Incremental learning algorithm based on support vector machine with Mahalanobis distance (ISVMM) for intrusion prevention

Conference or Workshop Item

Incremental Learning Algorithm based on Support Vector Machine with Mahalanobis distance (ISVMM) for Intrusion Prevention

Hnin Ohnmar Myint* and Phayung Meesad**
*Department of Information Technology, Faculty of Information Technology
**Department of Teacher Training in Electrical Engineering, Faculty of Technical Education
King Mongkut's University of Technology North Bangkok (KMUTNB)

***Abstract*- In this paper we propose a new classifier called an Incremental Learning Algorithm based on Support Vector Machine with Mahalanobis distance (ISVMM). Prediction of the incoming data type by supervised learning of Support Vector Machine (SVM), reducing the step of calculation and complexity of the algorithm by finding a support set, error set and remaining set, providing of hard and soft decisions, saving the time for repeatedly training the datasets by applying the incremental learning, a new approach for building an ellipsoidal kernel for multidimensional data instead of a sphere kernel by using Mahalanobis distance, and the concept of handling the covariance matrix from dividing by zero are various features of this new algorithm. To evaluate the classification performance of the algorithm, it was applied on intrusion prevention by employing the data from the third international knowledge discovery and data mining tools competition (KDDcup'99). According to the experimental results, ISVMM can predict well on all of the 41 features of incoming datasets without even reducing the enlarged dimensions and it can compete with the similar algorithm which uses a Euclidean measurement at the kernel distance.**

Keywords: Incremental Learning, Support Vector Machine, Mahalanobis, Euclidean, Kernel, Intrusion Prevention.

I. INTRODUCTION

Intrusion prevention is a relatively new field which tries to predict computer attacks by examining various data records. Early IPS was IDS that was able to implement prevention commands to firewalls and access control changes to routers. At the market of networking technology, there are a lot of commercially available tools. The majority problems of commercially available tools today are that they:

- need to regularly update the attack-type database
- are sensitive to unknown attacks that the system has never seen before
- in spite of being detected, it is still hard to prevent the new attacks
- rarely to detect the attack in actual real-time
- are cost sensitive and cannot extensible

Most of the powerful data mining algorithms can predict what is going on at the next stage after training enough amounts of old data records. Application of data mining technology in Intrusion Detection/Prevention becomes a hot spot in current research. There are many existing classifiers based on Incremental Learning such as Online Support Vector Regression [2], Adaptive Resonance Theory (ART) [3], Mahalanobis Distance-Based ARTMAP Network [4], Fuzzy ARTMAP [5], and Incremental Learning Algorithm based on Mahalanobis distance (ILM) [6], etc. Normally, the ability to learn a stream of input data is the common weak point of most of the existing classifiers.

In the real world, upcoming input data streams are arising continuously. In the process of building this new algorithm, the incremental learning is the most critical step. The main reason to use incremental learning is that there is no need to wait until all samples are ready.

Support Vector Machines (SVMs) have become a popular tool for learning with large amounts of high dimensional data. This is because of the fact that the generalization property of a SVM does not depend on all the training data but only a subset thereof, the so-called Support Vectors. The advantages of support vector machines are that can improve generalization ability and can predict well by supervised learning. However, one of the drawbacks of normal SVM algorithm is that it works in batch mode only. Most of the existing kernels employed in nonlinear SVMs measure the similarity between a pair of pattern images based on the Euclidean inner product or the Euclidean distance of corresponding input patterns [7].

