Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2016) 24: 863 – 873
© TÜBİTAK
doi:10.3906/elk-1302-68

A MapReduce-based distributed SVM algorithm for binary classification

Ferhat Özgür ÇATAK[1,*], **Mehmet Erdal BALABAN**[2]
[1]National Research Institute of Electronics and Cryptology, TÜBİTAK, Ankara, Turkey
[2]Quantitative Methods Program, Faculty of Business Administration, İstanbul University, İstanbul, Turkey

Received: 10.02.2013 • **Accepted/Published Online:** 11.12.2013 • **Final Version:** 23.03.2016

Abstract: Although the support vector machine (SVM) algorithm has a high generalization property for classifying unseen examples after the training phase and a small loss value, the algorithm is not suitable for real-life classification and regression problems. SVMs cannot solve hundreds of thousands of examples in a training dataset. In previous studies on distributed machine-learning algorithms, the SVM was trained in a costly and preconfigured computer environment. In this research, we present a MapReduce-based distributed parallel SVM training algorithm for binary classification problems. This work shows how to distribute optimization problems over cloud computing systems with the MapReduce technique. In the second step of this work, we used statistical learning theory to find the predictive hypothesis that would minimize the empirical risks from hypothesis spaces that were created with the Reduce function of MapReduce. The results of this research are important for the training of big datasets for SVM algorithm-based classification problems. We provided the iterative training of the split dataset with the MapReduce technique; the accuracy of the classifier function will converge to global optimal classifier function accuracy in finite iteration size. The algorithm performance was measured on samples from letter recognition and pen-based recognition of a handwritten digits dataset.

Key words: Support vector machine, machine learning, cloud computing, MapReduce, large-scale dataset

1. Introduction

Most machine-learning algorithms have problems with the computational complexity of the training phase of large-scale learning datasets. Applications of classification algorithms for large-scale datasets are computationally expensive to process. The computation time and storage space of the support vector machine (SVM) algorithm are largely determined by the large-scale kernel matrix [1]. Computational complexity and computation time are always a limiting factor for machine learning in practice. In order to overcome this complexity problem, researchers have developed the techniques of feature selection, feature extraction, and distributed computing.

Feature selection methods are used for machine-learning model construction with a reduced number of features. Feature selection is a basic approach for reducing feature vector size [2]. A new combination of feature subsets is obtained with various algorithms, such as information gain [3], correlation-based feature selection [4], Gini index [5], and t-statistics. Feature selection methods solve 2 main problems. The first solution is reducing the number of feature sets in the training set to effectively use computing resources, such as memory and CPU; the second solution is to remove noisy features from the dataset in order to improve the classification algorithm performance [6].

*Correspondence: ozgur.catak@tubitak.gov.tr

Feature extraction methods are used to remove the curse of dimensionality, which refers to the problems resulting from increase in dimensionality. In this approach, high-dimensional feature space is transformed into low-dimensional feature space. There are several feature extraction algorithms, such as principal component analysis (PCA) [7], singular value decomposition (SVD) [8], and independent component analysis (ICA) [9].

The last solution for overcoming the large amount of memory and computation power requirements for training large-scale datasets is chunking or distributed computing [10]. Graf et al. [11] proposed the cascade SVM to overcome very large-scale classification problems. With this method, the dataset is split into n parts in feature space. The nonsupport vectors of each subdataset are filtered, and only the support vectors (SVs) are transmitted. The margin optimization process only uses combined subdatasets to find out the SVs. Collobert et al. [12] proposed a new parallel SVM training and classification algorithm, where each subset of a dataset is trained with SVM, and then the classifiers are combined into a final single classifier function. Lu et al. [13] proposed a strongly connected network-based distributed SVM algorithm. In this method, the dataset is split into k roughly equal parts for each computer in a network. Then SVs are exchanged among these computers. Ruping et al. [14] proposed a novel incremental learning method with the SVM algorithm. Syed et al. [15] proposed another incremental learning method. In this method, a fusion center collects all SVs from the distributed computers. Caragea et al. [16] used the previous method. In this algorithm, the fusion center iteratively sends the SVs back to the computers. Sun et al. [17] proposed a novel method for parallelized SVM based on the MapReduce technique. This method is based on the cascade SVM model. Their approach is based on the iterative MapReduce model Twister, which is different from our implementation of Hadoop-based MapReduce. Their method is the same as the cascade SVM model. They only use the SVs of a subdataset to find an optimal classifier function. Another difference from our approach is that they apply feature selection with the correlation coefficient method for reducing the number of features in the datasets before training the SVM to improve the training time.

