A New Multi-class SVM Algorithm Based on One-Class SVM

Xiao-Yuan Yang, Jia Liu, Min-Qing Zhang, and Ke Niu

Network and Information Security Key Laboratory,
Engineering College of the Armed Police Forces, Xi'an 710086, China
twinlj77@gmail.com

Abstract. Multi-class classification is an important and on-going research subject in machine learning and data mining. In this paper, we propose a new support vector algorithm, called OC-K-SVM, for multi-class classification based on one-class SVM. For *k*-class problem, this method constructs *k* classifiers, where each one is trained on data from one class. OC-K-SVM has parameters that enable us to control the number of support vectors and margin errors effectively, which is helpful in improving the accuracy of each classifier. We give some theoretical results concerning the significance of the parameters and show the robustness of classifiers. In addition, we have examined the proposed algorithm on several benchmark data sets, and our preliminary experiments confirm our theoretical conclusions.

Keywords: Machine learning, Multi-class SVM, One-class SVM.

1 Introduction

Support Vector Machines [1] (SVM) were originally designed for binary classification. How to effectively extend it for multi-class classification is still an on-going research issue. Currently there are two types of approaches for multi-class SVM. One is the "decomposition-reconstruction" architecture approach [2, 3, 4, 5] that makes direct use of binary SVMs to tackle the tasks of multi-class classification, while the other is by directly considering all data in one optimization formulation [6, 7, 8].

The first approach divides the multiple class problems into a number of binary classifications. The generalization step is based on a voting among the binary classifiers to derive the winning class. There are different transformations into binary problems [2, 3], being the most widely used: one-vs.-all (OVA), in which each class is compared with all the other classes considered as one [2]; and one-vs.-one (OVO), in which each class is individually compared with all the others [3]. They do not consider the full problem directly. Particularly, the one-vs.-all approach unbalances the training sets (if the classes are balanced, the negative class in each binary classifier will have far more samples than the positive class), and the one-vs.-one will be using only information from two classes, losing each classifier the information from all the remaining classes.

Y. Shi et al. (Eds.): ICCS 2007, Part III, LNCS 4489, pp. 677–684, 2007.

The second trend considers the multi-class problem directly as generalization of the binary classification scheme [6] and [7]. This formulation is very promising because it deals with all the samples and classes at the same time, without losing any relevant information for arriving to the best solution for each problem. Besides, the resulting machines need a lower number of support vectors [9] and achieve higher performances in the case where the training set is separable. However, if the working set selection is not good, its training speed may be slow when using a large parameter C [10].

In this paper, we propose a new algorithm for multi-class classification, called OC-K-SVM, with decomposition-reconstruction architecture. This method constructs k one-class SVM (OC-SVM) [11, 12, 13] classifiers where k is the number of classes.

The rest of this article is outlined as follows. We first give a brief account of one-class SVM in Section 2. In Section 3, we present OC-K-SVM algorithm and then show some theoretical results on OC-K-SVM. Numerical experiments are in Section 4, where we show the performance of OC-K-SVM. Finally we have some conclusions in Section 5.

2 One-Class Support Vector Machines

We first introduce terminology and notation conventions. Consider n training data points in a d-dimensional space denoted as $\{\vec{x}_1,..., \vec{x}_n\}$. An OC-SVM first projects these data into a higher, potentially infinite, dimensional space with the mapping:φ: $R^d \rightarrow F$ .In this space, a bounding hypersphere is computed that encompasses as much of the training data as possible while minimizing its volume. Shown in Figure 1 is an example where OC-SVM was trained on the black dots.

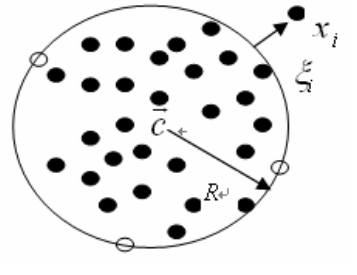

Fig. 1. The hypersphere contains the training data, described by the center $\vec{c}$ and radius R. Three white objects are on the boundary, the support vectors. One object x_i is outside and has $\xi_i > 0$.

