Fast Support Vector Data Description Using K-Means Clustering

Pyo Jae Kim, Hyung Jin Chang, Dong Sung Song, and Jin Young Choi

School of Electrical Engineering and Computer Science,
Automation and Systems Research Institute, Seoul Nat'l University,
San 56-1 Shillim-dong, Kwanak-ku Seoul 151-744, Korea
{pjkim,hjchang,dssong,jychoi}@neuro.snu.ac.kr
http://iccl.snu.ac.kr

Abstract. Support Vector Data Description (SVDD) has a limitation for dealing with a large data set in which computational load drastically increases as training data size becomes large. To handle this problem, we propose a new fast SVDD method using K-means clustering method. Our method uses *divide-and-conquer* strategy; trains each decomposed subproblems to get support vectors and retrains with the support vectors to find a global data description of a whole target class. The proposed method has a similar result to the original SVDD and reduces computational cost. Through experiments, we show efficiency of our method.

1 Introduction

Classification tasks, such as detecting a fault in the machine, image retrieval and surveillance system, need different approaches from general pattern classification or regression methods. These problems are called *one-class problems*, and the solutions have to aim for describing boundaries of each classes. Based on these boundaries, the classifier makes a decision whether the testing object is in a target class or not.

SVDD [1] is a well-known one class classification algorithm. This method finds a boundary of a target class in data space by assuming a hypersphere which has minimum volume enclosing almost all target class objects in feature space. SVDD uses *support vectors* to describe the boundary of target class as Support Vector Machine(SVM) [2] does. *Support vectors* are found by solving convex quadratic programming (QP).

Although QP has an advantage of avoiding a local minimum problem, it requires severe computation as the number of training objects increase. This problem has been known as 'Scale problem'. There is much literature for reducing computational cost. Those suggested studies can be roughly categorized into two groups.

One group is mainly focused on solving QP quickly. Chunking [3], decomposition [4] algorithms and Sequential Minimal Optimization (SMO) [5] are in this group. The chunking and decomposition methods break down a large problem into a series of small problems. By discarding non *support vector* data in small

D. Liu et al. (Eds.): ISNN 2007, Part III, LNCS 4493, pp. 506–514, 2007.

problems, the algorithms are able to train fast and reduce memory space. SMO is a more advanced version of decomposition method.

The other group is based on the *divide-and-conquer* strategy dealing with large scale data problems [6], [7], [8]. The large data set is decomposed into several simple sub-problems similar to decomposition methods, but each sub-problem is solved by its own local expert. Thus such approaches need additional decision rules to combine each local expert's decisions or to decide one of local experts to new test object. This decision rules cause additional computational cost. Parallel mixture of SVMs [6] and Bayesian Committee Support Vector Machine (BC-SVM) [7] can be the examples of this group.

In this paper, we propose a new method for a scale problem of SVDD. Our method is not aimed to solve QP quickly, but based on the *divide-and-conquer* strategy. We decompose a large data set into a series of small sub-data groups using K-means clustering algorithm [9]. Each small sub-problem is solved by its own local expert with SVDD, but it needs not an additional decision rule unlike the existing approaches. Instead of the decision rule, our method retrains using only *support vectors* which are found by each local experts. Through training decomposed data and retraining *support vectors* only, a global solution can be found faster than the original SVDD. Because our approach does not rely on the performance of a QP solver, it has a chance to be improved when faster QP solver (such as SMO) is used.

The proposed method is applied to data sets which are various in size and shape. Comparing with the original SVDD, we show our method has similar data description results with less computational load.

2 KMSVDD: Support Vector Data Description Using K-Means Clustering

In this section, we give detail explanation of the proposed method. We present the basic theory of SVDD and then introduce a model of KMSVDD.

2.1 Basic Theory of Support Vector Data Description

SVDD is similar to the hyperplane approach of Schölkopf et al. [10] which estimates decision plane to separate the target objects with maximal margin. However, instead of using a hyperplane, SVDD uses a minimum hypersphere to find an enclosed boundary containing almost all target objects [1]. Assume a hypersphere with center $\mathbf{a}$ and radius R. So the cost function is defined as below:

$$F(R, \mathbf{a}) = R^2 + C \sum_i \xi_i \ , \tag{1}$$

where ξ_i are slack variables $\xi_i \geq 0$, and the parameter C controls the trade-off between the volume and the errors [1]. Also (1) should be minimized under the following constraints:

$$\|\mathbf{x}_i - \mathbf{a}\|^2 \leq R^2 + \xi_i, \qquad \xi_i \geq 0, \quad \forall i \ . \tag{2}$$

