Incremental Learning and Online-Style SVM for Traffic Light Classification

by

Wen Liu

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Electrical and Computer Engineering

by

December 2015

APPROVED:

Professor Xinming Huang, Major Thesis Advisor

Professor Yehia Massoud, Department Head

Abstract

Training a large dataset has become a serious issue for researchers because it requires large memories and can take a long time for computing. People are trying to process large scale dataset not only by changing programming model, such as using MapReduce and Hadoop, but also by designing new algorithms that can retain performance with less complexity and runtime. In this thesis, we present implementations of incremental learning and online learning methods to classify a large traffic light dataset for traffic light recognition.

The introduction part includes the concepts and related works of incremental learning and online learning. The main algorithm is a modification of IMORL incremental learning model to enhance its performance over our application's learning process. Then we briefly discuss how the traffic light recognition algorithm works and the problem we encounter during training. Rather than focusing on incremental learning, which uses batch to batch data during training procedure, we introduce Pegasos, an online style primal gradient-based support vector machine method. The performance of Pegasos for classification is extraordinary and the number of instances it uses for training is relatively small. Therefore, Pegasos is the recommended solution to the large dataset training problem.

Acknowledgements

I would like to express my gratitude to my advisor, Professor Xinming Huang. He is really generous to provide help to his students. During the thesis project, he suggested a lot of brilliant ideas to support my research.

Many thanks also to my dear friends, Chao Li and Xiaotong Zhang. They gave me numerous supports both in daily life and academic study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the overall picture of incremental learning - a machine learning paradigm where the learning progress is activated to make adjustment when a batch of new examples emerges. It covers the motivation and background of incremental learning research, the general mathematical formulation, current researches that make significant contributions to the algorithm, and the objectives of using incremental learning in our work. Also, it briefly covers some research works of online learning - a typical machine learning technique when data becomes available in a sequence.

1.1 Motivation

Incremental learning has became an attractive topic in academic research fields such as artificial intelligent and data mining. It is more like the nature of human accessing and accumulating experience, associating current scenarios to past, making decisions to solve problem and learning from new experiences. Apparently, people tend to be good at summarizing and learning concepts from the past observations and incrementally refining those concepts when a new observation comes in. This

is a biological intelligent system to the artificial intelligence perspective. Incremental learning plays an important role in such biological intelligence research. The framework of incremental learning is expected to close to the biologically learning capability in the future.

In [2], two major issues that make human learn incrementally are: receiving information sequentially; limited memory and manipulation energy. Those above causes are very correlative with the serious data problem caused by large-scale dynamic stream data. Researchers in data mining perspectively focus on resolving large scale data computing difficulty and storage problem. Hence, the incremental learning method has become a potential solution to analyze large-scale streaming data and summarize useful information from data in some situations. More specifically, when the observations occur in a stream over time and the learning result is desired to be known at any moment, incremental learning becomes very useful. Besides, due to the unforeseeable characteristic of observations, it would be better if the existing hypothesis can be updated while receiving the new observations. A daily example for incremental learning is that when Apple Inc. releases a new version of iPhone, people add a new subcategory for iPhone to their cognitive scope. The recognition hypothesis is updated when this new instance is received.

1.2 Characteristics

One of the characteristics of incremental learning is that it does not necessarily need to keep original data. A simple but representative example mentioned in [3] is learning the mean of the observations from data stream. The ordinary way to compute the mean is by definition, to calculate the sum of all observations and then divide the result by the total number of observations. Assuming we have m

observations, the mean of those values is M. When n new observations are received, we have V denoted by the sum of those observations. Then, the new mean M' can be computed by $\frac{Mm+V}{m+n}$. In this way, the computing cost is reduced and the memory usage is much less since we only need to store the mean of old observations and the number of observations. Incremental learning algorithm may not aggregate the set of observations, but to modify the current knowledge. Besides, the result of incremental learning should also have the same performance with accumulated computation.

The distribution of stream data is unknown, which is contrast to the assumption of some machine learning algorithms. Hence, the concern of concept drifting caused by non-stationary environment is also a characteristic for some incremental learning methods. This is because the uncertainty of new coming observations are influenced by their previous observations. Therefore, it is really dangerous to combine current knowledge and learning from new data because the distribution is conditionally dependent on learner. Making the existing hypothesis adapt to new data in non-stationary environment is a big challenge for incremental learning.

To emphasize the concept of incremental learning, [2] and [4] both give a clear description of incremental learning approach:

1) the capability of learning and update knowledge with new batch of labeled or unlabeled data.

2) preserve previously acquired knowledge.

3) generate, divide or merge categories or clusters if required.

4) change dynamically in non-stationary environment.

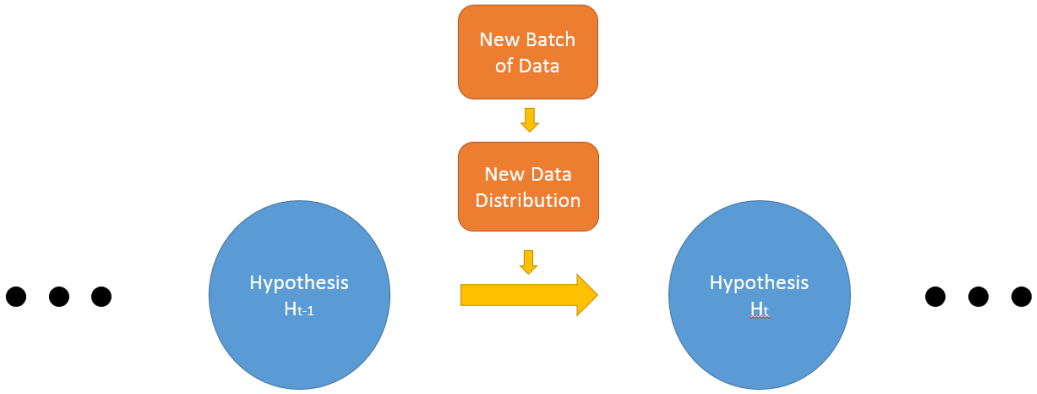

Figure 1.1: General incremental learning framework, where t is denoted as a point of time

1.3 Background

Rather than restricting with data sample size, incremental learning takes the advantage of learning from big data. In recent years, many approaches in machine learning and data mining have integrated with the concept of incremental learning. Incremental learning algorithms and their architectures have been well studied. Researchers have applied those algorithms to various application domains.

In order to realize the incremental learning ability over data flow, there are two major types of learning algorithms [5]. Data accumulation strategy is the first category of these approaches. It means that when a new data chunk comes in,

the new hypothesis is developed by all the data chunks arrived so far. This kind of methods applies previous hypotheses as the warm start to generate the new hypothesis. The benefit is that the new hypothesis can be generated more quickly when previous knowledge is introduced. The problem of data accumulation strategy is the limitation of memory for storing data. It needs to create a new area to store the data each time, which is not quite efficient in computation.

The other kind of incremental learning is called ensemble learning. The primary thought of this strategy is to formulate a series of hypotheses along with the data chunks and make a combination of this hypotheses to generate a unified hypothesis to fit all the prediction of data. This strategy only needs to store all the hypotheses generated in each moment and the previous data can be destroyed. The shortcoming of this strategy is that the hypothesis generated by the current dataset can not directly benefit to the dataset. It is because all the prediction is made by the final decision hypothesis. The voting of the final hypothesis is a knowledge integration process.

A series of incremental learning algorithms named after Learn++ is developed by the idea of ensemble learning. Learn++ [6] is an algorithm for incremental training of neural network pattern classifiers. The method of avoiding forgetting previous acquired knowledge in this algorithm is that it ensures the data instances that are misclassified by the current hypothesis have a higher probability of being sampled. The final classification hypothesis output is a combination of a weighted majority voting procedure. Learn++.MT [7] is a modified version of [6] and it solved the out-voting problem when [7] was asked for learning newly added classes. Learn++.NC [8] is designed for incremental learning new classes in data as well as keep previous classification stable. This algorithm introduces a new voting procedure, dynamically weighted consult and vote (DW-CAV). Learn++.MF [9] is also an ensemble

classifier based algorithm with the ability of addressing the missing feature problem by random selection of subspace. The majority voting mechanism can also be used to classify those data instances with missing features. The imbalance of data class size is one of the causes of non-stationary environment. Learn++.NSE [10] is developed to make the incremental learning procedure robust in such non-stationary environment. [11] is more concentrated on resolving the concept drifting problem in non-stationary environment.

[12] also proposed an ensemble incremental learning algorithm called IMORL. It can adaptively learn new principles in video stream data, such as recognizing a new objective with newly labeled image frame data. The majority voting mechanism is quite similar to Learn++. [5] proposed a more general adaptive incremental learning framework (ADAIN) and validated the efficiency of this framework. We will have more descriptions and discussions on this method in the next chapter.

