Evaluation of Min-Max Modular Based Methods and Parallel Processing Speedup Effect on Patent Classification Problem

Siyang Zhang
zhangsiyang-sjtu@sjtu.edu.cn

Wentao Dong
dongwentao@sjtu.edu.cn

Abstract

Patent classification is a typical large scale classification problem. Patent datasets usually have several millions of data samples, which is hard for traditional classification methods to solve efficiently. In this paper, we compared the traditional SVM classifier with the advanced Min-Max Modular Support Vector machine (M3-SVM) based on multi-thread parallel processing. We also compared the classification result of SVM and M3-SVM with traditional Multi-layer Perceptron (MLP) and advanced Min-Max Modular Multi-layer Perceptron (M3-MLP) based on CPU and GPU parallel processing speedup. The result shows that, with the help of prior information, parallel M3-SVM can achieve both better classification performance and better efficiency when dealing with large scale linearly separable problems. Besides, by making use of the GPU processing, low efficiency MLP can still get much better performance, and prior information based M3-MLP can achieve even better classification result than M3-SVM.

1. Introduction

According to the WIPO statistics, in a single year of 2015, there are about 2.9 million patent filings in the world. When accumulated by years, the number of patents is already to large to be efficiently classified by traditional classifiers, even the most efficient Support Vector Machine [10]. Other classifiers such as Multi-layer Perceptron can be much worse because of its high model complexity.

To solve this problem, several new approaches have been raised. For example, the Min-Max Modular Support Vector Machine [11] based on the traditional SVM improved the time efficiency by separating the large scale patent classification problem into several smaller independent sub-problems, which can be processed in parallel by multi-thread processing or cluster processing. After all the sub-problems have been solved individually, the final result can be derived according to the Min-Max rule. This Min-Max theory can be applied on any traditional classifier, for example SVM and MLP. Typically when handling linearly separable problems, MLP is less efficient than SVM, since the cost of computation of MLP is much larger than SVM. However, MLP can better handle more complex non-linear problems. It has better extendibility compared with linear SVM and kernel methods. Besides, with the help of faster GPU computation, nowadays training a MLP is much faster than before, making it and other neural networks more and more useful in practical use.

In our experiment, we compared different classification algorithms and the acceleration effect of multi-thread processing and GPU processing. In Section 2 we will introduce three different solutions to the Japanese patent classification problem: the traditional SVM, Min-Max Modular SVM with random sub-problem separation (M3-SVM-Rand), and Min-Max Modular SVM with prior information based sub-problem separation (M3-SVM-YC). We will compare the classification result of the three methods, and the acceleration effect of multi-thread processing on the two M3-SVM methods. The experiment of this part is based on LIBLINEAR [4] and OpenMP [2] . In Section 3, we will compare the traditional MLP and the Min-Max Modular MLP with both random (M3-MLP-Rand) and prior information based sub-problem separation (M3-MLP-YC) [8]. We will also compare the acceleration effect of GPU processing on the two M3-MLP methods. The experiment of the second part is based on TensorFlow [1].

2. Experiment Data

2.1. Dataset Introduction

In our experiment, we tried to solve a sub-task of the large scale Japanese patent classification problem to theoretically verify the efficiency of different algorithms. We use a subset of the whole Japanese patent data set, which includes the first four sections of the total eight sections. The number of data samples in each section is shown in Table 1. There are 113128 training data samples and 37786 testing data samples in total. Our task is to classify the first section (section

Section	A	B	C	D	Total
Size	27876	59597	23072	2583	113128

Table 1. Data sample numbers in each of the four sections of Japanese Patent Dataset

A) with the rest three sections (section B, C, and D). This design considered the fact that in reality most classification problems are unbalanced, which can increase the classification difficulty.

2.2. Data Pre-processing

Before data pre-processing, both the training data and the testing data are fully shuffled with Knuth-Durstenfeld Algorithm [3]. Each data sample contains 5000 features and several hierarchical labels which identify each data sample which section, class, subclass and group it belongs to. For example, a patent label can be expressed like this: A01C2100. This label means that this data belongs to section A, class A01, subclass A01C, and group A01C2100. Mention that each data sample can have several different labels, but they all belong to the same section. For example, a data sample can have two labels like: A01C2100, A01G0920.

