Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 34 (2008) 2013–2018
Expert Systems with Applications
www.elsevier.com/locate/eswa

An algorithm to cluster data for efficient classification of support vector machines

Der-Chiang Li *, Yao-Hwei Fang

Department of Industrial and Information Management, National Cheng Kung University, 1, University Road, Tainan 701, Taiwan

Abstract

Support vector machines (SVM) are widely applied to various classification problems. However, most SVM need lengthy computation time when faced with a large and complicated dataset. This research develops a clustering algorithm for efficient learning. The method mainly categorizes data into clusters, and finds critical data in clusters as a substitute for the original data to reduce the computational complexity. The computational experiments presented in this paper show that the clustering algorithm significantly advances SVM learning efficiency.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Density-based clustering algorithm; Machine learning; Support vector machines; Computational complexity

1. Introduction

On the basis of statistical learning theory, support vector machines (SVM) were proposed by Corinna and Vladimir (1995) for classification and regression purposes. They have been used in a wide range of applications, such as pattern recognition (Shigeo, 2005), test categorization (Edda & Jörg, 2002), bioinformatics (Komura, Nakamura, Tsutsumi, & Aburatani, 2005), and most financial forecasting (Shin, Lee, & Kim, 2005), etc.

Basically, an SVM builds an optimal geometric hyper-plane to separate the data into classes. To explain this mathematically, consider $\mathbf{x} \in \mathbb{R}^k$ and $y \in \{-1,1\}$, the hyper-plane function is presented as:

$$y = f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x}) + b$$

where $\mathbf{w} \in \mathbb{R}^k$ is the weight vector, $b \in \mathbb{R}$ is the bias. The function $\phi(\mathbf{x})$ maps a vector $\mathbf{x}$ into a higher dimension space (feature space) wherein they can be classified linearly. The optimal hyper-plane is obtained using the following optimization model:

$$\begin{aligned} &\text{minimize}_{\xi,\mathbf{w},b} \quad \langle w \cdot w \rangle + C\sum_{i=1}^{n} \xi_i \\ &\text{subject to} \quad y_i(\langle \mathbf{w} \cdot \phi(\mathbf{x}_i)\rangle + b) \geqslant 1 - \xi_i, \quad i = 1,\ldots,n \\ &\qquad\qquad\quad \xi_i \geqslant 0, \quad i = 1,\ldots,n \end{aligned}$$

When facing a large and complicated dataset, the accuracy of SVM classification is not as high as expected and the computation time increases rapidly. Therefore, improving the efficiency of SVM is one important area of study.

The computational complexity of an SVM depends on the number of support vectors used in the procedure (Tran, Zhang, & Li, 2003). The KM-SVM (Marcelo, Antônio, & João, 2000; Tran et al., 2003; Yang et al., 2003) is a method proposed to reduce the number of support vectors, and thus, increase computational efficiency. This method uses K-means clustering technique to assign the data of each class to k clusters, and trains the SVM using the new dataset consisting of only the centrals of clusters. The value of k is thus, the upper bound of the number of support vectors in each class. Experimental results demonstrate that the method improves the computational efficiency for an

* Corresponding author. Tel.: +886 6 2757575x50501; fax: +886 6 2374252.
E-mail addresses: lidc@mail.ncku.edu.tw (D.-C. Li), yaohwei_fang@hotmail.com (Y.-H. Fang).

0957-4174/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2007.02.016

SVM, but there is a trade-off between classifying efficiency and learning accuracy. In addition, the difficulty that exists with the KM-SVM method is that one needs to determine the k value by basically a trial-and-error approach. This work is time-consuming.

To overcome this shortcoming, this research used density-based spatial clustering of applications with noise (DBSCAN) as the base to develop a new algorithm, called C-SVM algorithm, to cluster the data for an SVM. This algorithm uses only centrals of clusters as a substitute for the original data in order to downsize the training dataset. Similar to most existing cluster algorithms, the DBSCAN algorithm is designed to cluster data of only one kind (one-class), and consequently no consideration is given to data purity for clusters. Uniquely, for the purpose of data classification, the proposed algorithm in this study creates two special parameters for enhancing the clustering procedure:Data Purity Level and Data Sufficiency, as explained in Section 3.

