International Journal of Database Theory and Application
Vol.9, No.8 (2016), pp.61-72
http://dx.doi.org/10.14257/ijdta.2016.9.8.07

Robust Machine Learning Approach for Large Data

Byung Joo Kim

Department of Computer Engineering Youngsan University, Korea
bjkim@ysu.ac.kr

Abstract

Machine learning is ideal for exploiting the opportunities hidden in big data. It delivers on the promise of extracting value from big and disparate data sources with far less reliance on human direction. It is data driven and runs at machine scale. It is well suited to the complexity of dealing with disparate data sources and the huge variety of variables and amounts of data involved. And unlike traditional analysis, machine learning thrives on growing datasets. The more data fed into a machine learning system, the more it can learn and apply the results to higher quality insights. In this paper we propose a robust machine learning approach for dealing with large data set. Through experimental results, proposed method performs well on large data sets.

Keywords*: Least Square Support Vector Machine, Large Data, Conjugate Method*

1. Introduction

Traditional analytics methods are not well suited to capturing the full value of big data. The volume of data is too large for comprehensive analysis, and the range of potential correlations and relationships between disparate data sources — from back end customer databases to live web based clickstreams —are too great for any analyst to test all hypotheses and derive all the value buried in the data. Basic analytical methods used in business intelligence and enterprise reporting methods reduce to reporting sums, counts, simple averages and running SQL queries. Online analytical processing is merely a systematized extension of these basic analytics that still rely on a human to direct activities specify what should be calculated. Machine learning is ideal for exploiting the opportunities hidden in big data. It delivers on the promise of extracting value from big and disparate data sources with far less reliance on human direction. It is data driven and runs at machine scale. It is well suited to the complexity of dealing with disparate data sources and the huge variety of variables and amounts of data involved. And unlike traditional analysis, machine learning thrives on growing datasets. The more data fed into a machine learning system, the more it can learn and apply the results to higher quality insights. Recently kernel trick has been applied to PCA and is based on a formulation of PCA in terms of the dot product matrix instead of the co-variance matrix [1]. Kernel PCA(KPCA), however, requires storing and finding the eigenvectors of a N × N kernel matrix where N is a number of patterns. It is infeasible method when N is large. This fact has motivated the development of incremental way of KPCA method which does not store the kernel matrix. It is hoped that the distribution of the extracted features in the feature space has a simple distribution so that a classifier could do a proper task. But it is point out that extracted features by KPCA are global features for all input data and thus may not be optimal for discriminating one class from others [2]. This has naturally motivated to combine the feature extraction method with classifier for classification purpose. In this paper we propose a new classifier for on-line and big data. Paper is composed of as follows. In Section 2 KPCA and eigen space update criterion is introduced. Conjugate based LS-SVM method is described in Section 3. Experimental results to evaluate the

ISSN: 2005-4270 IJDTA

performance of proposed classifier is shown in Section 4. Discussion of proposed classifier and future work is described in Section 5.

2. Incremental KPCA

A prerequisite of the incremental eigenspace update method is that it has to be applied on the data set. Please refer incremental eigenspace update method in [3-5]. In our previous papers [6] we outlined our approach to these problems, and also gave some preliminary results. Since the publication of our preliminary results, we have refined our algorithm. In this paper we will present these refinements, and the algorithm as a whole, in some detail.

2.1. Eigenspace Updating Criterion

The incremental method should include an additional eigenvector if necessary. In our previous research we can't set explicit rule for adding a eigenvector. In this section we will give a guide line for this problem. The incremental PCA represents the input data with principal components $a_{i(N)}$ and it can be approximated as follows:

$$\hat{x}_{i(N)} = Ua_{i(N)} + \bar{x} \tag{1}$$

To update the principal components $a_{i(N)}$ for a new input x_{N+1}, computing an auxiliary vector η is necessary. η is calculated as follows:

$$\eta = [U \hat{h}_{N+1}]^T (\bar{x} - \bar{x}') \tag{2}$$

then the computation of all principal components is

$$a_{i(N+1)} = (R')^T \begin{bmatrix} a_{i(N)} \\ 0 \end{bmatrix} + \eta, \quad i = 1, \cdots, N+1 \tag{3}$$