For traditional SVM and MLP, the whole training data set is used for training SVM or MLP. For random task separation based M3-SVM or M3-MLP, both positive and negative samples are randomly separated into several sub-classes which have the same size. Since the data samples have already be fully shuffled, we just separate the whole data set in order. For prior information based M3-SVM or M3-MLP, positive and negative samples are first classified by their class, for example, A01. Since the gap of number of samples between different classes is so large, small classes are merged together to make bigger classes and large classes are divided into smaller sub-classes. In our experiment, the size of both random based sub-classes and prior information based sub-classes is 5000. This subclass size is taken considering the computing efficiency and the processing ability of our devices. Finally we get 6 positive sub-classes and 17 negative sub-classes. This design makes sure that Min-Max Modular is fully achieved while not increasing the problem complexity to much.

3. SVM Experiment Design

In this section, we will explain the experiment design of the Japanese patent classification problem based on SVM and Min-Max Modular SVM. All the algorithms are implemented based on the LIBLINEAR SVM library and OpenMP multi-tread processing library. The LIBLINEAR SVM tool provides clear and easy-to-use API and is implemented with high efficiency. Besides, LIBLINEAR also supports distributed computation based on Hadoop and Spark, which is also part of our experiment.

3.1. Traditional SVM

The traditional SVM algorithm is quite straight forward. The data pre-processing program will shuffle the original training and testing data, and replace all the labels of a sample data with a single label 0 or 1. In this experiment, the first four sections of the Japanese patent data set is considered, which is the A to D sections. To separate section A with section B to D, all the data samples with section label A is allocated with new label 1, and all the other data samples are allocated with data sample 0. Under this circumstance, section A is the positive class, and section B, C, D together is the negative class. The processed training data is read by the SVM program to train a single SVM. After model training the Support Vectors and other model parameters are saved in files for future usage, and the testing data is then read to test the model.

3.2. Min-Max Modular SVM

The Min-Max Modular SVM method is based on the traditional SVM. Both positive and negative classes are separated into several sub classes, and a modular SVM is trained to classify each subclass of the positive classes and each sub class of the negative class. Suppose we have m positive sub classes and n negative sub classes, then there will be $m * n$ modular SVMs. After training all the modular SVM by serial processing or parallel processing, the Min-Max step is taken to derive the final result. Each testing data sample is fed into all the $m*n$ modular SVMs, and the returned $m*n$ results can form a matrix with m rows and n columns. For each row of the result matrix, if all the results are positive labels, that is, 1, we set the result derived from this row to positive, else negative. This is the Min step. After that, we get a column vector with m elements. Then the Max step is taken, which means as long as one of the elements is positive, then the final result is positive, else negative. This is the basic procedure of Min-Max Modular SVM. The only difference between random based M3-SVM and pior information based M3-SVM is the pre-processing of the training data set. The implementations of these two algorithms are the same.

3.3. Sub-class Size Selection

Mention that the size of each sub class should be decided by the scale of the problem and your processing ability. More and less sub classes can better make use of parallel processing, especially when you have enough computing resource, such as multi-core servers or clusters. However, if your devices are not powerful enough to support fully parallel processing, too many sub-classes will increase the total amount of data a lot. Basically, suppose the size of the positive class is x and the size of the negative class is y, and the size of each sub-class is s, then the number of positive sub-classes is $\frac{x}{s}$, while the number of negative sub-classes is $\frac{y}{s}$. The total number of data after modularization can be expressed by Eqn.(1).

$$TotalDataSize = 2 * s * (\frac{x}{s} * \frac{y}{s}) = \frac{2 * x * y}{s} \tag{1}$$

$$SpeedupRatio \geq \frac{2 * x * y}{s * (x + y)} \tag{2}$$

To make your algorithm more efficient, you should make sure that your speedup ratio satisfies Eqn.(2)

In our experiment on SVM and M3-SVM, the only parallel processing method we can achieve is multi-thread processing, so we implemented the multi-thread M3-SVM with OpenMP. OpenMP is widely supported by main stream compilers, and it doesnt requires much modification to the original serial processing version code. Limited by our devices, we only use four threads parallel processing in our implementation. Although the time efficiency is still lower than traditional SVM, its speed up performance is quite good compared with serial processing M3-SVM.