Besides the algorithms mentioned above, some other incremental learning approaches were proposed with different valuable ideas and structures. In [13], an extended incremental principal component analysis (IPCA) combined with some other classifiers models was proposed. It also introduced an extension of IPCA which was called chunk IPCA. A semi-supervised incremental learning aproach was presented in [14]. It proposed pattern based techniques which selected some unlabeled data to enhance the performance of classification in semi-supervised learning. [15] performed an online incremental learning method in semi-supervised learning. The algorithm firstly generated a learning topology of data distribution without prior knowledge, and then it uses a small amount of labeled data to generate nodes and divisions of the learning topology to represent data classes. [16] presented a survey of incremental learning using Hidden Markov Models (HMMs). It reviewed existing technologies which can resolve the restrictions of using limited training data

and prior knowledge in HMMs.

The term online machine learning often accompanies with incremental learning. It is also used when data is in a sequential order. It updates the mapping after the arrival of every new data point. The paradigm is really similar to incremental learning, so it is a point of confusion to call an algorithm incremental if its methodology also responses to new observations. [3] said the criterion depends on whether the result is the same that one would obtain when starting with all the observations in hand. If yes, it is incremental. [5] said that online learning also addressed to incremental learning. In order to draw a complete research background, selected literatures with online learning will be mentioned below as references.

Researchers have studied a lot on the online versions of current machine learning techniques. [17] discussed Passive-Agressive (PA) algorithm based online learning approaches, which construct the new support vector machine (SVM) style hypothesis that is close to the old hypothesis as well as meets a unit margin to new data points. [18] proposed an online SVM structure named LASVM to solve classification problem for very large dataset. The major contribution of this algorithm is the improvement of computational efficiency. After that, a series of LASVM was proposed, such as LASVM-NC, SVM-G. [19] proposed an online algrothm named PEGASOS for training SVM using training data in batchs or instance by instance. PEGASOS is a state-of-the-art SVM classifier. This method decreases the iterations steps for SVM training and fast converges to the solution under required accuracy. It can significantly shorten the training time for large scale data, such as text classification. In this work, we implement PEGASOS, Perceptron, Online Passive Agressive-1 and Online Independent SVM to solve the data training problem in our objective detection system.

Besides online style SVM, incremental SVM has also been studied. Incremental

SVM is applied to quickly update the model when a small number of instances is added into the dataset. The previous SVM becomes a warm start to the new SVM. The training time for the dataset with incremental data will be decreased. [20] also validated that for incremental linear SVM, a warm start setting towards a high-order optimization method for primal SVM formulations is more suitable than dual.

Other machine learning techniques also have online or incremental learning formulations. For instance, [21] modified classification and regression trees (CART) to incorporate data incrementally. In decision tree methods, ID3 (Iteractive Dichotomiser 3) is also a fundamental algorithm to generate decision tree towards dataset. A family of incremental decision trees was invented based on ID3. It includes ID3' [22], ID4 [22], ID5 [23], ID5R [24], ID6MDL and ITI [25]. Random decision forest is another popular technique for classification and regression. It constructs a multitude of decision tress for training data and outputs the classification result by calculating the mode or the mean of each tree's prediction. An online random forests algorithm was proposed by [26]. This algorithm is an extension of offline random forests and it is able to learning from online training observations. The authors used the ideas from online bagging, extremely randomized forests and formulated an online decision tree growing procedure.

For unsupervised learning, incremental clustering is also a research topic that attracts people's attention. [27] introduced an incremental hierarchical clustering algorithm named GRIN, which was based on gravitational theory in physics. The optimal parameter settings of GRIN are not really sensitive to the probability distribution of observations, so GRIN can perform high clustering quality to many datasets. [28] proposed a fast incremental clustering algorithm. This algorithm is based on distance measurement. It could change the radius threshold value of the clusters dynamically. Besides, the final cluster number should be restricted to a

certain level. [29] proposed an incremental clustering to analyze dynamic relational dataset. It uses representative objects and the balanced search tree to accelerate the learning process. The data only passes the model once. [30] proposed the first incremental clustering algorithm. This algorithm is based on DBSCAN, which is a powerful method to cluster data without specifying cluster number in advance. This incremental algorithm could yield to the same result as DBSCAN. Adaptive resonance theory (ART) is an unsupervised learning model which describes a number of unsupervised neural network models. It consists of a comparison field, a recognition field of neurons, a threshold parameter of recognition and a reset model. A revised ART model was proposed in [31] in order to analyze mixed data and enhance the effect on data clustering. [32] proposed a novel incremental approach to cluster interval data. This algorithm is designed to capture the inherent uncertainty relative to cluster analysis.

Incremental learning techniques have been used to solve many problems in several application domains. In computer vision domain, object recognition is involved with large scale datasets due to the demand of image information. [33] used IPCA for online face recognition. In [34], it also used incremental linear discriminant analysis (ILDA) for face recognition system. Based on ILDA and the generalized singular value decomposition, authors developed GSVD-ILDA, which could determine the projection matrix in full space and reduce the computational complexity. In object tracking, online learning is widely used. For instance, [35] proposed an online tracking method that uses online SVM and an explicit feature mapping method for updating the model. The experiment has shown that the proposed restoration scheme could improve the robustness of the base tracker. [36] proposed a vision based autonomous land vehicle (ALV) navigation approach associate with incremental learning. The incremental learning procedure makes the learned knowledge

of indoor environment could being utilized to update the model in each iteration. Text mining is also an application field with large amount of data and high dimension features. Therefore, it is not really practical to train all the data at once. [37] presented their academic work in incremental text document classification. This system includes text feature learning and incremental perception learning process. [38] used incremental learning technique to solve search problem in query database. [39] performed a incremental machine learning approach to segment medical image data. Also in medical research domain, [16] performed a new index oriented classification system for patient health surveillance. The method for analyzing the large scale dataset is Bayes tree.

1.4 Problem Formulation

Incremental learning can be used in many problems and applications. In this section, we only focus on classification problem with incremental learning approach because it is the major problem we are facing in our training work. Therefore, let's assume a data instance which is $\{x, y\}$, where x is the feature in N dimension space and y is the classified label along with instance x. Let's also assume that data is a stream in time sequence. A new batch of data D_t is received at time point t, and D_{t-1} is the previous data chunk at $(t-1)$. For each data chunk D_t, the sampling procedure of new observations follows a distribution P_t. It should be noted that not all the incremental learning methods consider to use P_t in their framework. The hypothesis of classification at t is denoted as h_t. The final classification hypothesis is H_f. Hence, we can generate incremental learning frameworks by these representatives.

1.4.1 Data Accumulation Strategy

The first framework of incremental learning is data accumulation. Usually, P_t is not necessary to be considered in data accumulation because all the data is used for updating the new hypothesis h_t. Concept drifting won't be able to happen. The dataset $\mathscr{D}_t$ is $\{D_1, \ldots, D_t\}$ at each time t. The hypothesis set H_t is $\{h_1, \ldots, h_t\}$. The pseudo-code of this framework is shown in Figure 1.2.

DATA ACCUMULATION
(1) **train** h_1 by D_1
(2) **for** $t \leftarrow 2$ **to** T **do**
    $\mathscr{D}_t \leftarrow \mathscr{D}_{t-1} \cup D_t$
    train h_t **by** $\mathscr{D}$, H_{t-1}
    $H_t \leftarrow H_{t-1} \cup h_t$
(3) **return** $H_f = h_T$

Figure 1.2: Data accumulation learning framework

1.4.2 Ensemble Learning Strategy

The second framework is ensemble learning strategy. It is widely used in incremental learning research and in some online learning works. This method discards the previous data when a new data instance or batch arrives. Therefore, in order to avoid concept drifting, the new data should compare with the previous data under certain methods. In order to enhance the weight of new observations, a proper probability distribution is generated to re-sample the new data batch. The new hypothesis is trained by the sampled data chunk. The final hypothesis is a combination of the hypotheses generated in previous stages by a voting mechanism $\mathcal{G}$. The general

framework of this strategy is shown in Figure 1.3.

ENSEMBLE LEARNING
(1) **train** h_1 by D_1, $P_1 = P(D_1)$
(2) **for** $t \leftarrow 2$ **to** T **do**
$P_t = \mathcal{F}(D_t,\ D_{t-1},\ P_{t-1},\ h_{t-1})$
Train h_t **by** P_t, D_t
$H_t \leftarrow H_{t-1} \cup h_t$
(3) **Caculate** $H_f = \mathcal{G}(H_t)$
(4) **Return** H_f

Figure 1.3: Ensemble learning framework

1.5 Project Overview

In this work, we conduct two major projects related to incremental machine learning and online learning techniques. The objective of implementing these methods is to train the large dataset from our traffic light application.

The first project is inspired by the incremental learning structure proposed in [12, 5]. The adaptive ensemble learning model in [12] is implemented with the random forest classifier as the basic learner in our work. We also run this modified model on the selected standard machine learning datasets for classification. The performance of binary class classification under three test datasets turns out to be better than the IMORL framework originally proposed in [12].