The above transformation produces a representation with (k + 1) dimensions. Due to the increase of the dimensionality by one, however, more storage is required to represent the data. If we try to keep a k dimensional eigenspace, we lose a certain amount of information. It is needed for us to set the criterion on retaining the number of eigenvectors. There is no explicit guideline for retaining a number of eigenvectors. Here we introduce some general criteria to deal with the model's dimensionality:

(a) Adding a new vector whenever the size of the residual vector exceeds an absolute threshold.

(b) Adding a new vector when the percentage of energy carried by the last eigenvalue in the total energy of the system exceeds an absolute threshold, or equivalently, defining a percentage of the total energy of the system that will be kept in each update. (c) Discarding eigenvectors whose eigenvalues are smaller than a percentage of the first eigenvalue

(d) Keeping the dimensionality constant.

In this paper we take a rule described in (b). We set our criterion on adding an eigenvector as $\lambda'_{k+1} \rangle 0.7\bar{\lambda}$ where $\bar{\lambda}$ is a mean of the λ. Based on this rule, we decide whether adding u'_{k+1} or not.

Incremental Kernel PCA Algorithm

$X = (x^1 x^2 ... x^n)$: ***matrix of training examples:***

λ : ***initial eigenvalue, U : initial eigenvector***

$K(x, y) := \Phi(x)^{'} \Phi(y)$: ***initial kernel matrix ,***

$\bar{x}$: ***initial mean***

for k=1:n : ***begin re-learning iteration***

$\bar{x}^{'} = \frac{1}{n+1}(n\bar{x} + x_{n+1})$: ***update the mean***

$h_{n+1} = (Ua_{n+1} + \bar{x}) - x_{n+1}$: ***compute orthogonal residual vector***

$\hat{h}_{n+1} = \frac{h_{n+1}}{\|h_{n+1}\|_2}$ ***for*** $\|h_{n+1}\|_2 > 0$, $\hat{h}_{n+1} = 0$ ***otherwise***

$D = \frac{n}{n+1}\begin{bmatrix} \Lambda & 0 \\ 0^T & 0 \end{bmatrix} + \frac{n}{(n+1)^2}\begin{bmatrix} aa^T & \gamma a \\ \gamma a^T & \gamma^2 \end{bmatrix}$:***Construct matrix D***

where $\gamma = \hat{h}_{n+1}^T (x_{n+1} - \bar{x})$, $a = U^T (x_{n+1} - \bar{x})$

$DR = R\Lambda^{'}$: ***solve the eigenproblem i.e computerotation matrix R***

If ((n+1) λ_{N+1} ***> 0.7*** λ***)***

Update eigenvector using the criterion rule

else

retain current eigenspce

end for : ***end of re-learning iteration***

3. SVM and LS-SVM

3.1. Support Vector Machine

Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. They belong to a family of generalized linear classifiers. In another terms, Support Vector Machine (SVM) is a classification and regression prediction tool that uses machine learning theory to maximize predictive accuracy while automatically avoiding over-fit to the data. Support Vector machines can be defined as systems which use hypothesis space of a linear functions in a high dimensional feature space, trained with a learning algorithm from optimization theory that implements a learning bias derived from statistical learning theory. Support vector machine was initially popular with the NIPS community and now is an active part of the machine learning research around the world. SVM becomes famous when, using pixel maps as input; it gives accuracy comparable to sophisticated neural networks with elaborated features in a handwriting recognition task. It is also being used for many applications, such as hand writing analysis, face analysis and so forth, especially for pattern classification and regression based applications. The foundations of Support Vector Machines (SVM) have been developed by Vapnik and gained popularity due to many promising features such as better empirical performance. The formulation uses the Structural Risk Minimization

