International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 12, December-2016

A Map Reduce - based SVM Ensemble with Stochastic Gradient Descent

Zhao Jin
Key Lab. of Machine Learning and Computational Intelligence
College of Mathematics and Information Science,
Hebei University
Baoding City, Hebei Province, 071002, China

Shuxia Lu *
Key Lab. of Machine Learning and Computational Intelligence
College of Mathematics and Information Science,
Hebei University
Baoding City, Hebei Province, 071002, China

Mi Zhou
Key Lab. of Machine Learning and Computational Intelligence
College of Mathematics and Information Science,
Hebei University
Baoding City, Hebei Province, 071002, China

***Abstract*—Stochastic Gradient Descent algorithm (SGD) is a simple and effective algorithm for SVM. It is particularly fast for linear classification and it is also adapted to the non-linear classification with Mercer kernel. The running time scales linearly with the number of iterations and does not depend on the number of the training size. In order to improve the convergence rate and classification accuracy with large data sets. This paper proposes a MapReduce-based SVM ensemble algorithm with SGD. We utilize Hadoop Distributed File System to store big training set and MapReduce parallel computing model to training several SVMs as SVM ensemble. The results show that our methods achieve a faster convergence rate than Pegasos that is a traditional SGD algorithm.**

Keywords—Stochastic Gradient Descent; Support Vector Machine; MapReduce; Voting Mechanism; Ensemble

I. INTRODUCTION

Support Vector Machines [1] is a machine learning algorithm for classification with better generalization ability. It is based on Statistical Learning Theory. VC dimension and structural risk minimization are the theoretical foundation of SVM.

Stochastic gradient descent is an effective approach for training SVM, where the objective is the native form rather than dual form. It proceed by iteratively choosing a labeled example randomly from training set and updating the model weights through gradient descent of the corresponding instantaneous objective function.

T.Zhang [2] proved that a constant learning rate in SGD would numerically achieve good accuracy, enabling a running time of $O(1/\varepsilon^2)$ for linear kernel. The algorithm Norma [3] suggests a learning rate proportional to $1/\sqrt{t}$. Shai Shalev-Shwartz et al [4] presented Pegasos algorithm, which is effective stochastic sub-gradient descent algorithm for solving SVM, which adopted a learning rate of $1/\lambda t$. Zhang Wang et al [5] presented a class of Budgeted SGD (BSGD) algorithms for large-scale kernel SVM training that have constant space and constant time complexity per update. Krzysztof Sopyla et al [6] presented SGD-BB algorithm that shows lower sensitivity to the choice of initial parameters.

According to Machine Learning Theory, training examples is not the more the better for recognizing the pattern. In fact, the proceed of SGD algorithm is not using all training examples but a subset which is chosen from training set randomly through the training process. Therefore, when the training set is too large to load into PC's memory, we can choose a subset of training set to achieve the optimal solution. There are some methods to deal with the "big date" problem. Nasullah Khalid Alham et al [7] presented MRESVM that is a MapReduce-based distributed SVM ensemble algorithm for scalable image annotation. Ferhat et al [8] proposed a different MapReduce-based distributed SVM algorithm for binary classification. There are two differences between [7] and [8]. First, the subsets to train a single SVM are selected using the method of bootstrapping in [7]. In the algorithm of [8], the training set is simply separated into different blocks that are treating as subsets. Second, [7] is an ensemble algorithm which has only one process of MapReduce. [8] is an iteration algorithm that has many times of MapReduce process. Both [7] and [8] use Sequential Minimal Optimization (SMO) [9] to train SVM.

In this paper, we proposed a MapReduce-based SVM ensemble using SGD algorithm that is called MR-SGD. In MR-SGD, we use the MapReduce programming model and Hadoop's Distributed File System to train several SVMs parallel. Each SVM of MR-SGD uses a subset of training set with SGD algorithm. Our proposed algorithm is evaluated with Pegasos [4]. The results show that MR-SGD achieves a fast convergence rate in almost all cases.