The second project is to use PEGASOS to solve the challenge of training the large-scale traffic light dataset created in our laboratory. We also compare the experimental results with some popular online learning algorithms.

The rest of paper is organized as follows. Chapter 2 introduces the combined

incremental learning method of [12] and random forest classification. The experimental results are also included in Chapter 2. Chapter 3 discusses the traffic light detection system, deep learning network and data features. Pegasos and the online learning algorithms are introduced in Chapter 3. The implementation results are also described in Chapter 3 as well. Then the conclusions and future works are summarized and discussed in Chapter 4.

Chapter 2

Classification by Adaptive Ensemble Learning

In this chapter, we introduce a general Adaptive Incremental Learning framework (ADAIN) and Random Forest (RF) classification method. We implement the first two methods together and formulate a system with a relatively high performance for classifying some benchmark testing datasets. The experimental results are presented near the end of this chapter.

2.1 ADAIN

ADAIN is proposed by [5], which is motivated by the adaptive boosting methods in [40, 41]. The aim of this method is to integrate knowledge from previous data to emphasize learning process in new data. The final decision of classification hypothesis is made by integrate hypotheses generated in each incremental learning procedure.

In this framework, the weight distribution function P_t plays a key role to realize the attribute of adaptiveness. Several computation steps are used to generate P_t

at first. For initialization, the distribution function of the new data chunk D_t is evaluated by the function:

$$\hat{P}_{t-1} = \varphi(P_{t-1}, D_{t-1}, D_t) \tag{2.1}$$

φ is called mapping function, which makes the linkage from the previous data and knowledge to the new data chunk. φ can be customized to a suitable estimation function.

The $\hat{P}_{t-1}$ still needs to be adjusted in order to describe the distribution of the new data. The pseudo-error of h_{t-1} is calculated by:

$$\epsilon_{t-1} = \sum_{j:h_{t-1}(x_j)\neq y_j} \hat{P}_{t-1}(j) \tag{2.2}$$

where j is the index of instance $\{x_j, y_j\} \in D_t$, and $j \in [1, m]$.

The scale of pseudo-error is:

$$\beta_{t-1} = \frac{\epsilon_{t-1}}{1 - \epsilon_{t-1}} \tag{2.3}$$

The pseudo-error measures how well the classification result of h_{t-1} to D_t performs to each new instance. If an instance is misclassified by h_{t-1}, it will gain a higher weight in the distribution function P_t. Therefore, the decision boundary is better adjusted by hard instances. The distribution function is finally determined by:

$$\hat{P}_t(j) = \frac{\hat{P}_{t-1}(j)}{Z_t} \times \begin{cases} \beta_{t-1}, & for h_{t-1}(x_j) = y_j \\ 1, & for h_{t-1}(x_j) \neq y_j \end{cases} \tag{2.4}$$

where Z_t is a normalization parameter to guarantee that P_t is a distribution.

Then, the new hypothesis h_t is calculated by D'_t, where D'_t is generated by re-

sampling D_t under distribution P_t,

After all the data chunks are processed, the final hypothesis is calculated by the voting mechanism of ensemble learning:

$$H_{final}(x) = \arg\max \sum_{t:h_T(x)=y} \log(\frac{1}{\beta_t}) \tag{2.5}$$

The mapping functions proposed in [5] are a multilayer perception neural network (MLP) and support vector regression (SVR).

In [12], the mapping function φ is relative to distance map between D_{t-1} and D_t. The format of distance map is $[I, Q]$, where I is the index of the closet point to each instance in D_t from D_{t-1} and Q is the Euclidean distance for that instance. Suppose $x_j \in D_t$, $x_i \in D_t - 1$ and the feature dimension of x is n. Hence, the distance map can be formulated:

$$DM_{ji} = \sqrt{\sum_{k=1}^{n}(x_{jk} - x_{ik})^2} \tag{2.6}$$

$$I_j = \arg\min(DM_{ji}) \tag{2.7}$$

$$Q_j = \min(DM_{ji}) \tag{2.8}$$

Then, Q is scaled to update the distribution of $\hat{P}_t$:

$$\hat{Q} = 1 - e^{-Q} \tag{2.9}$$

$$Q_s = 1/e^{\hat{Q}} \tag{2.10}$$

The initial distribtuion $\hat{p}_t$ is updated by Q_s. This procedure is also able to pass

the previous learning information to the new observations.

$$\hat{P}_t = \frac{P_{t-1}(I) \times Q_s}{Z_t} \tag{2.11}$$

The rest steps in [12] are similar to the ADAIN format. It uses h_{t-1} to test the new data chunk and update $\hat{P}_t$ to P_t by giving a higher weight to misclassified instances. Through this adjustment procedure, the instances which are hard to learn by the previous hypothesis is exaggerated and the new hypothesis is forced to the decision boundary.

The hypothesis h in the previous works is Classification and Regression Tree (CART). Here, we implement the Random Forest [42] classifier as the basic learning processor in order to enhance the classification result of this framework. The new implementation is called IMORL.RF in this paper.

2.2 Random Forest

Random forest is initially proposed by [42] in 2001. It is an ensemble learning technique which can be used for classification, clustering, regression and survival analysis. Here, we briefly introduce the classification procedure for random forest.

2.2.1 Single Decision Tree

Random forest algorithm is developed by multiple decision trees. The construction of a decision tree is described as the following steps:

1) Assume training set $D = d_1, \ldots, d_n$ and the samples in each set is $X = x_1, \ldots, x_n$, then the input samples for decision tree are randomly selected in each sample set with replacement.

2) Set each training sample $x_j, j \in [1, n]$ has M features. When training the decision tree, we randomly select $m(m < M)$ features as the input features of the tree. Then, we select the important feature in m for splitting. m is invariant when constructing the decision tree.

3) The splitting process continues until all the samples in one node are in the same class.

2.2.2 Random Forest Formulation

After generating t decision trees, random forest algorithm ensembles the outcomes from decision trees as the final outputs of classification results. There are two ways to integrate the classification results:

1) Set the mean of t decision trees.

2) The minority is subordinate to the majority.

Random forest method does not need cross validation to estimate the accuracy of the classification result. The base estimator for random forest is called out-of-bag (OOB). When constructing a decision tree, not all the samples are used for training. Hence, the remaining samples can be used as testing instances to test this decision tree. The proportion of this kind of testing samples can be customized. For each sample x, we use the decision trees which don't count this sample into training set to estimate its class. The false estimation percentage is the OOB error for the random forest.

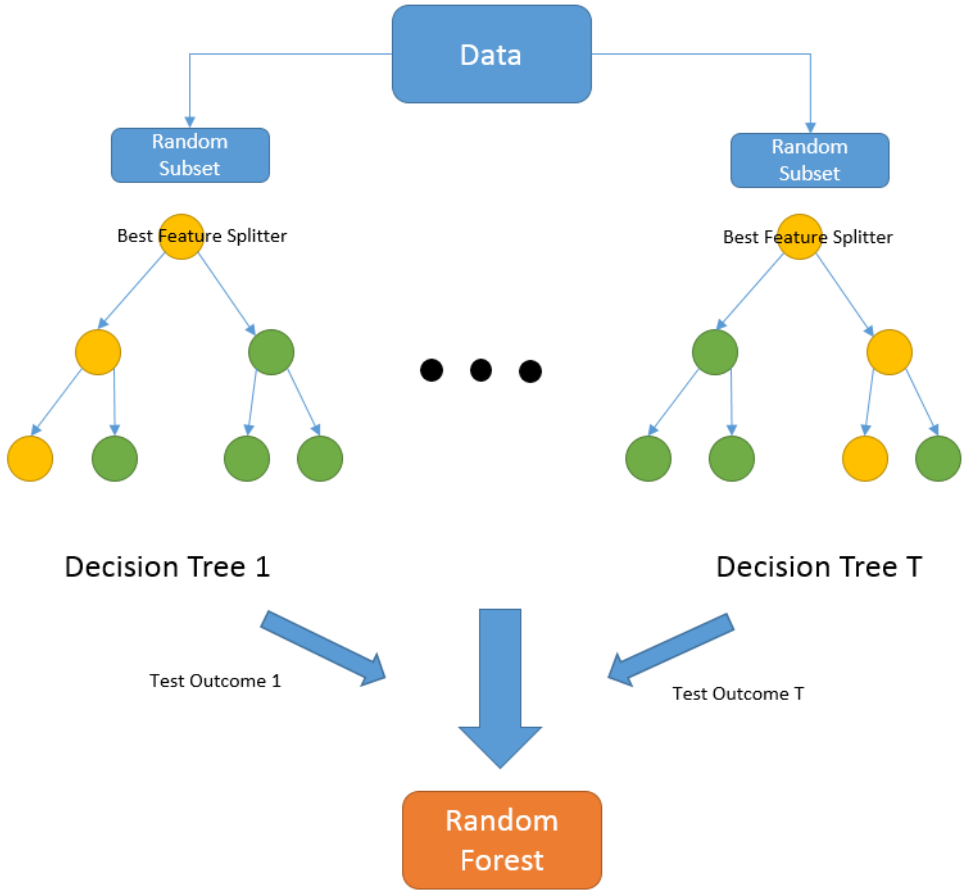

Figure 2.1: Random forest framework

2.3 Experimental Results

In this section, we compare the experimental performances with IMORL.RF and the proposed adaptive learning systems in previous works. In order to have a more valid comparison, we select two benchmark datasets from UCI machine learning repository and one traffic light dataset generated by our laboratory. These datasets are representative because their size and feature number are identical. The detailed

attributes are listed below.