(SRM) principle, which has been shown to be superior to traditional Empirical Risk Minimization (ERM) principle, used by conventional neural networks. SRM minimizes an upper bound on the expected risk, where as ERM minimizes the error on the training data. It is this difference which equips SVM with a greater ability to generalize, which is the goal in statistical learning. SVMs were developed to solve the classification problem, but recently they have been extended to solve regression problems [5]. The statistical learning theory provides a framework for studying the problem of gaining knowledge, making predictions, making decisions from a set of data. In simple terms, it enables the choosing of the hyper plane space such a way that it closely represents the underlying function in the target space. In statistical learning theory the problem of supervised learning is formulated as follows. We are given a set of training data $\{(\mathbf{x}_1,y_1)... (\mathbf{x}_l,y_l)\}$ in $R^n \times R$ sampled according to unknown probability distribution $P(\mathbf{x},y)$, and a loss function $V(y,f(\mathbf{x}))$ that measures the error, for a given $\mathbf{x}$, $f(\mathbf{x})$ is "predicted" instead of the actual value y. The problem consists in finding a function f that minimizes the expectation of the error on new data that is, finding a function f that minimizes the expected error $\int V(y,f(\mathbf{x}))\,P(\mathbf{x},y)\,d\mathbf{x}\,dy$. In statistical modeling we would choose a model from the hypothesis space, which is closest (with respect to some error measure) to the underlying function in the target space. More on statistical learning theory can be found on introduction to statistical learning theory.

Early machine learning algorithms aimed to learn representations of simple functions. Hence, the goal of learning was to output a hypothesis that performed the correct classification of the training data and early learning algorithms were designed to find such an accurate fit to the data . The ability of a hypothesis to correctly classify data not in the training set is known as its generalization. SVM performs better in term of not over generalization when the neural networks might end up over generalizing easily. Another thing to observe is to find where to make the best trade-off in trading complexity with the number of epochs; the illustration brings to light more information about this. The below illustration is made from the class notes.

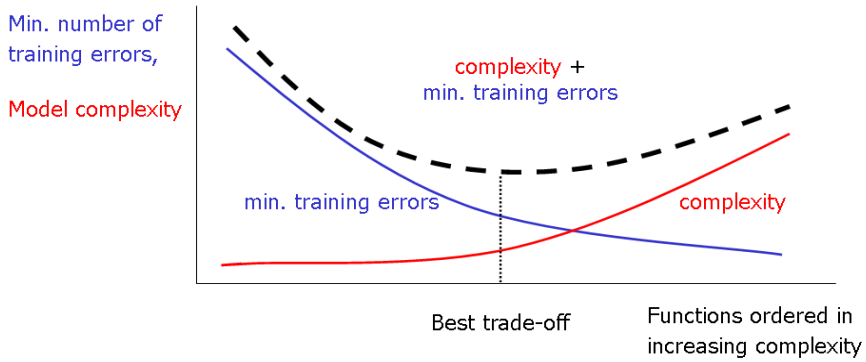

Figure 1. Number of Epochs Vs Complexity

3.1.1. Neural Network and SVM

Working with neural networks for supervised and unsupervised learning showed good results while used for such learning applications. MLP's uses feed forward and recurrent networks. Multilayer perceptron (MLP) properties include universal approximation of continuous nonlinear functions and include learning with input-output patterns and also involve advanced network architectures with multiple inputs and outputs.

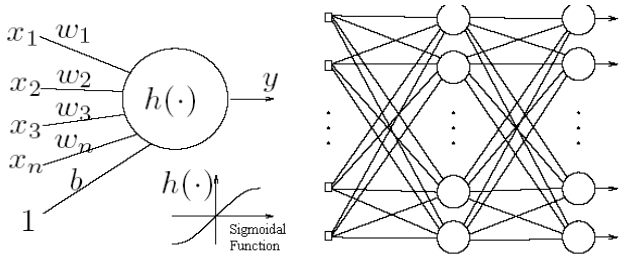

Figure 2. Simple Neural Network and Multilayer Perceptron

There can be some issues noticed. Some of them are having many local minima and also finding how many neurons might be needed for a task is another issue which determines whether optimality of that NN is reached. Another thing to note is that even if the neural network solutions used tends to converge, this may not result in a unique solution. Now let us look at another example where we plot the data and try to classify it and we see that there are many hyper planes which can classify it.

