MapReduce Reduced Support Vector Machine

Wei-Chih Lai
National Taiwan
University of
Science and Technology
Email:

Po-Han Huang
National Taiwan
University of
Science and Technology
Email:

Yuh-Jye Lee
National Taiwan
University of
Science and Technology
Email:
Telephone:

Abstract—**This paper propose a parallel algorithm named MapReduce Reduced Support Vector Machine(MRRSVM) which extend Reduced Support Vector Machine(RSVM), let non-liner SVM method able to handle super large scale data. There are many algorithms trying to extend non-linear SVM method. Most of them are based on SMO, shrinking technique or working set selection. Consider there are many training data, there is no necessary to enforce all instances joining in a single kernel. In this paper, we'll show that how to use a linear SVM to combine many small RSVM models. Our algorithm has two phase. The first phase is called "Map Phase", it splits dataset into many subsets and trains lots of RSVM models for each subset. The second phase is called "Reduce Phase", it treats each RSVM model as an expert for sub-data and use a linear SVM to vote their prediction in this phase. Combine these two phases, we'll get a binary classification for our big data. Finally, we apply non-linear SVM method on big data and speed up with parallel framework and reduced kernel trick. Experimental results show this algorithm is fast and fairly sharp for handling big data.**

Keywords. RSVM, map reduce, super large scale big data

I. Introduction

Data has became larger and larger due to data can be easily collected nowadays. People have to introduce new algorithms and structures to handle big data. The Support Vector Machine already showed good results on dealing with classic classification problem. There are several algorithms([1]–[8]) are trying to apply SVM method on big data. For linear SVM, Liblinear [8] presents a well performed library combines a dual coordinate descent method and other technique to implement large-scale linear SVM. For non-linear SVM, SMO [1] introduces an algorithm for solving the quadratic programming effectually. SMO-shrink [2] and SMO-wss2 [3] are extended SMO methods with selecting a good working set to reduce the size of quadratic programming problem. Those algorithms use lots of trick to squeeze instance and train a model like classic non-linear SVM.

Most algorithms are either linear method or approximate non-linear method because it is too difficult to build a full kernel for large data. By giving a data size n, having a linear combination solution of kernel function will require n^2 memory space. Since the data size goes millions upon millions, the n^2 space requirement is unreachable. Thankfully, we have RSVM that reduce the kernel size from n^2 to nm, which m is way smaller than n. If we look inside RSVM, m is actually rn, which r is our ratio. So the kernel size of RSVM is actually rn^2. Like we just mention that the size of data grows fast, reduced kernel will finally meet the limit of memory space. So the main idea of our method is to split the super large scale data into small pieces. Noticed that even we use "small" here, the size of these pieces may be large but can be compute by RSVM.

Parallel is very popular nowadays, more and more algorithms are applied into parallel structure. Like [9], [10] show that we can make SVM parallel itself. But in this paper, we focus on the map reduce structure which can be easily parallel because of every sub-set that we split in map phase are independent from each other. Due to the parallel structure and the small size of kernel, MRRSVM has a very good performance on speed as a non-linear method.

In Section II, we'll talk about RSVM which is the basic method of our algorithm. Then in Section III, we start to introduce the whole structure of our method. And in Section IV, we are going to show performance of MRRSVM.

In summary, the key point of this method is:

1) We split data into small pieces so that each we can make a RSVM model for it. This decision also make the whole architecture parallelable.
2) Combine the non-linear RSVM models by using a linear SVM.

II. RSVM

The Reduced Support Vector Machine(RSVM) aims to generate a non-linear separating hyperplane of a large dataset by keeping a little subset and throwing remainder. For data matrix $A \in R^{m\times n}$, the standard SVM will build kernel matrix with $K(A, A') \in R^{m\times m}$. Once the dataset has a lot of instance, standard kernel matrix will makes the computer running out of memory quickly. To avoid computing difficulty, RSVM uses a small subset, $\bar{A}$, which is randomly selected from the original dataset to build kernel. After cutting the problem size, kernel matrix $K(A, \bar{A}')$ is in $R^{m\times\bar{m}}$ and $\bar{m}$ is m multiplied by reduction ratio (typically less than 0.1). Computational results indicate that this data-reduction approach can work fairly well and reduced dataset is all needed in generating the final non-linear separating hyperplane. This is very important for handling the large dataset. The standard SVM can handle because building huge full kernel matrix for large dataset is impossible. With RSVM, we can use a reduced kernel matrix to generate a separating hyperplane instead of full kernel matrix. However, RSVM only reduces column number of kernel matrix. If the dataset has extreme number of instance, then the computing difficulty will back.