Dataset	Feature Num.	Instance Num.	Class Num.	Sample Ratio
Traffic Light	784	3000	2	2 : 1
Spambase	57	4601	2	1.54 : 1
Magic	10	19020	2	1.84 : 1

Table 2.1: Experiment datasets for incremental learning

In the experiments, we randomly order the observations in a dataset first and split one half of the instances as the test data. The rest of data is cut into 20 chunks to perform incremental learning procedure. The final accuracy is calculated by the average accuracy of 20 runs on each dataset. For the first experiment, we compare the performance between the simulation accuracy of IMORL and IMORL.RF.

Dataset	Method	Average Accuracy	p value	H_0
Traffic Light	IMORL	0.9371	5.8660e-11	Reject
	IMORL.RF	**0.9584**		
Spambase	IMORL	0.9052	2.9813e-10	Reject
	IMORL.RF	**0.9244**		
Magic	IMORL	0.8292	1.7177e-10	Reject
	IMORL.RF	**0.8533**		

Table 2.2: T-test and prediction accuracy

As shown in Table 2.2, the performance of IMORL.RF is better than IMORL. In order to have a more comprehensive comparison of the experimental results generated by these two methods, the significance test is performed to compare the accuracy of each 20 runs experiment on the same dataset.

Hotelling's T-square statistic test (t-test) is a statistical hypothesis test to measure if the null hypothesis can be accepted. The test statistical data should follow a normal distribution. We assume that our results for accuracy is under normal distribution and the standard deviation σ is the same in order to perform the t-test. The hypothesis we want to verify is that:

H_0: the pairwise difference between two methods' accuracy has $\mu = 0$.

H_1: the pairwise difference between two methods' accuracy has $\mu > 0$.

The data vector is the IMORL.RF's accuracy vector minus IMORLs accuracy vector. From the p value and the result of hypothesis test, it indicates that t-test rejects the null hypothesis significantly at confidence level of 0.05. In other words, the hypothesis that the prediction accuracy from IMORL.RF is better than IMORL's is tested to be accepted.

Dataset	Method	Average Accuracy
Spambase	IMORL.RF	**0.9244**
	ADAIN.MLP	0.9142 [5]
	ADAIN.SVR	0.9120 [5]
Magic	IMORL.RF	0.8533
	ADAIN.MLP	0.8549 [5]
	ADAIN.SVR	**0.8644** [5]

Table 2.3: Prediction accuracy comparison towards ADAIN series

Table 2.3 shows the comparison of prediction accuracy with IMORL.RF to ADAIN.MLP and ADAIN.SVR. We can see that IMORL.RF's prediction accuracy is really close to ADAIN.MLP and ADAIN.SVR's prediction accuracy. For spambase dataset, the performance of IMORL.RF reaches the best accuracy. And for magic dataset, IMORL.RF also reaches a similar average accuracy to ADAIN. [5] mentioned the reason that ADAIN has a better performance than IMORL is that ADAIN has a stronger initial probability distribution formulation (support vector regression and multilayer perception). Therefore, the usage of previous knowledge from data is more comprehensive than IMORL. From our experiment, we present that a superior base learner can also be used to improve the performance of IMORL. We implement random forest instead of CART as the base hypothesis classifier and the prediction accuracy towards bench mark datasets is increased significantly.

Let's take a look at the prediction accuracy curve through the number of data chunks involved in training procedure. In Figure 2.2, the x label means the number of data chunks that have used to generate the group of hypotheses in time sequence. The prediction accuracy is calculated by the ensemble hypotheses of n available trained hypotheses. From Figure 2.2, we can say that IMORL.RF's prediction accuracy is higher than IMORL's when receiving the same training data chunks.

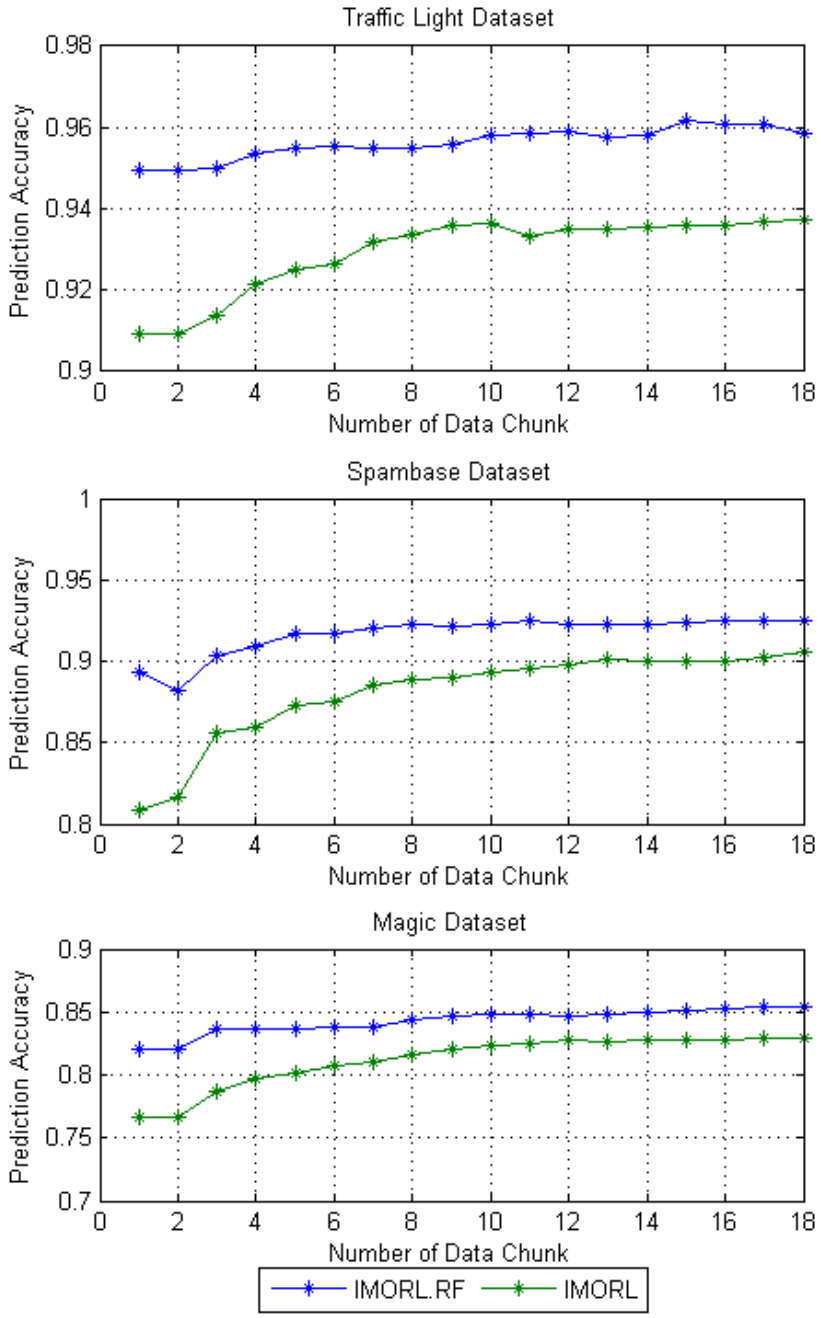

Figure 2.2: Prediction accuracy over number of data chunk

Chapter 3

Large Dataset Training Problem in a Traffic Light Recognition System

In this chapter, we first introduce the traffic light detection system proposed by our group. The detection system is a deep learning network based automatic recognition and classification system for traffic lights in video sequences.

It is the fact that image objective recognition problem is often hard to solve because data feature is in high dimension. A good detection system should be adaptive to detect traffic lights in high quality video streams, such as 1920×1080 resolution video stream. Moreover, it should detect small traffic light objects in the frame. Therefore, in order to design a good recognition system with high detection accuracy, not only the algorithms should be more complicated, our system also needs a lot of training data with high dimension features. Hence, we introduce PEGASOS, an online style SVM to solve the large dataset training problem in our traffic light detection system.

We also show some experimental results to demonstrate the efficiency of Pegasos for training the traffic light data. In addition, the comparison between Pegasos and

other online learning algorithms will be provided.