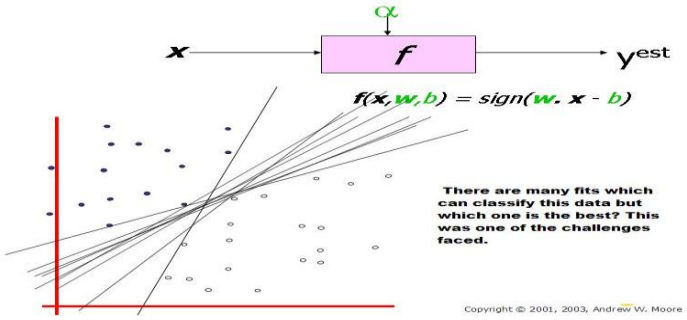

Figure 3. Linear Classifier

From Figure 3, there are many linear classifiers (hyper planes) that separate the data. However only one of these achieves maximum separation. The reason we need it is because if we use a hyper plane to classify, it might end up closer to one set of datasets compared to others and we do not want this to happen and thus we see that the concept of maximum margin classifier or hyper plane as an apparent solution. Figure 4 gives the maximum margin classifier example which provides a solution to the above mentioned problem.

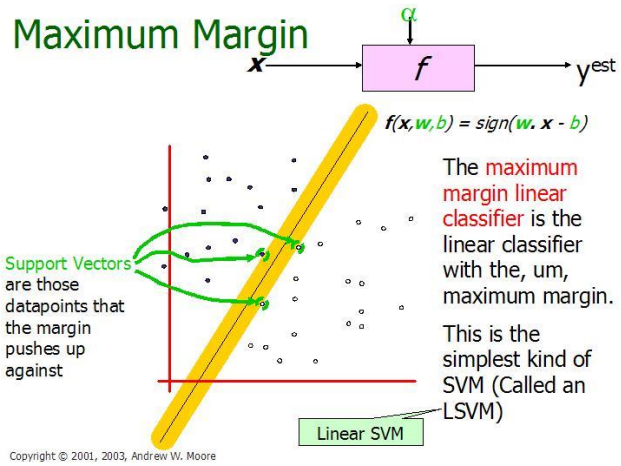

Figure 4. Maximum Margin of Linear SVM

Figure 4 is the maximum linear classifier with the maximum range. Another interesting question is why maximum margin? There are some good explanations which include better empirical performance. Another reason is that even if we've made a small error in the location of the boundary this gives us least chance of causing a misclassification. The other advantage would be avoiding local minima and better classification. Now we try to express the SVM mathematically and for this tutorial we try to present a linear SVM. The goals of SVM are separating the data with hyper plane and extend this to non-linear boundaries using kernel trick. Distance of closest point on hyperplane to origin can be found by maximizing the x as x is on the hyper plane. Similarly for the other side points we have a similar scenario. Thus solving and subtracting the two distances we get the summed distance from the separating hyperplane to nearest points. Maximum Margin = M = 2 / ||w||. Now maximizing the margin is same as minimum. Now we have a quadratic optimization problem and we need to solve for w and b. To solve this we need to optimize the quadratic function with linear constraints. The solution involves constructing a dual problem and where a Langlier's multiplier α_i is associated. Training SVM becomes quite challenging when the number of training points is large. A number of methods for fast SVM training have been proposed.