III. MRRSVM

To deal with super large scale data, we now introducing our algorithm. Map Reduced Reduced Support Vector Machine(MRRSVM) is an algorithm which can be paralleled due to its split structure. In this paper, to simplify the question, we focus on binary classification.

A. Training

Fig. 1 shows the whole structure of our algorithm. The first part which is called "Map Phase". In the figure is the part that data split into little pieces and each is trained by a RSVM. And the second part is "Reduce Phase". In this phase, we put all reduce sets which picked by our RSVMs as our new row element. This element will be combined as the new feature for our last linear SVM training with function f_{ij} which we will define later. The whole structure has only one propose, making non-linear SVM possible to train a super-large data. But in this propose, we can also have another advantage which is shorter training time. With building smaller kernels rather than one giant kernel, we can have a parallelable and quicker algorithm.

1) Map Phase : Generate experts for local data: In map phase, we're trying to make the super large-scale data fit in the kernel because we don't have infinite memory in real world. While using RSVM, we've already fix some problems that big kernel bring to us. But still, kernel has its own limit. Instead of changing kernel, we split data into small pieces. So that we can make a lot of fitting kernels to get RSVM models that we want for prediction.

Original data are split into n subsets. Each data subset are going to train a model by RSVM with a reduced subset distributively. We also make parallel this part that all RSVM models can compute in the same time because every subset is independent. After having these n RSVM models independently, models will be combined and concluded in out next phase, Reduce Phase.

In this phase, we have to make sure that our small RSVM models which make from split data can as approach the SVM model made by full data as possible. First, we let the data split in the same distribution as the original one.

2) Reduce Phase : Learning from experts estimation: In reduce phase, combining the results of small RVSMs is what we have to do. For this purpose, we design a structure that use a linear SVM to train a final model which input data is the things we receive from RSVMs. We collect all the instances from those n reduced subset we created in map phase. And next, we encode each of those instances as an n-dimensional vector by its estimated values from predicting them on RSVM models. To build this new instance, we create a function f_{ij} which i means the reduced set got from each split subset and j means which RSVM model is used to get our estimated values. We now defining this function as:

$$f_{ij} = K(ReduceSet_i, model_j.RS) \times model_j.w - model_j.b$$

This function is used to estimate the label in RSVM. The term $K(ReduceSet_i, model_j.RS)$ is to compute the kernel value between reduce set i and model j. Here the $model_j.RS$ is the reduce set we selected while creating RSVM model. These reduce set are used to build kernel with other set of instance to have the kernel value between other set and the model. So it is actually a representation of full set used by the model. Notice that we remain the same data size of all reduced subsets with the n-dimensional features described by those n RSVM models. Finally, we will get the same reduce set size which can represent the whole data set. We'll use these reduce set the get our estimation with our RSVM models. Once we have these estimated values, we can apply the linear SVM to conclude the final MRRSVM model by compressing the data. The detail of our algorithm is shown in Algorithm 1.

In this definition, each RSVM can be seen as an expert for its own set and each reduced sets stand for the point of view that expert specify in. What we do next is to let every expert(RSVM) to estimate all reduced sets, letting us get a table full of estimate number. Then we use it as new input data for linear SVM. While using estimate numbers produce by RSVMs as training data, we actually let the linear SVM weighting our experts.

Because every small RSVM only learn things from one subset, it is actually a local expert. We can not make sure that a local expert will be doing well in global field. Due to these, subset choosing maybe a very important issue in this method.

