory requirement is also reduced, showing that our approach is particularly useful on big data classification when data cannot fit in memory.

Finally, as the training set becomes larger, the number of support vectors increases significantly, especially when training multiple Competing One-Class SVM for a big data problem. We propose to reduce the number of support vectors to further reduce the computational cost. Our approach assumes that if the support vectors with higher weights, referred as dominant support vectors, are stored, the prediction result will remain competitive. By only keeping dominant support vectors, the model refinement in each iteration would be more efficient.

Experimental results show that the performance of our approach is competitive while our approach requires much less memory and is much more efficient than STOCS. The result reflects that it is promising to apply our approach for social media problems.

Social Media Application: In Chapter 6, we apply our approach for social media problems and conduct comparison with the state-of-the-art approaches, such as LibLinear and kNN algorithm. We choose the Yahoo! Large-scale Flickr-tag Image Classification Grand Challenge [2] as the social media problem to be solved. Quantitative evaluations are performed on different problem size levels. The proposed approach achieves superior classification performance on extremely large data, compared with two state-of-the-art methods, with faster convergence and lower memory usage.

We also employ the concept of confidence to make our approach more suitable to industry areas, such as advertisement and promotion, where the core idea is to only classify an example when we have high confidence. The incorporation of confidence will help to solve the "low-budget" promotion problems in the real world. For example, when a small company is trying to do promotion and the budget is not enough to cover the whole group of potential clients, then it will be promising to only cover the group of people with higher chance to buy its products.

In summary, this thesis addresses the big data classification problem for social media data under limited budget environment by extending and improving the existing

STOCS online learner. In the next chapter, the background and related work of our approach will be introduced. Chapter 3 presents the details of the foundation of our approach: STOCS. Chapter 4 shows the study of noise-resilient ability of STOCS. Our improvement and the specific design for social media classification are presented in Chapter 5. Experimental results and comparisons to existing methods are provided in Chapter 6. Finally, Chapter 7 concludes the thesis and suggests future research directions.

Chapter 2

Background and Related Work

In this chapter, we present the background and some related work of our approach and STOCS. Starting with the introduction of traditional classification algorithms. In section 2.2, some related work on label noise are discussed. Then the existing work of budget-driven classification is presented. At the end of this chapter, we provide some existing related work on the Large-scale Flickr-tag Image Classification Grand Challenge [2].

2.1 Well-Known Classification Algorithms

In this section, several well-known classification approaches are introduced: Naïve Bayes Classifier [15], Decision Tree [42] [43], k-Nearest Neighbor Algorithm [52], Adaptive Boosting Algorithm [20].

The Naïve Bayes Classifier is one of simplest classification methods, which is based on Bayes rule with the probabilistic knowledge about the data. As the Naïve Bayes Classifier has a simple model and does not need any complicated iterative parameter estimation schemes, it can be applied to huge data sets. Related research shows that the Naïve Bayes often performs surprisingly well: maybe not the best possible classifier in any particular application, but it can usually be relied on to be robust and effective [59].

Decision Tree is a popular classification method, which builds a tree system to classify a data example based on the different attributes of that example. Starting from the root, an example will pass the internal levels through the branches and finally reach one of the leaves. Here the leaves mean different classes, branches represent different value ranges of one attribute, and different levels mean different attributes. There are two types of decision tree widely applied, C4.5 and Classification and regression tree (CART). They both developed from the idea of decision tree but still are different in several respects: CART always produces binary tree, but C4.5 grows with multiway splitting; CART uses the Gini diversity index to select attribute where C4.5 employs the concept of entropy and information-based criteria.

Another traditional classification algorithm is k-Nearest Neighbor (k-NN) method, which is also among the simplest machine learning algorithms. The parameter k is used in the algorithm to classify an unlabeled example by assigning the most frequent label among the k nearest training examples. Euclidean distance is a commonly used distance measurement for k-NN. In the common cases, weights are added into k-NN to control the influence of different neighbors.

Ensemble learning deals with methods which employ multiple learners to solve a problem, and the generalization ability of an ensemble algorithm is usually much better than that of a single learner. The AdaBoost proposed by [20] is one of the most important ensemble learning methods with a solid theoretical foundation, very accurate prediction, great simplicity, and wide and successful applications. AdaBoost algorithm works with multiple learners and combines the weights to improve the performance. Boosted classifier is less susceptible to the overfitting problem than other learning algorithms, however, is more sensitive to noisy data and outliers.

2.2 Label Noise

When people are trying to solve real-world classification problems, they find that real-world datasets always contain noise, which is defined as anything that obscures the relationship between the features of an instance and its class. The ubiquity of noise becomes an important issue for practical machine learning. In the literature, two types of noise are distinguished: feature (or attribute) noise and label (or class) noise. Research shows that label noise is potentially more harmful than feature noise, which highlights the importance of dealing with this type of noise [63] [9]. The harmful impact of label noise is explained by the fact that there are many features in an example, whereas there is only one label. Besides, the importance of each feature for learning may vary, whereas labels always have a large impact on learning.

There are three main sources of label noise: first, the information which is provided to the expert may be insufficient to perform reliable labelling; second, errors can occur in the expert labelling itself; finally, when the labelling task is subjective, there may exist an important variability in the labelling by different experts. A new taxonomy of label noise is proposed by Benot Frenay [18]: the noise completely at random model (NCAR), the noise at random model (NAR), and the noise not at random model (NNAR).

In the literature, there exist three main approaches to take care of label noise [38] [35] [54] [55] [60]. The first approach relies on algorithms which are naturally robust to label noise, which means the learning of the classifier is assumed to be not too sensitive to the presence of label noise. Several studies have shown that some algorithms are less influenced than others by label noise even though label noise is not really taken into account in this type of approach. Some examples are ensemble methods like LogitBoost [21], BrownBoost [19] and split criteria for decision trees like the imprecise info-gain [1]. STOCS algorithm and our proposed extension belong to this first approach.

The second approach tries to improve the quality of training data using filtering methods, where noisy labels are typically identified and eliminated before training phase occurs. Filter methods are cheap and easy to implement, however, some of them tend to remove a substantial amount of data. Here are some examples of this approach: model predictions-based filtering [56], k nearest neighbors-based methods [58], neighborhood graph-based methods [45].

In the third approach, algorithms directly model label noise during learning or have been modified to take label noise into account in an embedded fashion. The advantage of this approach is to separate the classification model and the label noise model, which allows using information about the nature of label noise. Some examples are Bayesian approaches [39], frequentist methods [16], clustering-based methods [7] and model-based methods [44].

2.3 Budget-Driven Classification

Linear classification models have been shown to handle big data classification well [61] [12] [27]. The computational cost in big data is a great challenge, in both memory requirement and time cost. One solution for this challenge is to design the classification technique under limited budget environment. Existing work on training linear models under a limited-budget environment can be categorized into memory-driven [61] [12] [5] [62] and time-driven [27] [49] [37] approaches. Memory-driven approaches focus on solving the problem when data cannot fit in memory. LibLinear [17] is the most popular solver for linear SVM. Yu *et al.* [61] proposed a block optimization framework to handle the memory limitations of LibLinear. They split data into blocks and store them in compressed files. By training on one block of examples at a time, the required training memory is reduced. Following the idea of Yu [61], various block-optimization based approaches are also proposed [12] [5] [62].

Time-driven approaches try to solve the problem in linear time. Joachims [27] proposed SVMPerf by reformulating the original SVM function to a structural SVM, achieving linear time complexity for sparse training features. Unlike SVMPerf and many other methods, which formulate SVM as a constrained optimization problem, Shalev-Shwartz *et al.* [49] relied on an unconstrained optimization formulation of linear SVM and achieved liner training time. Very recently, Nie *et al.* [37] proposed a new primal SVM solver with linear computational cost for big data classification.

Another way to scale up big data classification is through parallelism [23]. However, the development of parallel classification is limited by not only parallel program implementation difficulties but also by data synchronization and communication overheads on distributed systems. More recent advances of big data classification are surveyed in paper [3].

Different from most existing approaches that focus on solving linear SVM, our approach relies on a competing One-Class SVMs model, and is designed specifically for social media classification that runs on a commodity PC with limited memory and processing power. In addition, our approach is not constrained to linear kernel, making it easily customizable for handling a variety of data.

2.4 Large-scale Flickr-tag Image Classification Grand Challenge

The 21st ACM International Conference on Multimedia 2013 [2] provides an image classification challenge called Large-scale Flickr-tag Image Classification Grand Challenge. The challenge is trying to solve large scale social media image classification problem, where images are collected from Flickr.com. In this challenge, there are hundreds of thousands of training images per class, much larger than any existing image classification challenge. Besides, each class is composed of visually diverse sub-classes, however, some of these sub-classes could be visually unrelated to the root class. Additionally, high amount of label noise is present in the data.

The conference provides both original images datasets and precomputed sets of features. The precomputed features are generated by employing the bag of word model, where there are 400 features in an example. There are two ways of entering the contest: one is using the provided feature vectors, and the other is to use source images. Two solutions have been provided for this chanllenge and they are both using source images. In this section, we will briefly describe both solutions [51] [34].

Flickr-tag Prediction using Multi-modal Fusion and Meta Information This solution [51] conducts experiments by combining multi-features and different classification models based on the original images datasets. In this approach, four feature combinations are adopted: Hessian Affine SIFT with Vector of Locally Aggregated Descriptors (VLAD), Grid Color Moment with VLAD, Dense SIFT with Locality constraint Linear Coding (LLC), Local Binary Patterns (LBP) with LLC. The authors first apply linear SVM and k-Nearest Neighbors classifier to deal with the classification problem separately and then integrate these two methods together. When integrating these two methods, the authors chose to use the late fusion strategy by simply adding the decision score of different classifiers. By combining multi-features and classifiers, the prediction accuracy is improved from approximately 35% to 58%, which demonstrates that combining multi-features and classifiers can noticeably improve the tag prediction performance.

Beside features and models fusion, the authors proposed to adopt meta information for tag-prediction, because of the observation that some tags are given mainly based on the meta information rather than the visual content. Without meta information, some tags are doomed to be ill predicted by visual content. For example, label 2012, is not tagged based on the visual information but based on the uploading time. Based on this observation we eliminated the class "2012" from the data set, because for such tags, visual content along cannot yield satisfactory performance and we choose to use the provided visual features only.

In summary, this approach integrates multi-features and different classification models and obtains promising results. However, this method is based on batch learning, which means it has high memory requirement and expensive time cost. When even only one new example is added into training set, this approach needs to re-train the whole dataset again. Hence, it is difficult to apply this approach for dynamic datasets.

Scalable Training with Approximate Incremental Laplacian Eigenmaps and PCA The main contribution of this solution [34] is the use of fast and efficient features with a highly scalable semi-supervised learning approach, the Approximate Laplacian Eigenmaps (ALEs) and its extension, by computing the test set incrementally for learning concepts in time linear to the number of images (both labelled and unlabeled) [34]. A combination of two local visual features, Scale-Invariant Feature Transform (SIFT) and RGB-SIFT, together with the Vectors of Locally Aggregated Descriptors (VLAD) feature aggregation method and Principal Component Analysis (PCA) are used to improve the efficiency and time complexity.

The proposed approach is based on Semi-Supervised Learning (SSL) by constructing Laplacian Eigenmaps (LEs) approximately and incrementally. The authors use a scalable manifold learning framework on top of ALE, called SMaL. The time complexity of SMaL is linear to the number of images, making it possible to use the graph Laplacian in large-scale problems. Results show that this methodology achieves competitive accuracy compared to the baseline (linear SVM) in small training sets, and the performance improves quickly as the size of training data increases. For instance, if 1K training images are used, the SMal yields an accuracy of 35.37% while linear

SVM achieves 35.43%, but if 100K images are used, the accuracy of SMal is 38.42%, higher than the linear SVM, which is 38.31%.

In summary, this method significantly decreases the computational requirements of training in view of large amounts of data by an approximate incremental semi-supervised learning approach, leveraging VLAD vectors that when 150K training images are used, the training time of SMaL is 3 mins, whereas it takes 71 mins for linear SVM. However, from the details of results, this approach is more sensitive to the presence of label noise. For the more noisy class, such as ”2012” and ”nature”, the performance of the SMaL is about 5% less accurate than the linear SVM.

Chapter 3

Prerequisites: SVM, One-Class SVM, and STOCS

In this chapter, the concepts of SVM and One-class SVM are presented, followed by STOCS algorithm, which the proposed approach is built upon. Besides, the properties of STOCS will be discussed to show its potential to solve the big data classification problem under limited budget environment by extending STOCS.

3.1 Support Vector Machine

Support Vector Machine (SVM) is one of the most important supervised learning methods, which was proposed by Boser, Guyon and Vapnik in 1992. The underlying motivation of SVMs is to construct a hyperplane or a set of hyperplanes in a high-dimensional or an infinite-dimensional space, which can be used for classification, regression, and other tasks. There are many possible separating hyperplanes in one problem, where SVM is trying to find out the optimal hyperplane, which means the one that has the largest distance to the nearest training data examples of any class (known as separating distance).

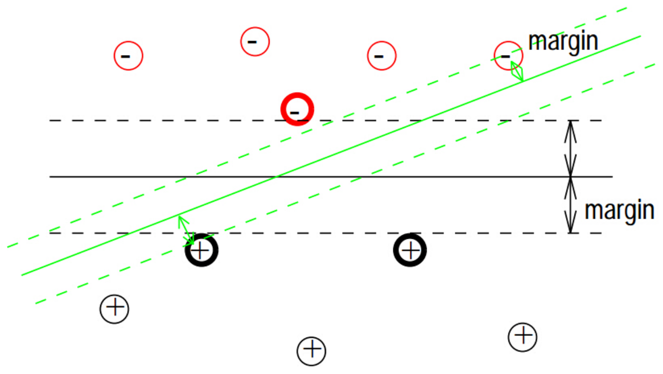


Figure 3.1: Given two classes of data points shown in graph, two possible separating planes are presented. It is clear that the black straight line is the optimal separating choice. Margins are shown in the graph and support vectors are the examples on the margins.

How the SVM uses a hyperplane to separate data examples is shown in Figure 3.1. Assuming that the training set is $\{(X_1, y_1), (X_2, y_2), \cdots, (X_n, y_n)\}$, where the class label is $y_i = +1$ or -1 and X_i is a p-dimensional vector. Any plane in a p-dimensional space can be expressed by equation $W \cdot X = b$, where b is a constant value, X, W is a p-dimensional vector. As shown in Figure 3.1, each solid plane showing the possible separating hyperplane is accompanied by two dashed planes, which represent the margins. By selecting appropriate values for W and b, two margins can be expressed as: $W \cdot X = b + 1$ and $W \cdot X = b - 1$. Then the separating distance becomes $2/\left|W\right|^2$, where SVMs try to minimize $\left|W\right|^2$. Then the classification problem can be converted to the following optimization problem by SVMs:

$$\begin{aligned} &\min \quad \frac{||W||^2}{2} \\ &\text{subject to} \quad y_i(w \cdot x_i + b) \geq 1, i = 1, \cdots n. \end{aligned} \tag{3.1}$$

However, there are many cases where the examples are not separable and a separating hyperplane does not exist. An example of a non-separable case is shown in Figure 3.2. In this case, SVMs try to find the hyperplane that split examples as well as possible. Then the problem is converted to:

$$\begin{aligned} &\min \quad \frac{||w||^2}{2} + C\sum_{i=1}^{m} \xi_i \\ &\text{subject to} \quad y_i(w \cdot x_i + b) \geq 1 - \xi_i, \xi_i \geq 0, i = 1, \cdots, m. \end{aligned} \tag{3.2}$$

where ξ_i is non-negative slack variable, which measures the degree of misclassification of the data and C is a constant value.

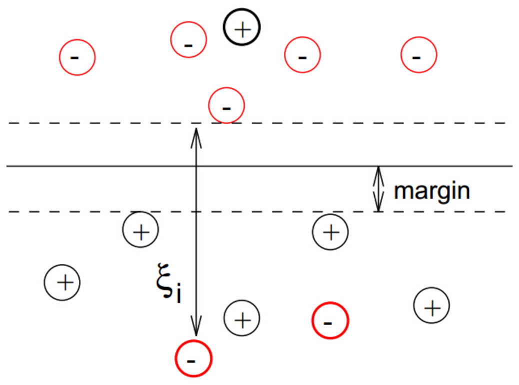


Figure 3.2: An example of a non-separable case, where a separating plane does not exist. In this case, SVMs try to find the hyperplane that split examples as best as possible, and this is still a quadratic problem.