3.1 Traffic Light Recognition System

3.1.1 System Overview

Traffic lights recognition is one of the topics in object recognition for intelligent transportation system. The reason of studying automatic red traffic light detection system is that it can be a useful assistance in case of driver's carelessness at some time. The main task of object recognition is finding and identifying objects in single image or video sequence. People can see and know a lot of objects in a very short time. The object might be scaled and rotated but human can recognize it quickly. However, those changes bring enormous challenges for computer vision systems. For example, as for red traffic light detection in daytime, the difficulty is that they often have intricate backgrounds because of tree shadows and surrounding buildings. Another difficulty is that they can be often confused with red tail lights of a car.

Many approaches have been implemented over the years in order to enhance traffic light recognition system. A lot of researches have been conducted based on the knowledge of image processing. For example, [43] introduced a system with a color pre-processing module to enhance the difference of red and green regions to recognize traffic light candidate. However, using the maximal/minimal localization is not a good strategy because it is lack of flexibility. [44] detected traffic light based on interest regions extraction. But the limit is that there are too many false positives if the thresholds are not accurate. [45] applied HSV color space information and then used Adaboosting for classification.

The common shortcoming is that those methods only use low level image features like color, shape and gradient and therefore the total detection rate is relatively low.

In visual recognition, deep learning network is a useful tool for image classification and recognition because it generates high dimension features abstracted from data. It can provide better feature for objects classification and recognition. Therefore, we implement a deep learning network model: PCA network (PCANet) [46], to solve traffic light problem.

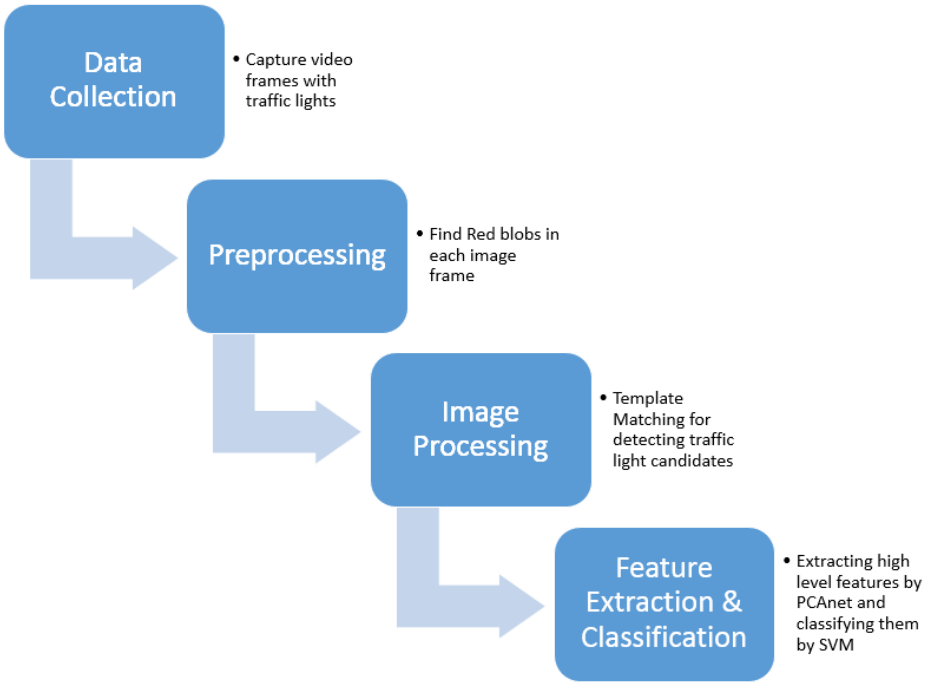

Figure 3.1: Traffic lights recognition system workflow

3.1.2 Traffic Light Preprocessing: ROI Extraction

The red circle traffic lights in daytime are our recognition objects. We concentrate on the traffic light with three lights in one column. It should have a red circle light on the top, a yellow light in the middle and a green light at the bottom. First of

all, we use color information to preprocess the images. We convert images RGB value to HSV (Hue, Saturation, Value) color space and extract red color according to the hue value thresholds. Then, an aspect ratio check is performed, which keeps all red blobs passing the check as candidates of red lights. The threshold of aspect ratio check is relatively loose because under different illumination conditions, the hue values of red traffic lights change somehow. By this preprocessing process, we obtain the candidates of traffic lights as many as we can.

Next, we follow the detection method proposed in [44] as the second step to extract region of interest (ROI) extraction. This method assumes that ROI is centered at one of the lamps of a traffic light. Also, it assumes all traffic lights consist of three lamps, red, yellow and green, with red light is on the top and at only one of them switched on at a certain time instant. This method hasn't consider other configurations beyond our dataset, but it has the adaptiveness of detecting other types of traffic lights by changing some template regions mentioned below.

By finding red blobs in each frame, we can get the locations of potential red light candidates. Then, we locate the off yellow and off green light areas beneath the red blobs. The two off lights are similar with each other but have obviously different color with their surrounding background around them. Therefore, we use the off yellow light area as ROI_{ref} to perform the template matching procedure. The corresponding off green light area is ROI_g. The result of template matching with ROI_{ref} and ROI_g is named as R_{in}. The corresponding ROI surrounding background is called ROI_{out} and the result of template matching with ROI_{ref} and ROI_{out} is recorded as R_{out}. Then, a threshold p is set as R_{in}/R_{out}. For each potential red light candidate, if $R_{in}/R_{out} < P$, we say it is still a traffic light candidate.

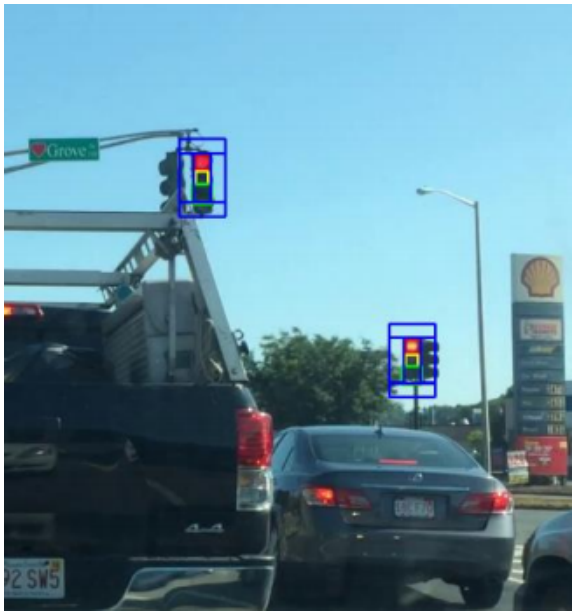

Figure 3.2: Traffic light image processing by ROI template matching

Figure 3.2 shows the template matching areas. The yellow rectangular area is represent of ROI_{ref}. The green rectangular area is ROI_g. The blue rectangular area is ROI_{out}.

The advantage of this method is that it is efficient for detecting traffic lights in any size. However, the threshold is such a weak classifier that there are still a great amount of false positive candidates. One can imagine that some false positives are really like traffic light in dark background, for example, rear red lights of cars. Therefore, we introduce deep learning network as the classifier for traffic light recognition in next step. By training deep learning network using detected candidates, we can extract higher level image features of traffic light. They are more reliable than hand-crafted features such as HOG or SIFT.

3.2 Classification Using PCAnet

We apply a two-stage cascaded classification system based on PCANet. [46] hopes to find a simpler training process which also can adapt to process different tasks and data types. The proposed PCANet comprises only basic data processing components: cascaded principal component analysis (PCA), binary hashing, and block-wise histograms. Specifically, the first two stage is PCA, which extracts image features. The output stage is hashing and histogram, producing the nonlinear output as the inputs features of Linear Support Vector Machine which trains a model for predicting labels.

3.2.1 Feature Extraction

PCANet training is divided into three stages. The first two stages are very similar: mean-removal, taking principle component analysis and convolution. The last step is binarization for producing a nonlinear output and quantification by histogram. We will describe PCANets training process step by step in the following paragraphs.

We conduct experiments for two PCA stages. First, we determine the window size as $k_1 \times k_2$ (k_1, k_2 are often set as 3,5,7 in pixel size), then slide the filter window to extract local features. After sliding $m \times n$ times, one image generates $m \times n$ patches and each patch size is $k_1 \times k_2$. Then we subtract the mean from each patch on these $m \times n$ patches and obtain mean-removed patches. Finally, single image feature extraction operation is completed. All the images should be applied the same operations. For T images, we get a new matrix X and each column represents a patch, including $k_1 \times k_2$ elements and the total is $T \times m \times n$ columns.

For the first stage, PCANet uses principle component analysis and takes the first L_1 feature vectors as filters. Each column is rearranged by these L_1 feature vectors

(each column contains $k_1 \times k_2$ elements) as a patch. Equivalently, we get L1 windows with the size of $k_1 \times k_2$. Then, the L_1^{th} filter output of the first stage applies 2D convolution with each images. The boundary of each images is zero-padded before convolving with the filter in order to make each new feature map having the same size of original images.

The second stage is similar to the first stage. At the first stage, the input images are the original image set. However, the second stage's inputs are the results of the first stage, which are the $L_1 \times T$ feature maps. In the second stage, we extract L_2 feature vectors, so the result of the second stage should have $L_1 \times L_2 \times T$ feature maps.