The hypersphere center $\vec{c}$ and radius R are computed by minimizing:

$$\min_{\vec{c}, R, \xi_1, ..., \xi_n} \quad R^2 + \frac{1}{n\nu}\sum_{i=1}^{n} \xi_i . \tag{1}$$

Where $\nu \in (0,1)$ is a parameterized constant that controls the fraction of training data that fall outside of the hypersphere, $\xi_i s$ are the "slack variables" whose values indicate how far these outliers deviate from the surface of the hypersphere. This minimization is subject to:

$$\left\|\phi(\vec{x}_i) - \vec{c}\right\| \le R^2 + \xi_i, \quad \xi_i \ge 0, \qquad i = 1,..., n. \tag{2}$$

Where $\|\cdot\|$ is the Euclidean norm. The objective function of Equation (1) embodies the requirement that the volume of the hypersphere is minimized, while simultaneously encompassing as much of the training data as possible. Equation (2) forces the training data to lie within the hyperspere.We can solve this optimization with Lagrangian multipliers.

In the following section we show how this basic framework can be extended to construct multi-class SVM with multiple hyperspheres.

3 OC-K-SVM

An OC-SVM with a single hypersphere, as described in the previous section, obviates the need for training classifiers on the other training sets. For *k*-class problem, we propose to cover the *k*-class training data sets with several hyperspheres, where each hypersphere encompasses one class subset of the training data. Shown in Figure 2 is a toy 2-D example where an OC-K-SVM was trained on three classes.

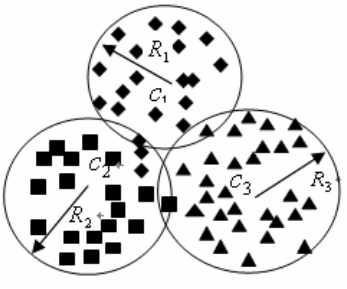

Fig. 2. Shown is a toy example of OC-K-SVM. The circles represent the OC-SVM classifiers. For example, The 2*th* one-class SVM are trained on only the black squares-notice that in the OC-K-SVM, the classifier is better able to generalize as both the black triangles and diamonds generally fall outside the support of the 2*th* bounding circle.

Note that, unlike the other multi-class SVM, an OC-K-SVM is trained on data from *k* classes by computing *k* bounding hyperspheres. We describe below the details behind the construction of such OC-K-SVM.

Given n_m training data $\left\{\vec{x}_1^m,..., \vec{x}_{n_m}^m\right\}$ for the m^{th} class, $\vec{x}_i \in R^d$, i=1,…,n_m where $m \in \{1,...,k\}$ is the class of $\vec{x}_i^m$, the OC-K-SVM solves the follows problems:

$$\min_{\vec{c}_m, R_m, \xi_i^m, \ldots, \xi_{n_m}^m} \quad R_m^2 + \frac{1}{n_m \nu_m} \sum_{i=1}^{n_m} \xi_i^m . \tag{3}$$

$$\textit{subject to} \quad \left\| \phi(\vec{x}_i^m) - \vec{c}_m \right\| \le R_m^2 + \xi_i^m . \tag{4}$$

$$\xi_i^m \ge 0, i = 1, \ldots, n_m, m = 1, \ldots k \tag{5}$$

where R_m and $\vec{c}_m$ are the radius and center of the m^{th} hypersphere, $\nu_m \in (0,1)$ is a parameterized constant that controls the fraction of training data that fall outside of the mth hypersphere, the training data $\vec{x}_i^m$ are mapped to a higher dimensional space by the function ϕ and ξ_i^ms are the "slack variables".

To determine $\vec{c}_m$ and R_m, the quadratic programming problem of Equations (3) are transformed into their dual form:

$$\min_{a_1^m, \ldots, a_{n_m}^m} \sum_{i=1}^{n_m} \sum_{j=1}^{n_m} \alpha_i^m \alpha_j^m \phi(\vec{x}_i^m)^T \phi(\vec{x}_j^m) - \sum_{i=1}^{n_m} \alpha_i^m \phi(\vec{x}_i^m)^T \phi(\vec{x}_j^m). \tag{6}$$

$$\textit{subject to} \quad \sum_{i=1}^{n_m} \alpha_i^m = 1 . \tag{7}$$

$$0 \le \alpha_i^m \le \frac{1}{n_m \nu_m}, \; i = 1, \ldots, n_m, \; m = 1, \ldots, k . \tag{8}$$

Where α_i^m s are Lagrange multipliers. Standard techniques from quadratic programming can be used to solve for the unknown Lagrange multipliers. The centers of the hyperspheres are then given by:

$$\vec{c}_m = \sum_{i=1}^{n_m} \alpha_i^m \phi(\vec{x}_i^m), \quad m = 1, \ldots, k . \tag{9}$$