For the non-separable cases, non-linear SVMs are proposed [48]. It is proposed that the original finite-dimensional space be mapped into a higher-dimensional space, presumably making the separation easier in that space. Formally, preprocess the data with: $x \to \Phi(x)$, then learn the map from $\Phi(x)$ to y: $f(x) = w \cdot \Phi(x) + b$. An example of polynomial mapping is shown in Figure 3.3, from two dimensional space to three dimensional space, with following rules:

$$\Phi : R^2 \to R^3 \tag{3.3}$$

$$(x_1, x_2) \to (z_1, z_2, z_3) := (x_1^2, \sqrt{2x_1x_2, x_2^2}) \tag{3.4}$$

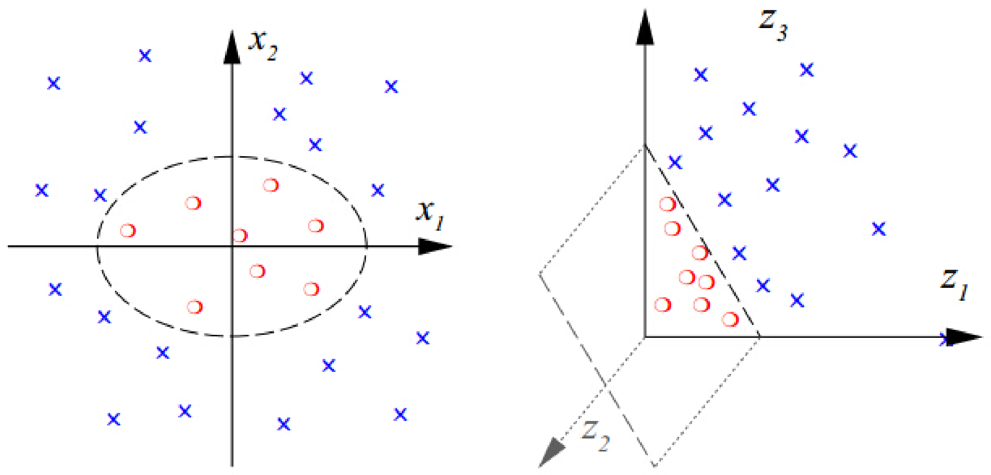


Figure 3.3: An example of polynomial mapping. In the left figure, the two classes cannot be separated by a plane. After mapping these two dimensional examples into three dimensional space, we find that the examples can be separated by a hyperplane. Mapping to higher dimensional space is a common way to solve non-separable cases.

The dimensionality of $\Phi(x)$ can be very large, making w hard to represent explicitly in memory, and hard for the quadratic program to solve. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function $k(x, y)$ selected to suit the problem. With the represented theorem $w = \sum_{i=1}^{m} \alpha_i \Phi(x_i)$, the decision function becomes:

$$f(x) = \sum_{i=1}^{m} \alpha_i \Phi(x_i) \Phi(x) + b \tag{3.5}$$

With kernel function:

$$f(x) = \sum_{i=1}^{m} K(x_i, x) + b \tag{3.6}$$

3.2 One-Class SVM

Traditionally, many classification algorithms try to solve the binary or multi-class classification situations. However, in some cases, problems have data examples for a single class and try to classify new examples as in the class or out of the class. A method for this task is the One-Class Support Vector Machine, which gained much popularity in the last two decades. There are two main approaches of One-Class SVM, the one proposed by Schölkopf [48] and that by Tax and Duin [53].

For the first approach, One-Class SVM basically separates all the examples from the origin and maximizes the distance between the hyperplane and the origin point in feature space [48]. This results in a binary function which captures the regions in the input space where the probability density of the data lies. The function returns +1 when a newly encountered example is in the class region, and -1 elsewhere.

The second approach takes a spherical boundary, instead of a planar approach, which is very similar to the fundamental method of our approach. The algorithm obtains a spherical boundary, in feature space, around the data, which means the examples inside spherical boundary will be classified as in the class. Then the optimal solution becomes the hypersphere with minimum volume containing all "in-class" training examples. The minimization problem can be expressed by the following mathematical expression:

$$\begin{aligned} \min_{R,a} \quad & ||R||^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & \|x_i - a\|^2 \leq R^2 + \xi_i, \\ \text{and} \quad & \xi_i \geq 0, i = 1, \cdots, n, \end{aligned} \tag{3.7}$$

where C is the penalty parameter and controls the trade-off between the volume and the errors, R means the radius of the hypersphere, a represents the center of hypersphere, and ξ_i is non-negative slack variable. After solving Equation 3.7 using Lagrange multipliers ω_i, a new data point can be tested if it is in or out of class. It is considered as in-class when the distance between data point and the center is smaller than or equal to the radius. Then the score function with kernel function becomes:

$$f(x) = \sum_{i=1}^{n}\omega_i k(x, x_i) \geq C - R^2/2, \tag{3.8}$$

where C only depends on support vectors x_i but not on testing point x when using Gaussian kernel.

Besides one class cases, One-Class SVM can also handle binary and multi-class problems. The key idea is to train and maintain multiple One-Class SVM models, where each model learns the data distribution of one class. Each model may label a testing example as inlier or outlier independently, and hence *competes* with all other One-Class SVM models. A testing example is *jointly* labeled by assigning it to the model returning the highest score.

3.3 Self-Tuning One-Class SVM

3.3.1 Online Learning

In most machine learning research, batch learning is the standard learning strategy for classification tasks. For batch learning, the training phase does not start until all training examples are collected. However, in social media problems, as the datasets become larger and larger, it becomes difficult to store all data into memory as the size of training data could reach tens of Gigabytes.

Online learning strategy is a promising way to overcome this challenge. Previous study by Bottou and Bousquet [6] demonstrated that a similar or even better generalization performance can be achieved using online learning with less computational cost than batch learning through minimizing a quadratic objective function. Gong *et al.* [13] proposed a method which incorporates One-Class SVM and online learning strategy. It is not only simple and efficient, but also work well with large scale datasets. Then Qian [40] improved the method by employing the idea of self-tuning and adjustable kernel function, called STOCS.

Online learning is a powerful and popular way of dealing with sequential prediction or classification problems, such as weather forecasting, predicting stock market trends, and deciding which ads to present on a web page. An online learning algorithm observes a stream of examples and data is processed for prediction (classification) immediately while it is observed, thus the user usually only has to wait a short time for the response. Online learning receives immediate feedback about each prediction and uses this feedback to update the classification model to improve its accuracy on subsequent predictions. Hence, it requires to only allocate memory for the current training example and the already-selected support vectors, rather than storing all the data.

When a new example is observed during training, only a simple score function computation is required to update models, allowing STOCS to react quickly to deal with data *velocity*. In contrast, batch learning requires to train on the whole new dataset again which is time consuming. Besides, for online learning, users may suspend the training process anytime by stop feeding data, and the partially trained model can be used to perform classification while achieving reasonable accuracy.

Based on these facts, we follow the online learning learner proposed by Qian [40]. Assuming that $f_t(\cdot)$ be a score function of example t, $k(\cdot,\cdot)$ be a kernel function, ω_t is a non-negative weight of example t and x_i is the i^{th} support vector. If an example x_t is collected to be trained, the score function is:

$$f_t(x_t) = \sum_{i=1}^{t-1} \omega_i k(x_i, x_t), \tag{3.9}$$

and the update rule for weights is:

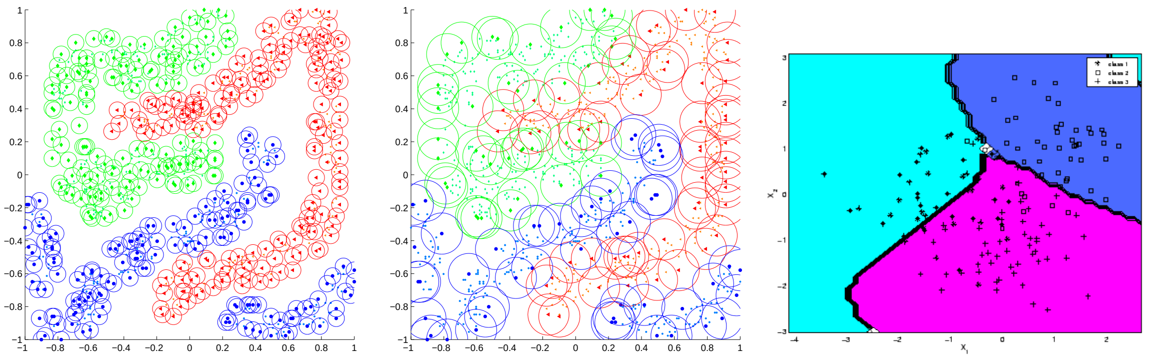
$$\begin{aligned} \omega_t &= \text{clamp}\left(\tfrac{\gamma-(1-\tau)f_t(x_t)}{k(x_t,x_t)}, 0, (1-\tau)\chi\right), \\ \omega_i &\leftarrow (1-\tau)\omega_i \qquad \forall i = 1,\ldots,t-1, \end{aligned} \tag{3.10}$$

in which $\gamma := 1$ is the margin, $\tau \in (0,1)$ is the decay parameter, and $\chi > 0$ the cut-off value which is used to handle noisy training data. clamp$(\cdot, A, B)$ is an identical function of the first argument bounded by A and B.

This approach builds One-Class SVM model for each class and keeps different support vectors lists for each class. When there is a newly encountered example, it will firstly calculate the scores of this example for each One-Class SVM model by using the score Equation 3.9 using the existing support vectors and weights. If the score of a new example calculated from the model of its corresponding class is high enough, we believe the existing models are good enough to represent this example, and this example will not be selected as a support vector; otherwise, it will be added into the corresponding support vectors list with a computed weight.

However, there are several parameters that need to be tuned well to achieve promising results, especially the parameter τ. The parameter τ has to be large enough to control the weights of support vectors, because at first, the support vectors lists are empty and each example receives score zero from the score function, then the first selected support vectors tend to have greater weights, which are affected by the parameter τ and need to be reduced in later iterations. On the other hand, the parameter τ requires to be close enough to zero to make the training phase converge. One solution for τ is proposed that $\tau = \exp(-\frac{t}{\xi})$, where parameter ξ controls how fast the decay parameter τ is decreasing.

In order to make Online learning One-Class SVM simpler and easier, Qian [40] propose to employ the concept of self-tuning and remove some parameters, such as τ. After the removal of parameter τ, the new score function and weight updating



(a) Result of our method with $\sigma = 0.1$, $\chi = 0.5$

(b) Result of our method with $\sigma = 0.25$, $\chi = 0.5$

(c) Example of conventional One-Class SVM

Figure 3.4: Examples of STOCS in (a),(b) and traditional One-Class SVM in(c). The classification results clearly show the difference in underlying motivation between these two methods. Each support vector in STOCS is accompanied with a circle showing its influence sphere by employing the idea of online learning and adaptive weighting. The collection of all circles jointly forms the separating hyperplane. When an example is in only one of these circles, it belongs to the same class as the corresponding support vector.

formula becomes:

$$f_t(x_t) = \sum_{i=1}^{t-1} \omega_i \delta(x_i \neq x_t) k(x_i, x_t, \sigma_i),$$
$$\omega_t = \text{clamp}\left(\gamma - f_t(x_t), 0, \chi_t\right), \tag{3.11}$$

where $\delta(\cdot)$ is an indicator function with $\delta(true) = 1$ and $\delta(false) = 0$.

The proposed approach is quite different from the traditional One-Class SVM, that it can use both positive and negative examples, which could make it better understanding the data structure. Figure 3.4 shows the difference between STOCS and conventional One-SVM.

3.3.2 Adjustable Kernel Functions

Kernel functions are of great importance in machine learning and have received a lot of attention, particularly due to the increased popularity of Support Vector Machines in recent years. Kernel functions provide a simple bridge from linearity to non-linearity for algorithms which can be expressed in terms of dot products. Kernel functions must be continuous, symmetric, and most preferably should have a positive (semi) definite Gram matrix [50]. In One-Class SVM, kernel reflects how much one example can support another example as a support vector. Choosing the most appropriate kernel highly depends on the problem at hand; for particular datasets, an appropriate kernel will output relatively higher score between two similar examples, and lower scores for those less similar examples.

In order to make the model of STOCS simpler, Qian [40] proposes to eliminate the denominator in the function of ω_t in Equation 3.10. The denominator is a kernel function and it is equal to 1 when the kernel function is normalized. A normalized kernel function satisfies the following properties:

$$\begin{cases} k(x_t, x_t) = 1, & \forall x_t \\ 0 \leq k(x_t, x_s) \leq 1, & \forall x_t, x_s. \end{cases}$$

where x_t, x_s are two examples.

Linear kernel (Eq. 3.12), Gaussian kernel (Eq. 3.13) and Histogram intersection kernel (Eq. 3.14) are three widely applied kernel functions in Support Vector Machines.

$$k(x, y) = x^T y + c \tag{3.12}$$

$$k(x, y) = exp(-\frac{\|x - y\|^2}{2\sigma^2}) \tag{3.13}$$

$$k(x, y) = \sum_{i=1}^{n} \min(x_i, y_i) \tag{3.14}$$

where x, y are two n-dimensional vectors , c is a constant value, σ is an adjustable parameter, and x_i, y_i means the i^{th} dimensional value in x and y.

It is clear that in these definitions that the linear kernel is not a normalized kernel, while the Gaussian kernel is normalized, as well as the histogram kernel when examples are histogram data. When datasets are normalized and non-negative, the linear kernel will also satisfy the normalized kernel properties, then it can be regarded as a normalized kernel.

Employing these generally designed standard kernels may not be enough for a particular problem. Sometimes it is required to control how strong the connections are between two examples or limit how much one support vector supports examples from different class. It is expected that each support vector has higher influence on the examples belonging to the same class, and vice versa. The STOCS employs the adjustable kernel function to meet these requirements. The adjustable kernel function is a slightly modified normalized kernel. The adjustable kernel function is a *normalized* kernel $k(u, v, \sigma)$ with parameter σ *adjustable*, if and only if it possesses the following two properties:

i) $\exists\sigma$, with which we can always satisfy $k(x_t, x_s, \sigma) \leq T, T \in (0, 1)$ for all $x_t, x_s, x_t \neq x_s$;

ii) if $k(x_t, x_s) \geq k(x_t, x_l)$, then $k(x_t, x_s, \sigma) \geq k(x_t, x_l, \sigma)$ holds regardless σ value.

where x_s is the closest negative example to x_t that x_s does not belong to the class of x_t . In this way, it is guaranteed that examples will not get kernel score greater than T from the One-Class SVM model of different classes.

Gaussian kernel satisfies adjustable kernel properties as the first requirement would be satisfied if we set $\sigma = \sqrt{-\|x_t - x_s\|^2/\log T}$, and adjusting σ does not alter the overall tendency of the Gaussian. If any normalized kernel function $k(x, y)$ does not satisfies the two properties, it can be modified to become adjustable by the following formula:

$$k(x, y, \sigma) = \max\left(1 - (1 - T)\frac{1 - k(x, y)}{1 - \sigma}, 0\right) \tag{3.15}$$

where x, y are two examples, σ is the adjustable parameter.

As shown in Figure 3.5, for a given σ, $k(x, y, \sigma)$ is a monotonically-increasing piecewise-linear function with respect to $k(x, y)$. Now we prove the $k(x, y, \sigma)$ is adjustable: for the first property of adjustable kernel, we can set $\sigma = k(x_t, x_s)$, then $k(x_t, x_s, \sigma) = T$, regardless of how the original kernel function $k(x, y)$ is defined; for the second property, since $k(x, y, \sigma)$ is a monotonic function, it does not change the score tendency and monotonicity defined by $k(x, y, \sigma)$.

As mentioned in the last section, STOCS is using self-tuning for its parameters, such as σ, T, etc. Our approach only inherits part of the self-tuning idea since the parameter T is tuned manually because the best choice of T is highly depending on the dataset at hand.

In summary, by employing the idea of online learning, self-tuning and adjustable kernel function, STOCS is simpler and easier than the traditional methods. Different from the classic One-Class SVM training, STOCS is trained using both positive and

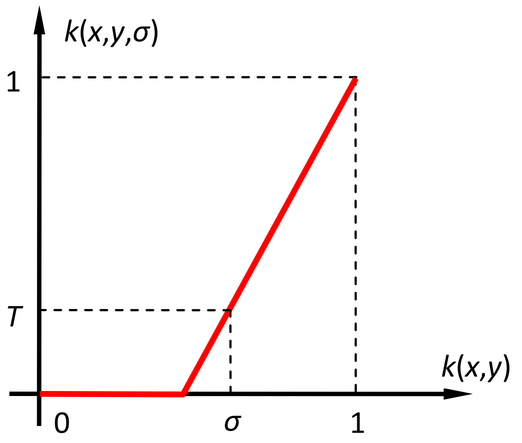


Figure 3.5: Define an adjustable kernel $k(x, y, \sigma)$ based on a normalized kernel $k(x, y)$.

negative examples in parameters tuning and could better learn the data structure. These properties convince us that STOCS seems to be a great fundamental methodology for our goal: to solve the big data classification problems for social media data under limited budget environment. In the next chapter, we present our study on the noise-resilient ability of STOCS, which further convinces us to extend STOCS for social media problem.