2. A brief introduction of DBSCAN

DBSCAN is suitable for a large amount of data of one class with high dimensions (Jiawei & Micheline, 2001; Martin, Kriegel, Jörg, & Xiaowei, 1996). DBSCAN gathers together high density data as a cluster and the shape of a cluster is arbitrary. DBSCAN mainly finds the clusters and deletes the data not belonging to any cluster. DBSCAN searches for clusters by checking the surroundings of each data within a scope called the ε-neighborhood. If the ε-neighborhood of a data contains more than a certain pre-defined number, parameter MinPts, a cluster with this data (called the **core object**) is created; otherwise, the data is treated as noise which will be deleted eventually. The DBSCAN iteratively collects **directly density-reachable** (within the ε-neighborhood of a core object) data until no new data can be added to any cluster, and this may involve merging some clusters.

3. The approach

The clustering algorithm presented in this paper is designed uniquely for an efficient classification. There are two key breakthroughs considered in this clustering procedure, as explained below.

3.1. Data Purity Level

It is believed that in the work of clustering data for classification, the complexity extent of a data set should have a signiticant effect on the determination of the cluster size. This research developed an effective index, the Purity Level, to measure data complexity. The parameters of the index are defined as follows:

n: the number of data and $n \geqslant 1$,
k: the number of attributes of data and $k \geqslant 2$,
A_{ij}: the value of j-th attribute of i-th data,
x_i^+, x_i^-: the data belongs to the positive and negative class, respectively,
$\overline{A}_j^+, \overline{A}_j^-$: the average value of the j-th attribute of the data in the positive and negative class, respectively and
$A_{j\max}, A_{j\min}$: the maximum and the minimum value of the j-th attribute, respectively.

Applying the parameters listed above, a couple of equations are formed:

$$M_a(x_i^+) = \sqrt{\frac{\sum_{j=1}^{k}\left(\frac{A_{ij}-\overline{A}_j^+}{A_{j\max}-A_{j\min}}\right)^2}{k-1}}$$

where $M_a(x_i^+)$ stands for the concept of within-class distance of data in the positive class.

$$M_a(x_i^-) = \sqrt{\frac{\sum_{j=1}^{k}\left(\frac{A_{ij}-\overline{A}_j^-}{A_{j\max}-A_{j\min}}\right)^2}{k-1}}$$

where $M_a(x_i^-)$ stands for the concept of within-class distance of data in the negative class.

$$M_a(S) = \sum_{i=1}^{n}\left(M_a(x_i^+) + M_a(x_i^-)\right)$$

where $M_a(S)$ stands for the sum of within-class distance for the whole data set.

$$M_b(x_i^+) = \sqrt{\frac{\sum_{j=1}^{k}\left(\frac{A_{ij}-\overline{A}_j^-}{A_{j\max}-A_{j\min}}\right)^2}{k-1}}$$

where $M_b(x_i^+)$ stands for the concept of between-class distance of data in the positive class to the center of data in the negative class.

$$M_b(x_i^-) = \sqrt{\frac{\sum_{j=1}^{k}\left(\frac{A_{ij}-\overline{A}_j^+}{A_{j\max}-A_{j\min}}\right)^2}{k-1}}$$

where $M_b(x_i^-)$ stands for the concept of between-class distance of data in the negative class to the center of data in the positive class.

$$M_b(S) = \sum_{i=1}^{n}\left(M_b(x_i^+) + M_b(x_i^-)\right)$$

where $M_b(S)$ stands for the sum of between-class distance for the whole data set.