Constraints can be incorporated into the cost function by using Lagrange multipliers:

$$L(R, \mathbf{a}, \alpha_i, \gamma_i, \xi_i) = R^2 + C \sum_i \xi_i \qquad (3)$$
$$- \sum_i \alpha_i \{R^2 + \xi_i - (\|\mathbf{x}_i\|^2 - 2\mathbf{a} \cdot \mathbf{x}_i + \|\mathbf{a}\|^2)\} - \sum_i \gamma_i \xi_i,$$
$$\alpha_i \geq 0, \quad \gamma_i \geq 0 \ .$$

L should be minimized with respect to R, $\mathbf{a}$, ξ_i and maximized with respect to α_i, γ_i. After setting partial derivatives of L to zero, solve each equation with a QP solver. Those objects $\mathbf{x}_i$ with $0 < \alpha_i \leq C$ are called *support vectors*, and used to describe a boundary description. Especially, objects $\mathbf{x}_i$ with $\alpha_i = C$ are called *outliers*.

To test whether an object $\mathbf{z}$ is within the hypersphere, the distance to the center of the hypersphere is used:

$$\|\mathbf{z} - \mathbf{a}\|^2 = \mathbf{K}(\mathbf{z}, \mathbf{z}) - 2 \sum_i \alpha_i \mathbf{K}(\mathbf{z}, \mathbf{x}_i) + \sum_{i,j} \alpha_i \alpha_j \mathbf{K}(\mathbf{x}_i, \mathbf{x}_i) \quad \leq R^2 \ , \qquad (4)$$

where $\mathbf{K}$ is a kernel function to solve non-separable problems. When the distance is equal to or smaller than the radius R, the test object $\mathbf{z}$ is accepted.

2.2 KMSVDD

KMSVDD, SVDD algorithm using K-means clustering, is described in Fig. 1. It can be summarized as the following three steps.

Step 1. Decompose a data set using K-means clustering

Assume k center points and partition a training data set into k sub-problems using K-means clustering algorithm.

Because a retraining process exists after this step, clustering quality is not a key factor of the proposed method's performance. Other clustering methods such as LVQ can be used, but we select K-means clustering algorithm because it has good performance with less computational load.

Step 2. Get local data descriptions using SVDD

Execute individual training on each k sub-problems.

As a result of training, k sub-problems' local descriptions and *support vectors* representing each local descriptions are obtained. We newly define a *working set* composed of these *support vectors* and will use it for finding a global data description in the next **Step 3**. Because many *inliers* which are within the hypersphere do not affect making a final data description [2], so they can be ignored and exclude from *working set*.

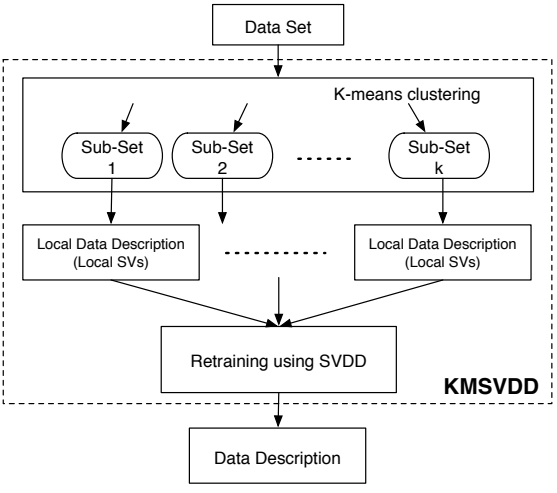

Fig. 1. The scheme of KMSVDD algorithm

Step 3. Retrain with working set using SVDD

To get a global description of a training data set, retrain only with *support vectors* which are obtained by **Step 2**.

In general, k sub-problems' local descriptions have lots of confronting sides with neighbors. It means that some *support vectors* are overlapped and generated uselessly so the description results are far from the result of using only the original SVDD. Through retraining process, useless *support vectors* (truly *inliers* of a target class) will be removed. Therefore the retrained result can be similar to that of the original SVDD.

The proposed algorithm adds two additional steps to the original SVDD: clustering and retraining step. So two learning parameters which user has to determine are added. One is the cluster number k and the other is the kernel parameter used in SVDD learning of **Step 2, 3**.