Chapter 4

Noise-Resilient Ability Study

In the last chapter, we discussed the details of STOCS: its online learning framework, adjustable kernel function and self-tuning feature. Besides, existing work has shown that One-Class SVM can better handle labeling ambiguities than conventional SVM [24]. Nevertheless, how well STOCS can cope with label noise in social media data is unknown. Hence, in this chapter, we study the noise-resilient ability of STOCS.

We first introduce the heuristics of chosen algorithms. Second, the recently proposed label noise taxonomy is discussed [18]. We then evaluate the capability of STOCS by conducting comparative experiments with selected well-known algorithms. Finally, we present our analysis on the ability of STOCS to handle label noise.

Experiments were conducted with several benchmark datasets, considering both binary problems and multi-class problems. Different types and amounts of label noise were also taken into account in our experiments. Several widely-used methods were included in our comparison: the Naïve Bayes Probabilistic Classifier [15], the C4.5 Decision Tree [42], the Classification and Regression Tree [8], kNN algorithm [52] and

Support Vector Machines (SVMs) [11], which are considered to be five of the top ten algorithms in data mining [59].

4.1 Selected Algorithms

In this section, we briefly describe the selected well-known classification algorithms, then go into a description of noise handling capabilities, if any, are inherent in each technique.

4.1.1 Naïve Bayes

The Naïve Bayes classifier is a simple probabilistic classifier based on Bayes rule with the probabilistic knowledge about the data. In simple terms, a Naïve Bayes classifier assumes that the value of a particular feature is unrelated to the presence or absence of any other features, given the class variable and there are no latent or hidden attributes which have influence on the prediction process. For example, a fruit may be considered to be an apple if it is red, round, and about 3" in diameter. A Naïve Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of the presence or absence of the other features.

The Naïve Bayes estimates the parameters of a simplified Bayes probabilistic model by considering the relative frequencies of each attribute and value per class in the training data set [36]. In many practical situations, the parameters can be estimated by maximum likelihood methods, which is an advantage of the Naïve Bayes method as it only requires few training examples. Although the Naïve Bayes classifier has apparently simple design and assumptions, it can achieve quite promising results in many complex real-world applications. As a result, the Naïve Bayes is a useful and popular method in data mining.

The version of Naïve Bayes we are employing is the "Naïve Bayes" classifier implemented in Weka 3.7 [25] [28]. Weka is a collection of machine learning algorithms for data mining tasks and developed by Machine Learning Group at the University of Waikato [25]. Parameters are tuned manually based on the recommended parameter settings in related work [36]. We applied the default setting for parameters that do not affect prediction results, such as `debug, displayModelInOldFormat`. For other numeric and boolean parameters, we tuned one parameter at a time. For example, we tried both "true" and "false" for parameter `useKernelEstimator` and the setting that gives better performance (false in this case) is used. The Weka parameters are set as: `debug= false; useKernelEstimator= false; displayModelInOldFormat= false; useSupervisedDiscretization= false; class= weka.classifiers.bayes.Naive Bayes.`

4.1.2 Decision Tree

Decision Tree is applied in machine learning as a predictive model which maps a new example to a target value based on several attribute values of this example. An example is shown in Figure 4.1, where each level represents an attribute of the example, and each leaf represents a different target value.

There are many specific versions of decision tree, such as Iterative Dichotomiser 3 (ID3), C4.5, Classification And Regression Tree (CART), Conditional Inference

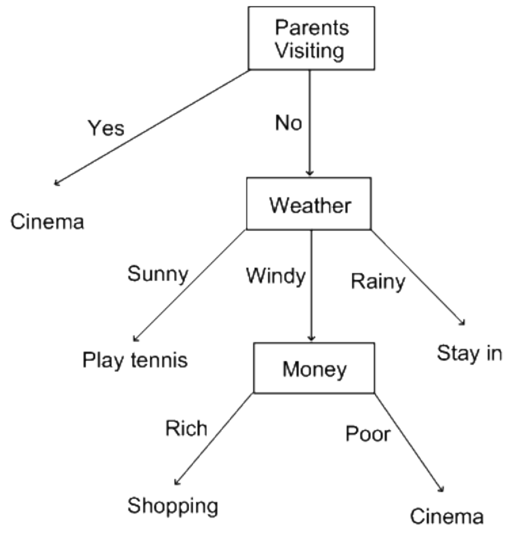


Figure 4.1: An example for decision tree that given three attributes: parents visiting, weather, and money, classify the activity. If an input example has "no parents visiting", "windy", "rich", the decision tree will follow the path "$No \rightarrow Windy \rightarrow Rich$", then classify this example as class shopping.

Trees (CIT). Our experiments employ C4.5 [42] and CART [8] as the decision tree generators. C4.5 is one of the most popular methods in data mining, however, as the version of C4.5 in Weka 3.7 does not support multi-class problem, then we chose another method instead, CART, for the comparisons of multi-class problems.

Two concepts are introduced in C4.5 for test attribute selection: entropy and information gain. The entropy of a random variable X measures the amount of uncertainty of X, and a small entropy of X implies low uncertainty for this random variable.

C4.5 is the successor of ID3, where the model is iteratively built. In each iteration of this method, a sub-model is executed for the remaining examples, and the incorrectly classified cases are included in the next learning window. As indicated in the paper of Fürnkranz [22], the C4.5 process incorporates all noisy examples into the learning window, as all noisy training examples are "misclassified by a good theory" [36]. Also, after a few iterations, there is a situation that in the learning window, the percentage of noisy data will rise to a relatively higher level, which makes learning more difficult.

In general, the C4.5 system tries to decrease the training error by completely fitting all the training examples. Therefore, the over-fitting should also be avoided. One solution for these two problems is tree pruning. There are two cases for tree prune: (1) prune while building the tree: stop growing the tree when the information is less reliable; (2) post-prune: in the growing phase, let the tree grow to its full height, then remove the leaves based on some criteria. C4.5 follows a standard Bernoulli-process-based method for pruning.

The concepts of "reliable" and "unreliable" are decided by the training examples, and the C4.5 algorithm is highly depending on the information gain calculation to choose which attribute will be the next one added to the tree. $Gain_Y(X)$ measurement is applied to calculate the correlation direction between X and Y, which is also called mutual information between X and Y. The formula is shown in Equation 4.1:

$$Gain_Y(X) = I(X;Y) = \sum_x \sum_y P(x,y) log \frac{P(x,y)}{P(x)P(y)} \tag{4.1}$$

where x,y is the attribute values of attributes X and Y, respectively.

The Classification and regression trees model is proposed by Breiman in 1984 [8], which uses a generalization of the binomial variance called the Gini index instead of entropy. CART is different from C4.5 that C4.5 is growing with multiway splitting but CART only makes binary splitting, which means the CART allows to overcome the bias of the splitting measure used and results in high depth decision tree. Both of them are highly dependent on the information gain, but with different criterion. Instead of using entropy, CART employs Gini index and conditional impurity, shown as following respectively:

$$I(Y) = -\sum_{k=1}^{K} \frac{n_k}{n} \times (1 - \frac{n_k}{n}) \tag{4.2}$$

$$I(Y/X) = -\sum_{l=1}^{L} \frac{n_l}{n} \sum_{k=1}^{K} \frac{n_{kl}}{n_l} \times (1 - \frac{n_{kl}}{n_l}) \tag{4.3}$$

where the n_k, n_{kl} represents the corresponding value in contingency table of X and Y, which shows the cross tabulation between Y and X [43].

Then the information gain becomes:

$$Gain_Y(X) = D(Y/X) = I(Y) - I(Y/X) \tag{4.4}$$

The version of C4.5 and CART decision tree generator we are using is the classifier implemented in Weka 3.7. Parameters are tuned manually based on the recommended parameter settings in related work [36]. We applied the default settings for parameters that do not affect prediction accuracy or are not related to current problems, such as `debug, saveInstanceData, Seed.` For other numeric and boolean parameters, we tuned one parameter at a time. Parameter `minNumObj` means the minimum number of instances per leaf and is tested from 1 to 5 with step size 1. Parameter `reducedErrorPruning` is set as "false" since C.4.5 pruning is used. The Weka parameters for C4.5 are set as: `binary splits = false, debug = false, minNumObj = 2, reducedErrorPruning = False, saveInstanceData = False, Seed = 1, sub-`

`treeRaising = True, unpruned = False, useLaplace= False, class is weka. classifiers.trees.J48.`

For the CART classifier, default settings are applied for some parameters since they do not affect prediction performance, such as `debug, doNotCheckCapabilities.` Parameter `maxDepth` is set equal to -1, which means there is no restriction for the maximum tree depth. Parameter `minNum` means the minimum total weight of the instances in a leaf and is tested from 0.5 to 3.0 with step size 0.5. Parameter `numFolds` determines the amount of data used for pruning. One fold is used for pruning, the rest for growing the rules. We test it from 3 to 7 with step size 1. The Weka parameters for CART are set as: `debug = false, doNotCheckCapabilities = false, initialCount=0, Seed = 1, maxDepth=-1, minNum=2.0, noPruning=false, num Folds=3, spreadInitialCount=False, class is weka.classi fiers.trees.REP tree.`

4.1.3 K-Nearest Neighbor Algorithm

One of the well-known and simplest data mining classification methods is the k-Nearest Neighbor Algorithm (kNN). Parameter k is used in the algorithm to classify an unlabeled example by assigning the label which is most frequent among the k nearest training examples. Euclidean distance is a commonly used distance measurement for kNN. However, there is a drawback of the basic voting method where the most frequent class tends to dominate the prediction of new examples, because it has larger population and tend to be more common among the neighbors. A solution for this drawback is that weights are taken into account to measure the contributions of the neighbors. An example of the kNN method is shown in Figures 4.2.

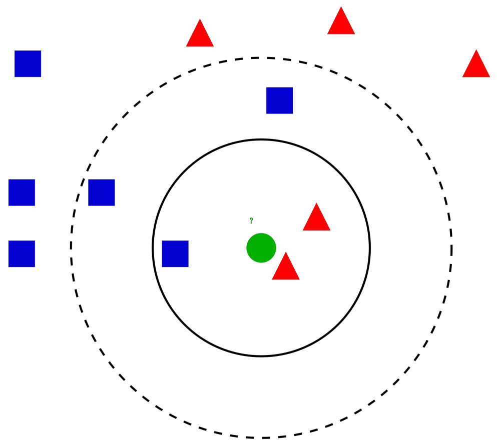


Figure 4.2: Example of k-NN classification, where the red triangles and blue squares represent the training examples. When we try to classify a new example, shown as the green circle in the graph, the prediction result depends on the value of k. If $k = 3$, the situation is shown as the solid line circle, in which there are 2 red triangles and 1 blue square, then the new example is classified to class red. If the k is set to 5 (dashed line circle), the classification result will be class blue. However, when considering weights of neighbors and $k = 5$, the new example may be classified to class red, because the two red triangles training examples are much closer to the green circle, but still depends on the definition of the weights.

It is of great importance to choose a proper value for k, and generally, a larger value of k can make kNN method more robust to noise; however, it can not solve boundary ambiguity well. The best choice of k usually depends on the data, but still, a good k can be selected by some heuristic methods, such as hyperparameter optimization. In our experiment, the k is tuned to avoid the effect of some noise but still keep boundary relatively distinct.

The kNN algorithm has some strong consistency results as well: with the number of training data increasing to infinity, kNN algorithm guarantees to yield the classification error rate better than twice the Bayes error rate, which is a statistical lower bound, on the achievable error rate for a given classification problem [57] [14] .

The version of k-NN algorithm we are using is the one implemented in Weka 3.7, which is also called "IBK" (The Instance Based Learning algorithm) classifier. Parameters are tuned manually based on the recommended parameter settings in related work [36]. We applied the default setting for parameters that do not affect prediction accuracy or are not related to current problems, such as `debug, doNotCheckCapabilities and meanSquared`. A value of 0 for `windowSize` signifies no limit to the number of training instances. For other numeric and boolean parameters, we tuned one parameter at a time. The value of `KNN` depends on the datasets, and we test it from 1 to 10 with step size 1. In the end, the parameters are set as following: `KNN=5; debug=false; distanceWeighting=no distance weighting; meanSquared=false; nearestNeighborSearchAlgorithm = LinearNN-weka.core.EuclideanDistance; windowSize = 0, class is weka.classifiers.lazy.IBk.`

4.1.4 Support Vector Machine

In Chapter 2, support vector machine (SVM) has been introduced. Support Vector Machines (SVMs) represent a group of very popular supervised machine learning algorithms. The motivation of SVMs is to build a hyperplane that separates examples of different classes and maximizes the separating distance with the nearest data points on the margin, which are defined as support vectors. When considering multiclass classification, the dominant approach is to reduce the single multiclass problem into multiple binary classification problems.

As SVMs classify new examples based on the support vectors from the training set, then the hyperplane can be easily changed by the inclusion or exclusion of a single noisy example. However,the presence of noise might disrupt the interrelations and correlations between the training data attributes, and decrease classification performance. Thus, SVM is sensitive to the presence of noisy data.

In our experiments, we are applying Lib-SVM, which is an integrated software for support vector classification and also supports multi-class classification. Lib-SVM is a very popular open source SVMs library, developed at the National Taiwan University by Dr.Chih-Chung Chang and Dr.Chih-Jen Lin [11], and has become a benchmark method in machine learning.Lib-SVM includes different SVM formulations, different kernel functions, cross validation for model selection, and probability estimates. Our experiment is using the default settings, Gaussian kernel with the multi-class classification function. In Lib-SVM software, the parameters are tuned automatically by itself using the provided cross validation approach.

4.2 Label Noise Models

In the survey of classification with label noise [18], a new taxonomy of label noise has been proposed based on the existing noise taxonomy provided by Schafer and Graham [47]. The survey provides three possible models of label noise which are shown in Figure 4.3. In order to model the label noise process, four random variables are depicted in Figure 4.3: X is the vector of features, Y represents the true class for an example, while $\tilde{Y}$ means the observed label of the example, and E is a boolean value that reflects whether the observed label of a example is error ($Y \neq \tilde{Y}$).

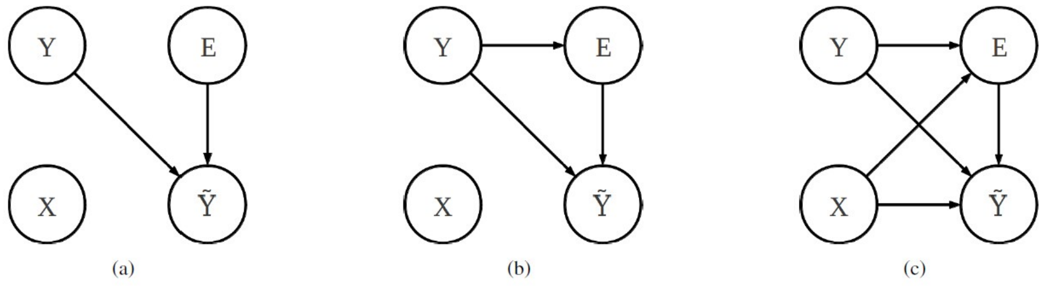


Figure 4.3: Statistical taxonomy of label noise proposed: (a) noise completely at random (NCAR), (b) noisy at random (NAR), (c) noisy not at random (NNAR). X, Y, E and $\tilde{Y}$ are random variables and the arrows indicate statistical dependencies. Note that from left to right, the complexity of statistical dependencies in the label noise generation models increases. The statistical link between X and Y is not shown for clarity.

The Noise Completely at Random Model

The noise completely at random (NCAR) model is the relationship between Y and $\tilde{Y}$ in which the occurrence of noise, or say the occurrence of an error E is independent from any other factors. In the condition of multi-class classification, it is usually assumed that the incorrect label is chosen randomly, regardless of the class distribution. For example, if there are four classes A, B, C, D and class A has 70%

population and B, C and D have 10% population each, then under the NCAR model when there is a noisy example, each of the other three class has the same probability, 1/3, to be chosen as the observed label.