4. MLP Experiment Design

4.1. MLP Introduction

Multi-Layer Perceptron is a much more complex classifier than SVM. It simulates the functionality of human nervous system to solve complex non-linear problem. Theoretically three layer MLP can fit any linear or non-linear model, but the number of neurons in the hidden layer may be too large to be implemented in reality. An alternative to increasing the number of neurons in the hidden layer is to increase the number of layers. However, when the neural network becomes too deep, the error generated in the output layer can hardly be passed forward, which is called gradient attenuation. Recently, some varieties of the tradition neuron network such as Convolutional Neural Network (CNN) [6] and Deep Belief Network (DBF) [5] has solved this problem by weight sharing or pre-training. These methods have been successfully used in multiple fields and have improved the results a lot.

The two classes patent classification problem we aim to solve is approximate linearly separable, so the simplest three-layer MLP and only a few hidden layer neurons is required. In our implementation, we use a 64 neurons hidden layer. However, since the 5000 dimension data feature is too high, the model complexity is still much higher than SVM, especially when using the Min-Max Modularization, in which each sub-task is a MLP. So parallel processing speedup is more significant in the implementation of M3-MLP.

4.2. Traditional MLP

In this section, all the algorithms are implemented with TensorFlow. TensorFlow is one of the most popular deep learning toolboxes at present. You can design your own neural network according to your demand. To implement the traditional MLP, we designed a three-layer neural network, in which the input layer has 5001 neurons corresponding to the 5000 dimension features and a bias neuron, and the hidden layer includes 64 neurons with a bias neuron, and the output layer has two neurons corresponding to the two classes. We use the traditional sigmoid function as the activation function, L2 regularized loss function, and use Gradient Descent Optimizer to find the optimum result. These functions are already been implemented in the TensorFlow models and are ready to use. We use batch learning to train the MLP, each time we randomly pick 100 data samples to train the model. Generally after 10000 iterations, the model is fully trained. All the testing results we will discuss about later are gathered after 30000 iterations.

4.3. Min-Max Modular MLP

As for Min-Max Modular MLP, the problem separation is the same as the Min-Max Modular SVM. The only difference is that each model is a three layer MLP rather than a SVM. Since TensorFlow has its own implementation of parallel computing, to fully make use the computing resource, each time a batch of models are trained together. Actually if the device has enough memory, all the models can be trained at the same time. However, limited by out devices, we can only batch training 6 models each time. Another advantage of using tendorflow to implement the algorithms is that the code can run with GPU with out modification, so we also compared the same code running wit CPU and GPU to find out the GPU accelerate effect on this classification task.

5. Parallel Processing and Distributed Computation

When handling massive data processing problems, there are always two bottlenecks that need to be taken into consideration. One of the bottlenecks is that when there is a massive of data, the speed of serial program execution is too slow for practical usage. The other is that the amount of data can be too large to be fit into the memory of a single computer or server. To solve these two problems, there are basically three approaches. The first is Muiti-core stand-alone CPU parallel computing which can improve the execution efficiency of the program. The second is GPU based parallel computing, since GPU is specially designed for massive processing. The third is distributed computing. It can

Classifier	TP	TN	FP	FN	TPR	FPR	Precision	Recall	F1
SVM	0.7452	0.2198	0.0224	0.0127	0.9833	0.0925	**0.9708**	**0.9833**	**0.9770**
M3-SVM-Rand	0.2174	0.7428	0.0150	0.0248	0.8976	0.0198	0.9353	0.8976	0.9161
M3-SVM-YC	0.2228	0.7428	0.0150	0.0193	0.9201	0.0198	0.9368	0.9201	0.9284
MLP	0.2139	0.7458	0.0120	0.0282	0.8835	0.0159	0.9467	0.8835	0.9140
M3-MLP-Rand	0.2170	0.7446	0.0133	0.0252	0.8960	0.0175	0.9423	0.8960	0.9185
M3-MLP-YC	0.2235	0.7423	0.0155	0.0186	0.9231	0.0205	0.9350	0.9231	0.9290

Table 2. Testing result of all the six classifiers

provide heterogeneous multi-processor environment which will accommodate high performance while providing high concurrency. For the three approaches discussed above, we implemented each of them with a classification algorithm. Specifically, we implemented the M3-SVM with Muiti-core stand-alone cpu parallel computing and distributed computing, and we implemented the M3-SVM with GPU based parallel computing. All the three approaches requires that there is no coupling between every two tasks. The experiment design will be described below.