Learning process of the proposed algorithm can be visualized as Fig. 2. Although, the number of training data is same, training k sub-problems is much faster than using whole data at a time.

2.3 Computational Complexity of KMSVDD

A general QP solver runs in $O(N^3)$ time, where N is the size of training data. We are not improving the performance of a QP solver, but cutting down the size of training data for each QP solver. So the complexity of KMSVDD with k clusters is

$$O(kN) + kO((N/k)^3) + O((\alpha k)^3) \approx kO((N/k)^3) \ , \tag{5}$$

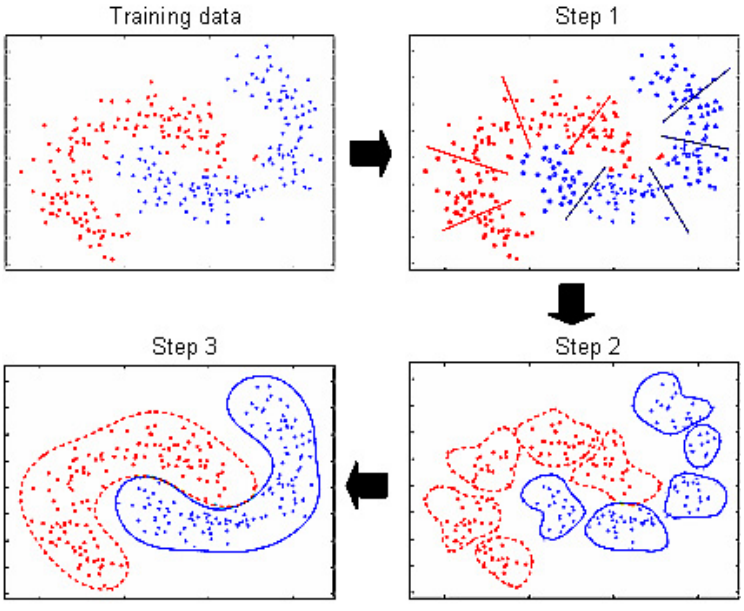

Fig. 2. Visualization of KMSVDD learning process. KMSVDD has three steps and gets a global data description finally.

where k is the cluster number and α is the average number of *support vector* for each sub-problems' description, so the size of *working set* is αk. The first term $O(kN)$ is for the complexity of K-means clustering algorithm. The second is for the complexity of local SVDD QP solver, provided the training data N is equally partitioned into k sub-problems. The third term $O((\alpha k)^3)$ is for retraining process. Generally under the sparse solution conditions, the size of *working set* αk is smaller than the initial size of training data N. So the influence of the new complexity increments caused by K-means clustering and retraining is negligibly small in contrast to the effect of decomposition. Therefore KMSVDD assures the reduction of computational cost for large N.

3 Experimental Results

Following three types of simulations are performed for comparison in training time and accuracy between KMSVDD and SVDD. Tax's data description toolbox [11] is used and simulations are run on a PC with a 3.0 GHz Intel Pentium IV processor and 1G RAM. We use quadratic program solver (*quadprog*) provided by MATLAB in all experiments so the training time difference caused by the QP solver doesn't occur. Optimal parameter values of Gaussian kernel are founded using 10-fold cross validation method.

First, we observe the effect of cluster number k on the training time of KMSVDD. We measure the training time by changing the cluster number using two artificial data sets (Banana shape, Donut shape [1]).