TABLE I
NOTATIONS AND VARIABLES IN THIS PAPER

Notation	Meaning
$\mathbf{x}_i$, y_i	the i^{th} training sample and class n^{th} dimension
l	the number of elements in the training set
$\mathbf{w}$	Normal vector
b	bias
ϵ	insensitive loss threshold
ξ_i and ξ_i^*	slack variable (error-epsilon)
$\mathbf{Q}$	kernel matrix
$\mathbf{W}$	the margin
$\eta_i, \eta_i^*, \delta_i, \delta_i^*, \mu_i, \mu_i^*$ and ζ	Lagrange multipliers
α and α^*	coefficients
θ	weight
C	the parameter
$f(\mathbf{x_i})$	function estimated
$h(\mathbf{x_i})$	margin function
S, E, R	support set , error set, remaining set
N	non support set
$\mathbf{R}$	matrix
$L_{c1}, L_{c2}, \mathbf{L_s}, \mathbf{L_e}, \mathbf{L_r}$	least variations of each sample to a new set

978-1-4244-3388-9/09/$25.00 ©2009 IEEE

This paper presents a new classifier namely Incremental Learning Algorithm based on Support Vector Machine with Mahalanobis distance (ISVMM). The proposed classifier is applied for Intrusion Prevention as an example application. It fills the weak point of normal SVM by incremental learning and improves its performance with Mahalanobis distance. This new algorithm can possibly add and delete elements in the machine without restarting the training from the beginning.

II. Derivation Of The New Algorithm

A. Support Vector Machines

When input data is two sets of vectors in an n-dimensional space, SVM will construct a separating hyperplane in the space which maximizes the margin between the two data sets. If there exists no hyperplane that can split the "yes" and "no" examples, the Soft Margin method will choose a hyperplane that splits the examples as cleanly as possible [8]. In this condition, it needs to increase the Quadratic Programming Problem of normal SVM by adding a bound in order to set the tolerance on errors number that can be committed. Because of minimization problem, it can be set all the constraints ≥ 0.

$$\min \frac{1}{2}\,\mathbf{w}^T \cdot \mathbf{w} + C\ \sum_{i=1}^{l}(\xi_i + \xi_i^*) \tag{1}$$

$$\text{subject to.}\quad y_i - (\mathbf{w}^T \cdot \boldsymbol{\phi}(\mathbf{x}) + b) \leq \epsilon + \xi_i$$
$$(\mathbf{w}^T \cdot \boldsymbol{\phi}(\mathbf{x}) + b) - y_i \leq \epsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \geq 0\ ,\ i = 1\ldots\ldots\ldots l$$

B. Calculation of Kernel Based on the Mahalanobis distance

For this Quadratic Optimization Problem, the prediction of $f(\mathbf{x})$ in the new sample x:

$$f(\mathbf{x}) = \text{sign}\ (\sum_{i=1}^{l} y_i\ \alpha_i\ (\mathbf{x} \cdot \mathbf{x_i}) + b\ ) \tag{2}$$

In this equation, the dot product of $(\mathbf{x} \cdot \mathbf{x_i})$ means each sample is multiplied by the value of the examples trained. If instead of a simple multiplication, other more complicated operations come into play the end result changes. Substitution of $(\mathbf{x} \cdot \mathbf{x_i})$ in the expression with another function, called the Kernel Function, which allows growth of the SVM solution [4]. To increase the reading facility, kernel function can be defined as:

$$\mathbf{K}(\mathbf{x_i}, \mathbf{x_j}) = \boldsymbol{\phi}(\mathbf{x}_i).\boldsymbol{\phi}(\mathbf{x}_j) = \mathbf{Q_{ij}} \tag{3}$$

The matrix **Q** contains the values of kernel function and it is called kernel matrix. It is useful to save kernel function values in this way for computational reasons. The similarity between the pattern images $\boldsymbol{\phi}(\mathbf{x}_i)$ and $\boldsymbol{\phi}(\mathbf{x}_j)$ measured by the kernel functions is based on Mahalanobis distance.

$$\text{dis}_{\text{Mah}}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}}\mathbf{K}^{-1}(\mathbf{x}_i - \mathbf{x}_j)} \tag{4}$$