Finally, the Purity Level is set as:

$$\text{PurityLevel} = \frac{M_b(S)}{M_a(S)}$$

When the value of $M_a(S)$ is large, this means that the data in the same class are more dispersive, and when $M_a(S)$ is small, this means that the data in the same class are condensed; when the value of $M_b(S)$ is large, it means

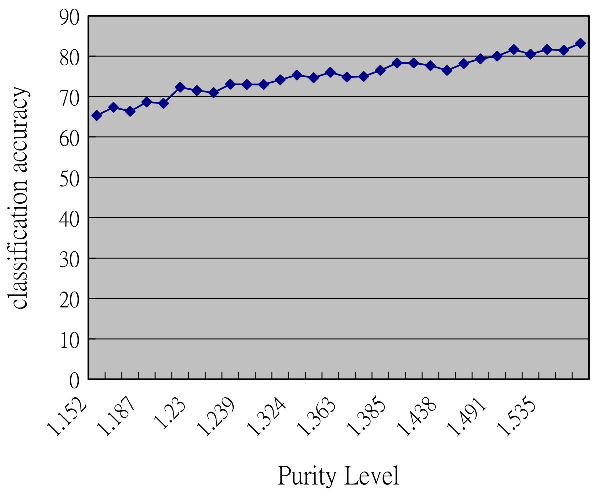

Fig. 1. The relation between the Purity Level and classification accuracy.

that the data in the different classes are more separate, and when $M_b(S)$ is small, it means that the data in the different classes are closer. Accordingly, the Purity Level has this relationship with data complexity: the smaller the Purity Level value, the higher is the data complexity, and vice versa.

According to our experience, Purity Level has a positive relationship with the classification accuracy of SVM (Kernel Function: RBF, $C = 1$). This is shown in Fig. 1:

When the dataset is not complex (i.e., the Purity Level is large), it usually means that the data are geometrically separable or easily classified. So the ε parameter (the scope of cluster) in the C-SVM should be set large in order to include more data into a cluster to improve the efficiency of an SVM. On the other hand, when the dataset is complex (the Purity Level is small), it means that the data are not separated geometrically. In other words, many misclassifications of data may occur in each class. So the parameter of ε should be set small in order to exclude noises from clusters.

3.2. The degree of data sufficiency (m/V)

We defined the degree of Data Sufficiency as m/V, where $V = \prod_j \text{range}(x_j)$ for $j = 1,\ldots,k$ stands for data range, and m is the amount of total data. When the dataset is insufficient, it theorefically means that the density of data (m/V) is small. Usually data collected in the initial stages of manufacturing systems are insufficient, meaning that the characteristic of the data is unripe. So, every additional data is more important and valuable in these early stages. In this case, the parameter of ε in the C-SVM should be set large to tolerate more data. In other words, the definition of a noise is set loosely this time. With this situation, when the dataset is sufficient, it means that the density of data (m/V) is large, and the parameter of ε in the C-SVM should be set small for each cluster for a precise representation of each.

3.3. Cluster radius

For this reason, when circling data into clusters, this research utilizes both Purity Level and Data Sufficiency (m/V) to determine radius ε, and radius ε is the principal consideration of the area size of a hypersphere. By apply the theorem of area of the hypersphere, the following is the formula set for ε:

$$\varepsilon_+ = \sqrt[k]{\frac{(p/m_+)V_+(\text{PurityLevel})\Gamma(k/2+1)}{\sqrt{\pi^k}}} \times \left(\frac{m}{V}\right)^{-1}$$

$$\varepsilon_- = \sqrt[k]{\frac{(p/m_-)V_-(\text{PurityLevel})\Gamma(k/2+1)}{\sqrt{\pi^k}}} \times \left(\frac{m}{V}\right)^{-1}$$

where ε_+ and ε_- are the radii used for the positive and negative class, repectively, k is the dimension of data, p is the number of MinPts (minimum points), Γ is the gamma function, m_+ and m_- are the amount of data in the positive and negative class, repectively, $V_+ = \prod_j \text{range}(x_j^+)$, and $V_- = \prod_j \text{range}(x_j^-)$ for $j = 1,\ldots,k$ are the data range in the positive and negative class, respectively.