5.1. CPU Parallel Computing

The patent classification problem itself is a typical parallelism problem. In the Min-Max Modular SVM algorithm, there is no data coupling between each base classifier. A simple idea is to assign each classifier to a single thread for training and testing by using the common parallel tools such as Pthread, OpenMPI and OpenMP. Both Pthread and OpenMPI are provided in the form of library functions. Pthread only exists in the unix-like systems, while the OpenMPI source code should be compiled before using. However, GNU compilers natively supports OpenMP as one of the compiler options. OpenMP is a memory shared, compiler oriented tool. It can also explore more data-level parallelism in addition to the task-level parallelism. Besides, it is super easy to modify the original M3-SVM code to adapt to OpenMP. The only thing we need to do is to add some declaration lines and keep the variables for each section private. In this experiment we use G++ 5.4.0 which supports OpenMP 3.0 to implement the CPU parallel computing.

To compile with OpenMP, pre-compiling instructions should be added before the part you want to make it run in parallel. When compiling, the compiler will automatically run the corresponding codes in parallel according to your wish. Since our device supports at most four threads, we use three threads parallelism and divide all the models into three groups so that they can run in three separated sections. In each section all the variables are decalred to be private, so that the compiler will make a copy of the variable for each section to prevent coupling between threads. The speedup result will be shown in Section 7.

5.2. GPU Parallel Computing

One of the advantages of developing with TensorFlow is that the code can run both with and without GPU. Although there is no need to do special modification to the original code, there is still some tricks to fully make use of the parallel processing ability. The execution of TensorFlow is based on session. When running a session with a variable, all the computation will be processed to derive the value of the variable. However, you can run session on several variables at one time, and the variables will be computed at the same time. As long as the memory size allows, you can run all the variables at the same time. In our implementation, we run six MLPs at one time while training, and run all the MLPs together while testing, since during the training procedure each time there is only one data sample fed into the MLP. The speedup comparation between CPU processing and GPU processing of the M3-MLP code will be shown in Section 7.

5.3. Distributed Computation

One of the problems of large scale machine learning is that the processing ability of a single device is always limited. While dealing with massive data processing problems, sometimes you can never run the whole task on a single computer or server. One of the solutions is to distribute the task to many computing nodes in a distributed system. We designed a simple experiment to verify the possibility to solve the patent classification problem in a distributed way.

The developers from NTU have already provided a distributed version of LIBLINEAR on its main page, which is called Spark LIBLINEAR [7]. Spark [12] cluster is a distributed computing framework based on memory, which is more efficient than the disk based Hadoop [9]. The basic principle is: the master Hadoop node on the local computer can transfer the data set into HDFS readable file blocks and put them under a certain path specified by HDFS. After that, all the Hadoop nodes and the Spark nodes can get access to the data. Since there is no coupling between tasks, the mape reduce function of Spark can be fully made use of so that