The Mahalanobis distance can be applied directly to modeling problems as a replacement for the Euclidean distance. When using the Mahalanobis distance measurement at the kernel, the problem of singularity covariance matrices (**K**) or NaN (Not-a-Number) result will be happened. To handle the covariance matrix (**K**) from dividing by zero, the ($\alpha\mathbf{I}_n$) concept was introduced to eliminate the singularity matrix problem.

$$\mathbf{K}_{j,initial} = \alpha\mathbf{I}_n \tag{5}$$

$$\alpha\mathbf{I}_n = \alpha \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & \ldots & 0 \\ 0 & \alpha & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \alpha \end{bmatrix} \tag{6}$$

The recurrent computation of covariant matrix in ISVMM is adapted from the ILM algorithm [6] and SFAM algorithm [9].

C. Calculation of Lagrange function [2,10, and 11]

From the minimization problem, it can be calculated a Lagrange function which includes all the constraints, introducing some Lagrange multipliers.

$$L_P = \frac{1}{2}\,\mathbf{w}^T \cdot \mathbf{w} + C\ \sum_{i=1}^{l}(\xi_i + \xi_i^*)$$
$$- \sum_{i=1}^{l}(\eta_i\xi_i + \eta_i^*\,\xi_i^*)$$
$$- \sum_{i=1}^{l}\ \alpha_i\ (\epsilon + \xi_i - y_i + \mathbf{w}^T \cdot \boldsymbol{\phi}(\mathbf{x}) + b)$$
$$- \sum_{i=1}^{l}\ \alpha_i^*\ (\epsilon + \xi_i^* + y_i - \mathbf{w}^T.\ \boldsymbol{\phi}(\mathbf{x_i}) - b)$$
$$\text{subject to.}\ \ \alpha_i, \alpha_i^*,\ \eta_i, \eta_i^* \quad \geq 0,\ i = 1\ldots\ldots\ldots l \tag{7}$$

After substituting the new definitions of w, η_i, η_i^* that derive by the partial derivatives of the Lagrangian and deleting some unnecessary parts, Lagrange functions becomes:

$$L_P = -\frac{1}{2}\ \sum_{i=1}^{l}\sum_{j=1}^{l}\boldsymbol{\phi}(\mathbf{x_i}).\boldsymbol{\phi}(\mathbf{x_j})(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)$$
$$+ \sum_{i=1}^{l}\ y_i\,(\alpha_i - \alpha_i^*)$$
$$- \sum_{i=1}^{l}\ \epsilon\,(\alpha_i - \alpha_i^*) \tag{8}$$

D. Calculation of Karush-Kuhn-Tucker (KKT) conditions [2,10]

All we need to find is the relation of h(x) and ϵ at the changing of α.

$$h(\mathbf{x}_i) \geq +\epsilon \qquad \theta_i = -\,C$$
$$h(\mathbf{x}_i) = +\epsilon \qquad \theta_i \in [-\,C, 0]$$
$$h(\mathbf{x}_i) \in [-\epsilon, +\epsilon] \qquad \theta_i = 0$$
$$h(\mathbf{x}_i) = -\epsilon \qquad \theta_i \in [\,0, +\,C\,]$$
$$h(\mathbf{x}_i) \leq +\epsilon \qquad \theta_i = +\,C \tag{9}$$

The Incremental Support Vector Machine algorithm main purpose is to verify these condition after that a new sample is added.

E. Calculation for Sample Moving

At the beginning of the algorithm, compute *f* and *h*.