3.4. The proposed procedure

For a binary classification problem, the C-SVM model is described in steps as follows:

Step 1: Find the Purity Level, Data Sufficiency (m/V), data range V, and the amount of total data m for the training data set.
Step 2: Calculate V_+, V_-, m_+, and m_- for each class of the training data set.
Step 3: Set up the number of MinPts p.
Step 4: Use the result of steps 1 to 3 to calculate parameters ε_+ and ε_- of the C-SVM.
Step 5: Use ε_+,ε_- and p to construct the clusters and erase the noise in the data set (a cluster with fewer elements than p).
Step 6: Figure out the centers of the clusters and substitute the centers for the original data.
Step 7: Apply SVM to train the critical data (centers) and test the learned model using the testing data.

4. Numerical experiment

This example is applied here to explain the C-SVM procedure and also used to compare the accuracies among the original SVM, KM-SVM, and C-SVM.

4.1. Data generation

The simulation dataset contains 600 data, of which 300 belong to the positive class and 300 to the negative class, the dataset contains two independent numerical attributes, and missing datum is assumed to not exist.

The simulated data set has a structure in which each class has three clusters. The positive class is designed with normal distribution of center (1, 9) with s.d. (standard deviation) 1, center (1, 6) with s.d. 0.6, and center (1, 4) with s.d. 0.3. The negative class is designed with normal distribution of center (2, 7) with s.d. 1, center (2, 4) with s.d. 0.6, and center (2, 9) with s.d. 0.3. The noise part contains 1%, 1.5%, 2% and 4% uniformly distributed noise with range (0,15) for each class.

We randomly select 400 data from the dataset as the training data and use the other 200 data as the testing data.

4.2. Experiment results

This research conducts five experiments for each noise level dataset and sets MinPts = 4 (a number used in the computer program which will be discussed in the conclusions). It records the average Purity Level, different kernel functions (linear, polynomial, RBF, sigmoid), the average accuracy of the original SVM, KM-SVM, and C-SVM, while the LIBSVM2.6 software (Chang & Lin, 2001; Hsu, Chang, & Lin, 2003) is used for the SVM model. The results are summarized in Tables 1–4.

In the four kinds of noise datasets, we simply set the k value of KM-SVM equal to the number of clusters found in C-SVM analysis (note that setting k values in the KM-SVM is usually time-consuming).

With regard to the training time, the number of training data of the original SVM is 400, and the number of training data in C-SVM is 10 to 20, so the process can improve the training time of SVM in C-SVM.

With regard to the prediction time, which is proportional to the number of support vectors, the original SVM is between 200 and 400. The number of support vectors of C-SVM and KM-SVM is between 5 and 20. The prediction time for C-SVM and KM-SVM are less than that for the original SVM by 10–40 times.

Though the learning accuracy is not the research target, it may also be worth noting that the accuracy of C-SVM is better then KM-SVM in all conditions, and is also better then the original SVM when the kernel function is sigmoid. However, when the kernel functions are linear, polynomial and RBF, the accuracies of C-SVM is inferior to the original the SVM.

Table 1
The experiment results of datasets with 1% noise

Datasets with 1% noise (average Purity Level: 1.04101)			
Method	Kernel function	Average accuracy	Average number of support vectors
Original SVM	Linear	72.02	241.2
	Polynomial	71.81	221.6
	RBF	74.94	208.4
	Sigmoid	49.59	399.2
KM-SVM	Linear	56.06	10.6
	Polynomial	60.31	9.8
	RBF	62.92	14.4
	Sigmoid	44.46	13.2
C-SVM	Linear	63.56	8.4
	Polynomial	61.64	8.8
	RBF	62.75	9.6
	Sigmoid	50.91	15.2

Table 2
The experiment results of datasets with 1.5% noise

Datasets with 1.5% noise (average Purity Level: 1.3392)			
Method	Kernel function	Average accuracy	Average number of support vectors
Original SVM	Linear	71.41	255.6
	Polynomial	70.90	238
	RBF	72.42	232.2
	Sigmoid	51.01	399.2
KM-SVM	Linear	58.98	9.6
	Polynomial	58.78	9.4
	RBF	61.71	14.6
	Sigmoid	50.50	14
C-SVM	Linear	67.16	8.4
	Polynomial	62.83	6.6
	RBF	62.93	9
	Sigmoid	50.00	14