The Noise at Random Model

The noise at random (NAR) model assumes that the probability of error depends on the true class Y. E is still independent of X, but the label noise is not equiprobable any more, which means for certain class, the instances have higher chance to be mislabeled or when mislabeling an example, certain class has higher chance to be chosen. For example, assuming that the error rate is linearly related to the population of class, if there are four classes A, B, C, D and class A has 70% population, and B, C and D has 10% population each, then when there is one noisy example with true label B, the probability ratio of the observed label is $A : C : D = 7 : 1 : 1$.

The Noise not at Random Model

The noise not at random (NNAR) model is more commonly existing in the real-world datasets, where the error E depends on both X and Y; for example, the examples may be more likely mislabeled when they are similar to instances of another class [31]. Although the NNAR model is the most general case of label noise, it is difficult to simulate this type of label noise because the inner relationship is highly dependent on the actual datasets. For this reason, we will only simulate the first two types of label noise in our experiments and used them to evaluate different methods.

4.3 Experiments and Results

In this section, we describe the experiments performed to reveal the capability of STOCS to handle label noise. Table 4.1 shows the datasets used in our experiments. These data sets are provided by the LIBSVM data sets [10] and UCI Machine Learning Repository [33], which are famous benchmark data sets for machine learning. We are using these data sets to evaluate STOCS and the five popular methods mentioned above. We first conducted experiment using the original datasets and then added 5%, 10%, 15% of label noise into the training set.

Both NCAR and NAR label noise models are considered in our experiment. For the NCAR noise model, we randomly selected an example to become a mislabeled example and then randomly assigned the observed label; while for the NAR noise model, we assumed that the class with higher population has greater chance to have mislabeled examples and to be selected as the observed label, which is linearly related.

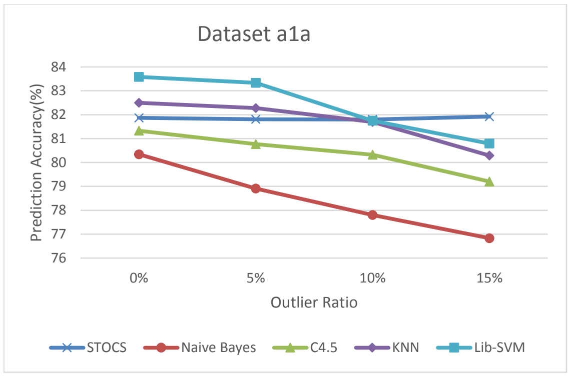
We are using the implementation available in Weka 3.7 for Naïve Bayes, C4.5 decision tree, CART, and kNN algorithm and using Lib-SVM for the implementation of SVMs. Note that the Naïve Bayes classifier and the C4.5 Tree generator in Weka 3.7 do not support multi-class problems. Hence in our experiments, we use Naïve Bayes and C4.5 for binary datasets only.

4.3.1 Binary Classification

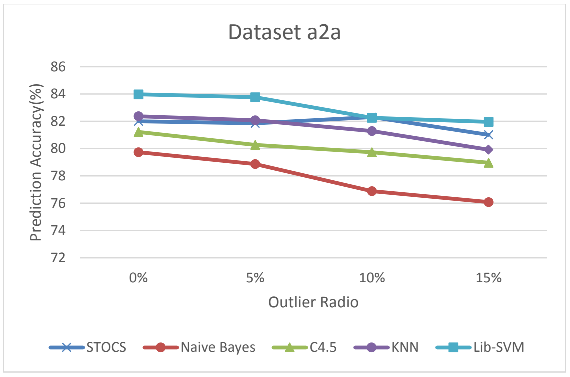
We first evaluated STOCS under binary classification situations. Note that the prediction accuracy we present in the thesis is computed using: Number of correctly classified examples divided by Number of testing examples. Figures 4.4 and 4.5 show

Dataset	**Classes**	**Training Examples**	**Testing Examples**	**Features**
optdigits	10	3823	1797	64
dna	3	2,000	1,186	180
monks-1	2	125	433	6
monks-2	2	170	433	6
monks-3	2	123	433	6
a1a	2	1,605	30,956	123
a2a	2	2,265	30,296	123
usps	10	7,291	2,007	256
vowel	11	528	462	10
letter	26	15,000	5000	16

Table 4.1: Datasets used for comparison



(a) Dataset a1a



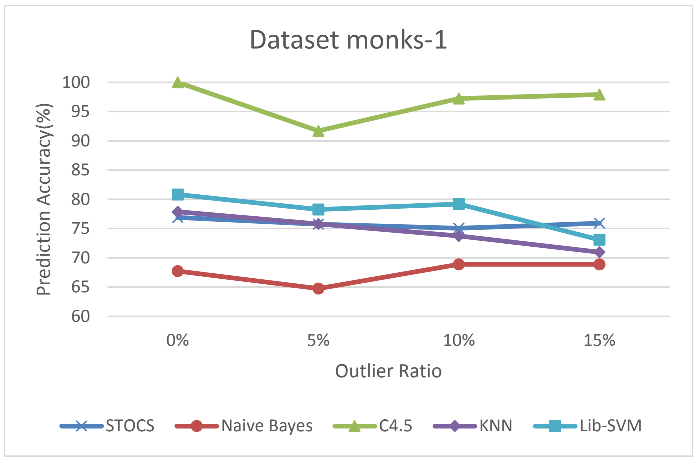
(b) Dataset a2a

Figure 4.4: Evaluation of noise handling ability of STOCS under binary datasets(1). Note that different axis ranges are used for different datasets, so that the performance differences can be better illustrated.

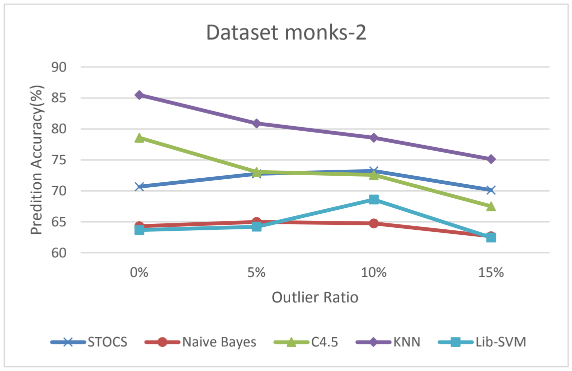
the results when conducting experiments using the binary datasets. For each dataset, different amount of label noise is added, and we only consider the NCAR noise model since there are only two classes in binary datasets. Note that to remove the impact of noise generating on the evaluation, results in this chapter are averaged on 3 runs. In each run, the noise is randomly generated and added into training sets.

We first analyzed the performance when label noise is not introduced, i.e., label noise ratio is equal to 0%. It reflects the prediction accuracy under the original datasets. It is easily concluded that the Lib-SVM performs best and wins 3 out of 5 predictions. Even though STOCS could not achieve the best accuracy in any of these datasets and is shown to be slightly inferior to the Lib-SVM, it does perform better than other traditional methods.

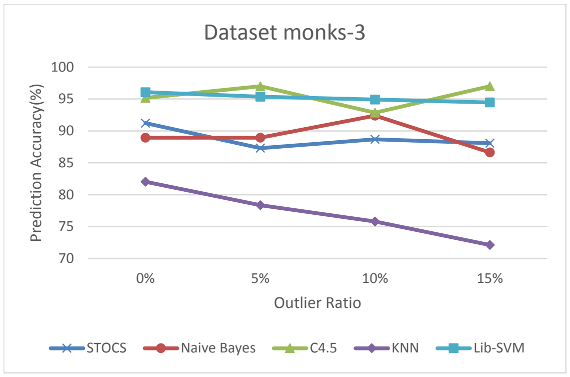
Generally speaking, as the amount of label noise increases, the prediction accuracy tends to decrease. When using the original datasets, STOCS can not achieve the



(a) Dataset monks-1



(b) Dataset monks-2



(c) Dataset monks-3

Figure 4.5: Evaluation of noise handling ability of STOCS under binary datasets(2).

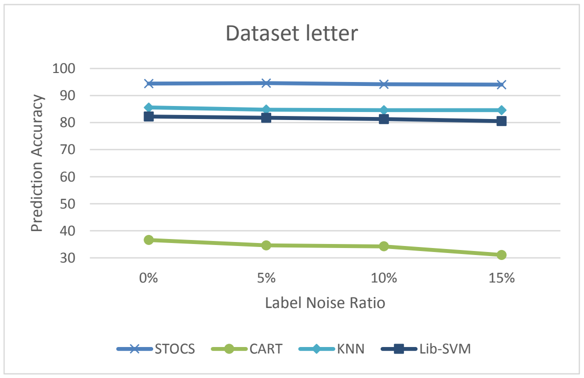
best accuracy in any prediction, whereas when 10% or 15% label noise is added, STOCS performs the best for some datasets. Even though there is a tendency that the prediction accuracy becomes worse with the amount of label noise increasing, STOCS turns out to be less sensitive to label noise as the performance of STOCS drops less than that of all other methods.

4.3.2 Multi-Class Classification

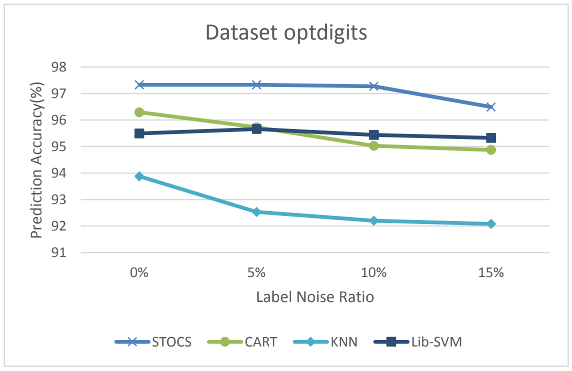
In this sub-section, we present the results achieved when considering multi-class problems. The datasets we are using are shown in Table 4.1 and all these benchmark datasets are obtained from UCI Repository. As we mentioned earlier, for the multi-class problems, we compare the CART, K-nearest Neighbor algorithm and SVMs with STOCS, using the Weka data mining software for the classifiers.

Figures 4.6 and 4.7 demonstrate that STOCS performs better in multi-class classification than binary cases since when using the original datasets, three out of five best classification results are achieved by STOCS, while the other two by the Lib-SVM. The figures also show that the CART obtains highly undesirable results with "usps" and "letter" datasets because the binary tree grown by CART is not always suitable and decision tree based methods can not handle large datasets very well [42] [8] [43].

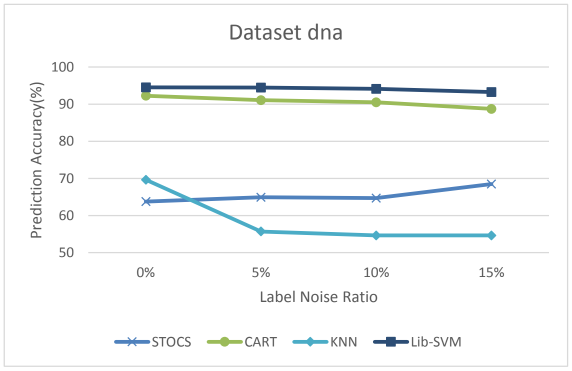
When different amount of label noise is added into datasets, there is a general tendency that prediction accuracy decreases with the increase of label noise. However, we conclude from the results that STOCS is the most robust one against label noise among these four algorithms. For example, when 15% label noise is added, the average accuracy loss for STOCS is 0.6045%, while the loss is 2.6931% for CART, 3.907% for KNN, and 1.1532% for LibSVM. When considering the impact of different type of label noise, it is shown that the NAR label noise is more harmful to the prediction accuracy than the NCAR label noise: with the same amount of noise generated, NAR label noise always leads to less accurate results. Besides, for the "letter" dataset, which is the largest dataset among these ten datasets, STOCS outperforms all algorithms in all cases no matter whether or how much label noise is



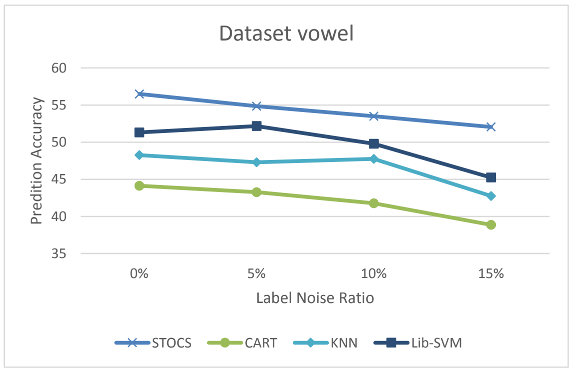
(a) Dataset letter



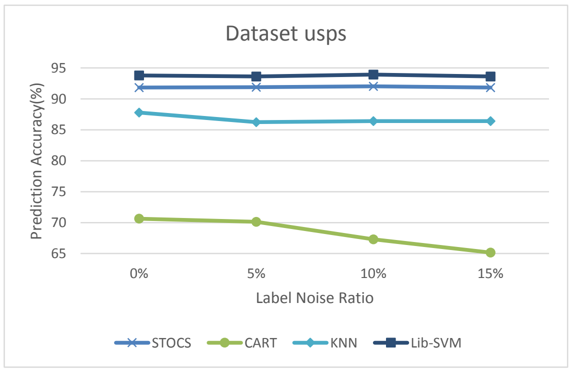
(b) Dataset optdigits



(c) Dataset dna

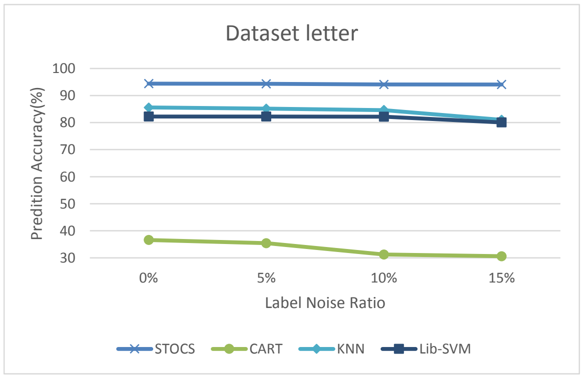


(d) Dataset vowel

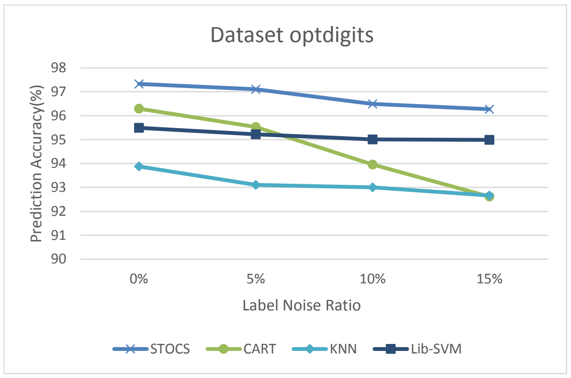


(e) Dataset usps

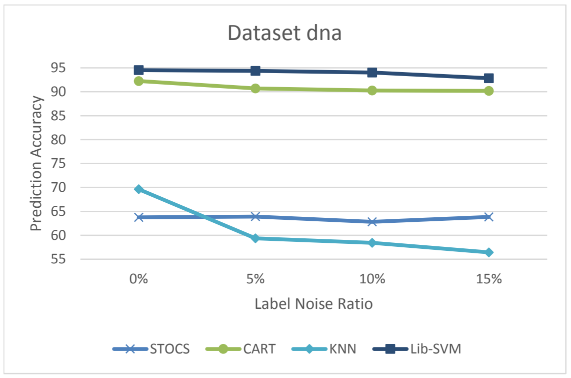
Figure 4.6: Evaluation of noise handling ability of STOCS under multi-class datasets with NCAR type of label noise.