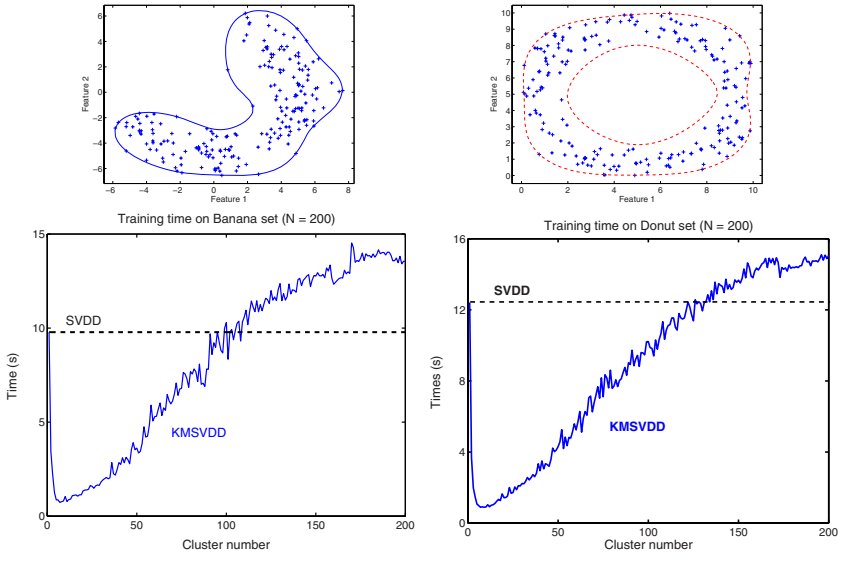

Fig. 3. Training time of KMSVDD on various cluster numbers using one class Banana and Donut shape set (data size $N = 200$). Upper figures show data distributions used in experiments. The training time varies convexly as k increases and the optimal cluster number k is different depending on the data distribution.

As shown in Fig. 3, the training time of KMSVDD varies depending on the cluster number k. If k increases, which means diminishing size of each subproblem, the computational load of local descriptions would be reduced, but the total number of *support vectors*, which are used in retraining, would increases. The increment of *working set*'s size makes the retraining process's computational cost expensive. To some extent, the training time is reduced as k increases, but when k is over a certain limit, the training time increases on the contrary. Therefore the training time varies convexly as k increases and an adequate selection of k is necessary.

We also perform similar experiment on the Banana shape data set having various data size. The change of an optimal cluster number k, which has minimum training time, for each data size is observed. The result is depicted in Fig. 4. From the results, we can find that the optimal cluster number k is different depending on the data distribution and training data size.

Second, we compare the training time of KMSVDD with the original SVDD on various size Banana shape data set. As in Table 1, the training time of SVDD grows

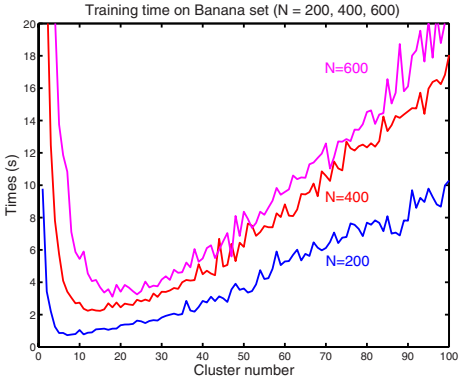

Fig. 4. Training time of KMSVDD using various size Banana shape data sets ($N = 200, 400, 600$ for each case). The optimal cluster number k varies depending on the data size.

drastically, whereas KMSVDD's training time grows relatively slowly. KMSVDD can reduce the training time even though the cluster number k is not optimal.

Table 1. Training time on two classes Banana shape data sets using the original SVDD and KMSVDD with various cluster numbers k. As the data size increases, the training time of SVDD grows drastically whereas the proposed method's training time grows relatively slowly.

		Data Size				
	k	200	600	1000	1200	1400
SVDD	1	16.8	488.4	3311.4	6527.4	11626.0
KMSVDD	2	4.4	167.7	1298.1	2157.4	4132.3
	10	0.8	12.0	112.7	327.9	458.3
	20	0.9	4.1	12.7	26.6	102.4
	30	1.3	3.1	6.2	13.9	11.9
	40	1.7	3.7	6.5	8.7	14.6
	50	2.0	4.8	6.9	10.9	13.5

Finally, we test training accuracy of both methods using real data set. UCI Iris and Wine data sets [12] are used as real data. Each data set has three classes respectively. For each class, one-against-all method [13] is used and test objects' false positive and false negative numbers are measured as the training accuracy. As we can see in Table 2, KMSVDD can result in similar accuracy with less computational load.

Also two methods results similar data descriptions as shown in Fig. 5(left), (right) though KMSVDD obtains k local data descriptions for each class before the retraining process as in Fig. 5(center) if proper learning parameters are used.

Table 2. Classification error of one-against-all classifiers for each class using UCI data set (Iris, Wine). The number of false positive and false negative objects are measured with 10-fold cross-validation method (values in brackets are standard deviations). KMSVDD shows comparable results with the original SVDD.

