Kernel Bisecting *k*-means Clustering for SVM Training Sample Reduction

Xiao-Zhang Liu[1,2], Guo-Can Feng[1]
[1]*Faculty of Mathematics and Computing, Sun Yat-sen University, Guangzhou 510275, PR China*
[2]*Heyuan polytechnic, Heyuan, Guangdong 517000, PR China*
liuxiaozhang@gmail.com, mcsfgc@mail.sysu.edu.cn

Abstract

This paper presents a new algorithm named Kernel Bisecting k-means and Sample Removal (KBK-SR) as a sampling preprocessing for SVM training to improve the scalability. The novel clustering approach Kernel Bisecting k-means in the KBK-SR tends to fast produce balanced clusters of similar sizes in the kernel feature space, which makes KBK-SR efficient and effective for reducing training samples for nonlinear SVMs. Theoretical analysis and experimental results on three UCI real data benchmarks both show that, with very short sampling time, our algorithm dramatically accelerates SVM training while maintaining high test accuracy.

1 Introduction

In recent years, a number of algorithms based on "divide-and-conquer" have been proposed to improve the scalability of SVMs, i.e., accelerate SVM training with large-scale samples, such as chunking algorithm [1], decomposition algorithm [2], sequential minimal optimization (SMO) [3], and *SVM* ^{light} [4]. These approaches decompose a large training task into a series of smaller sub-tasks so that the overall SVM training time can be reduced, but the time complexity still needs further improvement in practice.

While training samples close to decision boundaries have higher chances to be support vectors (SVs), samples far from decision boundaries have less effect when identifying SVs. Therefore, some algorithms have been proposed to sample a small portion of training data which are more likely to be SVs, such as active learning [5], random sampling [6], and clustering-based SVM [7]. All these methods need to train SVMs and/or scan the whole training set for many times to get the current selection of data, so their efficiency is still limited by the training speed of SVMs and the scale of training sets.

Wang et al. [8] proposed an algorithm named sample reduction by data structure analysis (SR-DSA), which selects potential SVs based on structure information extracted from the training set by agglomerative hierarchical clustering (AHC). However, the AHC procedure in this approach has two drawbacks: (1) the distance between every two clusters must be calculated and stored, which means a space requirement proportional to $\mathbb{N}^2$; (2) It is time-consuming to build the whole dendrogram to determine the appropriate number of clusters.

In view of the inefficiency of the *bottom-up* AHC, this paper proposes a new sample reduction approach, which executes a *top-down* hierarchical clustering named as kernel bisecting *k*-means (KBK), to produce balanced clusters of similar sizes in the kernel feature space, followed by a sample removal (SR) procedure.

The remainder of this paper is organized as follows. In section 2, we first provide a brief review of bisecting *k*-means, and then propose KBK. In section 3, we describe our algorithm KBK-SR, elaborating the proposed KBK clustering procedure. Experiments are reported in section 4, while we draw our conclusions in section 5.

2 Kernel bisecting *k*-means

2.1 Bisecting *k*-means algorithm

The traditional *k*-means provides no guarantee of producing balanced clusters. Therefore, it is quite possible that a single cluster contains a large portion of the entire datasets, limiting the usefulness of this algorithm for improving scalability. Bisecting *k*-means [9] can be used to enforce balancing constraints, which is a variant of *k*-means and repeatedly bisects the largest remaining cluster into two subclusters at each step, until the desired value of *k* is reached. This algorithm starts with the whole data set as a single cluster, and typically converges to balanced clusters of similar sizes. Readers are referred to [9] for the steps in detail.

2.2 KBK — kernel bisecting *k*-means

Like *k*-means, bisecting *k*-means yields piecewise linear borders among data, making the approach unsuitable for sample reduction preceding SVM training, because SVMs are developed mainly for nonlinear classification in applications. For this reason, we proposed kernel bisecting *k*-means (KBK) by integrating kernel methods with bisecting *k*-means.

Kernel methods are algorithms that, by replacing the inner product $< x, y >$ with an appropriate positive definite function $K(x, y)$, implicitly perform a nonlinear mapping $\Phi : \mathcal{X} \rightarrow \mathcal{F}$ from the input space $\mathcal{X}$ to a high dimensional feature space $\mathcal{F}$ endowed with an inner product defined as

$$K(x, y) =< \Phi(x), \Phi(y) > . \quad (1)$$

To perform bisecting *k*-means in the feature space, referring to the algorithm listed in [9], we formally need to solve two problems. One is how to compute what we call the kernel squared Euclidian distance, i.e., the squared Euclidian distance in the feature space; the other is how to calculate the kernel mean vector of a cluster, i.e., the centroid in the feature space.