Algorithm 1 MapReduce RSVM train

```
Require: The big dataset $A$.
Ensure: RSVM model $\{model_i\}_{i=1}^n$, the reduced set of each subset $\{\tilde{A}_i\}$, the linear SSVM model $\{model_{final}\}$ with new features generated by $\{model_i\}_{i=1}^n$
  1) Split $A$ into $n$ subsets and their associated reduced sets: $\{A_i\}_{i=1}^n$ and $\{\tilde{A}_i\}_{i=1}^n$
  2) Learn the RSVM $model_i$ for each subset $A_i$ with its reduced set $\tilde{A}_i$
  3) Generate a new representation $B \in \mathbb{R}^{(\Sigma\tilde{\ell}_i)\times n}$ for the reduced subsets and the $j$th row of $B$ is represented as follow:
     $\mathbf{x}_j^{new} = [model_1(\mathbf{x}_j), model_2(\mathbf{x}_j), \ldots, model_n(\mathbf{x}_j)]^T \in \mathbb{R}^n$ for $\mathbf{x}_j \in \{\tilde{A}_i\}_{i=1}^n$
  4) Learn the linear SSVM model with $B$:
     $model_{final} \leftarrow linear\_\mathrm{SVM}\_train(B)$
  5) Return $\{model_i\}_{i=1}^n$, $\{\tilde{A}_i\}_{i=1}^n$, and $model_{final}$
```

B. Predicting

Predict is also an important problem in binary classification. To efficiently predict label is the critical issue in this part. Thankfully, prediction doesn't need to change the model so that we don't have to worry about if each split subset have too few instances or the label in each split subset is unbalance.

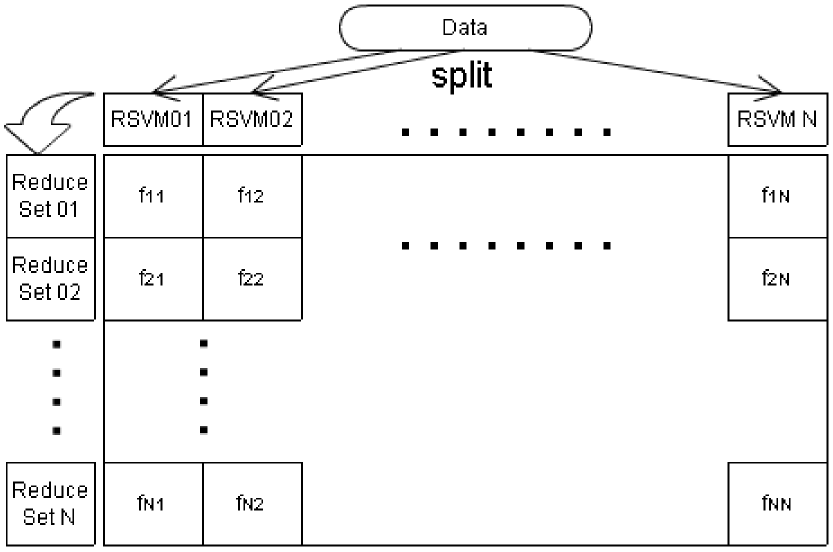

Fig. 1. MRRSVM structure

In this part, we use the same idea while we train. Map the test data into little pieces which allow us the compute paralleled and build smaller kernel to save more time. And then encode them into n-dimensional vector using n RSVM models estimated values, these RSVM models are what we learned in training part. Next we Reduce these results into a linear SVM which is the final model we learned before. The detail for prediction is shown in Algorithm 2.

```
Algorithm 2 MapReduce RSVM predict
Require: A testing instance $\mathbf{x}_t$, $\{model_i\}_{i=1}^n$, and $model_{final}$
Ensure: Predicted label
    1) Generate the new representation for the testing instance $\mathbf{x}_t$:
       $\mathbf{x}_t^{new} = [model_1(\mathbf{x}_j), model_2(\mathbf{x}_j), \ldots, model_n(\mathbf{x}_j)]^T \in \mathbb{R}^n$
    2) Predicted label $\leftarrow$ sign$(model_{final}(\mathbf{x}_t^{new}))$
```