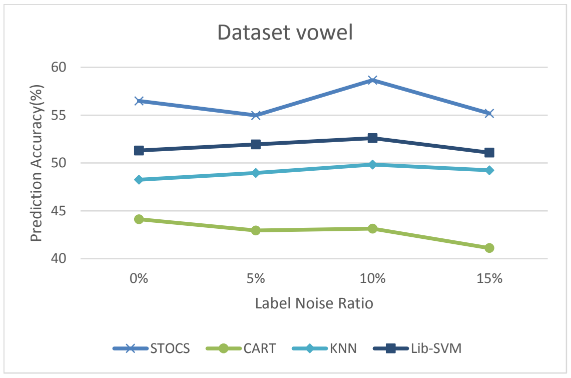
(a) Dataset letter



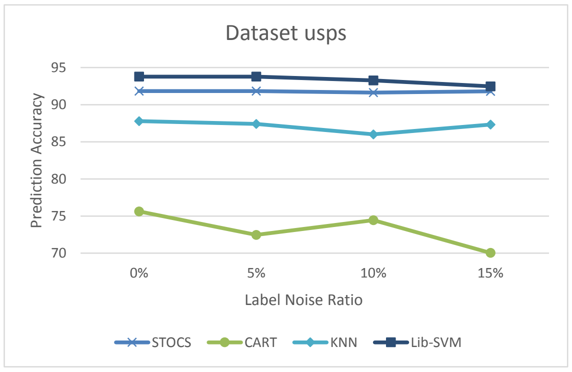
(b) Dataset optdigits



(c) Dataset dna

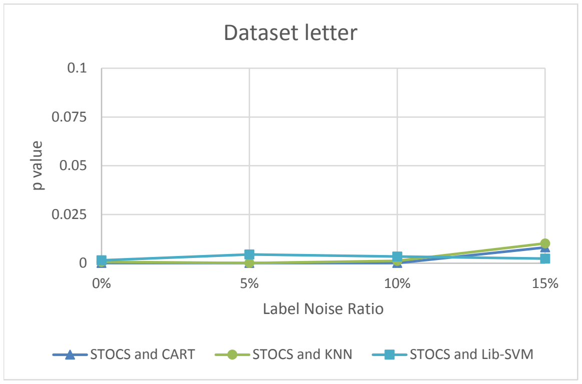


(d) Dataset vowel

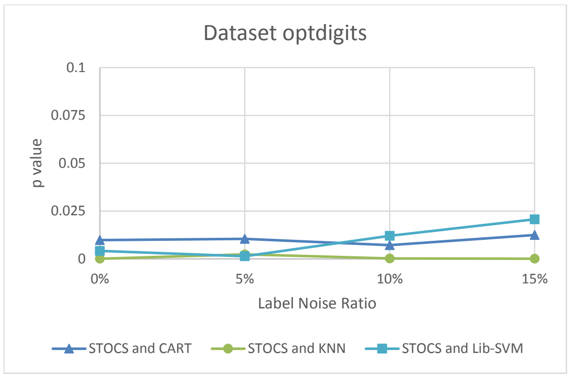


(e) Dataset usps

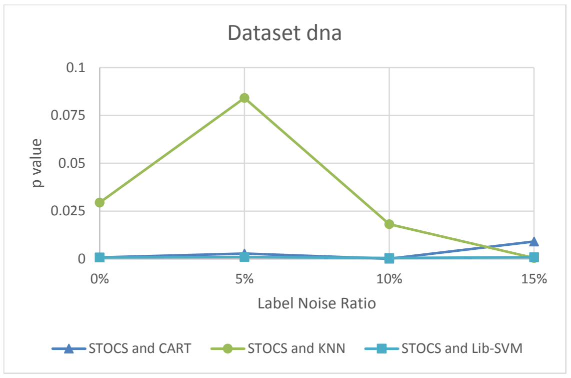
Figure 4.7: Evaluation of noise handling ability of STOCS under multi-class datasets with NAR type of label noise.



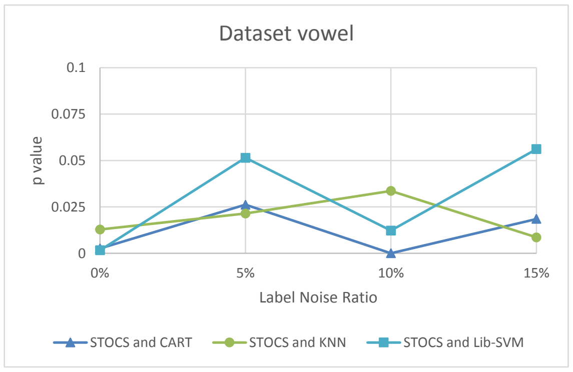
(a) Dataset letter



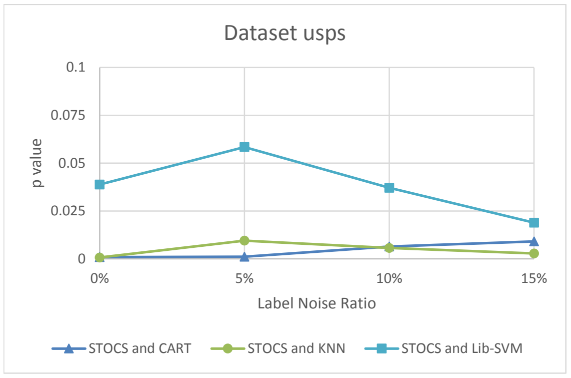
(b) Dataset optdigits



(c) Dataset dna



(d) Dataset vowel



(e) Dataset usps

Figure 4.8: Statistical evaluation on the results obtained under multi-class datasets with NCAR type of label noise. Only 4 out of the total 60 cases have $p > 0.05$.

added, showing the ability of STOCS to handle large scale datasets with label noise.

In order to indicate whether the difference in accuracy is statistically significant, we conducted statistical tests on the obtained prediction results by using the t-test approach. We chose 5% as the significance level of the test and applied the paired t-test model. Figure 4.8 shows the statistical evaluation for the results when considering NCAR label noise. It reflects that most of the differences are statistically significant.

The above findings make us believe that STOCS could better handle classification problems with the presence of label noise. As our goal is to solve the big data classification problems for social media data which contain high amount of noise, extending STOCS algorithm is a promising direction.

4.4 Robustness Analysis

As we are trying to solve the big data classification problems for social media data with the existence of label noise by extending and improving STOCS algorithm, we needed to analyze the robustness of STOCS against label noise. Experimental results in Section 4.3 show that the STOCS could handle label noise better in comparison with other well-known algorithms. We believe there are two main features contributing to this promising performance.

First, the adaptive parameters help to reduce the impact of label noise. STOCS is different from conventional One-Class SVM in that each support vector only affects other examples inside its hypersphere by incorporating the online learning frame and adaptive parameters, which is shown in Figure 3.4. The adaptive parameters will not only limit the chance of a noisy example to be selected as support vectors, but also reduce the harmful impact of a noisy example when it is in support vector sets.

Training examples can be corrupted by different label noises. Assigning a large weight on the corrupted examples would increase the chance of adding them into support vector sets, thus distorting the decision boundary. To address this, a cut-off value χ is used to bound w_t, thus limiting the effects of label noise. The adaptive cutoff value in STOCS further helps to reduce the chance of a mislabeled example to be added into support vector sets by guaranteeing that a support vector should have higher χ if there are many positive examples within its support region and a smaller χ if there are many negative examples surrounding it. Hence, STOCS tunes the cutoff value χ for each individual support vector to better limit the impacts of mislabeled examples.

In addition, by automatically selecting different Gaussian support (parameter σ) for different support vectors, STOCS allows that 1) support vectors that are far away from the decision boundaries having larger influence area; and 2) assigning small influence areas to support vectors that are close to the decision boundaries help to reduce the level of misclassification. In this way, even if a mislabeled example is selected as a support vector, the adaptive σ will reduce the impact of the mislabeled example in decision boundaries.

Secondly, STOCS employs the idea of adjustable kernel function, where the parameter T limits the kernel scores of examples from different classes. Even if a mislabeled example is selected as a support vector, it is guaranteed that examples of other classes in its neighborhood will not get kernel score greater than T from this mislabeled example support vector. For example, assuming that there are two very similar examples, A and B, where the example A is a mislabeled example and is selected as a support vector, then when STOCS tries to classify example B, the kernel score calculated with A is less than T. In this way, example B has relatively lower chance to be mislabeled to the class of example A, compared with conventional methods.

4.5 Conclusions

In this chapter, we first presented the heuristics of the chosen algorithms and talked about the details of how the Weka software was used to employ these methods, including all parameters settings.

Second, we described a newly proposed taxonomy of label noise [18], where label noise is divided into three types: Noise Completely at Random (NCAR), Noise at Random (NAR), Noise not at Random (NNAR).

Then, we conducted comparison experiments for both binary and multi-class cases under the benchmark datasets provided by Lib-SVM datasets and UCI Repository. Several conclusions can be drawn from the results: (1) STOCS performs better in multi-class problems than binary problems; (2) with the increasing of label noise, prediction performance tends to become less accurate; (3) STOCS is the most robust approach among all selected algorithms against label noise; (4) in most cases, the results of STOCS are competitive with the best performance; (5) NAR label noise is shown to be more harmful than NCAR label noise.

Finally, we analyzed the robustness of STOCS against label noise. There are two main reasons contributing to the strong performance of STOCS: the adaptive parameters and the adjustable kernel parameter. They can not only reduce the chance of a mislabeled example to become a support vector, but also limit the harmful influence of a mislabeled example support vector.

Our goal is to design an algorithm for solving large scale social media classification problems under limited budget environment. The above-mentioned observations make us believe that extending STOCS has great potential to achieve our goal. Besides, we find out that Lib-SVM, and kNNs algorithm are the other two best performing methods considering both binary and multi-class cases, while decision tree based methods can not handle large datasets well and the Naïve Bayes classifier is outperformed by all other methods in binary cases. Based on these facts, we choose the Lib-SVM and kNN algorithms as the baseline approaches and compare our approach with them in our chosen social media big data challenge.

Chapter 5

Budget-Driven Big Data Classification

In the previous chapter, the noise-resilient ability of STOCS is studied by conducting several experiments between well-known algorithms. Experimental results show that STOCS is the most robust method against label noise among all chosen methods for multi-class problems, which convinces us to extend STOCS for our goal: the big data classification problem for social media data under limited budget environment. In this chapter, we first extend STOCS and discuss our specific design for social media classification. Then we introduce the training algorithm in our approach and present related experimental results. Finally, the confidence-driven classification is discussed, which can be applied in real-world "low-budget" business promotion problem.

5.1 Budget-Driven Online Learning Model

Our approach is designed based on STOCS, inheriting the online learning strategy and the advantages in dealing with label noise. In order to solve classification problem under low budget environment, our approach is designed considering two important aspects: the redundancy and duplication in big data and the impact of support vectors.

5.1.1 Online Learning Framework

As an extension of STOCS, our approach inherits the following distinct advantages of online learning for budget-driven problems. Examples are observed by the learner one by one in a time sequence. The online learner is gradually refined based on its current partial model and the newly observed example. The training process terminates when the user stops presenting examples, and the partially trained model can be used to perform classification. As a result, training time is controlled by the user, which is promising when trying to classify big data under limited training time while still aiming for good performance. Furthermore, our approach can easily handle dynamic data such as video stream and online user generated webdata, etc. When a new training example (e.g. a new image uploaded to social media) is observed, the minimization function of the batch One-Class SVM is changed, and thus it should be solved again to obtain the new solution. Hence, batch learning needs to start over, which is time-consuming. In comparison, by extending from STOCS, our approach refines the training model only using the newly observed example. The refinement process only involves a simple computation of score function, as shown in Algorithm

1.

5.1.2 Redundancy in big data

Compared with our fundamental methodology, STOCS, our approach differs in several ways to realize our specific goal: budget-driven big data classification. First, in social media classification problem, the training sets are large scale and provided by different users; as a result, we believe large amount of redundancy is existing in the data set. Different users may upload the same or very similar images to social media. However, in STOCS, all examples are trained repetitively by the online learner, which requires a large number of iterations to converge. When it is a big data problem, the converging time of STOCS is expensive. In contrast, our approach takes the redundancy and duplication into account, and trains on only a portion of training examples. We believe that the classification accuracy is still competitive. In each iteration, a randomly selected example is read and then the Competing One-Class SVM model is updated. Reading data and training classifier happen at the same time in our approach, allowing to only store the training example and support vectors in the memory during training. Notice that even though STOCS incorporates the online learning framework, it requires to read all data into memory since all examples are trained multiple times. Hence, our approach is particularly useful on big data classification when data cannot fit in memory. Our approach not only reduces the memory requirement for big data problem but also relaxes the convergence condition, making large scale training more efficient. Convergence condition is defined as when the preset condition is satisfied, the alogorithm is considered as converged. In our approach, the convergence condition is defined as support vectors lists do not change for at least $\beta * n$ iterations, where n is the number of examples in the training set. It is shown that 0.001 will be a good choice for β, achieving promising accuracy and fast convergence. Experimental results in Figure 5.2 also show that our approach can converge after training a small portion of the training set when the problem size is large, which further confirms our assumption of redundancies in big data.

5.1.3 Impact of Support Vector Number

As the training set becomes larger, the number of support vectors increase significantly, especially for a big data problem. As mentioned in last chapter, STOCS approach results in more support vectors than classic One-Class SVM algorithm, so storing all these support vectors in STOCS would require vast memory. To further reduce memory requirement, our approach assumes that the classification decision of STOCS is primarily determined by the dominant support vectors, i.e. those with high supporting weights. In practice, we set the number of support vectors of each class as a fixed value and store only the most dominant support vectors in memory, which is also controlled by the user. Compared with the conventional One-Class SVM algorithm that keeps all support vectors, our approach achieves competitive performance as reported in Figure 6.3. Moreover, controlling the number of support vectors helps reduce the converging time since the computational cost for evaluating the score function is linearly dependent on the number of support vectors; see Equation 3.3. By only keeping dominant support vectors, the model refinement in each iteration would be more efficient. Besides, in most cases, algorithms can not effectively handle both the noise and the overfitting problems at the same time, whereas our approach avoids this issue since only dominant support vectors are stored.

We extend STOCS with these two main modifications. As a result, our approach has lower computational requirements ; as demonstrated later in Chapter 6. Experimental result shows that our approach could achieve promising classification accuracy with limited computational budget. Moreover, our approach avoids the overfitting problem by only keeping dominant support vectors, while being robust against label noise.

5.2 The Training Algorithm

As an extension of STOCS, our approach inherits most formulae and functions from it. When a new example and its label $\{x_t, y_t\}$ is randomly observed/read, our approach first evaluates the score of x_t based on the existing dominant support vectors from the same class and computes the weight of x_t. Following the decision function of SVM, the score function is defined as:

$$f_t(x_t) = \sum_{j=1}^{n} w_j \chi\left(S_{y_t}(j) \neq x_t\right) K(S_{y_t}(j), x_t), \tag{5.1}$$

and its supporting weight:

$$w_t = \max\left(0, \min\left(\frac{\gamma - f_t(x_t)}{K(x_t, x_t)}, C\right)\right), \tag{5.2}$$

where S_{y_t} is the support vector set of the class y_t, w_j is the weight of the j_{th} support vector in S_{y_t}, $\gamma := 1$ is the margin, C is the cut-off value, and $\chi(\cdot)$ is an indicator function with $\chi(true) = 1$ and $\chi(false) = 0$. $K(\cdot, \cdot)$ is the kernel function, and different kernels can be used for different applications.

Algorithm 1 Budget-driven Competing One-Class SVM

```
Input: training examples with corresponding labels $\{x_t, y_t\}$, kernel function $K(\cdot,\cdot)$,
    cut-off value $C$, support vector size $n$, convergence condition
Output: support vector set $S$
 1: Initialize each $S_i$ as an empty set for each $i_{th}$ class
 2: repeat
 3:     randomly read an example $(x_t, y_t)$
 4:     compute score $f_t(x_t) \leftarrow \sum_{j=1}^{n} w_j \chi\left(S_{y_t}(j) \neq x_t\right) K(S_{y_t}(j), x_t)$
 5:     $w_t \leftarrow \max\left(0, \min\left(\frac{\gamma - f_t(x_t)}{K(x_t,x_t)}, C\right)\right)$
 6:     if $x_t$ already exists in $S_{y_t}$ then
 7:         update the weight of $x_t$ with $w_t$
 8:     else if $w_t >$ the minimal weight in $S_{y_t}$ then
 9:         replace the SV with minimal weight in $S_{y_t}$ with $\{x_t, w_t\}$
10:     end if
11: until User Termination or $S$ convergence condition is satisfied
```

The training algorithm of our approach is presented in Algorithm 1, which is very easy to implement. Note that our approach employs the idea of reweighting, which is shown in the algorithm from line 6 to line 9. Reweighting is a popular approach to deal with duplication, which is also applied in STOCS since STOCS may train the examples over several rounds. However, our approach can converge after only training a portion of examples and we are employing the idea of reweighting in a different way from STOCS. We believe duplication is common in big data problem as people may upload same or very similar pictures or videos. In the conventional online learning, all duplicates are added into support vector sets as long as their supporting weights are large enough. These duplicate support vectors come from the same example but have different weights. In contrast, we define two examples are duplications if the difference between the features of the examples is smaller than a threshold. Then when observing a duplicate example x_t that is already in the corresponding support vector set, our approach computes the score with the duplicate x_t excluded and the original weight of x_t is substituted by w_t. By summing the supports of the remaining support vectors, $f_t(x_t)$ can better shows how well the current model can predict x_t. The supporting weight w_t is computed by comparing the margin and the score. When our approach tries to classify a newly encountered example, it just simply calculates the scores for each class model and then selects the class whose model obtains the highest score.