In our previous research [18], we developed a novel approach for MapReduce-based SVM training for binary classification problems. We used several UCI datasets to show the generalization property of our algorithm.

In this paper, we propose a novel approach and formal analysis of the models that are generated with the MapReduce-based binary SVM training method. We distribute the whole training dataset over the data nodes of the cloud computing system. At each node, the subset of the training dataset is used for training in order to discover a binary classifier function. The algorithm collects SVs from every node in the cloud computing system and then merges all the SVs to be saved as global SVs. Our algorithm is analyzed with letter recognition [19] and pen-based recognition of handwritten digits [20] dataset with Hadoop streaming, using the mrjob Python library. Our algorithm is built on the LibSVM and is implemented using the Hadoop implementation of MapReduce.

The structure of this article is as follows. In the next section, we will provide an overview of SVM formulations. In Section 3, we will present the MapReduce pattern in detail. Section 4 explains the system model with our implementation of the MapReduce pattern for SVM training. In Section 5, the convergence of our algorithm is explained. In Section 6, the simulation results with letter recognition and pen-based recognition of handwritten digit datasets are shown. Finally, we will make concluding remarks in Section 7.

2. Support vector machine

In the machine-learning field, SVM is a supervised learning algorithm for classification and regression problems depending of the type of output. SVM uses statistical learning theory to maximize the generalization property of the generated classifier model. SVM prevents overfitting of the training dataset. Statistical learning theory generalizes the quality of fitting the training data (empirical error). Empirical risk is $R = \frac{1}{n}\sum_{i=1}^{n} l(f_\theta(x_i),y_i)$, which is the average loss l of the chosen estimator over the training set $(x_i y_i)\}$. SVM uses a set of training data and predicts, for each given input, one of 2 possible classes $-1, 1\}$. As shown in Figure 1, the hyperplane is defined by $w^T x + b = 0$, where $w \in R^n$ is orthogonal to the hyperplane and $b \in R^n$ is the bias. Giving some training data $\mathbf{D}$, a set of point of the form

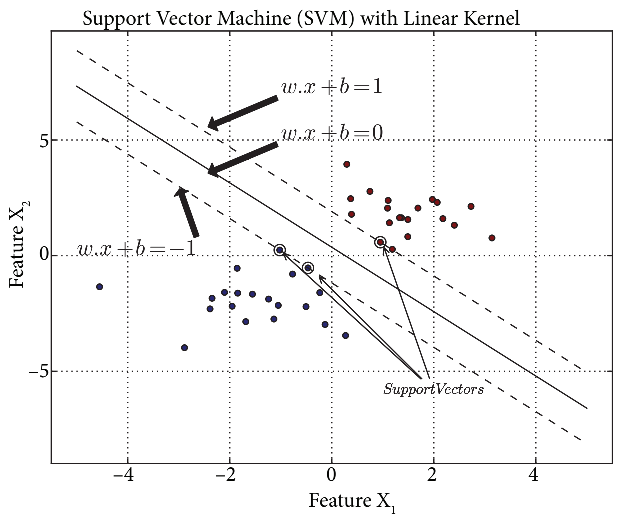

Figure 1. Classification of an SVM with maximum-margin hyperplane trained with samples from 2 classes.