Similar to the above, R^2 is computed from such points, $\vec{x}_i$ with $0 < \alpha_i^m < 1/(n_m \nu_m)$. Any such data point $\vec{y}_m$ that lies on the surface of the m^{th} optimal hypersphere satisfies the following:

$$R_m^2 = \left\| \phi(\vec{y}_m) - \vec{c}_m \right\|^2, \quad m = 1, \ldots, k . \tag{10}$$

Substituting the solution of Equation (9) into the above yields a solution for the hypersphere radius:

$$R_m^2 = \sum_{i=1}^{n_m} \sum_{j=1}^{n_m} \alpha_i^m \alpha_j^m \phi(\vec{x}_i^m)^T \phi(\vec{x}_j^m) - 2 \sum_{i=1}^{n} \alpha_i^m \phi(\vec{x}_i^m)^T \phi(\vec{y}_m) + \phi(\vec{y}_m)^T \phi(\vec{y}_m). \tag{11}$$

After solving (3), there are k decision functions:

$$f_m(\vec{x}) = R_m^2 - \left\|\phi(\vec{x}) - \vec{c}_m\right\|^2, \quad m = 1,...,k. \tag{12}$$

Here we can say $\vec{x}$ is the class which has largest value of the decision function:

$$\arg\max_{m=1,...,k}\left(R_m^2 - \left\|\phi(\vec{x}) - \vec{c}_m\right\|^2\right). \tag{13}$$

However, there are k hyperspheres with k radiuses and k centers, which makes large errors. So we redefine the decision function:

$$f_m(\vec{x}) = \frac{R_m^2 - \left\|\phi(\vec{x}) - \vec{c}_m\right\|^2}{R_m^2} = 1 - \frac{\left\|\phi(\vec{x}) - \vec{c}_m\right\|^2}{R_m^2}, \quad m = 1,...,k. \tag{14}$$

$$\arg\max_{m=1,...,k}\left(1 - \frac{\left\|\phi(\vec{x}) - \vec{c}_m\right\|^2}{R_m^2}\right). \tag{15}$$

Substituting the solutions of Equation (9) and (11) into the above decision function (15) yields the decision function:

$$\arg\max_{m=1,...,k}\left(1 - \frac{\sum_{i=1}^{n_m}\sum_{j=1}^{n_m}\alpha_i^m\alpha_j^m\phi(\vec{x}_i^m)^T\phi(\vec{x}_j^m) - 2\sum_{i=1}^{n_m}\alpha_i^m\phi(\vec{x}_i^m)^T\phi(\vec{x}) + \phi(\vec{x})^T\phi(\vec{x})}{\sum_{i=1}^{n_m}\sum_{j=1}^{n_m}\alpha_i^m\alpha_j^m\phi(\vec{x}_i^m)^T\phi(\vec{x}_j^m) - 2\sum_{i=1}^{n_m}\alpha_i^m\phi(\vec{x}_i^m)^T\phi(\vec{y}_m) + \phi(\vec{y}_m)^T\phi(\vec{y}_m)}\right). \tag{16}$$

If an appropriate kernel function is introduced, the re-formulated objective function takes the form:

$$\min_{\alpha_1^m,.....\alpha_n^m}\sum_{i=1}^{n_m}\sum_{j=1}^{n_m}\alpha_i^m\alpha_j^m k(\vec{x}_i^m,\vec{x}_j^m) - \sum_{i=1}^{n_m}\alpha_i^m k(\vec{x}_i^m,\vec{x}_j^m) \qquad m = 1,..,..,k. \tag{17}$$

Then the class of point $\vec{x}$ is determined by the largest value of the decision function:

$$\arg\max_{m=1,...,k}\left(1 - \frac{\sum_{i=1}^{n_m}\sum_{j=1}^{n_m}\alpha_i^m\alpha_j^m k(\vec{x}_i^m,\vec{x}_j^m) - 2\sum_{i=1}^{n_m}\alpha_i^m k(\vec{x}_i^m,\vec{x}) + k(\vec{x},\vec{x})}{\sum_{i=1}^{n_m}\sum_{j=1}^{n_m}\alpha_i^m\alpha_j^m k(\vec{x}_i^m,\vec{x}_j^m) - 2\sum_{i=1}^{n_m}\alpha_i^m k(\vec{x}_i^m,\vec{y}_m) + k(\vec{y}_m,\vec{y}_m)}\right) \tag{18}$$

Note that the idea is similar to the one-vs.-all approach. The one-vs.-all approach also constructs k one-class classifiers. The m^{th} classifier constructs a hyperplane between one class and the k-1 other classes. However, for each classifier, the OC-SVM is trained on data from only one class by computing a bounding hypersphere (in the projected high-dimensional space) that encompasses as much of the training data as possible, while minimizing its volume.