Table 3
The experiment results of datasets with 2% noise

Datasets with 2% noise (average Purity Level: 1.04101)			
Method	Kernel function	Average accuracy	Average number of support vectors
Original SVM	Linear	69.59	242.2
	Polynomial	69.79	223.4
	RBF	72.42	214.6
	Sigmoid	48.38	399.6
KM-SVM	Linear	52.12	9
	Polynomial	61.81	10.6
	RBF	65.95	15.4
	Sigmoid	48.28	12.4
C-SVM	Linear	70.51	6.8
	Polynomial	65.16	7.2
	RBF	67.38	8.6
	Sigmoid	55.06	14.4

Table 4
The experiment results of datasets with 4% noise

Datasets with 1% noise (average Purity Level: 1.04101)			
Method	Kernel function	Average accuracy	Average number of support vectors
Original SVM	Linear	69.59	276.4
	Polynomial	69.79	251
	RBF	72.42	237.8
	Sigmoid	48.38	399.6
KM-SVM	Linear	56.06	9
	Polynomial	64.24	8.4
	RBF	66.06	13
	Sigmoid	54.14	10.8
C-SVM	Linear	67.08	7.8
	Polynomial	66.07	6.6
	RBF	67.08	9
	Sigmoid	52.03	12

Table 5
The experiment results of data from the Pima database

Pima database (average Purity Level: 1.12676)			
Method	Kernel function	Average accuracy	Average number of support vectors
Original SVM	Linear	76.96	359.4
	Polynomial	65.10	429.4
	RBF	75.91	412.2
	Sigmoid	70.70	430.8
KM-SVM	Linear	70.58	30.4
	Polynomial	70.19	40.2
	RBF	69.93	42.2
	Sigmoid	68.49	30.6
C-SVM	Linear	74.25	17.2
	Polynomial	70.99	21
	RBF	71.51	23
	Sigmoid	74.51	15.8

4.3. Case example

Pima Indians Diabetes Database (Available at: http://www.ics.uci.edu/~mlearn/MLSummary.html) has a total of 768 data, and each data has 8 attributes. There is no missing value in all the data. Because the units of the attributes are different, the data is normalized before analyzing. This experiment sets MinPts = 2 and records the average Purity Level, different kernel functions, the average accuracy of the original SVM, KM-SVM and C-SVM. All experiments are under five-fold cross validation and Table 5 describes the results of this case analysis.

In the Pima case, we also set the k value of KM-SVM equal to the number of clusters in the C-SVM analysis.

With regard to the training time, the number of training data of the original SVM is 614 and the average number of training data in C-SVM is 89.6, so training time is improved with SVM in C-SVM.

With regard to the prediction time, the number of support vectors of the original SVM is between 350 and 450. The number of support vectors of C-SVM is between 15 and 25, and the number of support vectors of C-SVM is smaller than that of KM-SVM. The predicted time for C-SVM is less than the original SVM by 15 to 30 times, so it can obviously improve the prediction time of SVM.

With regard to the accuracy, when the kernel functions are linear and RBF, the accuracies of the original SVM are better than C-SVM and KM-SVM. When the kernel functions are polynomial and sigmoid, the accuracies of C-SVM are better than the original SVM.

5. Conclusions and suggestions

This research proposed an improved method of clustering for classification called C-SVM. Facing a large and complicated dataset, the original SVM has lengthy computation time, whether for training or predicting. C-SVM is developed to decrease the number of data required, thus improving the training time of SVM. The number of support vectors is also reduced, and this reduces the prediction time by 10 to 40 times in the experiments.