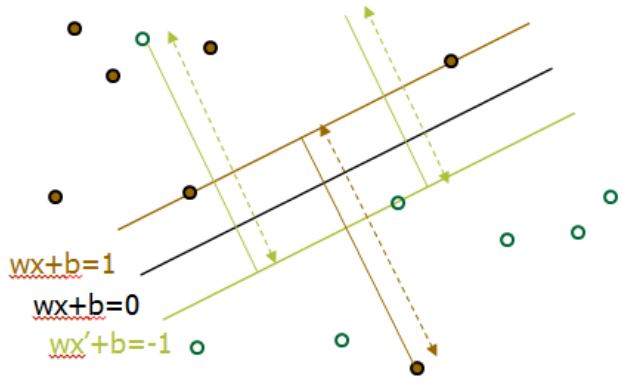

Figure 5. Representation of Hyper Planes

3.2. LS-SVM for Big Data

Support vector machines(SVM) developed by Vapnik [7] and it is a powerful methodology for solving problems in nonlinear classification. Originally, it has been introduced within the context of statistical learning theory and structural risk minimization. In the methods one solves convex optimization problems, typically by quadratic programming(QP). Solving QP problem requires complicated computational effort and need more memory requirement. LS-SVM[8] overcomes this problem by solving a set of linear equations in the problem formulation. LS-SVM method is computationally attractive and easier to extend than SVM. But traditional batch way LS-SVM requires storing (N+1) × (N+1) matrix where N is a number of patterns. It is infeasible method when dealing with big data. For big data sets the use of iterative methods is recommended. In principle, various methods can be used at this point including SOR(Successive Over-Relaxation), CG(Conjugate Gradient), GMRES(Generalized Minimal Residual) *etc.* However, not all of these iterative methods can be applied to any kind of linear system. For example, in order to apply CG the matrix should be positive definite. Due to the presence of the *b* bias term in the LS-SVM model the resulting matrix is not positive definite. So before we can apply such methods we have to transform the linear system into a positive definite system. The LS-SVM KKT system is of the form

$$\begin{bmatrix} 0 & Y^T \\ Y & H \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \quad (4)$$

More specifically with $H = \Omega + I/\gamma$, $\xi_1 = b, \xi_2 = \alpha$, $d_1 = 0, d_2 = I_v$. This can be transformed into

$$\begin{bmatrix} s & 0 \\ 0 & H \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 + H^{-1} y \xi_1 \end{bmatrix} = \begin{bmatrix} -d_1 + y^T H^{-1} d_2 \\ d_2 \end{bmatrix} \quad (5)$$

with $S = y^T H^{-1} y > 0$ ($H = H^{-T} > 0$) Because *s* is positive and *H* positive definite the overall matrix is positive definite. This form is very suitable because different kinds of iterative methods can be applied to problems involving positive definite matrices. This leads to the LS-SVM classifier with conjugate gradient algorithm LS-SVM for big data is as follows.

1. Solve η, ν from $H\eta = Y$ and
   $H\nu = 1_v$
2. Compute $s = Y^T \eta$
3. Find solution
   $b = \eta^T 1_v / s$
   $\alpha = \nu - \eta b$

4. Experiment

To evaluate the performance of proposed classification system, experiment is performed on big data. First we evaluate the proposed system to YouTube comedy slam preference data set. Next we will extend our experiment to KDD CUP 99 data.

4.1. YouTube Comedy Slam Preference Data Set

YouTube Comedy Slam[11] is a video discovery experiment running on YouTube's version of labs (called TestTube) for a few months in 2011 and 2012. This database

contains 1138,562 instances and number of attributes are 3. In this experiment, a pair of videos were shown to the user and the user was asked to vote for the video that they found funnier. Left/right positions of the videos were randomly selected before being presented to the user to eliminate position bias. Videos were selected from a large pool of weekly updated sets of videos. Each line in this dataset corresponds to one vote over a pair of YouTube videos. Each video is represented by its YouTube video ID. User preference over a pair of videos is presented in the form of string "left" if the left video was deemed funnier, and "right" otherwise. The first 80% are provided here as the training dataset and the remaining 20% as the testing dataset. In [10] it is shown that the use of 10-fold cross-validation for hyperparameter selection of LS-SVMs consistently leads to very good results. In this problem RBF kernel has been taken and hyperparameter $\gamma_1 = 3.3651, \gamma_2 = 64.174, \gamma_3 = 34.427, \gamma_4 = 18.438$ and $\sigma_1 = 89.361, \sigma_2 = 565.371, \sigma_3 = 43.472, \sigma_4 = 68.594$ are obtained by 10-fold cross-validation technique.