$$\mathbf{D} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in R^m,\, y_i \in \{-1,1\}\}_{i=1}^{n} \tag{1}$$

where x_i is an m-dimensional real vector and y_i is the class of input vector x_i, either –1 or 1. SVMs aim to search a hyperplane that maximizes the margin between the 2 classes of samples in D with the smallest empirical risk [22]. For the generalization property of SVM, 2 parallel hyperplanes are defined, such that $w^T x + b = 1$ and $w^T x + b = -1$. One can simplify these 2 functions into a new one:

$$y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 \tag{2}$$

The SVM aims to maximize the distance between these 2 hyperplanes. One can calculate the distance between them with $\frac{1}{\|w\|}$. The training of the SVM for the nonseparable case is solved using the quadratic optimization problem that is shown in Eq. (3).

$$\begin{aligned} \text{minimize}: \quad & P(\mathbf{w}, b, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{subject to}: \quad & y_i\left((\langle\mathbf{w}, \phi(\mathbf{x}_i)\rangle + b\right) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned} \tag{3}$$

for $i = 1, ..., m$, where ξ_i are slack variables and C is the cost variable of each slack. C is a control parameter for the margin maximization and empirical risk minimization. The decision function of SVM is $f(x) = w^T \phi(x) + b$, where w and b are calculated by the optimization problem P in Eq. (3). Using Lagrange multipliers, the optimization problem P in Eq. (3) can be expressed as:

$$\begin{aligned} \min \ : \quad & F(\alpha) = \frac{1}{2}\alpha^T \mathbf{Q}\, \alpha^T - \alpha^T \mathbf{1} \\ \text{subject to} \ : \quad & 0 \leq \alpha \leq C \\ & y^T \alpha = 0 \end{aligned} \tag{4}$$

where $[Q]_{ij} = y_i\, y_j\, \phi^T(x_i)\, \phi(x_j)$ is the Lagrangian multiplier variable. It is not necessary to know function ϕ, although it is necessary to know how to compute the modified inner product, which will be referred to as the kernel function, represented as $K(x_i x_j) = \phi^T(x_i)\, \phi(x_j)$. Thus, $[Q]_{ij} = y_i y_j K(x_i, x_j)$ [23].

3. MapReduce model

MapReduce is a programming model, derived from the map, which reduces function combination from functional programming. The MapReduce model is widely used to run parallel applications for large-scale dataset processing. MapReduce uses the key/value pair data type in Map and Reduce functions [24]. An overview of the MapReduce system is shown in Figure 2.

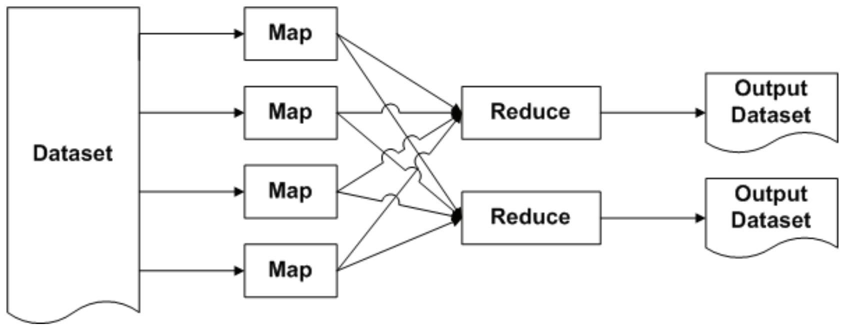

Figure 2. Overview of the MapReduce system.

The MapReduce pattern is divided into 2 functions, Map and Reduce. These 2 functions are separated by a shuffle step of the intermediate key/value data. The MapReduce framework executes these functions in a parallel manner over any number of computers [25]. Simply put, a MapReduce job executes 3 basic operations on a dataset distributed across many shared-nothing cluster nodes. The first task is a Map function that processes each node in parallel without transferring any data to the other nodes. In the next operation, the data processed by the Map function is repartitioned across all nodes of the cluster. Lastly, the Reduce task is executed in parallel by each node with partitioned data.