The value $1/(\nu_m n_m)$ gives the upper boundary for the parameters α_i^m (see equation (8)) where $m \in \{1,...,k\}$. Similar to the statements in [12], the following statements hold:

i. ν_m is an upper bound on the fraction of outliers, that is, training point outside the m^{th} estimated region.

ii. ν_m is lower bound on the fraction of support vectors.

iii. Suppose the data $\{\vec{x}_1,...,\vec{x}_{n_m}\}$ were generated independently from a distribution $P_m(\vec{x})$, which does not contain discrete components. Suppose, moreover, that the kernel is analytic and non-constant. With probability 1, asymptotically, ν_m equals both the fraction of SVs and fraction of outliers.

4 Numerical Experiments

In this section we tested the proposed method on the benchmark data sets selected from the UCI data repository [14] and Statlog data collection. We scaled all the data to rage [-1, 1]. Table 1 summarizes the data sets used. Note that for problems glass and satimage, there is one missing class. That is, in the original application there is one more class but in the data set no examples are with this class.

Table 1. Benchmark datasets used for testing

Problem	Training data	testing data	class	attributes
Iris	150	0	3	4
Wine	178	0	3	13
glass	214	0	6	13
Segment	2310	0	7	19
Satimage	4435	2000	6	36
letter	15000	5000	26	16
shuttle	43500	14500	7	9

The most important criterion for evaluating the performance of this method is its accuracy rate. As a comparative approach, we also cite the result of comparing five methods which presents on [9] In order to reduce the search space of parameters, practically, we set parameters $\nu_1 = \nu_2 = ... = \nu_m = \nu$ for OC-K-SVM. We only trained the classifiers using the Radial Basis Function (RBF) kernel $k(\vec{x}_i, \vec{x}_j) = \exp(-\gamma \|\vec{x}_i - \vec{x}_j\|)$ with $\gamma = \{2^{-3}, 2^{-2}, ..., 2^3\}$ and $\nu = \{0.01, 0.05, 0.1, 0.15, 0.2\}$, where γ is the width parameter of RBF kernel and ν is the penalty parameter. We use similar stopping criteria for our method. For each problem we stop the optimization algorithm if the KKT violation is less than 10^{-3}. We use two criteria to estimate the generalized accuracy. For data sets satimage, letter and shuttle where both training and testing sets are available, for each pair of (γ, ν), the performance is measured by training the 80% of training set and testing the other 20% of the training set. Then we train the whole training set using the pair of

(γ, ν) that achieves the best validation rate and predict the test set. For the other four smaller datasets where test data may not be available, we simply conduct 5-fold cross-validation on the whole training data and report the best cross-validation rate.

We report best testing rate, training time, testing time and number of support vectors in Table 2. These experiments were carried out using LIBSVM [15] on Intel Pentium Ⅳ 2.00GHz PC with 256M RAM. However, LIBSVM did not provide one-class SVM algorithm based on hypersphere. We modified the program to output the radiuses and the value of decision function defined in our OC-K-SVM.

Table 2. The result of the numerical experiment. Measured: best rate, training time, testing time, and number of support vectors

problem	rate	training time	testing time	number of SVs
iris	90.67	0.0156	---	119
wine	54.49	0.0156	---	45
glass	71.03	0.0156	---	129
segment	98.57	0.0781	---	672
satimage	90.26	0.4063	4.35	949
Letter	90.00	0.5600	43.09	3584
shuttle	99.08	2.1820	7.37	1352

It can be observed that, for large problem, OC-K-SVM has a good performance. Comparing to earlier results listed in [9], the accuracy obtained by OC-K-SVM is competitive. Specifically, for the "segment" set, OC-K-SVM outperforms the others. For "shuttle" set, OC-K-SVM shows a similar performance to all the others. Unfortunately, we note that for smaller problems, the accuracy rate of the OC-K-SVM is lower than all the others. This is because one-class SVM is proposed in [11] as the data domain description problem. If the number of the objects is too small, it is likely that the data domain description has poor performance.