The output layer process is like the function below:

$$T_i^l = \sum_{l=1}^{L_2} 2^{l-1} H(I_i^l \times W_l^2) \tag{3.1}$$

Each image convolves with L_2 filters respectively, multiplies by a weight matrix and is added together. $H(\star)$ is a Heaviside step function whose value is one for positive entries and zero for otherwise. Ascending weights are corresponding to the filter in the order of small to large. We partition the matrix into B blocks, so the block matrix size is $k_1 \times k_2 \times B$. Then, the histogram for each blocks is computed and the range of histogram is $[0, 2^{L2-1}]$. And then, we concatenate all the B histograms into one vector which is denoted as $B_{hisht}(T_i^l)$.

After this encoding process, the feature of each input image is then defined to be a set of block-wise histograms as:

$$f_i = [B_{hisht}(T_i^l), \ldots, Bhist(T_i^{L_1})]^T \in R^{(2^{L2})L_1 B} \tag{3.2}$$

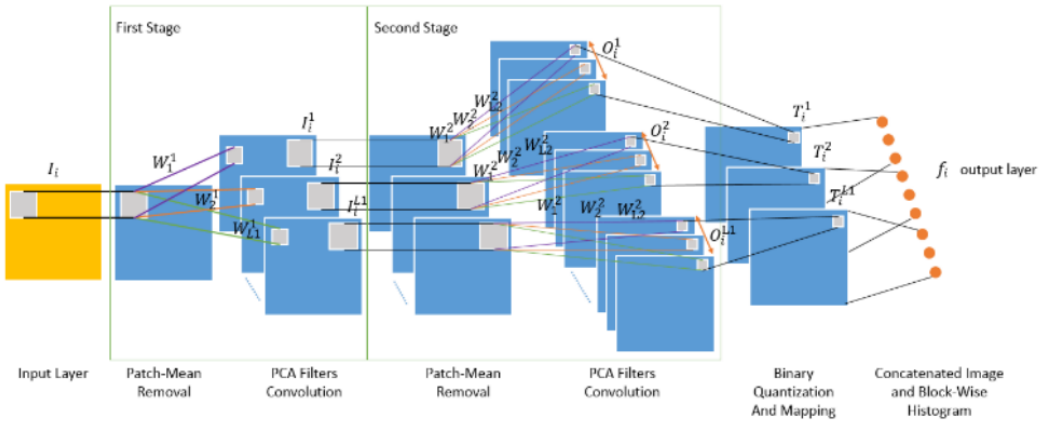

Figure 3.3: PCAnet structure

3.2.2 SVM classification

In the proposed approach, the final of PCAnet concatenates to a linear SVM for classifying PCAnet features. Due to the high volume data candidates for training, we want to find a way to train the data but don't necessarily store all the data in memory for training. Hence, we consider online learning structure and we propose to use Pegasos [19], a primal estimated sub-gradient solver for SVM to solve the training data size problem. Pegasos is especially suitable to train SVM for large scale dataset and has the similar speed and simplicity with online learning method. Furthermore, it can reach to the accuracy of state-of-art SVM solvers.

3.3 Pegasos

3.3.1 Introduction

For the primal formulation of SVM, assume a training dataset:

$$S = (x_i, y_i)_{i=1}^{m} \tag{3.3}$$

where $x_i \in \mathcal{R}^n$ and $y_i \in \{+1, -1\}$.

The soft margin with $L2$ regularization form is:

$$\min_{\mathbf{w}} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{m} \sum_{(\mathbf{x},y)\in S} \ell(\mathbf{w}; (\mathbf{x}, y)) \tag{3.4}$$

in this form, the bias b for standard SVM has been omitted. For here

$$\ell(\mathbf{w}; (\mathbf{x}, y)) = \max\{0, 1 - y \langle \mathbf{w}, \mathbf{x} \rangle\} \tag{3.5}$$

$\langle \mathbf{w}, \mathbf{x} \rangle$ is the inner product of $\mathbf{w}$ and $\mathbf{x}$. $\ell(*)$ is called hinge loss in SVM.

For the objective function, an ϵ level accurate solution $\hat{\mathbf{w}}$ can be difined as:

$$f(\hat{\mathbf{w}}) \leq \min_{\mathbf{w}} f(\mathbf{w}) + \epsilon \tag{3.6}$$

This algorithm is based on simple stochastic sub-gradient descent. It means that at each time, a training instance is sampled randomly from S. Then, the algorithm takes a step with certain length to the oppose direction of gradient. After a number of steps, it finds an ϵ level accurate solution $\hat{\mathbf{w}}$ then the iteration for randomly pick examples is stopped.

The basic mathematical formulation of Pegasos is listed as follows. For initialize,

$\mathbf{w}_1$ is randomized or set to $\mathbf{0}$. On iteration t, the random training instance is picked as (x_{i_t}, y_{i_t}). Then current objective value is $f(\mathbf{w}; i_t)$.

The sub-gradient for $f(\mathbf{w}; i_t)$ is calculated as:

$$\nabla_t = \lambda \mathbf{w}_t - \begin{cases} y_{i_t} x_{i_t}, & for y \langle \mathbf{w}, \mathbf{x} \rangle < 1 \\ 0, & for\ otherwise \end{cases} \tag{3.7}$$

when a sample's classification result is true, it yields to zero loss, otherwise $\mathbf{w}_t$ is updataed to $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \nabla_t$. Here, the step length is $\eta_t = 1/(\lambda t)$. After that, an optional step can be taken:

$$\mathbf{w}_{t+1} \leftarrow \min \left\{ 1, \frac{1/\sqrt{\lambda}}{||\mathbf{w}_{t+1}||} \right\} \mathbf{w}_{t+1} \tag{3.8}$$

where $1/\sqrt{\lambda}$ is the radius of the ball for admissible solutions.

Pegasos also can be applied to mini batch iterations, which means for iteration t, we uniformly random pick k instances for training. The training subset is $A_t \subset S$ and $|A_t| = k$. The approximate objective function changes to:

$$f(\mathbf{w}; A_t) = \frac{\lambda}{2} ||\mathbf{w}||^2 + \frac{1}{k} \sum_{i \in A_t} \ell(\mathbf{w}; (\mathbf{x}, y)) \tag{3.9}$$

Also, the sub-gradient function becomes:

$$\nabla_t = \lambda \mathbf{w}_t - \frac{1}{k} \sum_{i \in A_i} H(y_i \langle \mathbf{w}, \mathbf{x} \rangle) y_i x_i \tag{3.10}$$

where $H(y_i \langle \mathbf{w}, \mathbf{x} \rangle)$ is one when $y \langle \mathbf{w}, \mathbf{x} \rangle < 1$, otherwise it is 0. The update and the step length is still same for $\mathbf{w}_{t+1}$.

Although Pegasos can implement $k > 1$, the main focus of this algorithm is for $k = 1$. The computation process is suitable for sparsity feature vectors as

well. Besides, we also can implement kernel functions on Pegasos. This is also an attraction for Pegasos algorithm.

One should note that Pegasos shares a same feature with online learning, that is for each iteration, it can use only one example for training. Many papers said that Pegasos is in online style or online SVM. However, online learning algorithms also have significant differet with it. For example, they don't optimizing SVM's objective function directly. They have an online-to-batch conversion scheme. Therefore, the solutions generated online learning algorithms don't yield to the error level that Pegasos can guarantee.

3.3.2 Computational Performance

[19] proves that the number of iterations for getting ϵ-solution is $\widetilde{O}(1/\epsilon)$, which is shorter than previous proposed stochastic gradient descent methods (SGD) for solving SVM in $\widetilde{O}(1/\epsilon^2)$. The runtime of Pegasos is $\widetilde{O}(s/(\lambda\epsilon))$, where s denotes to the average number of non-zero features in data vector. We also want to remark that the state-of-art linear SVM solver - LibLinear[47] has the runtime of $\widetilde{O}(nd/(\lambda\epsilon))$, where n is the number of training examples and d is the number of features. LibLinear is a library for large linear classification. It has very good performance towards large scale training problem. The coverage iteration number is $O(\log 1/\epsilon)$, which is very efficient. The performance of LibLinear and Pegasos are really comparative. One may ask, except that Pegasos takes advantage of its online style, if it has better performance in time or accuracy than Liblinear? Figure 3.4 shows the comparison of SGD and LibLinear towards optimization accuracy and training time.