This paper is organized as the following: section II discussed the basic stochastic gradient descent algorithm for SVM, both for linear kernel and non-linear kernel. In section III, our proposed algorithm and its details of implementation are discussed. The section IV shows the experimental results on datasets and in the last section conclusion are discussed.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 12, December-2016

II. STOCHASTIC GRADIENT DESCENT (SGD) FOR SVMS

A. SGD for linear kernel

Consider a binary classification problem with data set $S = \{(x_i, y_i), i = 1,...,m\}$, where instance $x_i \in R^n$ is an n-dimensional input vector and $y_i \in \{+1,-1\}$ is the corresponding label of that instance. Training an SVM using S is formulated as solving the following optimization problem

$$\min P(W) = \frac{\lambda}{2}\|W\|^2 + \frac{1}{m}\sum_{(x,y)\in S} l(W;(x,y)),$$

(1)

where $\lambda \geq 0$ is a regularization parameter used to control model complexity and

$$l(W;(x,y)) = \max\{0, 1 - y\langle W, x\rangle\} \quad (2)$$

is the hinge loss function where $\langle W, x\rangle$ denotes the standard inner product between the vectors W and x.

The classifier is expressed as

$$f(x) = W^T x \quad (3)$$

where W is a vector of weights associated with each input. We study the case where the bias is set to zero.

SGD operates as follows. Initially, we set W_1 to be zero vectors. On iteration t of the process, we first choose a training example (x_{i_t}, y_{i_t}) by picking an index $i_t \in (1,...,m)$ uniformly at random. We then replace the objective in Eq. (1) with an approximation based on the training example (x_{i_t}, y_{i_t}). We can get

$$\min P(W) = \frac{\lambda}{2}\|W\|^2 + l(W;(x_{i_t}, y_{i_t})).$$

(4)

We consider the sub-gradient of the above approximate objective, given by:

$$\nabla_t = \lambda W_t - \alpha_t y_{i_t} x_{i_t}, \quad (5)$$

$$\text{where } \alpha_t = \begin{cases} 1, & \text{if } y_{i_t}\langle W_t, x_{i_t}\rangle < 1 \\ 0, & \text{otherwise} \end{cases}. \quad (6)$$

Then we update the norm W using the formula below

$$W_{t+1} \leftarrow W_t - \eta_t \nabla_t, \quad (7)$$

where $\eta_t > 0$ is a learning rate. Then we can rewrite the update formula as follows:

$$W_{t+1} \leftarrow \left(1 - \frac{1}{t}\right) W_t + \eta_t \alpha_t y_{i_t} x_{i_t} \quad (8)$$

After a predetermined number T of iterations, we can acquire the norm of SVM classifier W_{T+1}. The pseudo-code of linear kernel SGD is below.

Algorithm 1 linear kernel SGD

1. Input: S, λ, T
2. Initialize: $W_1 = \vec{0}$;
3. For $t = 1,...,T$
4. Choose $i_t \in \{1,...,|S|\}$ uniformly at random;
5. $\eta_t = \frac{1}{\lambda t}$
6. $W_{t+1} \leftarrow (1 - \frac{1}{t}) W_t$
7. If $y_{i_t}\langle W_t, x_{i_t}\rangle < 1$
8. $W_{t+1} \leftarrow W_{t+1} + \eta_t y_{i_t} x_{i_t}$
9. Output: W_{T+1} .

B. SGD for non-linear kernel

Represented Theorem [10] implies that the optimal solution of Eq. (1) can be expressed as a linear combination of the training instances. So SGD for SVM can be used to solve non-linear problems with Mercer kernels rather than with direct access to the feature vectors.

This section we consider predictors, which are linear functions of some implicit mapping $\Phi(x)$ of the instances. The minimization problem is below:

$$\min P(W) = \frac{\lambda}{2}\|W\|^2 + \frac{1}{m}\sum_{(x,y)\in S} l(W;(\Phi(x), y)) \quad (9)$$

where the loss function is:

$$l(W;(\Phi(x), y)) = \max\{0, 1 - y\langle W, \Phi(x)\rangle\} \quad (10)$$

We then replace the objective in Eq. (9) with an approximation based on the training example (x_{i_t}, y_{i_t}) just as before:

$$\min P(W) = \frac{\lambda}{2}\|W\|^2 + l(W;(\Phi(x_{i_t}), y_{i_t})). \quad (11)$$

Rather than explicitly calculating W by using $\Phi(x)$, it saves training examples to implicitly represent W and using the kernel function $K(x, x') = \langle \Phi(x), \Phi(x')\rangle$ when calculating prediction $W^T x$.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 12, December-2016