Table 1. Training and Generalization Result on YouTube Comedy Slam Preference Data

	Training	Generalization	Eigenvalue update criterion
Standard LS-SVM	100%	98.02%	none
Proposed method	100%	97.8%	$\lambda^{'} \rangle 0.7\overline{\lambda}$

The results on the YouTube data are given in Table 1. Generalization ability in proposed method is similar to standard LS-SVM. But in standard LS-SVM (1138,563) x (1138,563) matrix is needed. It is not infeasible for big data.

4.2. KDD CUP 99 Data

We extend our experiment to more big data. The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records. A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. Attacks fall into four main categories:

- DOS: denial-of-service, *e.g.* syn flood;
- R2L: unauthorized access from a remote machine, *e.g.* guessing password;
- U2R: unauthorized access to local superuser (root) privileges, *e.g.*, various "buffer overflow" attacks;
- Probing: surveillance and other probing, *e.g.*, port scanning.

It is important to note that the test data is not from the same probability distribution as the training data, and it includes specific attack types not in the training data. This makes the task more realistic. The datasets contain a total of 24 training attack types, with an additional 14 types in the test data only. Table 2 shows the results of the classification performance and computing time for training and testing of data by the proposed system using all features. Table 3 shows the results of the classification performance and computing time for training and testing data by the proposed system using extracted features. We can see that using important features for classification gives similar accuracies compared to using all features and reduces the training and testing time. Comparing Table 2 with Table 3, we obtain the following results. The performance when

using the extracted features does not show any significant differences to when using all features. This means that the proposed on-line feature extraction method has a good performance in extracting features. The proposed method has another merit in memory requirement. The advantage of the proposed feature extraction method is more efficient in terms of memory requirement than a batch KPCA because the proposed feature extraction method does not require the whole N X N kernel matrix where N is the number of the training data. A second one is that the proposed on-line feature extraction method performance is comparable in performance to a batch KPCA.

Table 2. Performance of Proposed System Using all Features

Class	Accuracy	Training Time (Sec)	Testing Time (Sec)
Normal	98.55	5.83	1.45
Probe	98.59	28.0	1.96
DOS	98.10	16.62	1.74
U2R	98.64	2.7	1.34
R2L	98.69	7.8	1.27

Table 3. Performance of Proposed System Using Extracted Features

Class	Accuracy	Training Time (Sec)	Testing Time (Sec)
Normal	98.43	5.25	1.42
Probe	98.63	25.52	1.55
DOS	98.14	15.92	1.48
U2R	98.64	2.17	1.32
R2L	98.70	7.2	1.08

4.3. Wine Data

We take training data as wine data obtained from the UCI Machine Learning Repository. Wine data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivates. The analysis determined the quantities of 13 constituents found in each of the three types of wines. Detailed attributes are available from http://www.ics.uci.edu/\~mlearn/MLSummary.html. The number of instances per class is 59 for class 1, 71 for class 2, and 48 for class 3 respectively. In this case we use a classifier as a multiclass LS-SVM proposed by Suykens[68]. The outputs can in principal encode 2^{*}different classes. In this case there exists 3 classes so we set *m*=2. We don't discuss the issue of optimal coding in this thesis. A RBF kernel has been taken with $\sigma_1^2 = 0.3154$, $\sigma_2^2 = 0.2928$ and $v_1 = 32.9019$, $v_2 = 14.733$ by 10-fold cross-validation technique.

Table 4. Training and Generalization Result on Wine Data

	Training	Generalization	Eigenvalue update criterion
Standard LS-SVM	100%	98.04%	none
Proposed method	100%	97.5%	$\lambda_i' \rangle 0.7\bar{\lambda}$

4.4. NIST Handwritten Data Set

To validate the above results on a widely used pattern recognition benchmark database, we repeated the same experiment on the NIST data set. This database originally contains 15,025 digit images. For computational reasons, we decided to use a subset of 2000 data set, 1000 for training and 1000 for testing. Figure 21 shows the sample images of training data. A RBF kernel has been taken with $\sigma_1^2 = 67.416$, $\sigma_2^2 = 56.351$, and are obtained by 10-fold cross-validation.