The prediction time of SVM is affected by the number of support vectors. The number of data after applying C-SVM is small, so the number of support vectors is less and the prediction time is shorter. Mathematically, in the calculation of the computational complexity of the training time, let the original number of data be n_1 and the number of data after C-SVM be n_2, and $n_1 \geqslant n_2$. The time complexity C-SVM can be fomulated as $O(n_1 \log n_1) + O((n_2)^5 \log n_2/\varepsilon_{n_2})$, where the time complexity of DBSCAN is $O(n_1 \log n_1)$ (Martin et al., 1996), and the time complexity of SVM is $O((n_2)^5 \log n_2/\varepsilon_{n_2})$ (Don & Clint, 2003). Therefore, we can form the computational complexity as:

$$\begin{aligned} &O(n_1 \log n_1) + O((n_2)^5 \log n_2/\varepsilon_{n_2}) \\ &\quad = O(n_1 \log n_1) + O((n_2)^5 \log n_2^{1/\varepsilon_{n_2}}) \\ &\quad \leqslant O(n_1 \log n_1) + O((n_1)^5 \log n_1^{1/\varepsilon_{n_1}}) \\ &\quad = \max\left\{O(n_1 \log n_1), O((n_1)^5 \log n_1^{1/\varepsilon_{n_1}})\right\} \\ &\quad = O((n_1)^5 \log n_1^{1/\varepsilon_{n_1}}) = O((n_1)^5 \log n_1/\varepsilon_{n_1}) \end{aligned}$$

from the above equation, C-SVM is theoritically faster than the original SVM.

The research also investigated the accuracies of the original SVM, KM-SVM, and C-SVM. Based on the results, the performance of C-SVM is better than KM-SVM. Furthermore, the difficulty of determining k value, the drawback of KM-SVM, is eliminated by C-SVM. In addition, another drawback of KM-SVM is that the results become unstable due to the variation of the chosen initial centers. Through the proposed procedure, the results that C-SVM attains are much more stable from the change of radius ε_+ and ε_-.

In our experience, the setting of MinPts in C-SVM affects the classified performance. We set MinPts = 2, because using a large MinPts may yield results similar to that using small ε in performance of C-SVM. Therefore, it seems advisable to fix MinPts = 2 and concentrate attention on the control of the value of parameter ε. However, analysis on MinPts and ε are considered for further research in future.

There is still a difficulty when applying nominal attributes to a problem, thus C-SVM for nominal attributes can also be a new focus of future research.

References

Chang, C. C. & Lin, C. J. (2001). LIBSVM: A library for support vector machines. Available at: <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.

Corinna, C., & Vladimir, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 273–297.

Don, H., & Clint, S. (2003). Polynomial-time decomposition algorithms for support vector machine. *Machine Learning, 51*(1), 51–71.

Edda, L., & Jörg, K. (2002). Text categorization with support vector machines. how to represent texts in input space? *Machine Learning, 46*(1–3), 423–444.

Hsu, C. W., Chang, C. C. & Lin, C. J. (2003). A practical guide to support vector classification. Available at: <http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>.

Jiawei, H., & Micheline, K. (2001). *Data mining: Concepts and techniques*. San Francisco: Morgan Kaufmann Publishers.

Komura, D., Nakamura, H., Tsutsumi, S., & Aburatani, H. (2005). Multidimensional support vector machines for visualization of gene expression data. *Bioinformatics, 21*(4), 439–444.

Marcelo, B. D. A., Antônio, D. P. B. & João, P. B. (2000). SVM-KM: Speeding SVMs learning with a priori cluster selection and k-means. *Neural Networks, 2000*. In: *Proceedings of the sixth Brazilian symposium*, November 22–25, pp. 162–167.

Martin, E., Kriegel, H. P., Jörg, S. & Xiaowei, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of second international conference on knowledge discovery and data mining*, Portland, pp. 226–231.

Shin, K. S., Lee, T. S., & Kim, H. J. (2005). An application of support vector machines in bankruptcy prediction model. *Expert Systems with Applications, 28*(1), 127–135.

Shigeo, A. (2005). *Support vector machines for pattern classification*. London: Springer.

Tran, Q. A., Zhang, Q. L. & Li, X. (2003). Reduce the number of support vectors by using clustering techniques. In: *Proceedings of the second international conference on machine learning and cybernetics*, November 2–5, pp. 1245–1248.

Yang, X., Lin, D., Hao, Z., Liang, Y., Liu, G., & Han, X. (2003). A fast SVM training algorithm based on the set segmentation and k-means clustering. *Progress in Natural Science, 13*(10), 750–755.