$$f(\mathbf{x}_c) = \sum_{i=1}^{l}\ \theta_i\mathbf{Q}_{ic}\ +\ b \tag{10}$$

Margin function is defined as:

$$h(\mathbf{x}_c) = f(\mathbf{x}_c) - y_i \tag{11}$$

When the least variation is found, it is possible to update θ_c, $\theta_i \mid i \in S$, b and $h(\mathbf{x}_i) \mid i \in E \cup R$:

$$\theta_c = \theta_c + \Delta\theta_c \tag{12}$$
$$\theta_i = \theta_i + \Delta\theta_i \tag{13}$$
$$b = b + \Delta b \tag{14}$$

$$h(\mathbf{x}_i) = h(\mathbf{x}_i) + \Delta h(\mathbf{x}_i) \quad (15)$$

$$\begin{bmatrix} \Delta h(\mathbf{x}_{n_1}) \\ \vdots \\ \Delta h(\mathbf{x}_{n_{l_n}}) \end{bmatrix} = \gamma \Delta\theta_c \quad (16)$$

$$\begin{bmatrix} \Delta\beta \\ \Delta\theta_{s_1} \\ \vdots \\ \Delta\theta_{s_{l_s}} \end{bmatrix} = \beta \Delta\theta_c \quad (17)$$

Each iteration, β and γ are useful for least variations calculus:

$$\beta = \begin{bmatrix} \beta_b \\ \beta_{s1} \\ \vdots \\ \beta_{s_{l_s}} \end{bmatrix} = -\mathbf{R} \begin{bmatrix} 1 \\ Q_{s_1 c} \\ \vdots \\ Q_{s_{l_s} c} \end{bmatrix} \quad (18)$$

$$\gamma = \begin{bmatrix} \Delta Q_{n1c} \\ \vdots \\ \Delta Q_{n l_n c} \end{bmatrix} + \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & Q_{n1s1} & \cdots & Q_{n1sl_s} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Q_{nl_n s1} & \cdots & Q_{nl_n sl_s} \end{bmatrix}^{-1} \beta \quad (19)$$

where:

$$\mathbf{R} = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & Q_{s1s1} & \cdots & Q_{s1sl_s} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Q_{sl_s s1} & \cdots & Q_{sl_s sl_s} \end{bmatrix}^{-1} \quad (20)$$

Using Karush-Kuhn-Tucker (KKT) conditions, the samples can be divided into three sets. The goal is to find a way to add a new sample to one of the three sets maintaining KKT conditions consistent.

Support Set

$$S = \{i \mid (-C < |\theta_i| < 0 \wedge h(\mathbf{x}_i) = +\epsilon) \vee (0 < |\theta_i| < C \wedge h(\mathbf{x}_i) = -\epsilon)\} \quad (21)$$

Error Set

$$E = \{i \mid (\theta_i = +C \wedge h(\mathbf{x}_i) < -\epsilon) \vee (\theta_i = -C \wedge h(\mathbf{x}_i) > +\epsilon)\} \quad (22)$$

Remaining Set

$$R = \{i \mid \theta_i = 0 \wedge h(\mathbf{x}_i) < \epsilon\} \quad (23)$$

These following equations are necessary for considering least variation for the samples movement.

When a sample goes from remaining to support set,

$$\Delta\theta_c = \frac{\epsilon - h(\mathbf{x}_i)}{\gamma_i} \text{ or } \frac{-\epsilon - h(\mathbf{x}_i)}{\gamma_i} \quad (24)$$

When a sample goes from support to error set,

$$\Delta\theta_c = \frac{C - \theta_i}{\beta_i} \text{ or } \frac{-C - \theta_i}{\beta_i} \quad (25)$$

When a sample goes from support to remaining set,

$$\Delta\theta_c = \frac{-\theta_i}{\beta_i} \quad (26)$$

When a sample goes from error to support set,

$$\Delta\theta_c = \frac{\epsilon - h(\mathbf{x}_i)}{\gamma_i} \text{ or } \frac{-\epsilon - h(\mathbf{x}_i)}{\gamma_i} \quad (27)$$

III. ISVMM ALGORITHM

After long math proofs and analysis ISVMM, the algorithm is finally shown by pseudo-code format.