A file in the distributed file system (DFS) is split into multiple chunks, and each chunk is stored in different data nodes. The input of a Map function is a key/value pair from the input chunks of the dataset, which creates an output in the list of key/value pairs:

$$map\,(key_1, value_1) \Rightarrow list(key_2 value_2).$$

The Reduce function takes a key value, and this value is listed as input. Then it generates a list of new values as output:

$$reduce\,(key_2, list\,(value_2)) \Rightarrow\, list(value_3)\,.$$

4. System model

The cloud computing-based binary class SVM algorithm works as follows. The training set of the algorithm is split into subsets. Each node within a cloud computing system classifies the subdataset locally via the SVM algorithm and gets α values (i.e. SVs), and then passes the calculated SVs to the global SVs to merge them. In the Map stage of the MapReduce job, the subset of the training set is combined with global SVs. In the Reduce step, the merged subset of training data is evaluated. The resulting new SVs are combined with the global SVs in the Reduce step. The algorithm can be explained as follows. First, each node in a cloud computing system reads the global SVs set, and then it merges it with the subsets of the local training dataset and classifies them using the SVM algorithm. Finally, all the computed SVs set in cloud nodes are merged. Thus, the algorithm saves the global SV set with new ones. Our algorithm consists of the following steps (our terminology is shown in Table 1).

Table 1. The notation used in our work.

Notation	Description
t	Iteration number
L	Number of computers (or MapReduce function size)
h^t	Best hypothesis at iteration t
D_l	Subdataset at computer l
SV_l	Support vectors at computer l
SV_{Global}	Global support vector

1. At initialization, the global support vector is set as $t = 0$, $SV^t\ = \emptyset$
2. t = t + 1
3. For any computer in $ll = 1, ..., L$ reads the global SVs and merges them with the subset of the training data
4. Train SVM algorithm with merged new dataset
5. Find out support vectors
6. After all computers in the cloud system complete their training phase, merge all calculated SVs and save the result to the global SVs
7. If $h^t\ =\ h^{t-1}$, stop; otherwise go to Step 2

Algorithm 1. Map function of binary SVM algorithm.

```
$SV_{Global} = \emptyset$   // Empty global support vector set
while $h^t \neq h^{t-1}$
  for $l \in L$ do // For each subset loop
      $D_l^t \leftarrow D_l^t \cup SV_{Global}^t$
  end for
end while
```

Algorithm 2. Reduce function of binary SVM algorithm.

```
while $h^t \neq h^{t-1}$ do
  for $l \in \mathbf{L}$
// Train merged dataset to obtain support
// Vectors and binary class hypothesis
    $SV_l h^t \leftarrow binarySvm(D_l)$
  end for
  for $l \in \mathbf{L}$
    $SV_{Global} \leftarrow SV_{Global} \cup SV_l$
  end for
end while
```

The pseudocodes of our algorithm's Map and Reduce functions are given in Algorithms 1 and 2.

For training SVM classifier functions, we used LibSVM with various kernels. Appropriate parameters C and γ values were found by cross-validation test. We used the 10-fold cross-validation method. The entire system was implemented with Hadoop and streaming Python package mrjob library.

5. Convergence of the algorithm with statistical learning theory

Let S denote a subset of training dataset D. $F(S)$ is the optimal classifier function over dataset S, h^* is the global optimal hypothesis that has a minimal empirical risk $R_{emp}(h)$ over dataset D, and Y_S is the vector space of all possible outputs over subdataset S. Our algorithm's aim is to find a classifier function $f : X \to Y$ such that $f(x) \sim y$. Let H be the hypothesis space of functions $f : X \to Y$. Our algorithm starts with $SV^0_{Global} = \emptyset$ and generates a nonincreasing sequence of a positive set of vectors SV^t_{Global}, where SV^t_{Global} is the vector of the SV at the tth iteration. We used hinge loss for testing our models trained with our algorithm. Hinge loss is effective in SVM as a classifier, since the more the margin is violated, the higher the penalty is [26]. The hinge loss function is the following:

$$l\left(f\left(x\right), y\right) = \, max\left\{0, 1 - y.f(x)\right\} y_i \tag{5}$$

Empirical risk can be computed with an approximation:

$$R_{emp}\left(h\right) = \frac{1}{n} \sum\nolimits_{i=1}^{n} \left(l\left(h\left(x_i\right), y_i\right)\right) \tag{6}$$

According to the empirical risk minimization principle, the binary class learning algorithm should choose a hypothesis $\hat{h}$ in hypothesis space H, which minimizes the empirical risk:

$$\hat{h} = \arg\, R_{emp}(h) \tag{7}$$

A hypothesis is found in every cloud node. Let X be a subset of training data at cloud node i , where $X \in R^{mxn}$, SV^t_{Global} is the vector of the support vector at the tth iteration, and $h^{t,i}$ is the hypothesis at node i with iteration t.