The process is the same as the case of linear kernel. Set W_1 to zero vector initially. According Eq. (8), we can get:

$$W_2 = \left(1-\frac{1}{1}\right)W_1 + \frac{1}{\lambda}\alpha_1 y_1 \Phi(x_1) = \frac{1}{\lambda}\alpha_1 y_1 \Phi(x_1)$$

$$W_3 = \left(1-\frac{1}{2}\right)W_2 + \frac{1}{2\lambda}\alpha_2 y_2 \Phi(x_2)$$

$$= \frac{1}{2\lambda}\alpha_1 y_1 \Phi(x_1) + \frac{1}{2\lambda}\alpha_2 y_2 \Phi(x_2)$$

$$= \frac{1}{2\lambda}(\alpha_1 y_1 \Phi(x_1) + \alpha_2 y_2 \Phi(x_2))$$

$$W_4 = \left(1-\frac{1}{3}\right)W_3 + \frac{1}{3\lambda}\alpha_3 y_3 \Phi(x_3)$$

$$= \frac{1}{3\lambda}(\alpha_1 y_1 \Phi(x_1) + \alpha_2 y_2 \Phi(x_2)) + \frac{1}{3\lambda}\alpha_3 y_3 \Phi(x_3)$$

$$= \frac{1}{3\lambda}(\alpha_1 y_1 \Phi(x_1) + \alpha_2 y_2 \Phi(x_2) + \alpha_3 y_3 \Phi(x_3))$$

. . . . . .

Apparently we can induce the presentation of W_{t+1}:

$$W_{t+1} = \frac{1}{\lambda t}\sum_{j=1}^{t}\alpha_j y_j \Phi(x_j) \quad (12)$$

Note that the instance $x_j, j \in \{1,...,t\}$ is chosen by uniformly at random at the jth iteration. Then we can get the predictor with Mercer kernel function.

$$f_t(x) = W_t^T \Phi(x) = \frac{1}{\lambda t}\sum_{j=1}^{t}\alpha_t y_t K\langle x_t, x\rangle \quad (13)$$

It is obvious from Eq. (12) that the training examples can be ignored if the corresponding α is zero because they have no contribute to the predictor.

The pseudo-code of non-linear kernel SGD is below.

Algorithm 2 non-linear kernel SGD

1. Input: S, λ, T
2. Initialize: $W_1 = empty$
3. For $t = 1,...,T$
4. Choose $i_t \in \{1,...,|S|\}$ uniformly at random
5. If $y_{i_t}\frac{1}{\lambda t}\sum_{j=1}^{length(W_t)}\alpha_j y_j K\langle x_j, x_{i_t}\rangle < 1$
6. If x_{i_t} in W_t
7. Increment corresponding α by 1
8. Else
9. Put x_{i_t} in W_t
10. Set corresponding α to 1
11. Output: W_{T+1}

Let W be a List where each element records two information: training example that α is not zero and it's selected times so far.

III. A MAPREDUCE-BASED SVM ENSEMBLE WITH STOCHASTIC GRADIENT DESCENT

Since the run-time of SGD algorithm does not depend directly on the size of the training set, it is especially suited for learning from large datasets. However, there is some difficulty for traditional algorithms when the file of training set is too big to load into the memory of PC.

We proposed an algorithm which is a MapReduce-based SVM ensemble using SGD (MR-SGD). In MR-SGD, we utilize the Hadoop's Distributed File System and the MapReduce programming model to conquer the "big data" problem and to achieve a faster convergence rate than single SVM.

A. *MapReduce Model*

Here we introduce Apache Hadoop [11], which has two core components: Hadoop Distributed File System (HDFS) [12] and MapReduce parallel computation programming model [13].

HDFS is used to store large data. It split data file into multiple chunks. Each chunk is stored in different data nodes.

MapReduce is a parallel processing programming model can handle the large data sets which are stored in HDFS. The basic function of the MapReduce model is iterate over the input, compute key/value pairs from each part of input, group all intermediate values by key, then iterate over the resulting groups and finally reduce each group. The model process in parallel. Map task is an initial transformation step, in which individual input records are processed in parallel. The system shuffles and sorts the map outputs and transfers them to the reduce tasks. Reduce task is a summarization step, in which all associated records are processed together by a single entity.

B. *MR-SGD*

There are two properties about SGD algorithm:

1. The run-time scales linearly with the number of iterations and does not depend on the number of the training size;
2. It proceeds by iteratively choosing a labeled example randomly from training set. If the number of iteration is less than the training set size, all of the instances that are chosen to train classifier are just a partition of the training set.