Input

1. Training set $\{\mathbf{x}_i, y_i,\ i = 1 \ldots\ldots\ldots l\}$
2. Weights $\theta_i,\ i = 1 \ldots\ldots\ldots l$
3. Bias b
4. Training set partition into Support set (S), Error set (E) and Remaining set (R)
5. Parameters : ϵ, C, Kernel Type and Kernel parameters
6. R Matrix
7. New Sample C = ($\mathbf{x}_c$, y_c)

A. *New Sample Training Pseudo-code*

1. *Add* ($\mathbf{x}_c, y_c$) at the Training set
2. Set $\theta_c = 0$
3. Compute $f(\mathbf{x}_c)$ and $h(\mathbf{x}_c)$
4. If ($|h(\mathbf{x}_c)| < \epsilon$)
   Add New Sample to the R and exit
5. Compute $h(\mathbf{x}_i)$, $i = 1 \ldots\ldots\ldots l$
6. While (New Sample is not Added into a set)
   - 6.1. Update the values β and γ
   - 6.2. Find least variations ($L_{c1}, L_{c2}, \mathbf{L_s}, \mathbf{L_e}, \mathbf{L_r}$)
   - 6.3. Find Min Variations
     $\Delta\theta_c = \min(L_{c1}, L_{c2}, \mathbf{L_s}, \mathbf{L_e}, \mathbf{L_r})$
   - 6.4. Let Flag the case number that determinates $\Delta\theta_c$
     ($L_{c1} = 1, L_{c2} = 2, \mathbf{L_s} = 3, \mathbf{L_e} = 4, \mathbf{L_r} = 5$)
   - 6.5. Let $\mathbf{x}_1$ the sample that determines $\Delta\theta_c$
   - *6.6.* Update $\theta_c, \theta_i, i = 1 \ldots\ldots\ldots l$ *and* b
   - 6.7. Update $h(\mathbf{x}_i)$, i ϵ E U R
   - 6.8. Switch Flag
     - 6.8.1 (Flag = 1)
       Add New Sample to S
       Add New Sample to **R** Matrix and Exit
     - 6.8.2 (Flag = 2)
       Add New Sample to E and Exit
     - 6.8.3 (Flag = 3)
       If ($\theta_i = 0$)
       Move Sample l from S to R
       Remove Sample l from **R** Matrix
       Else [$\theta_i = |C|$]
       Move Sample l from S to E
       Remove Sample l from **R** Matrix
     - 6.8.4 (Flag = 4)
       Move Sample l from E to S
       Add Sample l from **R** Matrix
     - 6.8.5 (Flag = 5)
       Move Sample l from R to S
       Add Sample l to **R** Matrix

B. Output

1. Training set $\{ \mathbf{x}_i, y_i, \ \mathrm{i} = 1 \ldots \ldots \ldots l\}$
2. New Coefficients $\theta_i, \ i = 1 \ldots \ldots \ldots . l + 1$
3. New Bias b
4. New Training set partition
5. New **R** matrix

C. L_{C1} Variation

This first variation considered is that new sample of $| h (\mathbf{x}_i)|$ reaches $| \epsilon |$ and consequently the Support set.

D. L_{C2} Variation

This second variation measures the distance of the new sample from the Error set.

E. L_s Variation

This third variation measures the distance of each Support sample to Error or Remaining set. The Sample's direction depends on the direction of the new sample and by the value of β_i . If it is negative, the sample goes in the opposite direction.

F. L_E Variation

The fourth variation measures the distance of each error sample to Support Set. The samples' direction depends on the direction of the new sample and by the value of γ_i. If it is negative, the sample goes in the opposite direction.

G. L_R Variation

The last variation measures the distance of each remaining sample to Support set. The samples' direction depends on the direction of the new sample and by the value of γ_i. If it is negative, the sample goes in the opposite direction.

H. Stabilization Procedure

For each sample *S* in the training set, if *S* doesn't verify KKT conditions, the algorithm will forget *S* and train *S* again.