The algorithm's stop point is reached when the hypothesis's empirical risk is the same as the previous iteration. That is:

$$R_{emp}\left(h^t\right) \, = \, R_{emp}\left(h^{t-1}\right) \tag{8}$$

begin{lemma} The accuracy of the classifier function of our algorithm at iteration t is always greater than or equal to the maximum accuracy of the classifier function at iteration $t-1$. That is:

$$R_{emp}\left(h^t\right) \, \leq \, argR_{emp}\left(h\right) \tag{9}$$

Proof Without loss of generality, the iterated MapReduce binary class SVM monotonically converges to an optimum classifier:

$$SV_{Global}^{t} = SV_{Global}^{t-1} \cup \{ SV_{i}^{t-1} \mid i = 1, ...n \}, \tag{10}$$

where n is the dataset split size (or cloud node size). Then the training set for SVM algorithm at node i is

$$d = X \cup SV_{Global}^{t} \tag{11}$$

Adding more samples cannot decrease the optimal value. The generalization accuracy of the subproblem in each node monotonically increases in each iteration step.

6. Simulation results

Our experimental datasets consist of real handwriting data. The first dataset, a pen-based recognition of a handwriting digit dataset [20], contains 250 samples from 44 different writers. All input features are numerical. The classification feature of the dataset is in the range of 0 to 9. The second dataset is a letter recognition dataset that contains capital letters in 20 different fonts.

Linear kernels were used with optimal parameters (C, γ). Parameters were estimated with the cross-validation method. In our experiments, datasets were randomly partitioned into 10 subdatasets of approximately equally sized parts. We ensured that all subdatasets were balanced and that the classes were uniformly distributed. We fit the classifier function with 90% of the original dataset. Then, using this classifier function, we predicted the class of the 10% remaining test dataset. The cross-validation process was repeated 10 times, with each part used once as a test sample. We added the errors of all 10 parts to calculate the overall error.

6.1. Computation time comparison between SVM and MapReduce-based SVM

In our experiments, we compared the single-node SVM-training algorithm to the MapReduce-based SVM training algorithm. We used the single-node training model as the baseline to find the speedup. The calculation of the speedup is computation time with MapReduce divided by the single-node training model computation time. We show the different node size computation results in Tables 2 and 3.

Table 2. Letter recognition dataset SVM training speedup using MapReduce with different node size.

Number of MapReduce job	Speedup
1	1.00
2	3.39
4	4.45
6	4.76
8	5.97
10	6.42

The speedups in both data sets range from $6\times$ to $7\times$. The speedup shown in the tables is the average of 50 runs.

6.2. Results with MapReduce-based SVM

Figure 3 shows the average accuracy of the test error for each dataset. The figure shows the improvement in MapReduce-based SVM at each iteration and stability on large datasets. Figure 4 shows the average number of SVs for each dataset. The figure shows the stability of the number of SVs with MapReduce-based SVM at each iteration.

Table 3. Pen-based recognition of handwriting digit dataset SVM training speedup using MapReduce with different node sizes.