Using Eq.(1), we can compute the kernel squared Euclidian distance as

$$\begin{aligned}\|\Phi(x) - \Phi(y)\|^2 & \\ &= K(x, x) - 2K(x, y) + K(y, y). \quad (2)\end{aligned}$$

However, it is not so easy to solve the second problem. Let C be a cluster of l patterns $x_1, x_2, \ldots, x_l$, $C^\Phi = \{\Phi(x_j) | j = 1, \ldots, l\}$, and μ_{C^Φ} be the mean vector of C^Φ. Then we have

$$\mu_{C^\Phi} = \frac{1}{l} \sum_{j=1}^{l} \Phi(x_j). \quad (3)$$

Clearly μ_{C^Φ} cannot be computed directly in the case of an unknown mapping Φ. Fortunately, there is no need to calculate μ_{C^Φ}, since it is not necessarily the case that μ_{C^Φ} belongs to C^Φ. What we really need in practice is a pattern $m_1 \in C$, whose image $\Phi(m_1)$ is the closest element to μ_{C^Φ} in C^Φ. From Eqs.(2)–(3), we have

$$\begin{aligned}\|\Phi(x_i) - \mu_{C^\Phi}\|^2 = K(x_i, x_i) - \frac{2}{l} \sum_{j=1}^{l} K(x_i, x_j) \\ + \frac{1}{l^2} \sum_{i=1}^{l} \sum_{j=1}^{l} K(x_i, x_j), \quad (4)\end{aligned}$$

where the final summation term is a constant. Let

$$r_i = K(x_i, x_i) - \frac{2}{l} \sum_{j=1}^{l} K(x_i, x_j), \quad (5)$$

then we can easily find i_1 subject to

$$r_{i_1} = \min_{1 \le i \le l} r_i, \quad 1 \le i_1 \le l. \quad (6)$$

It can be easily seen that

$$m_1 = x_{i_1}. \quad (7)$$

Thus the bisecting *k*-means algorithm can be performed in the feature space, which is what we call KBK. The detailed algorithm is seen in the following.

3 KBK-SR for SVM sample reduction

3.1 The KBK-SR algorithm

The kernel bisecting *k*-means and sample removing (KBK-SR) algorithm can be described as three steps, of which the first is KBK clustering procedure.

S1 Perform KBK clustering for positive class and negative class independently. For positive class P, given a certain threshold τ, the clustering $P = P_1 \bigcup P_2 \bigcup \ldots \bigcup P_{S_P}$ is achieved as follows:

```
$C_0 \leftarrow P$, $CS \leftarrow \{C_0\}$ ($CS$ means set of clusters)
do
    select the largest remaining cluster $C_j$ in $CS$
    *bisecting step*:
    use the method shown by Eqs.(5)–(7) to find $m_1$ in $C_j$;
    find $m_2$ in $C_j$, subject to $\|\Phi(m_2)-\Phi(m_1)\|^2 = \min\{\|\Phi(x) - \Phi(m_1)\|^2 | x \in C_j\}$, where the kernel squared Euclidian distance is computed by Eq.(2);
    do
        *calculation step*: for each data point of $C_j$, use Eq.(2) to compute the kernel squared Euclidian distance with $m_1$ and $m_2$, respectively, and assign the data point to its closest choice, forming two subclusters $C_{j_1}, C_{j_2}$;
        *update step*: use the method shown by Eqs.(5)–(7) to update $m_1$ in $C_{j_1}$ and $m_2$ in $C_{j_2}$, respectively;
    until no change in $m_1$ and $m_2$
    $CS \leftarrow CS \bigcup \{C_{j_1}, C_{j_2}\}$
until size of the largest cluster in $CS$ is below $\tau$.
```

For negative class N, the clustering $N = N_1 \bigcup N_2 \bigcup \ldots \bigcup N_{S_N}$ is achieved in a similar way.

S2 Remove the interior samples in each relatively large cluster. For each cluster C, if $|C| \ge \tau_0$, calculate the kernel squared Mahalanobis distance (the *distance* for short) from each data pattern in

C to the cluster itself, sort the *distances*, and according to a given η ($0 < \eta < 1$), pick out $\eta|C|$ patterns with the largest *distances*, and remove all other data points, where $|C|$ denotes the cardinal of C, and τ_0 is a certain small natural number (e.g., 5). Thus, the cluster C shrinks to C', which only contains $\eta|C|$ patterns.

S3 **For each relatively large cluster, remove exterior samples that are distant from the opposite class.** If $|C'| \geq \tau_0$, remove the points whose *distances* to the clusters in the opposite class are greater than the average *distance*, to get C''.

S2 and **S3** composes the SR procedure, which aims at removing points that are impossible to have influence on the final decision boundary. The role of τ_0 is to skip very small clusters.