	Iris ($N = 150$)			Wine ($N = 178$)		
Class index	1	2	3	1	2	3
SVDD	2.3(0.48)	7.5(0.97)	6.1(1.28)	5.5(1.78)	11.7(2.11)	7.0(1.56)
KMSVDD	0.8(0.94)	8.2(1.57)	5.6(2.17)	4.7(2.86)	10.8(1.39)	6.1(2.57)

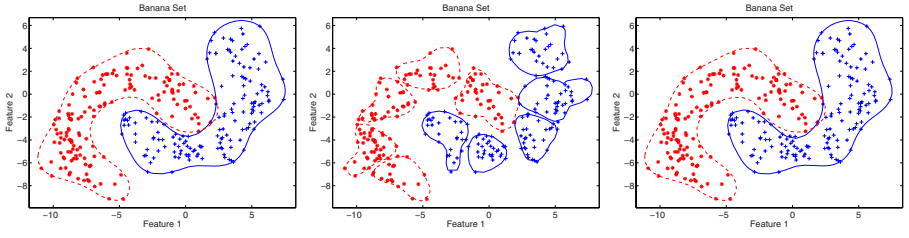

Fig. 5. Data descriptions using two classes Banana shape set $N = 200$. Data description by the original SVDD (*left*). Local (*center*) and Global (*right*) data descriptions using KMSVDD. Both methods have similar data description results.

4 Conclusion

In this paper, we present a fast one-class classifier, called KMSVDD, by combining the K-means clustering method with SVDD. SVDD has a problem that as the number of training data increases, training time also increases drastically. Most of training time is consumed calculating QP problems. To reduce the training time, we decompose a large training data set into small and compact subproblems using K-means clustering method. Training each small k sub-problem to get *support vectors* is much faster than using a whole data at a time. Subgroup training makes many local descriptions, so an additional decision rule is needed to classify (or to get a global description). To solve this problem we retrain data only with *support vectors* of every local descriptions. The simulation results of KMSVDD show remarkable training time reduction comparing with SVDD, and the training performance is comparable.

As for further research, we will deal with as following issue.

Through the mathematical analysis on the relationship between the cluster number and training time of the proposed algorithm, we will give the guideline to determine proper (or optimal) cluster number k.

The effectiveness of the proposed algorithm may be further investigated by using large scale real data sets including the UCI Forest database [12] or MNIST handwritten digit database [13].

Acknowledgments. This work was supported by the Brain Korea 21 Project and ASRI (Automation and Systems Research Institute).

References

1. Tax, D.M.J.: Support Vector Data Description. Machine Learning **54** (2004) 45–46
2. Vapnik, V.N.: Statistical Learning Theory. Wiley, New York (1998)
3. Joachims, T.: Making Large-scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning. MIT Press, Cambridge, MA (1999) 169–184
4. Osuna, E., Freund, R., Girosi, F.: Training Support Vector Machines. Conf. Computer Vision and Rattern Recognition (1997) 130–136
5. Platt, J.C.: Fast Training of Support Vector Machines using Sequential Minimal Optimization. Advances in Kernel Methods - Support Vector Learning. MIT Press, Cambridge, MA (1999) 41–65
6. Collobert, R., Bengio, S. Bengio, Y.: A Parallel Mixture of SVMs for Very Large Scale Problems. Neural Computation **14** (5) (2002) 143–160
7. Schwaighofer, A., Tresp, A.: The Bayesian Committee Support Vector Machine. Proc. Int. Conf. Artificial Neural Networks (2001) 411–417
8. Rida, A., Labbi, A., Pellegrini, C.: Local Experts Combination through Density Decomposition. Proc. Seventh Int. Workshop AI and Statistics (1999)
9. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley-Interscience (2001)
10. Schölkopf, B., Burges, C., Vapnik, V.N.: Extracting Support Data for a Given Task. Proc. of First Int. Conf. Knowledge Discovery and Data Mining (1995) 252–257
11. Tax, D.M.J.: DDtools, the Data Description Toolbox for Matlab. http://www-ict.ewi.tudelft.nl/∼ddavidt/dd_tools.html (2006)
12. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI Repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html, Univ. of California, Irvine, Dep. of Information and Computer Science (1998)
13. Bottou, L., Cortes, C., Denker, J., Drucker, H., Guyou, I., Jackel, L., LeCun, Y., Muller, U,. Sackinger, E., Simard, P., Vapnik, V.: Comparison of Classifier Methods: A Case of Study in Handwriting Digit Recognition. Proc. Int. Conf. Pattern Recognition (1994) 77–87