Number of MapReduce job	Speedup
1	1.00
2	2.72
4	4.39
6	4.56
8	6.46
10	7.78

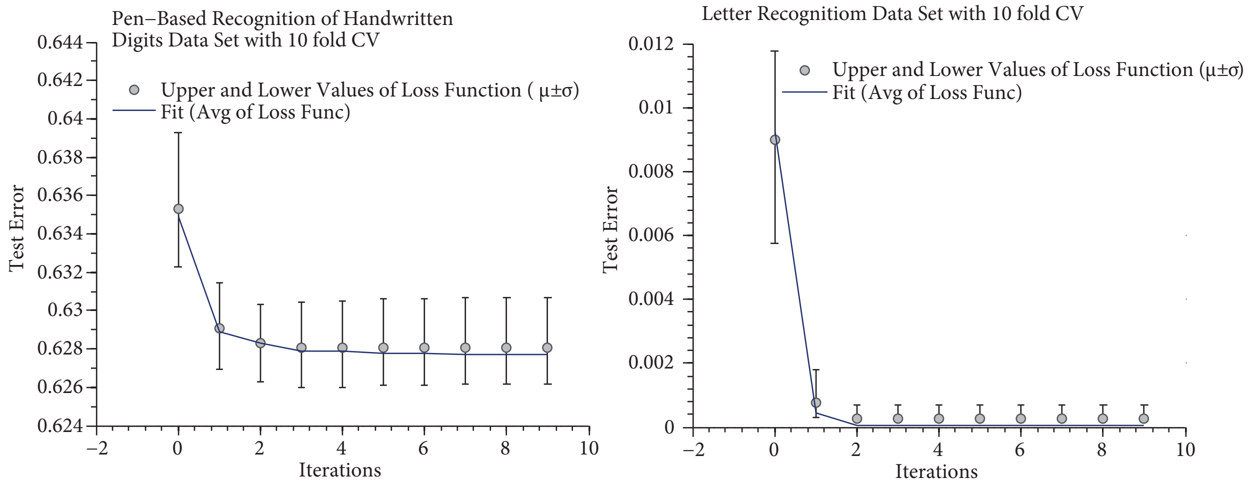

Figure 3. Hinge loss values over iterations with 2 datasets.

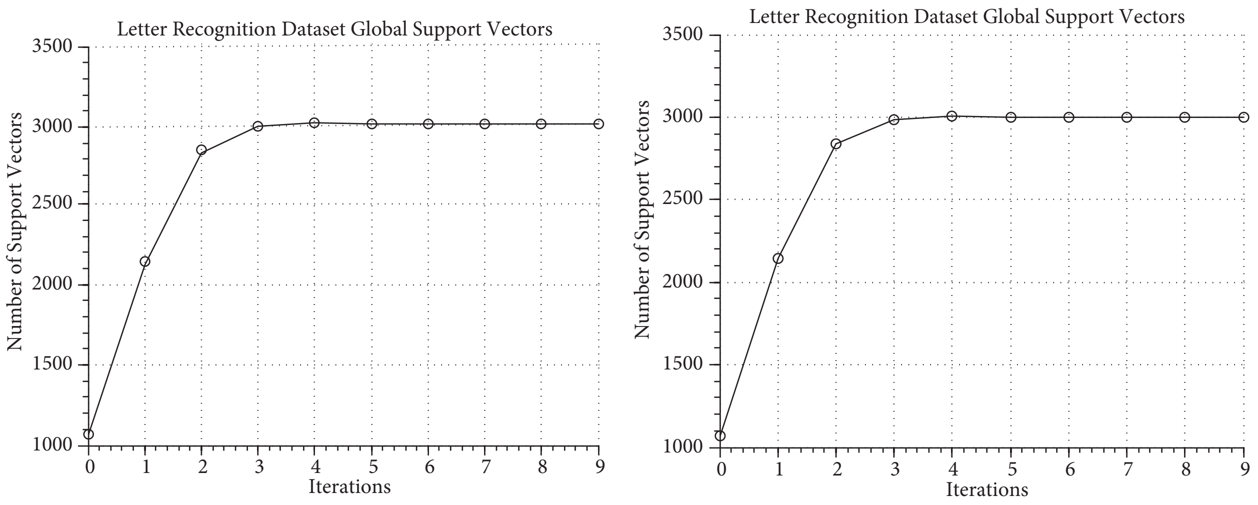

Figure 4. Support vector sizes over iterations with 2 datasets.