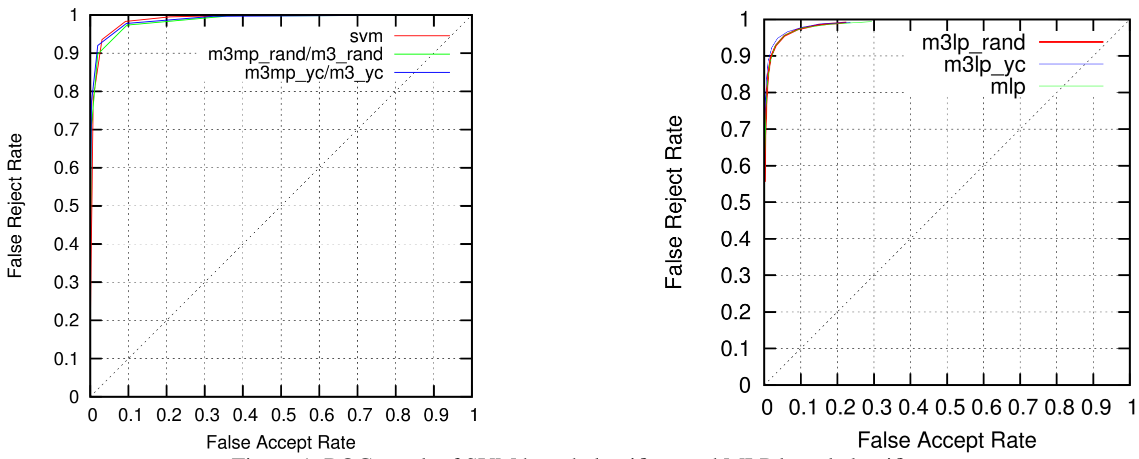

Figure 1. ROC graph of SVM based classifiers and MLP based classifiers

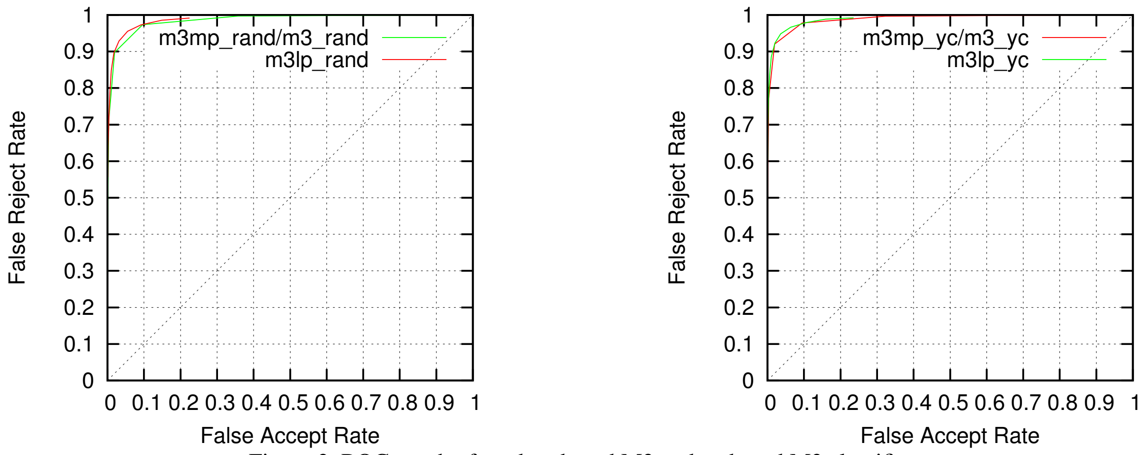

Figure 2. ROC graph of random based M3 and yc based M3 classifiers

data blocks can be mapped to different nodes for different models' training.

Mention that this experiment is based on the HDFS distributed file system, the advantage of which is its high throughput. However, it can also cause high delay. When tested on a local computer, the speedup ratio achieved by using the API provided by Spark LIBLINEAR and Map Reduce provided by Spark can hardly reach the theoretical value of cluster computing. This can be explained by the fact that in this two-class patent classification task, the bottleneck is not the computing but the IO. When the data set is too large for a single computer to handle, distribued computing can achieve better performance with less overhead.

6. Min-Max Threshod Selection

According to the Min-Max Modular theory, during the Min procedure, only when all the results are positive can the derived resault be positive. And during the Max procedure, as long as one of the results is positive, then derived result can be positive. In our implementation, instead of being absolutely strict, we tried to grid search the best Min-Max decision threshod. Both the result of SVM based Min-Max and the MLP based Min-Max showed that, when the sub-classes are generated according to prior information, the best Min-Max decision threshod is the same as the theory. However, when it comes to the random based Min-Max, the optimum decision threshod is not the most strict one. Instead, in our experiment, when there are 6 positive sub-classes and 17 negative sub-calsses, and we take Min on each negative subclass and all the positive classes, the Min threshod is 15 and the Max threshod is 2. It means that when taking the Min procedure, when more than 15 results are positive, the derived result should be positive, and when taking the Max procedure, only when there are more than 2 results are positive can the derived result be positive.