Table 5. Training and Generalization Result on NIST Handwritten Data

	Training	Generalization	Eigenvalue update criterion
Standard LS-SVM	100%	97.02%	none
Proposed method	100%	96.5%	$\lambda' \rangle 0.7\overline{\lambda}$

Figure 6. NIST Handwritten Data Set

The results on the NIST data are given in Table 5. For this widely used pattern recognition problem, we can see that proposed classification system classifies well on NIST handwritten data set.

4.5. Comparison with SVM

Recently SVM has been a powerful methodology for solving problems in nonlinear classification. To evaluate the classification accuracy of the proposed system it is desirable to compare with SVM. Generally a disadvantage of the incremental method is its accuracy compared to the batch method even though it has the advantage of memory efficiency. According to Table 6 and Table 7 we can see that the proposed method has better classification performance compared to batch SVM. Through this result we can show that the proposed classifier has remarkable classification accuracy, although it is worked in an incremental way.

Table 6. Performance Comparison of Proposed Method and SVM Using all Features

	Normal	Probe	DOS	U2R	R2L
Proposed method	98.76	98.81	98.56	98.92	98.86
SVM	98.55	98.70	98.25	98.87	98.78

Table 7. Performance Comparison of Proposed Method and SVM Using Extracted Features

	Normal	Probe	DOS	U2R	R2L
Proposed method	98.67	98.72	98.56	98.88	98.78
SVM	98.59	98.38	98.22	98.87	98.78

5. Conclusion and Remarks

A conjugate based LS-SVM which combining incremental KPCA was presented for dealing with big data. Such classifier has following advantages. Proposed classifier is more efficient in memory requirement than batch LS-SVM. In batch LS-SVM the (N+1) × (N+1) matrix has to be stored, while for our proposed method does not. It is very useful when dealing with big data. Experimental results on huge data from UCI machine learning repository, proposed method shows lead to good performance.

Acknowledgment

This study was supported by a grant of Youngsan University (in 2016).

References

[1] V. N. Vapnik, “Statistical learning theory”, John Wiley & Sons, New York, **(1998)**.
[2] H. Gupta, A. K. Agrawal, T. Pruthi, C. Shekhar and R. Chellappa, “An Experimental Evaluation of Linear and Kernel-Based Methods for Face Recognition”, accessible at http://citeseer.nj.nec.com.
[3] P. Hall, D. Marshall and R. Martin, “Incremental eigenalysis for classification”, In British Machine Vision Conference, vol. 1, **(1998)**, pp. 286-295.
[4] J. Winkeler, B. S. Manjunath and S. Chandrasekaran, “Subset selection for active object recognition”, In CVPR, IEEE Computer Society Press, vol. 2, **(1999)**, pp. 511-516.
[5] H. Murakami and B. V. K. V. Kumar,” Efficient calculation of primary images from a set of images”, IEEE PAMI, vol. 4, no. 5, **(1982)**, pp. 511-515.
[6] K. B. Joo, S. J. Yong, H. C. Ha and K. I. Kon, “Incremental Feature Extraction Based on Emperical Feature Map”, Foundations of Intelligent Systems of Lecture Notes in Artificial Intelligence, vol. 2871, **(2003)**, pp. 440-444.
[7] V. N. Vapnik, “Statistical learning theory”, John Wiley & Sons, New York, **(1998)**.
[8] J. A. K. Suykens and J. Vandewalle, “Least squares support vector machine classifiers”, Neural Processing Letters, vol. 9, **(1999)**, pp. 293-300.
[9] G. H. Golub and C. F. Van, “Large scale LS_SVM Matrix Computations”, Baltimore MD: Johns Hopkins University.
[10] G. V. Suykens, T. J. A. K. Lanckriet, G. Lambrechts, A. B. D. Moor and J. Vandewalle, “A Bayesian Framework for Least Squares Support Vector Machine Classifiers”, Internal Report 00-65, ESAT-SISTA, K.U. Leuven.
[11] Accesible at http://archive.ics.uci.edu/ml/