To analyze our algorithm, we randomly distributed all the training data to a cloud computing system with 10 computers and pseudo-distributed Hadoop. We developed a Python script for the distributed SVM algorithm with scikit, SciPy, NumPy, mrjob, Matplotlib, and LibSVM. Dataset prediction accuracies with iterations are shown in Tables 4 and 5.

Table 4. Average, maximum, and minimum values of hinge loss for the pen-based recognition of handwriting digit dataset with 10-fold cross-validation.

Iteration no.	Loss (μ)	Loss (μ+σ)	Loss (μ−σ)
1	0.02550	0.03605	0.01736
2	0.00961	0.01602	0.00401
3	0.00801	0.01335	0.00267
4	0.00694	0.01335	0.00134
5	0.00681	0.01335	0.00134
6	0.00654	0.01335	0.00134
7	0.00654	0.01335	0.00134
8	0.00641	0.01335	0.00134
9	0.00641	0.01335	0.00134
10	0.00641	0.01335	0.00134

Table 5. Average, maximum, and minimum values of hinge loss for the letter recognition dataset with 10-fold cross-validation.

Iteration no.	Loss (μ)	Loss (μ+σ)	Loss (μ−σ)
1	0.00925	0.01201	0.00600
2	0.00045	0.00150	0.00000
3	0.00005	0.00050	0.00000
4	0.00005	0.00050	0.00000
5	0.00005	0.00050	0.00000
6	0.00005	0.00050	0.00000
7	0.00005	0.00050	0.00000
8	0.00005	0.00050	0.00000
9	0.00005	0.00050	0.00000
10	0.00005	0.00050	0.00000

The total number of SVs is shown in Table 6. When iteration size becomes 5, the test accuracy values of all datasets reach their highest values, i.e. the smallest value of the hinge loss of empirical error. If the iteration size increases, the value of the test accuracy falls into a steady state. The value of the test accuracy does not change with a large enough number of iteration sizes.

7. Conclusion

In this article, we proposed a new MapReduce-based distributed and parallel binary class SVM classification implementation in cloud computing systems with a MapReduce model. We showed the generalization property of our algorithm with the 10-fold cross-validation method. The results of the empirical analyses show that our algorithm reaches a steady state condition in approximately 5 iterations. Our research differs from the previous distributed or parallel works in 2 main ways. First, we used full datasets for training the SVM algorithm.

Second, we used binary class classification to obtain a classifier function using the structural risk minimization property of statistical learning theory. Our approach is simple to implement in other development environments such as Java, MATLAB, etc.

Table 6. Average SV size for pen-based recognition of handwriting digit and letter recognition dataset with 10-fold cross-validation.

Iteration no.	Pen digit	Letter recognition
1	1068.7	186.9
2	2147.6	314.9
3	2837.7	418.2
4	2981.1	487.6
5	3003.8	520.4
6	2995.8	541.0
7	2996.7	550.1
8	2996.5	553.8
9	2997.5	556.9
10	3001.0	558.2

At present, the term 'big data' is used quite frequently. Most datasets used in machine learning fields, such as human genomes, social networks, and complex physics simulation, can be classified as big data. The results of this research are important for the training of big datasets for SVM algorithm-based classification problems. In our future work, we plan to use this algorithm in multiclass classification problems with an iterative approach of MapReduce with Twister.

References

[1] Bakır GH, Planck M, Bottou L, Weston J. Breaking SVM complexity with cross training. In: Proceedings of the 17th Neural Information Processing Systems Conference; 5–8 December 2005; Vancouver, BC, Canada. Cambridge, MA, USA: MIT Press. pp. 81–88.

[2] Weston J, Mukherjee S, Chapelle O, Pontil M, Poggio T, Vapnik V. Feature selection for SVMs. In: Proceedings of the 14th Annual Neural Information Processing Systems Conference; 1–4 December 2000; Denver, CO, USA. Cambridge, MA, USA: MIT Press. pp. 668–674.

[3] Kullback S, Leibler RA. On information and sufficiency. Ann Math Stat 1951; 22: 79–86.

[4] Hall MA. Correlation-based feature selection for machine learning. PhD, University of Waikato, Hamilton, New Zealand, 1999.