For the training time, our method is the best. This is due to that we only need to train k classifiers, each problem is smallest (only data from one class). Although one-vs.-one has to train as many as $k(k-1)/2$ classifiers, as each problem is smaller, the total training time is still less.

Regarding the testing time, the experience in [9] show that in general the testing time is still dominated by the kernel evaluations and is proportional to the number of support vectors. We also observe that among these methods, OC-K-SVM is really faster on the testing time.

We then discuss the number of support vectors. We can see that for larger problems, the methods from [6], [7] returns fewer support vectors than all three binary-based approaches. On the other hand, we cannot draw any conclusions about the OC-K-SVM method by us. Sometime it needs very few support vectors but sometimes the number is huge.

Finally we would like to draw some remarks about the implementation of our method. As can be seen in the Table 2, for the larger problems, the OC-K-SVM has smallest training and testing time. Moreover, the resulting accuracy is also acceptable. Therefore, if the training and testing time is very important, this method can be an option.

5 Conclusion and Future Work

In this paper, we propose a new algorithm, OC-K-SVM, for the multi-class classification based on one-class classification. This procedure has the advantage of providing solutions that are as good as the previously proposed schemes or better with a decrease in the training time. We have confirmed the established theoretical results and good behavior of algorithm through experiments on benchmark data sets.

It is worthwhile to investigate the proposed kernel with other efficient algorithms which can solve the one-class problems, e.g. The Nearest Point Algorithm or Successive Over-relaxation (SOR) algorithm. Future research subjects include more comprehensive testing of the algorithm and application to real-world problem.

References

1. C.cortes and V. Vapnik, J. (ed.): Support-vector network, Machine learning, Vol. 20, （1995） 273-297.
2. L. Bottou, C. Cortes, and J. Denker, A.: Comparison of classifier methods: a case study in handwriting digit recognition, in Proc. of the International Conference on Pattern Recognition, IEEE Computer Society Press, Vol. 2. (1994) 77-87.
3. S. Knerr, L. Personnaz, and G. Dreyfus, In Fogelman-Soulie and Herault, (eds.): Single-layer learning revisited: a stepwise procedure for building and training a neural network. Neurocomputing: Algorithms, Architectures and Applications, Vol. F68 of NATO ASI Series. Springer-Verlag, Berlin Heidelberg New York (1990) 41-50.
4. J. C. Platt, N. Cristianini, and J. Shawe-Taylor, J. (ed.): Large margin DAGs for multi-class classification, Advances in Neural Information Processing Systems, Vol. 12. (2000)547-553.
5. Ping Zhong and Masao Fukushima, J. (ed.): A new multi-class support vector algorithm, Optimization Methods and Software, to appear.
6. V. Vapnik. Z.: Statistical Learning Theory, Wiley, New York, NY(1998).
7. J.Weston and C. Watkins, Multi-class support vector machines, Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, Egham, TW20 0EX, UK(1998).
8. Jeronimo, Arenas-Garcia, and Fernando Perez-Cruz, A: Multi-class support vector machines: a new approach. In: ICASSP, Hong Kong (2003) 6-10.
9. C. W. Hsu and C. J. Lin, B.C.: A comparison of methods for multi-class support vector machines, IEEE Transactions on Neural Networks, Vol. 13, no. 2.(2002)415-425.
10. C.-W. Hsu and C.-J. Lin, J. (ed.): A simple decomposition method for support vector machines, Machine Learning, Vol. 46. (2002)291-314.
11. Tax, D.M.J., and Duin, R.P.W, A: Data domain description by support vectors, Proceedings ESANN, Brussels: D Facto, (1999)251-256.
12. B. Scholkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, J. (ed.): Estimating the support of a high-dimensional distribution, Neural Computation, Vol. 13, no. 7. (2001)1443-1471.
13. Siwei Lyu and Hany Farid, A: Steganalysis using color wavelet statistics and one-class support vector machines, in SPIE Symposium on Electronic Imaging, San Jose, CA(2004).
14. UCI-benchmark repository of artifical and real data sets, University of California Irvine, http://www.ics.uci.edu/˜mlearn.
15. C.-C. Chang and C. J. Lin, LIBSVM: a library for support vector machines, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm, (2001).