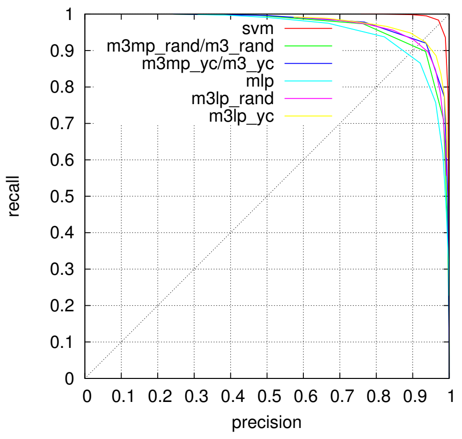

Figure 3. PR graph of all the six classifiers

Because of the high complexity of MLP and the huge testing data set, if we use the whole testing dataset to do optimum threshod grid searching, it will take several hours to finish the task. Since the threshod values are not related to the number of testing dataset, instead of using all the 37786 testing data samples, we take the first 1000 samples to search the best threshod. After the threshods are settled, we use the whole testing data set to evaluate the performance of the classifiers so that we can have more generalized observation result.

7. Evaluation

In this section, we will first evaluate the classification accuracy of all the six classifiers: SVM, M3-SVM-Rand, M3-SVM-YC, MLP, M3-MLP-Rand and M3-MLP-YC. Then we will compare the accelerate effect of multi-thread processing on M3-SVM, and the accelerate effect of GPU processing on M3-MLP.

7.1. Classification Accuracy Analysis

The testing results of all the six classifiers are listed in Table 2. Mention that for all the MLP based algorithms, we took 30,000 iterations to train the models. From the results we can see that traditional SVM has the highest score in all the three evaluation indexes. This can be explained by the fact that the two-class patent classification problem we used for evaluation is linearly separable, so the simplest linear SVM can have pretty good result. M3-MLP-YC has the second highest score in F1 measurement, which suggests the potential of MLP in classification problem. The third place of F1 measurement goes to M3-SVM-YC. For both SVM based Min-Max Modular method and MLP based Min-Max Modular method, the prior information based version gets better result than the random version. F1 value means we can get the best balance between recall and precision. If we prefer better precision or recall, we can also use F-beta for evaluation.

To make the comparation more intuitive, we can refer to the ROC graph of the six classifiers. In each of the four graphs, all the curves are all so close to each other, which means that these classifiers have so similiar performances. However, if we have a closer look, we can see that in Figure 1, the traditional SVM has the best performance, and prior information based M3-SVM is better than random based M3-SVM. As for the MLP based classifiers, the prior information based M3-MLP has slightly better performance than the others. Considering all the Min-Max Modular methods, MLP based M3 classifiers generally have better performance than SVM based M3 classifiers.

Classifier	Before	After	Speedup Ratio
M3-SVM	68.92	32.70	2.11
M3-MLP	1261	486	2.59

Table 3. Speedup result

The evaluation of the PR graph is almost the same as the ROC graph. Closer to the top right corner means better performance. The observation result is the same as the reault of ROC. However, except for the traditional SVM curve, all the other curves are a little bit worse than the PR graph. This is because the PR graph is sensitive to the unbalanced classification problem. Since in this patent classification problem, the negative class is about three times the size of the positive class, this problem is slightly unbalanced. As the result, the PR curves are less perfect than the ROC curves.

7.2. Confusion Matrix Analysis

Since we have enought testing result data, we also calculated the confusion matrix of all the classifiers. From each confusion matrix, 6 classifier measurement data can be derived. ACC is the percentage of correctly predicted positive and negative labels. The sum of ACC and ERR is 1. PPV is the percentage of correctly predicted positive labels, and NPV is the percentage of correctly predicting the negative labels. TPR is actually recall, and SPC is the percentage of correctly prediced negative labels of all the negative testing data. All the data can be used for model selection.