[5] Raileanu LE, Stoffel K. Theoretical comparison between the Gini Index and Information Gain criteria. Ann Math Artif Intel 2004; 41: 77–93.

[6] Mladenić D, Brank J, Grobelnik M, Milic-Frayling N. Feature selection using linear classifier weights: interaction with classification models. In: Proceedings of the 27th Annual International ACM SIGIR Conference; 25–29 July 2004; Sheffield, UK. New York, USA: ACM. pp 234–241.

[7] Jolliffe IT. Principal Component Analysis. New York, NY, USA: Springer, 2002.

[8] Golub GH, Reinsch C. Singular value decomposition and least squares solutions. Numer Math 1970; 10: 403–420.

[9] Common P. Independent component analysis, a new concept? Signal Process 1994; 36: 287–314.

[10] Vapnik V. The Nature of Statistical Learning Theory. New York, NY, USA: Springer, 1995.

[11] Graf HP, Cosatto E, Bottou L, Durdanović I, Vapnik V. Parallel support vector machines: the cascade SVM. Adv Neur In 2005; 17: 521–528.

[12] Collobert R, Bengio S, Bengio Y. A parallel mixture of SVMs for very large scale problems. Neural Comput 2002; 5: 1105–1114.

[13] Lu Y, Roychowdhury V, Vandenberghe L. Distributed parallel support vector machines in strongly connected networks. IEEE T Neural Networ 2008; 7: 1167–1178.

[14] Rüping S. Incremental learning with support vector machines. In: IEEE 2001 International Conference on Data Mining; 29 November–2 December 2001; San Jose, CA, USA. New York, NY, USA: IEEE. pp. 641–642.

[15] Syed NA, Huan S, Kah L, Sung K. Incremental learning with support vector machines. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery; 15–18 August 1999; San Diego, California, USA. Cambridge, MA, USA: ACM. pp. 317–321.

[16] Caragea C, Caragea D, Honavar V. Learning support vector machine classifiers from distributed data sources. In: Proceedings of the 20th National Conference on Artificial Intelligence; 9–13 July 2005; Pittsburgh, PA, USA. Palo Alto, CA, USA: AAAI. pp. 1602–1603.

[17] Sun Z, Fox G. Study on parallel SVM based on MapReduce. In: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications; 16–19 July 2012; Las Vegas, NV, USA. Sterling, VA, USA: CSREA. pp. 495–561.

[18] Catak FO, Balaban ME. CloudSVM: training an SVM classifier in cloud computing systems. In: ICPCA/SWS 2012 Joint Conference on Pervasive Computing and the Networked World; 28–30 November 2012; İstanbul, Turkey. Dordrecht, Germany: Springer. pp. 57–68.

[19] Frey PW, Slate DJ. Letter recognition using Holland-style adaptive classifiers. Mach Learn 1991; 2: 161–182.

[20] Alimoglu F, Alpaydin E. Methods of combining multiple classifiers based on different representations for pen-based handwriting recognition. In: Proceedings of the 4th International Conference on Document Analysis and Recognition; 18–20 August 1997; Washington, DC, USA. New York, NY, USA: IEEE. pp. 637–640.

[21] Stoyanov V, Ropson A, Eisner J. Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics; 11–13 April 2011; Fort Lauderdale, FL, USA. Chicago, IL, USA: JMLR. pp. 725–733.

[22] Vapnik V. An overview of statistical learning theory. IEEE T Neural Netw 1999; 10: 988–999.

[23] Mercer J. Functions of positive and negative type and their connection with the theory of integral equations. Philos T R Soc1909; 209: 415–446.

[24] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. In: Proceedings of the 6th conference on Symposium on Operating Systems Design and Implementation; 6–8 December 2004; Berkeley, CA, USA. New York, NY, USA: ACM. pp. 107–113.

[25] Schatz M. CloudBurst: Highly sensitive read mapping with MapReduce. Bioinformatics 2009; 25: 1363–1369.

[26] Rosasco L, Vito ED, Caponnetto A, Piana M, Verri A. Are loss functions all the same? Neural Comput 2011; 16: 1063–1076.