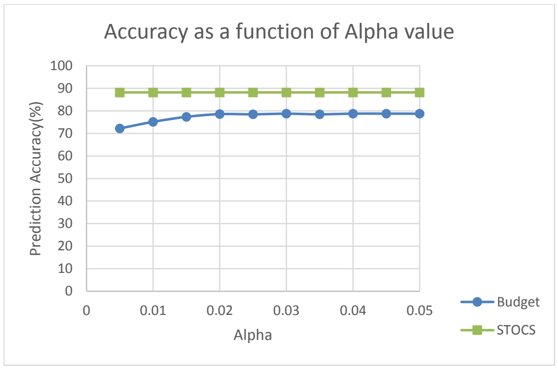
It is worth noting that in our approach, examples with high scores tend to be assigned with low weights and it always replaces the minimum-weighted support vector. There are two main reasons: 1) if existing models return high scores for an example, it means the current models/ support vectors are good enough for representing this example. Then if this example is selected as a support vector, it does not need to be assigned high weight; 2) if a support vector has the minimum weight, it means other support vectors are already good enough to represent it and this support vector is the least useful one among all support vectors. If an example is trained but the returned scores are low, reflecting that current models are not good, then it is of great importance for it to be added into the corresponding support vectors list with high weight. If current support vector set is full, our approach will replace the one with minimum weight.

With the benefits of online learning, the training process can be terminated by stopping feeding training examples. Users can also selectively evaluate the partially trained model using a validation dataset to check its effectiveness. Besides user controllable termination, our algorithm itself can converge fast and achieve competitive accuracy at the same time. The training procedure may terminate after a fix number of iterations, after changes in the support vectors lists fall below a threshold, or after the support vector lists stay constant for some number of iterations.

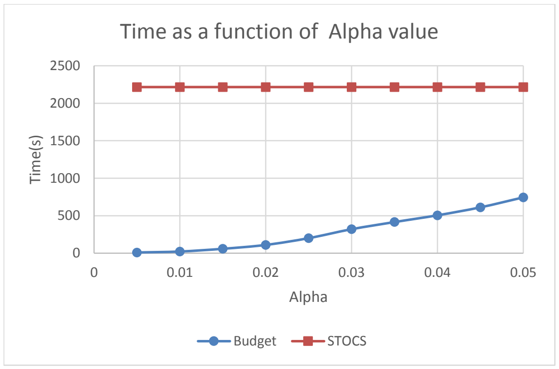
5.3 Evaluation of Proposed Budget-Driven Features

In this section, we present the experimental results of our two specific modifications to STOCS. We are using the dataset “MNIST” from the Lib-SVM dataset, in which there are 60K training examples, 10K testing examples, 10 classes and 780 features. We compare our approach with the original STOCS approach that we got inspired from, which is robust against label noise but has a high computational cost.

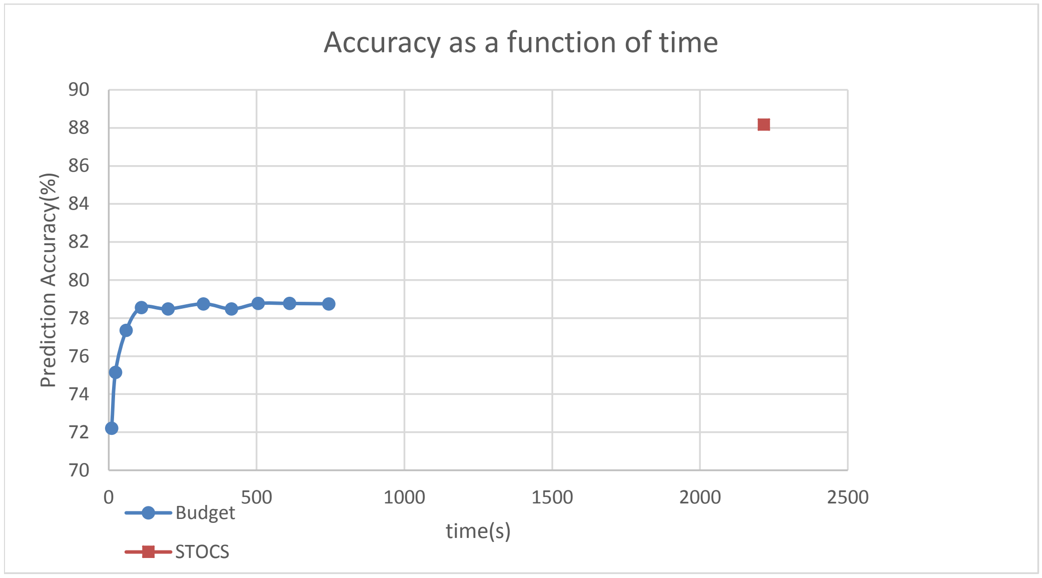
For the support vector buffer size, we observe that the optimal buffer size is related to the number of examples in each class. Denote the ratio of examples selected as support vectors as α. For the i_{th} class, we set the support vector buffer $n_i = \alpha N_i$ in our experiments, where N_i is the number of the observed examples belonging to the i_{th} class. Figure 5.1 plots the classification accuracy of our approach under different α values on the dataset “MNIST”. We can conclude from the results that 1) support vectors help to capture complex data structure, hence, keeping only a small number of support vector in the buffer produces lower accuracy; 2) as the α value increases, the classification accuracy improves rapidly at first and then remains



(a) Classification Accuracy



(b) Training Time

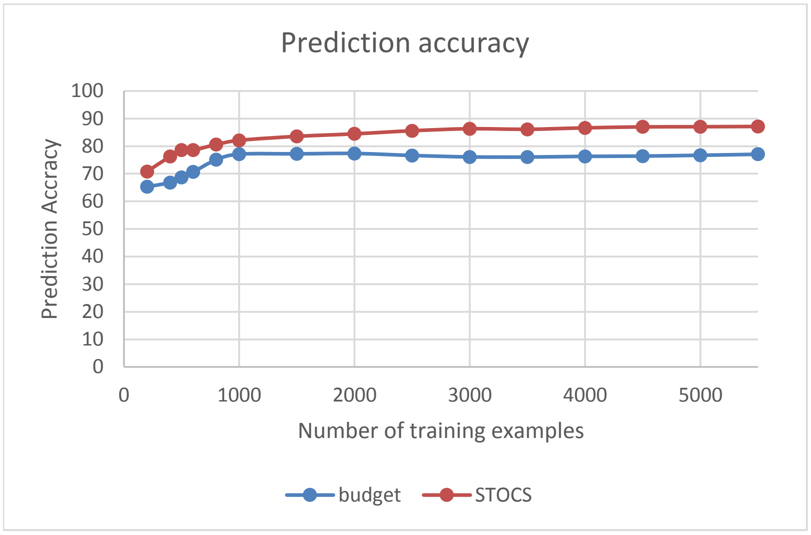


(c) Accuracy as a Function of Training Time

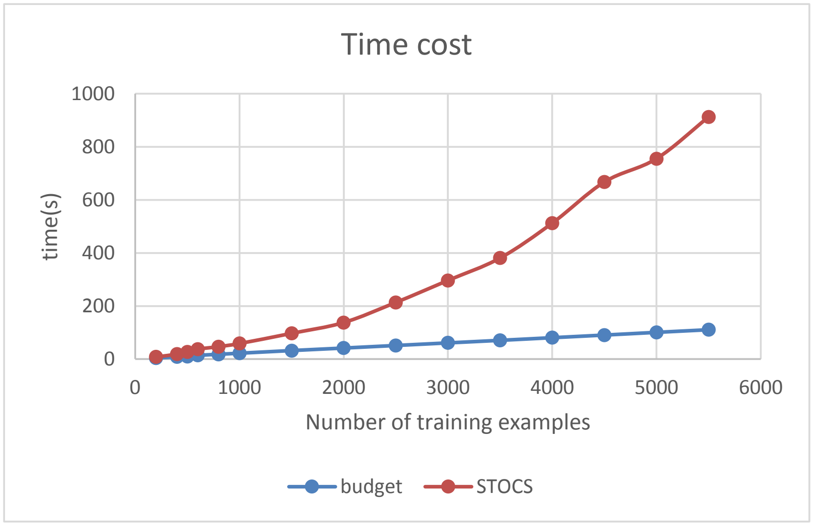
Figure 5.1: Evaluation of support vector number on data set of $60K$ training examples from the "MNIST" data. Figure (a)-(b) show the classification accuracy, training time used under different α values, respectively. Figure (c) shows the relation between accuracy and time cost.

stable. It is shown that our approach with a relatively smaller α value can achieve competitive performance compared with the result obtained by larger α value, but has lower memory requirement and faster converging time. The comparison between our approach and STOCS reflects that STOCS could obtain higher prediction accuracy but with much higher computational cost. Besides, the observations confirm our assumption that the decision boundaries are mainly influenced by dominant support vectors, while competitive performance can still be achieved no matter how large the training size is. We observe that setting $\alpha = 0.02$ provides a good trade-off for the dataset "MNIST".

We present the performance of our approach and STOCS when training on a portion of examples. In each iteration, our approach trains one example. For our approach, we trained on the original "MNIST" dataset and then evaluated our model every 200 iterations at first and then every 500 iterations. However, for STOCS, we generated different sizes of training set, and then STOCS trains those datasets multi-rounds, until fully converged. As shown in Figure 5.2, as the number of examples in training portion increases, the classification accuracy improves rapidly at first, then stays relatively stable. STOCS could achieve better prediction accuracy but has higher time cost since STOCS trains on all examples over rounds until fully converged whereas our approach only trains on each example once. The result demonstrates that for 60K size dataset "MNIST", our approach could achieve a good trade-off when training on 1K examples, approximate 1/60 of the training set. It is shown that our approach can converge after training on only a portion data when the problem size is large, which further confirms our assumption of redundancies in big data.



(a) Classification Accuracy



(b) Training time

Figure 5.2: The performance of partial training. We compare our approach with STOCS where we evaluate the training models every 200 iterations and then every 500 iterations.

5.4 Budget-based Selective Labeling

In many real-world applications, classification is used to identify the group of entities that fit particular features. For example, social media data classification can be used to identify the type of sport that people are most interested in, such as soccer, basketball, etc. In many of these cases, the goal is not to classify each individual, but rather to select the ones that fit the selection criteria the best.

The experimental result above shows that the best classification performance can only achieve approximately 78% prediction accuracy. Hence, using the classification results to guide tasks such as advertisement and marketing promotion may not be very effective. To address this problem, we here explore the possibility of selective labeling the examples for which the classifier is more confident about their class. We argue that, in real-world cases, it is common that promotion activities are restricted by 1 budget, i.e., the available funding. It is helpful to find out the group of potential customers who are more likely to be persuaded by the promotion. We call it "budget-driven" selective labeling problem and by incorporating the confidence, our approach can be applied.

Employing the concept of confidence, our approach classifies one example if and only if has high confidence in the classification. When classifying an example, conventional Competing One-Class calculates the scores from different classes models, and chooses the class returning the largest score as the prediction label. In contrast, our approach will compute both the largest score and the second largest score, and classify the example only if the difference between these two scores is greater than the confidence value, i.e., the ambiguity of labeling the example is low. The suitable confidence value is highly depending on the problem and the kernel selection.

After employing the confidence, we generated the accuracy and recall curve, which is shown in Figure 5.3. We can conclude that, generally, the lower percentage of examples we classify, the higher accuracy we achieve, where the highest the accuracy

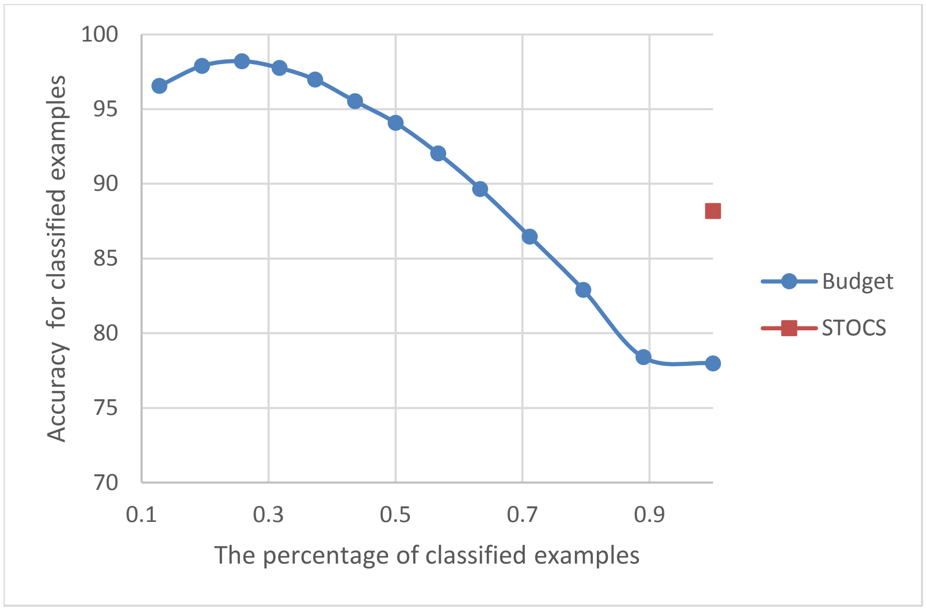


Figure 5.3: The accuracy as a function of the number of classified examples.

we can reach is 99%, higher than the best performance of STOCS. At the highest the accuracy, 27% of the examples are classified.

This experimental result shows that our approach could help the "budget-driven" selective labeling problem by only classifying a part of the examples, and achieving higher accuracy. We believe this property is useful in real-world problems.

5.5 Conclusion

In this Chapter, we introduce our approaches for our budget-driven classification and the training algorithm. It is shown that by only keeping dominant support vectors and training on a portion of data, our approach could handle the big data classification problem with lower memory requirements and higher efficiency, achieving reasonable classification accuracy. These facts confirm our assumption of the redundancies in big data and that the decision boundaries are mainly influenced by dominant support vectors. Finally, we propose to employ the idea of confidence to help solve real world “budget-driven” selective labeling problems.

Chapter 6

Application to Real Social Media Data

This chapter presents experiments performed to evaluate our approach when applied to the social media problem Yahoo! Large-scale Flickr-tag Image Classification Grand Challenge. We introduce the dataset first and then show the parameter tuning and kernel function chosen. Finally, comparisons with existing approaches are presented. We implemented the proposed approach in C++ and all experiments were run on a Intel Core i5 (3.20GHz) machine with 4GB RAM.

6.1 Flickr-tag Image Datasets

The Flickr-tag image dataset used in our experiment is shown in Table 6.1, which is provided by Yahoo! Multimedia Grand Challenge [2]. Most of the effort in current image classification work has been dedicated to building systems that can scale up when the number of classes is large. In this dataset, there are 2 million images, 200,000

Dataset	**Classes**	**Training Examples**	**Testing Examples**	**Features**
Fickr	10	1500000	500000	400

Table 6.1: The Fickr-tag Image Datasets

images per class. The ten images classes include nature, food, people, wedding, music, sky, london, beach, 2012, and travel. These are amongst the most commonly used tags by the Flickr users to annotate images. However, as the annotations are provided by the users, the tags may not always be accurate, which will result in the existence of label noise.

What makes the challenge more difficult is that each class is composed of visually diverse sub-classes, for example, the class "nature" contains images from sub-classes such as "Beach", "Mountains", and "Sky". Moreover, some of these sub-classes could be "visually" unrelated to the root class; for example, the class "Nature" may contain some images of nature journals.

In summary, the Fickr-tag image datasets are collected from social media, with large scale training and testing sets, containing relatively higher percentage of label noise. This is what our approach tries to solve, the big data classification problem for social media data. Besides, based on the contribution of Yu-Chuan Su *et al* [51], instances tagged with “2012” are removed from the dataset before conducting experiments since the tag “2012” is assigned based on the image uploading time and is unrelated to visual information. Unlike previous works on the Flickr-tag dataset which concentrate on extracting and exploring multiple image features, we are interested in the classification technique itself. Hence, the provided bag-of-words features are used in our implementation so that we can compare different classification techniques.