Given the properties of SGD described above, we can utilize the HDFS to store training set and use MapReduce programming model to select multiple subset of the training set and to train SVM on each subset in parallel using a cluster of computers. Finally, we aggregate all the trained SVMs using Voting Mechanism to predict a testing instance. Fg.1 shows the process of MR-SGD.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 12, December-2016

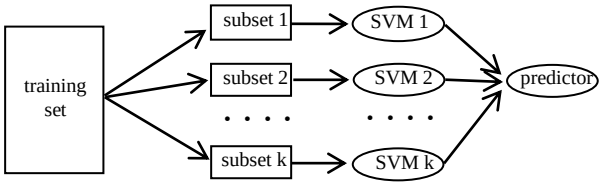

Fig. 1 Workflow of MR-SGD

The implementation details of MR-SGD describes below:

1) First, we pick up *k* subsets from training set where *k* should be odd to support the voting Mechanism. According to Machine Learning Theory, the number of training examples is not the more the better. The size of subset that required training an effective SVM ensemble is different with different training set.

Assume we have *m* training examples and want to randomly choose *d* examples as a subset. Map task deal with each instance in the training set. It loops *k* times to decide whether this instance should be chosen into the corresponding subset. In the *i*th($i \in \{1,...,k\}$) loop, it will generate a random integer from *1* to *m*. If the random integer is less than *d*, then the instance will be put into the *i*th subset to train an SVM in Reduce task.

2) Then Reduce tasks use the subsets which are chosen in Map tasks to train SVMs with SGD algorithm which is mentioned in section II. There is *k* SVMs just as we set in the first step.

3) Finally, we aggregate *k* SVMs using Voting Mechanism to decide the label of the test samples.

IV. EXPERIMENTS

In this section we present experimental results that demonstrate the merits of our algorithm. The basic SGD algorithm is Pegasos [4]. To evaluate the classification accuracy and convergence rate of Pegasos and MR-SGD, several benchmark datasets are used to illustrate both the linear kernel and the non-linear kernel situations. MR-SGD experiments were carried out on a cluster contains 6 machines. Each of machines has four E5-2609 2.50GHz processors and 4GB RAM memory. The CentOS-6.4 was used as an operating system. The version of Apache Hadoop is 2.4.1.

The experiments in this section were performed on four datasets that are downloaded from LIBSVM web page [15]. The Usps and Mnist datasets were used for the task of classifying digits 0, 1, 2, 3, 4 versus the rest of the classes. The original Letter dataset's labels represent 26 alphabets and we set the former 13 alphabets as positive class and the rest as negative class. We use both the linear kernel and Gaussian kernel $K(x,y)=e^{-\gamma\|x-y\|^2}$ in our experiments. The parameters include the regularization parameter λ1 for linear kernel and λ2 for Gaussian kernel and the parameter γ which controlling the width of the Gaussian kernel. The datasets characteristics and the parameters are shown in Table 1 and Table 2 respectively.

TABLE I. DATASETS

Dataset	#Training	#Testing	#Features
Usps	7,291	2,007	256
Mnist	60,000	10,000	780
Letter	15,000	5,000	16
Adult	32,561	16,281	123

TABLE II. PARAMETERS

	Usps	Mnist	Letter	Adult
λ1	8E-3	2E-4	6E-4	6E-3
λ2	1.35E-5	1.67E-5	1.55E-6	3.07E-5
γ	0.1	0.2	40.0	0.05

For the experiments on Mnist dataset, the size of subset to training each SVM is set to 30,000. However, on Usps, Letter and Adult datasets, we used all examples in original training set because the original training set is not a "big data" set. We should better use all of the training examples to complete the training process. Results are shown as follows.