7.3. Speedup Evaluation.

We also evaluated the speedup effect of multi-thread processing compared with single thread processing, and the speedup effect of GPU computing compared with CPU computing. We tested the running time of single thread M3-SVM and four-thread parallel M3-SVM, and the training time of M3-MLP on CPU and GPU. From Table 3 we can

Classifier	ACC	ERR	PPV	NPV	TPR	SPC
SVM	0.9650	0.0350	**0.9456**	0.9708	0.9075	**0.9833**
M3-SVM-Rand	0.9602	0.0389	0.9353	**0.9677**	**0.8976**	0.9801
M3-SVM-YC	0.9656	0.0344	0.9368	0.9746	0.9201	0.9802
MLP	0.9599	0.0401	**0.9091**	**0.9766**	**0.9271**	**0.9704**
M3-MLP-Rand	**0.9578**	**0.0422**	0.9423	0.9755	0.9209	0.9824
M3-MLP-YC	**0.9658**	**0.0342**	0.9350	0.9755	0.9231	0.9795

Table 4. Confusion matrix result of the six classifiers

see that both muiti-thread processing and GPU processing have achieved speedup ratio higher than 2. Since our experiments are limited by the processing ability of our devices, we can hardly achieve higher speedup ratio. However, with multi-core servers or faster GPUs with larger memory, it is possible that the speedup ratio is much higher. According to our experiments, then total training time of traditional SVM is about 1s. However, for the prior information based M3-SVM, the average training time of each thread is about 0.15s. If we have enough processing resource, the speedup effect can be obvious.

8. Conclusion

From all results and analysis of the experiments above, there are several conclusions we can safely draw. First of all, all the Min-Max Modular classifiers can be as powerful as the traditional SVM and MLP classifier, which means there is no need to worry about possible accuracy loss when using Min-Max Modular methods. Secondly, both CPU based multi-thread parallel processing and GPU based acceleration can achieve pretty good speedup result. However, multi-thread parallelism on a single computer or server can only handle light weight parallel processing problems, such as M3-SVM. Parallel processing of classifiers with super high complexity such as MLP may need more advanced and powerful parallel computing framworks, such as distributed computation and GPU computation. Finally, although MLP is much more comples than SVM and is quite hard to train, when dealing with complex problems, fully trained MLP can have much better performance than linear classifier SVM.

9. Future Work

This experiment compared the performance of six different classifiers on the Japanese patent classification task. We compared SVM and MLP and their Min-Max Modular varieties. In fact we also tried to implement kernel functions based on LIBLINEAR, however we were not able to finish the work in time. Since SVM with kernel functions such as polynomial kernel or RBF kernel can also have pretty good classification results on linearly non-separable problems, the performance comparision of SVM with kernel function and MLP also deserves a closer look.

In this experiment, we only use a very simple patent classification task to test the performance of all kinds of classifiers. The task is so easy that the efficiency gaps between different classifiers are not fully revealed. It is possible that when the data set becomes much larger, or when the simple two-class classification problem is replaced by a more complex multi-class classification problem, some of the methods tested in our experiments may fail to perform that well.

Besides, although we noticed the optimum Min-Max threshod difference caused by different subclass generation methods, we can hardly explain this phenomenon. More related experiments should be designed to discover the hidden relationship under this phenomenon.

The MLP design is also an interesting topic. In our implementation, we choose to use three-layer MLPs with 64 neurons hidden layer only by experience. When facing more complex classification tasks, the structure of the base classifier MLP may need some modification to adapt to new problems.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] L. Dagum and R. Menon. Openmp: an industry standard api for shared-memory programming. *IEEE computational science and engineering*, 5(1):46–55, 1998.

[3] R. Durstenfeld. Algorithm 235: random permutation. *Communications of the ACM*, 7(7):420, 1964.

[4] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[5] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In

Advances in neural information processing systems, pages 1097–1105, 2012.

[7] C.-Y. Lin, C.-H. Tsai, C.-P. Lee, and C.-J. Lin. Large-scale logistic regression and linear support vector machines using spark. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 519–528. IEEE, 2014.

[8] C. Ma, B.-L. Lu, and M. Utiyama. Incorporating prior knowledge into task decomposition for large-scale patent classification. *Advances in Neural Networks–ISNN 2009*, pages 784–793, 2009.

[9] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on*, pages 1–10. IEEE, 2010.

[10] J. A. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[11] Z.-F. Ye, B.-L. Lu, and C. Hui. Patent classification using parallel min-max modular support vector machine. In *Autonomous Systems–Self-Organization, Management, and Control*, pages 157–167. Springer Netherlands, 2008.

[12] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.