6.2 Parameter Tuning

As we mentioned in the previous chapter, our approach extends STOCS by reducing memory and time requirements. Our approach limits the support vectors buffer size to reduce computational cost and avoid overfitting. Besides, due to the redundancy in big data, we believe competitive classification accuracy can be obtained by training only on a portion of training examples. In this section, we describe how parameters, the adjustable kernel function parameter T and cut-off value, were tuned for our approach.

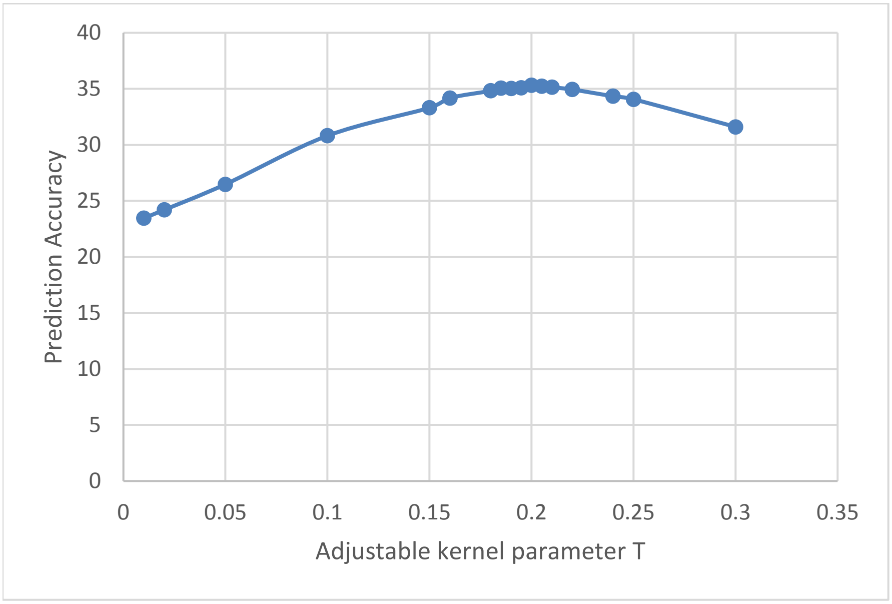


Figure 6.1: The accuracy as a function of the adjustable kernel function parameter T with 10K training set. The choice of parameter T has high influence on the prediction performance.

Training dataset of 10K is chosen for tuning, then dataset of 50K is used to verify

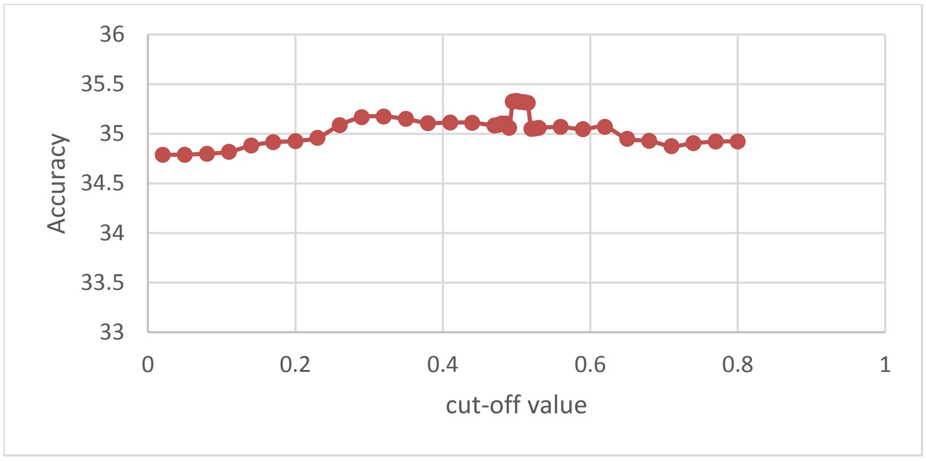


Figure 6.2: The accuracy as a function of cut-off values.

the suitability of the value selected. As shown in Figure 6.1, the prediction accuracy is sensitive to the value of T parameter. With increasing values of T parameter, the accuracy reaches a peak at $T = 0.2$ and then has a pronounced drop. To confirm the suitability of this value for T, we tested this setting with the 50K training set and obtained good results as well.

Second, the cut-off value was also considered, as shown in Figure 6.2. Compared with the impact of T parameter, the cut-off value affects accuracy very slightly. From the figure, we find that the highest and the lowest accuracy is approximately 35.3% , 34.8%, respectively. Then the cutoff $= 0.5$ is used in further experiments.

6.3 Kernel Selection

Kernel functions can be used in many applications as they provide a simple bridge from linearity to non-linearity for algorithms which can be expressed in terms of dot products. Choosing the most appropriate kernel highly depends on the problem at

Dataset size	Linear Kernel	Lib-Linear SVM	Histogram Kernel
10K	26.7048%	34.2691%	34.2802%
50K	31.185%	36.7276%	36.5738%

Table 6.2: Comparison between linear kernel and histogram kernel

hand, and will highly affect the algorithm's performance and time complexity. As our approach is solving the budget-driven classification problem, the kernel function can not be complex.

6.3.1 Linear Kernel

We would like to employ a simple kernel function to reduce the computational cost, while the linear kernel is the simplest kernel function. It is given by the inner product $< x, y >$ plus an optional constant c. Kernel algorithms using a linear kernel are often equivalent to their non-kernel counterparts, i.e. Kernel Principal Component Analysis (KPCA) with linear kernel is the same as standard PCA [26]. The formula is :

$$k(x, y) = x^T y + c \tag{6.1}$$

By incorporating the linear kernel, our approach produces suboptimal result, shown in Table 6.2, as compared with a benchmark approach, the Lib-Linear SVM [11]. We conduct the evaluation under training sets of 10K, and 50K. Experimental results reflect that with linear kernel, our approach is outperformed by Lib-linear SVM. Based on this results, we decided to look for another simple kernel function.

6.3.2 Histogram Kernel

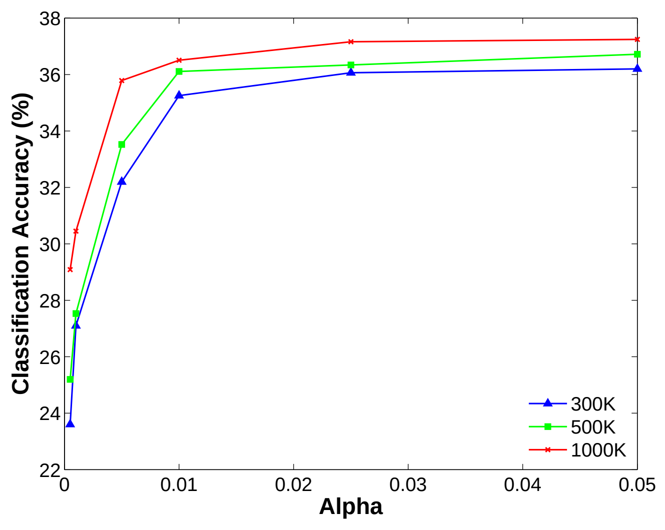
The histogram intersection kernel function fulfills the mathematical requirements for it to be used as a kernel for SVMs. Experiments show that it performs well, compared with standard kernels and with other state-of-the-art color kernels [4]. It also has the nice property of being easy to tune, since it only depends on fewer parameters, and it has been proven useful in image classification. The formula of Histogram Kernel is :

$$k(x, y) = \sum_{i=1}^{n} \min(x_i, y_i) \tag{6.2}$$

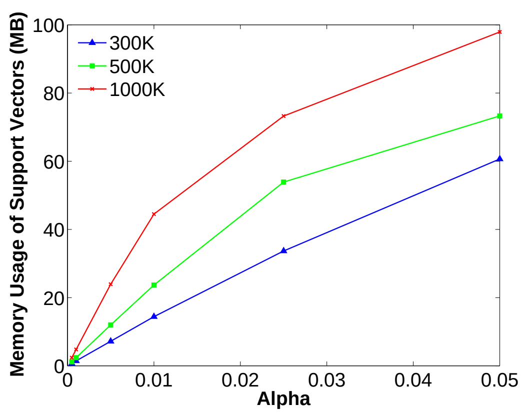
Experimental results of incorporating histogram kernel are shown in Table 6.2, compared with linear kernel and Lib-Linear SVM. The dataset and parameters setting are the same as previous experiments. The result reflects that by employing histogram intersection kernel function, our approach achieves promising performance, and is competitive with Lib-Linear SVM. Hence we incorporated histogram intersection kernel to solve this big data classification problem.

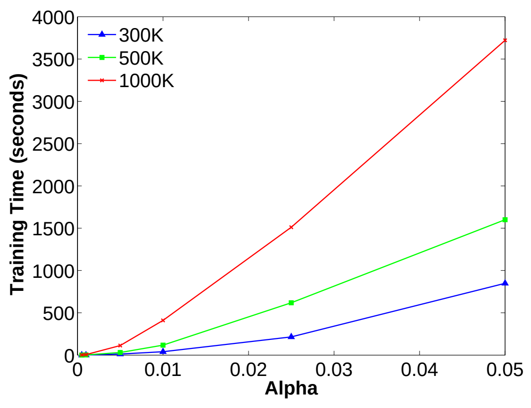
6.4 Evaluation of Proposed Budget-Driven Features

In this section, we present the evaluation of our approach in this particular social media set. We first evaluate the impact of the number of support vectors on accuracy, memory usage and training time. For the i_{th} class, we set the support vector buffer $n_i = \alpha N_i$, where N_i is the number of the observed examples belonging to the i_{th} class, and α is the parameter that controls support vector buffer size. The classification accuracy of our approach under different α values is shown in Figure 6.3. Keeping only a small number of support vector in the buffer produces lower accuracy, since



(a) Classification Accuracy





(b) Memory Usage of SVs

(c) Training Time

Figure 6.3: Evaluation of support vector number on three sets of $300K$, $500K$ and $1000K$ training examples from the Flickr-tag data. Figure (a)-(c) show the classification accuracy, memory usage of support vectors and training time used under different α values, respectively.

the limited number of support vectors cannot capture complex example distributions. As the α value increases, the classification accuracy improves rapidly at first and then levels off. It is worth noting that when using a large α value, e.g. 0.05, our approach becomes the conventional Competing One-Class SVM training, because the support vector buffers are large enough to store all support vectors. Our approach with a smaller α value can achieve similar performance to the conventional One-Class SVM training, while it only uses a fraction of support vectors and saves time and memory, as shown in Figure 6.3. We observe that setting $\alpha = 0.01$ provides a good trade-off regardless of the training size. Hence, we use $\alpha = 0.01$ for further comparisons.

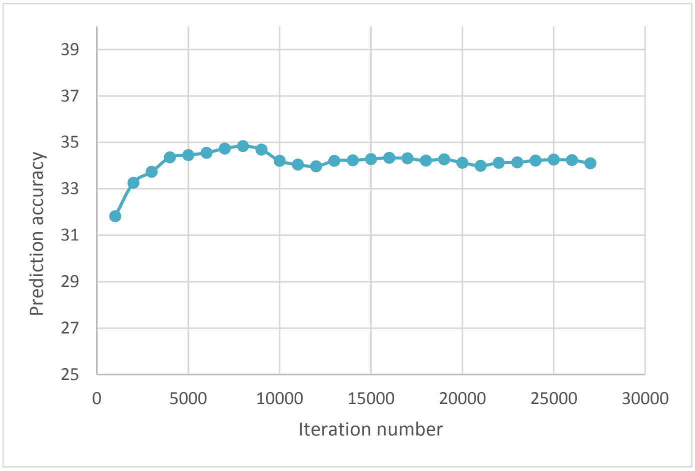


Figure 6.4: Analysis of only training a portion of examples. We evalute the model every 1000 iterations.

Then we present the performance of our approach when training on a portion of examples to show the faster convergence of our approach. In this experiment, we are using the dataset of 50K training examples from the Flickr-tag data. It is not feasible to evaluate our model after every iteration to find out the exact convergence condition. We apply our assumption about the convergence condition that support vectors lists do not change for at least $\beta * n$ times, where n is the number of examples in the training set and we evaluate our learner model every 1000 iterations. As shown in Figure 6.4, with the number of training examples increasing, the classification accuracy improves rapidly at first, then stays relatively stable. We find that our approach converges after only training on a small portion of examples when the problem size is large, which further confirms our assumption of redundancies and duplications in big data. From our experiments, we find that 0.001 is a suitable choice for β to result in a promising accuracy, which is very close to the peak accuracy value.

6.5 Comparisons with Existing Approaches

Finally we compare the performance of our approach using histogram kernels with LIB-Linear SVM and kNN algorithm in this section. LibLinear [17] is the most popular solver for large-scale data classification, and its extension [61] can handle data that cannot fit in memory using a block optimization method. The KNN classifier is used in recent work [51] [34] and is shown to be efficient on the Flick-tag prediction task.

To evaluate budget-driven features of our approach, performance on different problem sizes is evaluated. The comparison among our approach, LibLinear, and KNN classifier is shown in Figure 6.5. We use the original LibLinear for $< 500K$ dataset. When data cannot fit in memory, the extension of LibLinear is used for $\geq 500K$ dataset. Our approach performs competitively with LibLinear on small datasets, and outperforms LibLinear's extension on large datasets.

The following budget-driven features of our approach are demonstrated in Figure 6.5 : First, the training process can be controlled by the user. Users can terminate the training process anytime because of their time or memory limitations, and the

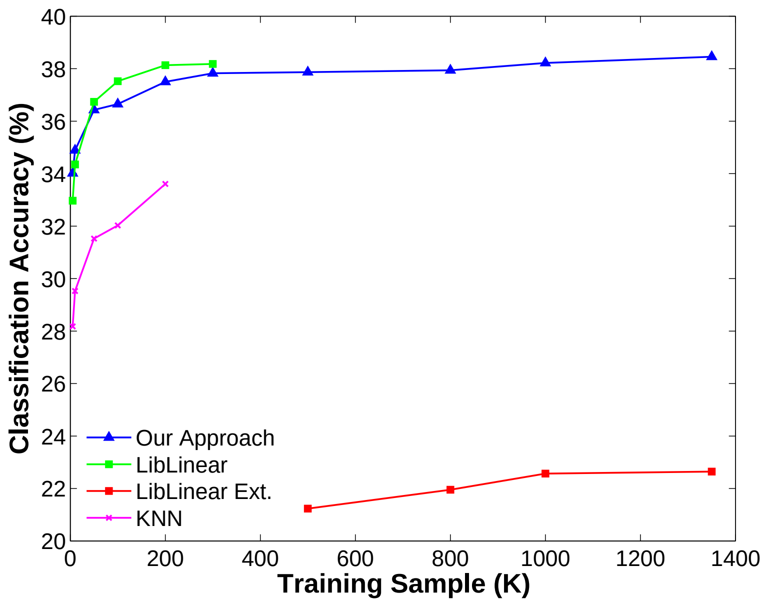


Figure 6.5: Comparisons between the proposed approach and LibLinear, KNN classifier. As the training set increases, KNN classifier and LibLinear can handle $200K$ and $300K$ examples at most because of memory limitation, respectively. In contrast, our approach can achieves superior performance than the extension of LibLinear in large sets.

partially trained model can still obtain competitive performance compared with LibLinear. Second, the accuracy of our approach stabilizes after observing only $300K$ examples. Reading and training with more examples does not significantly improve performance. This confirms our assumption of redundancy of big data, and a fraction of instances can produce encouraging results when training set is large enough. Note that the performance of LibLinear Extension is quite poor, our hypothesis is that the available version of LibLinear Extension is an experimental version and hence may not fully optimized.

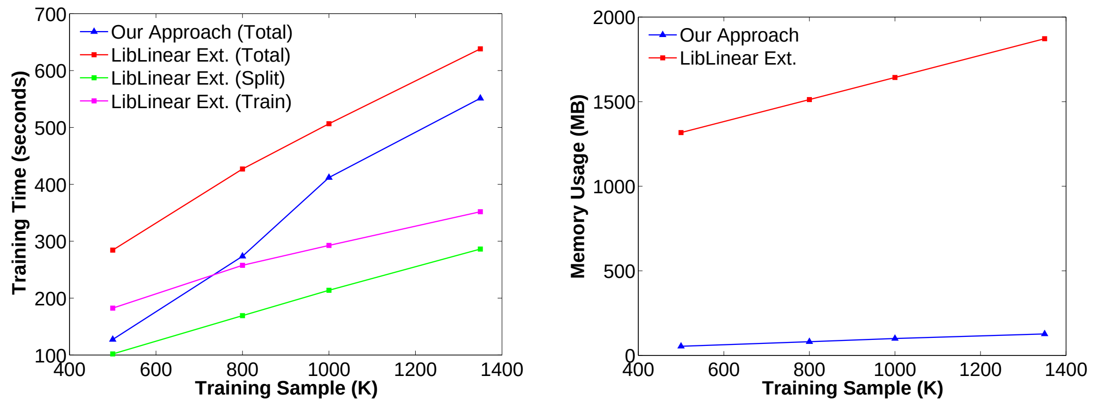


Figure 6.6: Processing seconds and memory usage on large training datasets. Left and right side show comparisons on training time and memory usage, respectively. The processing time of LibLinear extension consists of data splitting and training.