IV. Experiments

In this section, we focus on the general of our method and the strength of our classification. Also we're interesting in the performance for each RSVM models and the difference between those RSVM models and final linear SVM model. First, we use a toy problem called checkerboard to proof that our method remain the same ability as RSVM. Instead of using the original checkerboard which size is 2,000, we create a new checkerboard size 1,000,000. Second, we're trying to approach the well turned model as possible as we can. For this propose, we implement uniform design to choose the parameters for us. Third, we design an experimental that all methods only use the default parameters while training. This experiment is to show the general of our tool. Even through there is uniform design, you can't expect every user waiting for such a long time for the well tuned model. Especially for those hard turned dataset like covertype. Since we compare our method with libsvm and RSVM, we use both default parameters. So that we get two pair of parameters and three method to test, we'll have total six results for each dataset. Notice that our method is fast and parallel, waiting time for well tuned model will effectively decrease. But making things easier doesn't mean that this thing will become cost free. Still, tuning is a hard work and don't know where to ended.

A. Experiment Environment

B. Preprocessing

The preprocessing for each dataset is the same. First, normalizing each attribute column between -1 and 1. Second, calculating the frequent of classes. Final, split dataset to several files with same size. Also, we make sure that each subset will remain the same proportion of labels. We let each training subset not more than 25,000 instances and for each testing set not more than a specified size of instances. These bounds are to going to limit our kernel size but not to lose too much information after splitting.

C. Experiment Setting

While training RSVM model in map phase, we need to choose the value of cost, gamma and ratio. And for final linear SVM, we need to decide the value of cost. Usually, ratio is setted to 0.1. The rest three parameters will choose by uniform design [11] which is also introduced in Section II.

D. Experiment Results

1) Checkerboard: A checkerboard dataset is to show the ability of a non-linear SVM. To proof that we remain the feature of RSVM and have the ability to handle large scale data, we use a data with size $1,000,000$ for training and $2,000$ for testing. Fig.(ref) shows the result of the test data. Even those it is a toy problem, it still represents the power of non-linear method.

2) Well Tune Model Experiment: In this part, we use uniform design to find the approach of the best tune models for some datasets. Including svmguide1, w3a, a9a, ijcnn1, zero1 and covtype.

Svmguide1 is a dataset size is 3,089 and has a test set which size is 4,000.

3) General Experiment: Subsubsection text here.

4) Discover During Experiment: Subsubsection text here.

V. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

References

[1] J. Platt *et al.*, "Sequential minimal optimization: A fast algorithm for training support vector machines," 1998.

[2] T. Joachims, "Making large scale svm learning practical," 1999.

[3] R.-E. Fan, P.-H. Chen, and C.-J. Lin, "Working set selection using second order information for training support vector machines," *The Journal of Machine Learning Research*, vol. 6, pp. 1889–1918, 2005.

[4] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 116.

[5] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *Signal Processing, IEEE Transactions on*, vol. 52, no. 8, pp. 2165–2176, 2004.

[6] 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.

[7] H. Ouyang and A. G. Gray, "Fast stochastic frank-wolfe algorithms for nonlinear svms." in *SDM*. SIAM, 2010, pp. 245–256.

[8] 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.

[9] H. P. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, and V. Vapnik, "Parallel support vector machines: The cascade svm," in *Advances in neural information processing systems*, 2004, pp. 521–528.

[10] R. Collobert, S. Bengio, and Y. Bengio, "A parallel mixture of svms for very large scale problems," *Neural computation*, vol. 14, no. 5, pp. 1105–1114, 2002.

[11] K.-T. Fang, D. K. Lin, P. Winker, and Y. Zhang, "Uniform design: theory and application," *Technometrics*, vol. 42, no. 3, pp. 237–248, 2000.

TABLE I
RESULT OF WELL TUNED MODEL

DataSet	SFW 1 epo.	SFW 2 epo.	MSFW 1 epo.	MSII-W 2 epo.	SMO- L2	SMO- L1	SMO- shrink	SMO- wss2	MapReduce RSVM
svmguide1	96.85	96.98	97.00	97.00	97.00	97.00	97.00	97.00	97.05
	0.30	0.86							