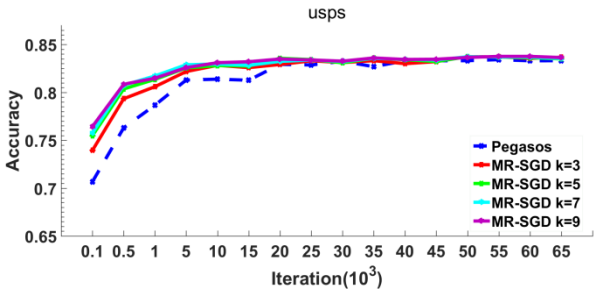

Fig. 2 Testing accuracy on Usps dataset with linear kernel

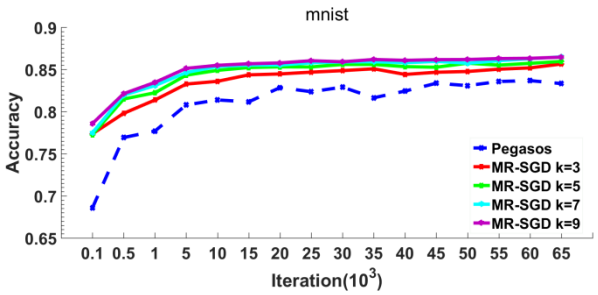

Fig. 3 Testing accuracy on Mnist dataset with linear kernel

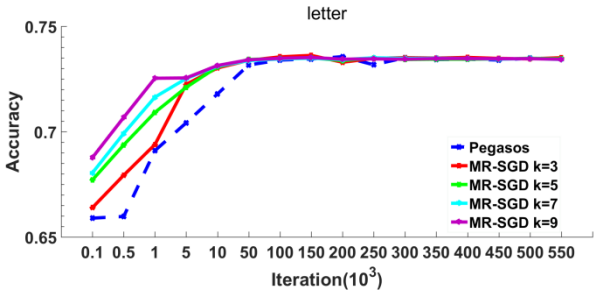

Fig. 4 Testing accuracy on Letter dataset with linear kernel

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 12, December-2016

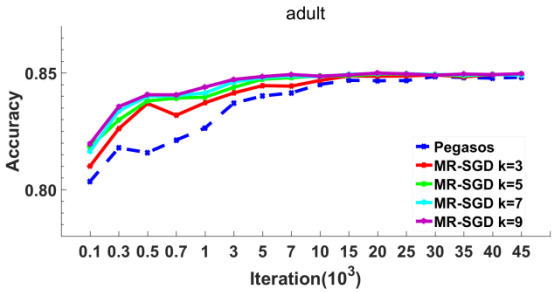

Fig. 5 Testing accuracy on Adult dataset with linear kernel

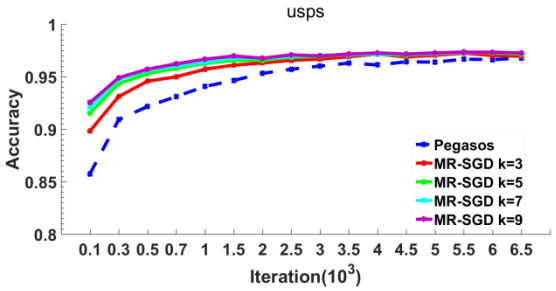

Fig. 6 Testing accuracy on Usps dataset with Gaussian kernel

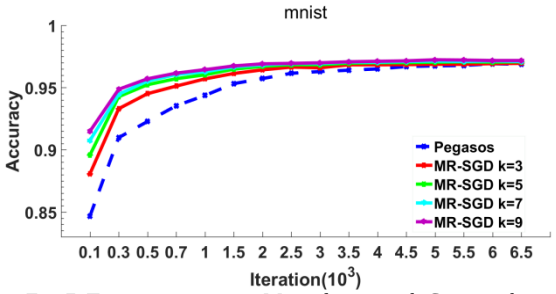

Fig. 7 Testing accuracy on Mnist dataset with Gaussian kernel

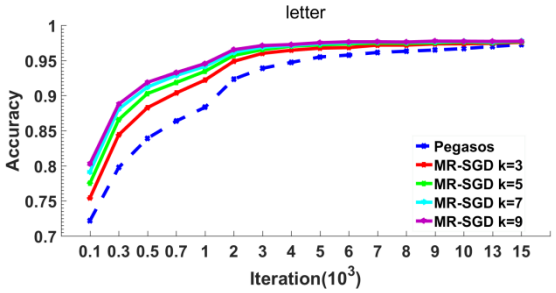

Fig. 8 Testing accuracy on Letter dataset with Gaussian kernel

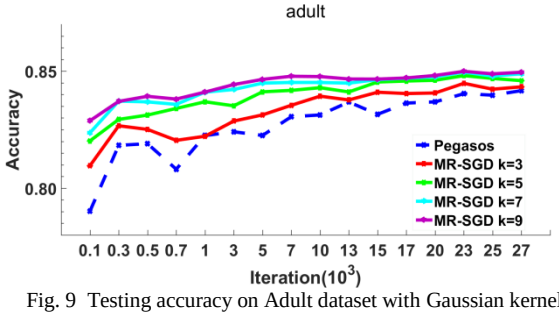

Fig. 9 Testing accuracy on Adult dataset with Gaussian kernel

From Figure 2 to Figure 5, it shows that the convergence rate both Pegasos and MR-SGD with the number of iteration growing. And it also shows that MR-SGD with linear kernel has a faster convergence rate than Pegasos on four datasets. The classification accuracy of MR-SGD is slightly higher than that of Pegasos on four datasets.