Besides achieving promising classification accuracy on large datasets, our approach can converge fast and save vast memory. In Figure 6.6, we compare memory usage and training time of our approach with that of LibLinear extension, where only $\geq 500K$ datasets are used to illustrate the effectiveness of our budget-driven approach on extreme large datasets. It is worth noting that LibLinear extension needs to split training set into blocks, a process that is time-consuming itself. Default parameters are used for LibLinear extension, where dataset is split into 8 blocks. These promising results reflect the fact that our approach could handle well the big data social media classification problem with limited budget.

6.6 Budget-driven Selective Labeling

From the experimental results, we found that even the best performance can only achieve approximately 38% prediction accuracy. We believe this result is not sufficient for real world applications. Hence, we applied the idea of "budget-driven" selective labeling proposed in the previous chapter. For the example in Flickr dataset, our approach calculates the largest score and the second largest score, if the difference is greater than the confidence value, the example is classified. Here, after conducting experiments, we find out that the range of the confidence value is 0 to 1.

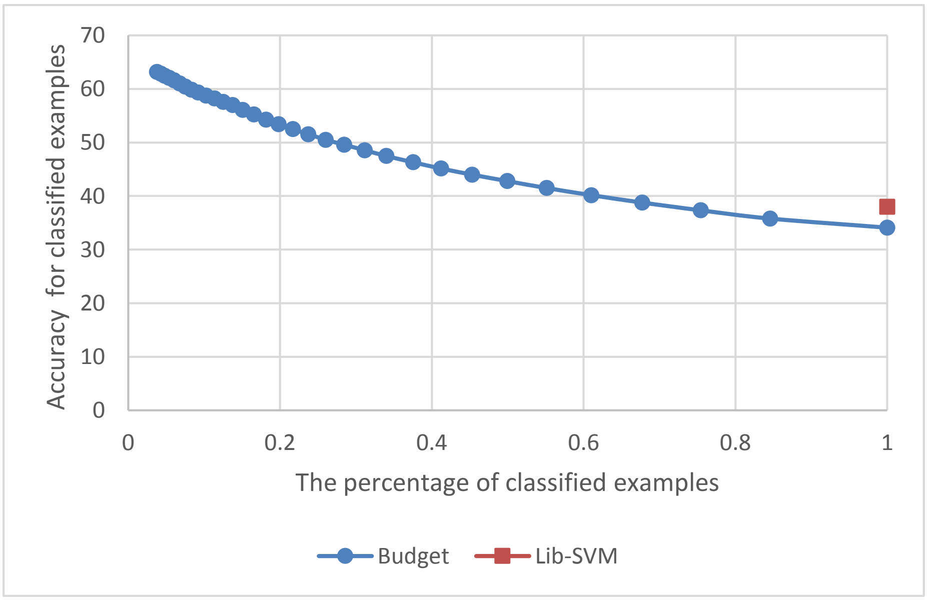


Figure 6.7: The accuracy as a function of the proportion of classified examples.

After employing the confidence, we generated the accuracy and recall curve, which is shown in Figure 6.7. We can conclude that, generally, the lower the percentage of examples get classified, the higher the accuracy our approach can achieve, where the highest accuracy we can reach is 63%. When our approach classifies 30% of the testing examples, it can obtain 50% classification accuracy, which is better than the original performance. Experimental results show that the proposed "budget-driven" classification performs well in this large scale Flickr dataset, further convincing us that the “budget-driven” classification will be useful in real world “limited-budget” applications.

6.7 Discussions

In this chapter, our approach is applied to Yahoo! Large-scale Flickr-tag Image Classification Grand Challenge and is shown to outperform other state-of-the-art approaches in term of accuracy, training time and memory requirement. We present our experiments to show the parameter tuning and kernel function selection. Experimental result demonstrates that our approach possesses superior performance comparing to the state-of-the-art approaches, such as LibLinear, especially on large-scale data, with much lower memory requirement and faster convergence time. Encouraging facts convince us that our approach could well handle the large scale social media classification under a limited-budget environment.

Chapter 7

Conclusions

The challenges of large scale social media data classification problems are studied in this thesis. To handle these challenges, this thesis proposes to extend the existing STOCS so that big data corrupted by labeling noise can be processed under limited budget environment. The proposed technique is finally evaluated using the real-world dataset from the Large-scale Flickr-tag Image Classification Grand Challenge, who has large scale, high dimensional, multiple classes with the existence of label noise.

We first present the underlying motivation of our fundamental method, STOCS, which incorporates the online learning frame and adjustable kernel function. When a new example is observed during training, only a simple score function computation is required to update the One-Class SVM models, allowing STOCS to react quickly to deal with data velocity. In contrast, batch learning requires to train the whole new datasets again, which is time consuming. As an extension of STOCS, our approach inherits properties of online learning framework, which are useful for budget-driven problems. In our approach, reading and training data happen at the same time, and we only need to store one training example and the support vectors.

Second, we study the noise-resilient ability of STOCS by conducting comparison experiments with well-known algorithms, Naïve Bayes, Decision Tree C4.5, Classification and Regression Tree, K-Nearest Neighbor Algorithm and Support Vector Machine. We concluded from experimental results that: (1) STOCS performs better in multi-class problems than binary problems; (2) with the increasing of label noise, prediction performance tends to become less accurate; (3) STOCS is the most robust approach among all selected algorithms against label noise; (4) in most cases, the results of STOCS are competitive with the best performance; (5) NAR label noise is shown to be more harmful than NCAR label noise.

Then, based on STOCS, we present our specific modifications for limited-budget classification. We consider the redundancies in the big data, where we believe competitive classification accuracy can be obtained by training only on a portion of training examples. In addition, we propose to only store the dominant support vectors to further reduce memory requirement and the convergence time. Our approach assumes the hyperspheres of STOCS are primarily determined by the dominant support vectors, i.e. those with high supporting weights. Experimental results reflect that our approach could achieve promising prediction accuracy with lower computational cost, which also confirms our assumption of redundancies and dominant support vectors.

Finally, we apply our approach to the selected social media problem and conduct comparison with state-of-the-art approaches, kNN algorithm and Lib-linear SVM. We give a brief introduction to the chosen grand challenge and discuss about the parameters tuning. Quantitative evaluations are performed on different problem size levels. Our approach achieves superior classification performance on extremely large data. In terms of computational resources needed, our approach only requires keeping a fraction of examples in memory as support vectors, and the training can converge in only a portion of training examples. In addition, only the most dominant support vectors are used in our approach, further saving memory.

In summary, our approach can be applied to the big data classification problems for social media data under limited budget environment by extending STOCS algorithm. Experiment results show that our approach outperforms other state-of-the-art approaches in term of accuracy, training time and memory requirement. It is worth noting that our work has been accepted by the Canadian AI 2015 and wins the Best Paper Award [41]. In the future, we would like to apply our approach to more social media datasets and explore the possibility of implementing our technique on GPUs to further reduce the training time needed.

Bibliography

[1] J. Abelln and A. Masegosa. Bagging decision trees on data sets with classification noise. In S. Link and H. Prade, editors, *Foundations of Information and Knowledge Systems*, volume 5956 of *Lecture Notes in Computer Science*, pages 248–265. Springer Berlin Heidelberg, 2010.

[2] ACM2013. Yahoo! large-scale flickr-tag image classification grand challenge. `http://webscope.sandbox.yahoo.com/catalog.php?datatype=i`, 2013. [Online, accessed February-2015].

[3] Aggarwal and C. Charu. *Data classification: algorithms and applications.* CRC Press, 2014.

[4] A. Barla, F. Odone, and A. Verri. Histogram intersection kernel for image classification. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 3, pages III–513–16 vol.2, Sept 2003.

[5] M. Blondel, K. Seki, and K. Uehara. Block coordinate descent algorithms for large-scale sparse multiclass classification. *Machine Learning*, 93(1):31–52, 2013.

[6] O. Bousquet and L. Bottou. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168, 2008.

[7] C. Bouveyron and S. Girard. Robust supervised classification with mixture models: Learning from data with uncertain labels. *Pattern Recognition*, 42(11):2649 – 2658, 2009.

[8] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[9] F. Breve, L. Zhao, and M. Quiles. Semi-supervised learning from imperfect data through particle cooperation and competition. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8, July 2010.

[10] Chang and Lin. LibSVM dataset. `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`. [Online; accessed July-2014].

[11] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011.

[12] K.-W. Chang and D. Roth. Selective block minimization for faster convergence of limited memory large-scale linear models. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 699–707, New York, NY, USA, 2011. ACM.

[13] L. Cheng, M. Gong, D. Schuurmans, and T. Caelli. Real-time discriminative background subtraction. *Image Processing, IEEE Transactions on*, 20(5):1401–1414, May 2011.

[14] T. Cover and P. Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, January 1967.

[15] P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130, 1997.

[16] E. Eskin. Detecting errors within a corpus using anomaly detection. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, NAACL 2000, pages 148–153, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[17] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, pages 1871–1874, 2008.

[18] B. Frenay and M. Verleysen. Classification in the presence of label noise: A survey. *Neural Networks and Learning Systems, IEEE Transactions on*, pages 845 – 869, 2013.

[19] Y. Freund. An adaptive version of the boost by majority algorithm. *Machine Learning*, 43(3):293–318, 2001.

[20] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, Aug. 1997.

[21] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998.

[22] J. Fürnkranz. Noise-tolerant windowing. In *Proceedings of the Fifteenth international joint conference on Artifical intelligence*, pages 852–857. Citeseer, 1997.

[23] A. Ghoting, R. Krishnamurthy, E. Pednault, B. Reinwald, V. Sindhwani, S. Tatikonda, Y. Tian, and S. Vaithyanathan. Systemml: Declarative machine learning on mapreduce. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*, ICDE '11, pages 231–242, Washington, DC, USA, 2011. IEEE Computer Society.

[24] K.-S. Goh, E. Chang, and B. Li. Using one-class and two-class SVMs for multiclass image annotation. *Knowledge and Data Engineering, IEEE Transactions on*, 17(10):1333–1346, Oct 2005.

[25] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[26] H. Hoffmann. Kernel PCA for novelty detection. *Pattern Recognition*, 40(3):863 – 874, 2007.

[27] T. Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 217–226, New York, NY, USA, 2006. ACM.

[28] G. H. John and P. Langley. Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, UAI'95, pages 338–345, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[29] A. M. Kaplan and M. Haenlein. Users of the world, unite! the challenges and opportunities of social media. *Business Horizons*, 53(1):59 – 68, 2010.

[30] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, pages 3–24, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press.

[31] P. A. Lachenbruch. Discriminant analysis when the initial samples are misclassified ii: Non-random misclassification models. *Technometrics*, 16(3):pp. 419–424, 1974.

[32] J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *J. Mach. Learn. Res.*, 10:777–801, June 2009.

[33] M. Lichman. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2013.

[34] E. Mantziou, S. Papadopoulos, and Y. Kompatsiaris. Scalable training with approximate incremental laplacian eigenmaps and PCA. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, pages 381–384, New York, NY, USA, 2013. ACM.

[35] N. Manwani and P. Sastry. Noise tolerance under risk minimization. *Cybernetics, IEEE Transactions on*, 43(3):1146–1151, 2013.

[36] D. F. Nettleton, A. Orriols-Puig, and A. Fornells. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artif Intell Rev*, 2010.

[37] F. Nie, Y. Huang, X. Wang, and H. Huang. New primal SVM solver with linear computational cost for big data classifications. *The 31st International Conference on Machine Learning (ICML)*, 2014.

[38] M. Pechenizkiy, A. Tsymbal, S. Puuronen, and O. Pechenizkiy. Class noise and supervised learning in medical domains: The effect of feature extraction. In *Computer-Based Medical Systems, 2006. CBMS 2006. 19th IEEE International Symposium on*, pages 708–713, 2006.

[39] C. J. Pérez, F. J. G. González-Torre, J. Martín, M. Ruiz, and C. Rojano. Misclassified multinomial data: a Bayesian approach. *Revista de la Real Academia de Ciencias Exactas, Físicas y Naturales. Serie A: Matemáticas (RACSAM)*, 101(1):71–80, 2007.

[40] Y. Qian, M. Gong, and L. Cheng. Stocs: An efficient self-tuning multiclass classification approach. In *Advances in Artificial Intelligence*, volume 9091 of *Lecture Notes in Computer Science*, pages 291–306. Springer International Publishing, 2015.

[41] Y. Qian, H. Yuan, and M. Gong. Budget-driven big data classification. In *Advances in Artificial Intelligence*, volume 9091 of *Lecture Notes in Computer Science*, pages 71–83. Springer International Publishing, 2015.

[42] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[43] R. Rakotomalala. Decision tree learning algorithms. `http://www.data-mining-tutorials.blogspot.fr`. [Online; accessed July-2014].

[44] M. Rantalainen and C. C. Holmes. Accounting for control mislabeling in case–control biomarker studies. *Journal of proteome research*, 10(12):5562–5567, 2011.

[45] J. S. Sánchez, F. Pla, and F. J. Ferri. Prototype selection for the nearest neighbour rule through proximity graphs. *Pattern Recogn. Lett.*, 18(6):507–513, June 1997.

[46] N. V. Sawant, K. Shah, and V. A. Bharadi. Survey on data mining classification techniques. In *Proceedings of the International Conference ; Workshop on Emerging Trends in Technology*, ICWET '11, pages 1380–1380, New York, NY, USA, 2011. ACM.

[47] J. L. Schafer and J. W. Graham. Missing data: our view of the state of the art. *Psychological methods*, 7(2):147, 2002.

[48] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt. Support vector method for novelty detection. In *Advances in Neural Information Processing Systems 12*, pages 582–588. 1999.

[49] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, pages 3–30, 2011.

[50] C. R. Souza. Kernel functions for machine learning applications. *Creative Commons Attribution-Noncommercial-Share Alike*, 3, 2010.

[51] Y.-C. Su, T.-H. Chiu, G.-L. Wu, C.-Y. Yeh, F. Wu, and W. Hsu. Flickr-tag prediction using multi-modal fusion and meta information. In *Proceedings of*

the 21st ACM International Conference on Multimedia, MM '13, pages 353–356, New York, NY, USA, 2013. ACM.

[52] P. Tan. *Introduction To Data Mining.* Pearson Education, 2007.

[53] D. Tax and R. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, 2004.

[54] C. Teng. Evaluating noise correction. In *PRICAI 2000 Topics in Artificial Intelligence*, volume 1886 of *Lecture Notes in Computer Science*, pages 188–198. Springer Berlin Heidelberg, 2000.

[55] C.-M. Teng. A comparison of noise handling techniques. In *Proceedings of the Fourteenth International Florida Artificial Intelligence Research Society Conference*, pages 269–273. AAAI Press, 2001.

[56] J. Thongkam, G. Xu, Y. Zhang, and F. Huang. Support vector machine for outlier detection in breast cancer survivability prediction. In *Advanced Web and Network Technologies, and Applications*, volume 4977 of *Lecture Notes in Computer Science*, pages 99–109. Springer Berlin Heidelberg, 2008.

[57] K. Tumer and J. Ghosh. Estimating the bayes error rate through classifier combining. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 2, pages 695–699 vol.2, Aug 1996.

[58] D. R. Wilson and T. R. Martinez. Reduction techniques for instance-based learning algorithms. *Mach. Learn.*, 38(3):257–286, Mar. 2000.

[59] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 14(1):1–37, Dec. 2007.

[60] H. Yin and H. Dong. The problem of noise in classification: Past, current and future work. In *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, pages 412–416, May 2011.

[61] H.-F. Yu, C.-J. Hsieh, K.-W. Chang, and C.-J. Lin. Large linear classification when data cannot fit in memory. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 833–842, New York, NY, USA, 2010. ACM.

[62] K. Zhang, L. Lan, Z. Wang, and F. Moerchen. Scaling up kernel SVM on limited resources: A low-rank linearization approach. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS-12)*, volume 22, pages 1425–1434, 2012.

[63] X. Zhu and X. Wu. Class noise vs. attribute noise: A quantitative study. *Artif. Intell. Rev*, 2004.