The above graph in Figure 3.4 shows how the test error drops when the opti-

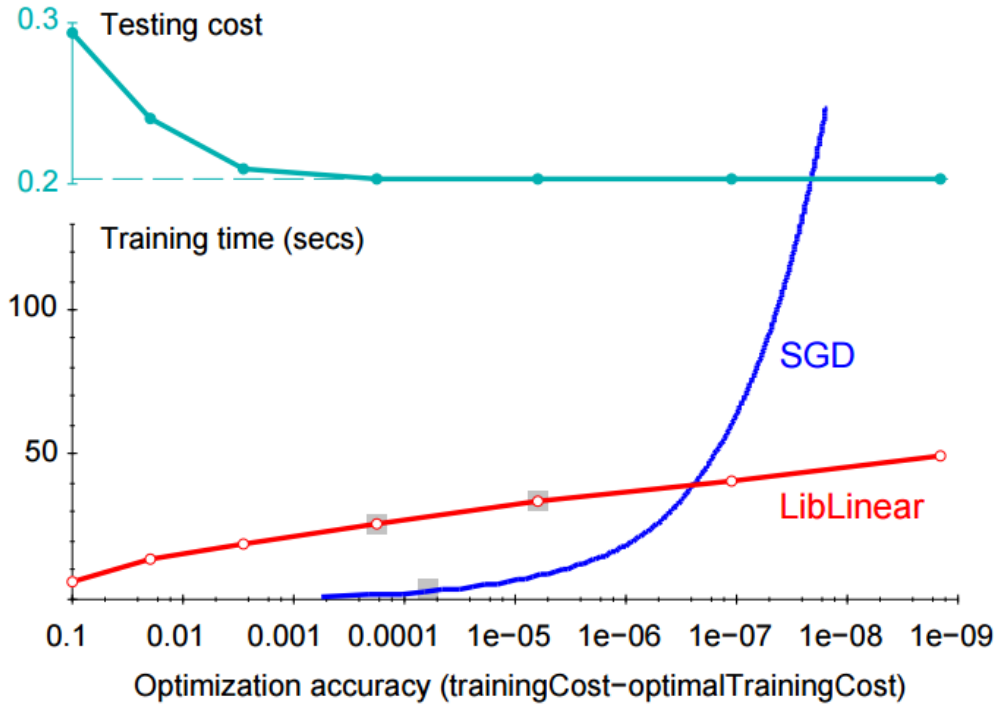

Figure 3.4: Comparison of Liblinear and SGD from [1]

mization accuracy increases. We can infer that after 0.0001, we reach the held-out accuracy. Higher optimization accuracy after 0.0001 may be unnecessary. From the bottom graph in Figure 3.4, at the efficient range of optimization accuracy, SGD based algorithms have a shorter traning time than LibLinear. If the opimization accuracy (tolerance ϵ) is set too high, SGD likely takes a long time in iterations for convergence.

3.4 Experimental Results

The attributes of our traffic light dataset with PCAnet features are shown in Table 3.1. The total number of training data is 15,000. The number of testing data is 5,296. The original distribution of our data samples are in time series. Therefore, we separate the last 20.9% data as the unseen testing data. The positive data is the red circle traffic light and the negative data is the false detected object by the preprocessing procedure and template matching processing. We can see that by the image processing method, there are still a lot of potential candidates needed to be classified. For PCAnet attributes, we set patch size as 7×7, number of filters as 8×8, histogram block size as 7×7 and block overlap ratio as 0.5.

Dataset	Train		Test		Class Ratio	Figure Size	# Feature	Sparsity
	Pos	Neg	Pos	Neg				
Traffic Light	3210	11790	1133	4163	0.27	28×28	73728	10.84%

Table 3.1: Traffic light dataset information

When using Pegasos to train the dataset, the first step is to randomly permute the order of data instances. Then, training data is separated into 15 chunks and the iteration is begin with the first data in chunk 1.

For our experiment, we perform the basic formulation of Pegasos ($k = 1$). It is because despite that [19] said the framework could be suitable for $k > 1$, their main argument were in the case of $k = 1$. Also, the training SVM doesn't have variable bias because an unregularized bias doesn't have serious impact on the performance accuracy.

Table 3.2 shows the evaluation test results using Pegasos for training. Precision

and recall are calculated by the following equations:

$$precision = \frac{true\ positives}{true\ positves + false\ positives} \tag{3.11}$$

$$recall = \frac{true\ positives}{true\ positves + false\ negatives} \tag{3.12}$$

Dataset	λ	ϵ	Num of iterations	Precision	Recall
Traffic Light	0.0001	0.01	3799	99.82%	99.1%

Table 3.2: Evaluation results of Pegasos

Our experiment is running on MATLAB R2014a. The computer has a i7-4700MQ CPU with 2.40G Hz. The installed memory is 8.0 GB. Due to the limitation of reading large scale data with MATLAB, we can not load all the data into workspace and train a SVM model with Liblinear for comparison. Therefore, the algorithm performance in time won't be compared with Liblinear's.

In Table 3.2, Pegasos only uses 3,799 instances to find the 0.01 level optimization accuracy solution. The precision and recall values are extremely high, which valid that PCAnet features are really good deep learning features for image classification problem. Pegasos helps us to train large dataset in a more efficient way. For high volume data, one does not need to evaluate how large the training set should be. While running this algorithm, one does not need to load all the data into memory. It is more feasible than dropping all the data into LibSVM or LibLinear.

3.5 Online Learning Algorithms

3.5.1 Introduction

Online learning method uses data sequentially and for every moment it updates its predictor as best hypothesis for the incoming data. PEGASOS and online learning algorithm share some common features. First of all, both of them use only a single example at each iteration. Second, they share the simplicity and computation speed. Furthermore, some of online learning algorithms can also be viewed as using SGD steps. However, one distinction should be notified is that Pegasos uses sampling with replacement strategy while online learning does not. The main reason of using Pegasos in our project is that it yeilds an ϵ-accurate solution which typical online learning methods don't necessarily guarantee.

There are two types of online learning models. The first is statistical learning model, which assumes data instances are i.i.d. The second model is adversarial model. It views learning procedure as a chess game and the objective is to minimize the losses of a player versus the other. Here, we present three online learning algorithms (Perceptron Learning, Online Passive Agressive-1[17] and Online Independent SVM[48]) for training our traffic light dataset, in order to compare their performances of accuracy with Pegasos. All those online learning methods should be categorized in statistical learning models.

3.5.2 Experimental Results

Table 3.3 shows experimental results of training and testing the traffic light dataset by the online learning algorithms mentioned above. Here, the column of PA-1 shows testing errors by Online Passive Agressive-1. The column of OISVM is representative of Online Independent SVM. We use 3,799 instances to perform online learning,

which is the same number when Pegasos meets to 0.001 tolerance with 0.0001 regularized parameter. We can find that all the online learning algorithms have good performance and OISVM gets the lowest error rate. This may demonstrates that PCANet features are easily linear separable. Also, we can demonstrate these three online learning algorithms are also suitable for training this large dataset. Figure 3.5 shows the average online errors during the training process of each algorithms. The algorithms are implemented by DOGMA MATLAB toolbox [49].

	Perceptron	PA-1	OISVM	Pegasos
Error 1 (Last Hyperplane)	0.0025	0.0025	**0.0008**	0.0023
Error 2 (Average Hyperplane)	0.0045	0.0021	-	-

Table 3.3: Evaluation results of online learning algorithms

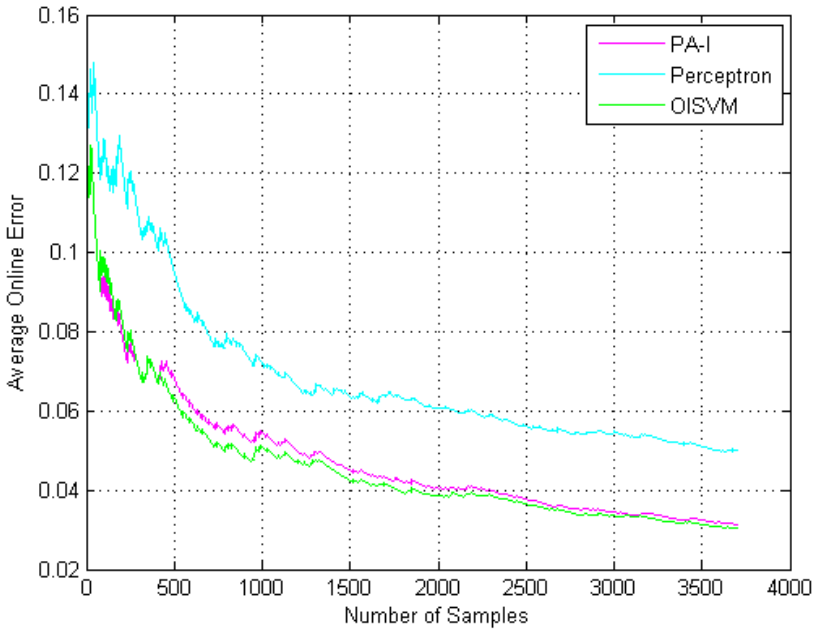

Figure 3.5: Average online error

Chapter 4

Conclusions

In this thesis, we present two projects related to binary data classification. The first project is an implementation of incremental learning technique. We propose to modify the base learner of IMORL framework in order to enhance the performance for object classification. The inspiration is to explore how the base learner can affect the classification performances. We use random forest classifier instead of CART and the results of classification have been improved. The second project is based on the traffic light recognition system using deep learning network. From this system, we obtain a lot of data for training and testing through video streams. Hence, we implement Pegasos to solve the large dataset training problem. People intuitively tend to use as mush as the amount of data they have for training, however, pegasos gets the $\epsilon = 0.01$ solution only within 4,000 data samples.