From Figure 6 to Figure 9, it shows that MR-SGD with non-linear kernel has a faster convergence rate than Pegasos on four datasets. The classification accuracy of MR-SGD is slightly higher than that of Pegasos on four datasets.

From Figure 2 to Figure 9, it shows that the classification accuracy of MR-SGD with Gaussian kernel are 11 percent higher than that of MR-SGD with linear kernel on Usps and Mnist datasets. The classification accuracy of MR-SGD with Gaussian kernel is 24 percent higher than that of MR-SGD with linear kernel on Letter dataset. However, the classification accuracy of MR-SGD with Gaussian kernel is similar to that of MR-SGD with linear kernel on Adult dataset.

V. CONCLUSIONS

We proposed a MapReduce-based SVM ensemble with SGD algorithm, which we called MR-SGD. The idea of the proposed algorithm is to improve the convergence rate and classification accuracy of basic SGD algorithm for SVM. The experiments show that MR-SGD can achieve faster convergence rate and higher classification accuracy in almost all cases of binary classification. One of the future works is to find a better aggregate strategy of SVM Ensemble. On the other hand, we will exploit different sampling method to enhance performance of MR-SGD.

ACKNOWLEDGMENT

This research is supported by the natural science foundation of Hebei Province No. F2015201185.

REFERENCES

[1] Cortes C, Vapink V, “Support Vector Networks”, Machine Learning, vol 20, 1995, pp. 273-297.
[2] Tong Zhang, “Solving Large Scale Linear Prediction Problems Using Stochastic Gradient Descent Algorithms”, International Conference, 2004, pp. 919-926.
[3] Jyrki Kivinen, Alexander J. Smola, and Robert C. Williamson, “Online Learning with Kernels”, in IEEE Transactions on Signal Processing, vol 52, pp. 2165-2176.
[4] Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, “Pegasos: Primal Estimated Sub-gradient Solver for SVM”, Mathematical Programming, vol 127, 2011, pp. 3-30.
[5] Zhuang Wang, Koby Crammer, Slobodan Vucetic, “Breaking the Curse of Kernelization: Budgeted Stochastic Gradient Descent for Large-Scale SVM Training”, Journal of Machine Learning Research, vol 13, 2012, pp. 3103-3131.
[6] Krzysztof Sopyla, Pawel Drozda, “Stochastic Gradient Descent with Barzilai-Borwein update step for SVM”, Information Sciences, vol 316, 2015, pp. 218-233.
[7] Nasullah Khalid Alham, Maozhen Li, Yang Liu, Man Qi, “A MapReduce-based distributed SVM ensemble for scalable image classification and annotation”, Computers and Mathematics with Applications, vol 66, 2013, pp. 1920-1934.
[8] Ferhat Ozgur CATAK, Mehmet Erdal BALABAN, “A MapReduce-based distributed SVM algorithm for binary classification”, Turkish Journal of Electrical & Computer Science, 2013, pp. 863-873.
[9] JC Platt, “Fast Training of Support Vector Machines using Sequential Minimal Optimization”, Advances in Kernel Methods, 1999, pp. 185-208.
[10] G. Kimeldorf, G. Wahba, “Some results on Tchebycheffian spline functions”, Journal of Mathematical Analysis & Applications, vol 33, 1971, pp. 82-95.
[11] Apache Hadoop. http://hadoop.apache.org/, 2016.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 12, December-2016

[12] Honnutagi, Pooja S, "The Hadoop Distributed File System", International Journal of Computer Science & Information Technolo, vol 5, 2014, pp. 6238-6243.
[13] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters", Communications of the ACM, vol 51, 2008, pp. 107–113.
[14] S. Sonnenburg, V. Franc, E.Y. Tov, M. Sebag, PASCAL large scale learning challenge, 2008.
[15] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm, 2016.