TABLE II
PREDICTION RESULTS OF ISVMM APPLIED ON INTRUSION PREVENTION

Mean Square Error (MSE) of algorithm using Radial Basis Function kernel (RBF): $\mathbf{K}(\mathbf{x}, \mathbf{x}') = \exp(-\boldsymbol{\gamma} \|\mathbf{x} - \mathbf{x}'\|^2)$, for $\boldsymbol{\gamma} > 0$		
ISVMM		**OnlineSVR** Result
Covariance matrix variable (α)	Result	
0.9	2.2787	
0.8	2.2763	
0.7	2.2746	
0.6	2.2737	
0.5	2.2736	**2.2819**
0.4	2.2736	
0.3	2.2729	
0.2	**2.2711**	
0.1	2.2725	

IV. EXPERIMENTAL RESULTS

To test the classification performance of ISVMM, it was applied on Intrusion Prevention by employing the KDD dataset. The attack types were assigned into DOS- 1, R2L- 2, U2R- 3, Probing- 4 and Normal- 5 according to their categories. Moreover, the protocol types were assigned to tcp- 1, udp- 2, icmp-3 and smtp-4 and also all of the other nominal values such as service, and flag were assigned to each numeric value respectively. For preventing the loss of information, we used all of the 41 features of these datasets without removing any dimension. For implementation, 3882 samples were randomly selected for training and 1957 for testing. There is completely without overlapping the data with the training and testing dataset. To demonstrate the performance as shown in Table II, we compared the experimental results of ISVMM against with the similar algorithm, OnlineSVR which uses Euclidean distance. For this experiment, we set the value of user definable parameter of C, Epsilon, kernel type and kernel parameter to be 10, 0.1, RBF and 30 respectively.

ACKNOWLEDGMENT

The author would like to express our special thanks to Francesco Parrella for providing experienced guidance, and supporting us in all stages for building this new algorithm.

REFERENCES

[1]. KDD'99 datasets, The UCI KDD Archive, Irvine, CA, USA, 1999. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[2]. F. Parrella, "Online support vector regression," *Department of Information Science, University of Genoa, Italy,* June 2007, http://onlinesvr.altervista.org/.

[3]. Gail A. Carpenter and S. Grossberg, "*Adaptive resonance theory*," Cambridge, MA: MIT Press, April 2002.

[4]. H. Xu and M. Vuskovic, "Mahalanobis distance-based ARTMAP Network," *IEEE Trans.,* 0-7803-8359, 2004.

[5]. Gail A. Carpenter, S. Grossberg, N. Markuzon, John H. Reynolds, and David B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *EEE Trans. Neural Netw*., vol. 3, no. 5, Sept. 1992.

[6]. S. Sodsee, M. Komkhao, P. Meesad, "An incremental learning algorithm based on mahalanobis distance for breast cancer diagnosis", *Department of Information Technology, Faculty of IT, King Mongkut's University of Technology*, 2006.

[7]. D. Wang, Daniel S. Yeung and Eric C. C. Tsang, "Weighted mahalanobis distance kernels for Support Vector Machines", *IEEE Trans. Neural Netw.,* vol. 18, No. 5, pp 1453-1562, September 2007.

[8]. Corinna Cortes and V. Vapnik, "Support-Vector Networks", *Machine Learning, SpringerLink,* ISSN 0885-6125, pp.273-297, 1995.

[9]. Kasuba, T, "Simplified fuzzy ARTMAP." *AI Expert*, Nov., pp. 18–25, 1993.

[10]. Ma, J. Theiler,S. Perkins, "Accurate on-line support vector regression", Neural Computation, vol.15, pp.2683–2703, 2003.

[11]. Alex J.Smola and B Scholkope, "A tutorial on support vector regression", *Kluwer Academic Publishers, Statistics and Computing,* vol.14, pp.199–222, 2004.

[12]. W.Lee, Salvatore J. Stolfo "A Framework for constructing features and models for intrusion detection systems", *ACM Transl. Information and System Security*, vol.3, No. 4, pp. 227-261, Nov. 2001.