For future work, we want to attempt implementing Pegasos into data stream with warm start at each stage to see if it is suitable for incremental learning. Furthermore, multi-classification needs to be investigated beyond binary classification.

Bibliography

[1] L. Bottou, "Learning with large datasets," *NIPS Tutorial*, p. 31, 2007.

[2] M. Ade, P. GHRIET, P. Deshmukh, and A. SCOE&T, "Methods for incremental learning: A survey," *International Journal of Data Mining & Knowledge Management Process*, vol. 3, no. 4, pp. 119–125, 2013.

[3] P. Utgoff, "Incremental learning," in *Encyclopedia of Machine Learning* (C. Sammut and G. Webb, eds.), pp. 515–518, Springer US, 2010.

[4] P. Joshi and P. Kulkarni, "Incremental learning: areas and methods–a survey," *International Journal of Data Mining & Knowledge Management Process*, vol. 2, no. 5, pp. 43–51, 2012.

[5] H. He, S. Chen, K. Li, and X. Xu, "Incremental learning from stream data," *Neural Networks, IEEE Transactions on*, vol. 22, no. 12, pp. 1901–1914, 2011.

[6] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 31, no. 4, pp. 497–508, 2001.

[7] M. Muhlbaier, A. Topalis, and R. Polikar, "Learn++. mt: A new approach to incremental learning," in *Multiple Classifier Systems*, pp. 52–61, Springer, 2004.

[8] M. D. Muhlbaier, A. Topalis, and R. Polikar, "Learn. nc: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes," *Neural Networks, IEEE Transactions on*, vol. 20, no. 1, pp. 152–168, 2009.

[9] R. Polikar, J. DePasquale, H. S. Mohammed, G. Brown, and L. I. Kuncheva, "Learn++. mf: A random subspace approach for the missing feature problem," *Pattern Recognition*, vol. 43, no. 11, pp. 3817–3832, 2010.

[10] G. Ditzler, R. Polikar, and N. Chawla, "An incremental learning algorithm for non-stationary environments and class imbalance," in *Pattern Recognition (ICPR), 2010 20th International Conference on*, pp. 2997–3000, IEEE, 2010.

[11] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *Neural Networks, IEEE Transactions on*, vol. 22, no. 10, pp. 1517–1531, 2011.

[12] H. He and S. Chen, "Imorl: Incremental multiple-object recognition and localization," *Neural Networks, IEEE Transactions on*, vol. 19, no. 10, pp. 1727–1738, 2008.

[13] S. Ozawa, S. Pang, and N. Kasabov, "Incremental learning of chunk data for online pattern classification systems," *Neural Networks, IEEE Transactions on*, vol. 19, no. 6, pp. 1061–1074, 2008.

[14] R. Zhang, A. Rudnicky, *et al.*, "A new data selection principle for semi-supervised incremental learning," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 2, pp. 780–783, IEEE, 2006.

[15] F. Shen, H. Yu, Y. Kamiya, and O. Hasegawa, "An online incremental semi-supervised learning method.," *JACIII*, vol. 14, no. 6, pp. 593–605, 2010.

[16] W. Khreich, E. Granger, A. Miri, and R. Sabourin, "A survey of techniques for incremental learning of hmm parameters," *Information Sciences*, vol. 197, pp. 105–130, 2012.

[17] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *The Journal of Machine Learning Research*, vol. 7, pp. 551–585, 2006.

[18] A. Bordes, S. Ertekin, J. Weston, and L. Bottou, "Fast kernel classifiers with online and active learning," *The Journal of Machine Learning Research*, vol. 6, pp. 1579–1619, 2005.

[19] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: Primal estimated sub-gradient solver for svm," *Mathematical programming*, vol. 127, no. 1, pp. 3–30, 2011.

[20] C.-H. Tsai, C.-Y. Lin, and C.-J. Lin, "Incremental and decremental training for linear classification," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 343–352, ACM, 2014.

[21] S. L. Crawford, "Extensions to the cart algorithm," *International Journal of Man-Machine Studies*, vol. 31, no. 2, pp. 197–217, 1989.

[22] J. C. Schlimmer and D. Fisher, "A case study of incremental concept induction," in *AAAI*, pp. 496–501, 1986.

[23] P. E. Utgoff, "Id: An incremental id3," 1987.

[24] P. E. Utgoff, "Incremental induction of decision trees," *Machine learning*, vol. 4, no. 2, pp. 161–186, 1989.

[25] P. E. Utgoff, N. C. Berkman, and J. A. Clouse, "Decision tree induction based on efficient tree restructuring," *Machine Learning*, vol. 29, no. 1, pp. 5–44, 1997.

[26] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof, "On-line random forests," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pp. 1393–1400, IEEE, 2009.

[27] C.-Y. Chen, S.-C. Hwang, and Y.-J. Oyang, "An incremental hierarchical data clustering algorithm based on gravity theory," in *Advances in knowledge discovery and data mining*, pp. 237–250, Springer, 2002.

[28] X. Su, Y. Lan, R. Wan, and Y. Qin, "A fast incremental clustering algorithm," in *International Symposium on Information Processing*, pp. 175–178, Citeseer, 2009.

[29] T. Li and S. S. Anand, "Hirel: An incremental clustering algorithm for relational datasets," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pp. 887–892, IEEE, 2008.

[30] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu, "Incremental clustering for mining in a data warehousing environment," in *VLDB*, vol. 98, pp. 323–333, Citeseer, 1998.

[31] C.-C. Hsu and Y.-P. Huang, "Incremental clustering of mixed data based on distance hierarchy," *Expert Systems with Applications*, vol. 35, no. 3, pp. 1177–1185, 2008.

[32] S. Asharaf, M. N. Murty, and S. K. Shevade, "Rough set based incremental clustering of interval data," *Pattern Recognition Letters*, vol. 27, no. 6, pp. 515–519, 2006.

[33] S. Ozawa, S. L. Toh, S. Abe, S. Pang, and N. Kasabov, "Incremental learning for online face recognition," in *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, vol. 5, pp. 3174–3179, IEEE, 2005.

[34] H. Zhao and P. C. Yuen, "Incremental linear discriminant analysis for face recognition," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 38, no. 1, pp. 210–221, 2008.

[35] J. Zhang, S. Ma, and S. Sclaroff, "Meem: Robust tracking via multiple experts using entropy minimization," in *Computer Vision–ECCV 2014*, pp. 188–203, Springer, 2014.

[36] G.-Y. Chen and W.-H. Tsai, "An incremental-learning-by-navigation approach to vision-based autonomous land vehicle guidance in indoor environments using vertical line information and multiweighted generalized hough transform technique," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 28, no. 5, pp. 740–748, 1998.

[37] Z. Chen, L. Huang, and Y. L. Murphey, "Incremental learning for text document classification," in *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*, pp. 2592–2597, IEEE, 2007.

[38] E. Demidova, X. Zhou, and W. Nejdl, "A probabilistic scheme for keyword-based incremental query construction," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 24, no. 3, pp. 426–439, 2012.

[39] A. Misra, A. Sowmya, and P. Compton, "Incremental learning for segmentation in medical images," in *Biomedical Imaging: Nano to Macro, 2006. 3rd IEEE International Symposium on*, pp. 1360–1363, IEEE, 2006.

[40] Y. Freund, R. E. Schapire, *et al.*, "Experiments with a new boosting algorithm," in *ICML*, vol. 96, pp. 148–156, 1996.

[41] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[42] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[43] G. Siogkas, E. Skodras, and E. Dermatas, "Traffic lights detection in adverse conditions using color, symmetry and spatiotemporal information.," in *VISAPP (1)*, pp. 620–627, 2012.

[44] S. Salti, A. Petrelli, F. Tombari, N. Fioraio, and L. Di Stefano, "Traffic sign detection via interest region extraction," *Pattern Recognition*, vol. 48, no. 4, pp. 1039–1049, 2015.

[45] J. Gong, Y. Jiang, G. Xiong, C. Guan, G. Tao, and H. Chen, "The recognition and tracking of traffic lights based on color segmentation and camshift for intelligent vehicles," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pp. 431–435, Ieee, 2010.

[46] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "Pcanet: A simple deep learning baseline for image classification?," *arXiv preprint arXiv:1404.3606*, 2014.

[47] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[48] F. Orabona, C. Castellini, B. Caputo, L. Jie, and G. Sandini, "On-line independent support vector machines," *Pattern Recognition*, vol. 43, no. 4, pp. 1402–1412, 2010.

[49] F. Orabona, *DOGMA: a MATLAB toolbox for Online Learning*, 2009. Software available at `http://dogma.sourceforge.net`.