3.2 Further explanation and complexity analysis

Note that a modified version of bisecting *k*-means is used in the KBK clustering procedure (3.1 **S1**), whose peculiarity is embodied by two details. First, the initial assignment of m_1 and m_2 in the outer loop is deterministic rather than stochastic, which makes the initial two subclusters well separated in the feature space and fast leads to local optima of better quality than those produced from a random initialization. Second, the stopping criterion for **S1** is, the size of the largest cluster is below a certain threshold τ, rather than a certain number of clusters are obtained. This provides safer guarantee of balanced clusters of similar sizes, which controls the computational burden of both the KBK clustering and the SR procedures, as can be seen in the following.

Now we get down to the complexity analysis of our KBK-SR algorithm. As each *bisecting step* in **S1** tends to produce two subclusters of similar sizes, for analytical convenience, we assume the two are of the same size. In the KBK clustering procedure, the major computational load derives from the calculation of the kernel squared Euclidian distances in the *calculation step* and the computation of Eq.(5) in the *update step*, which is both referred to as *basic operations* of this procedure. The repeated *bisecting steps* split the original dataset of n points from 2^{t-1} subclusters of size $\frac{n}{2^{t-1}}$ into 2^t subclusters of size $\frac{n}{2^t}$, which involves $2^{t-1} \cdot \overline{T}_t \cdot 2 \cdot \frac{n}{2^{t-1}}$ distance calculations and $2^{t-1} \cdot \overline{T}_t \cdot 2 \cdot \frac{n}{2^t}$ computations of Eq.(5), the two terms summing to $3\overline{T}_t n$, where $\overline{T}_t$ denotes the average number of inner loops for one subcluster of size $\frac{n}{2^{t-1}}$ bisected into two of size $\frac{n}{2^t}$. Given the size threshold τ, when the outer loop stops, we have $\tau = \frac{n}{2^t}, t = \log \frac{n}{\tau}$. The way we initially assign m_1 and m_2 makes the initial two subclusters well separated in the feature space, which means $\overline{T}_t \ll n$, so we replace all $\overline{T}_t, t = 1, 2, \ldots, \log \frac{n}{\tau}$, with a constant T. Therefore, the number of *basic operations* in all can be approximated by $3Tn \log \frac{n}{\tau}$, and the overall time complexity for the KBK clustering procedure is $O(n \log \frac{n}{\tau})$, which greatly reduces the time complexity $O(n^2)$ of AHC procedure in SR-DSA [8].

In the SR procedure, the kernel squared Mahalanobis distance calculations account for the major computational burden. It has been proven that the calculation of kernel squared Mahalanobis distance from a sample to a population X_m amounts to computing m^2inner products [10] (where $m = |X_m|$), i.e., the computational load depends on the number of samples rather than the dimensionality of the feature space, which is favorable for computing in a high- or even infinite-dimensional feature space. This fact leads us to perform the kernel squared Mahalanobis distance calculations to populations of smaller sizes, which is achieved by setting the size threshold τ of the largest cluster in the stopping criterion. Again we start with the ideal condition, i.e., the original dataset of n points is partitioned by KBK into $\frac{n}{\tau}$ clusters of the same size τ. Thus the number of inner products computed in **S2** is $\frac{n}{\tau} \cdot \tau \cdot \tau^2 = \tau^2 n$. It can be easily seen that if not all the clusters produced by KBK are of size τ, i.e., some of sizes less than τ, the number of inner products is even smaller. As for **S3**, the computational load is minor to that of **S2**, because a sampling has been performed before **S3**. Therefore the overall time complexity for the SR procedure is $O(\tau^2 n)$.

4 Experimental results

Our KBK-SR algorithm as a sampling preprocessing of SVM training has been tested on three benchmark datasets for binary classification applications from UCI machine learning repository[1], GERMAN, DIABETES and LIVER-DISORDERS, whose sizes differ in a descending order. The performance comparison of KBK-SR, SR-DSA and no sampling has been made, all the three followed by an SMO-type algorithm with Gaussian kernel. The algorithms have been implemented in MATLAB and executed on a Celeron 1.4 GHz laptop with 512 MB memory.

We empirically used $2\sqrt{n}$ as the value of the parameter τ, where n stands for the number of patterns fed to the corresponding KBK clustering procedure. In all the experiments, we set the parameter η to 0.3. The regularization parameter C and the width parameter σ of Gaussian kernel were tuned using 5-fold cross-validation on the whole dataset.

[1]http://www.ics.uci.edu/ mlearn/MLRepository.html

Table 1. performance comparison of sampling methods

Dataset	Sampling Method	No. of samples	Sampling time(s)	Training time(s)	Test accuracy(%)
GERMAN	KBK-SR	156	6.48	60.42	70.83
	SR-DSA	134	98.42	32.81	70.77
	No sampling	500	0	986.93	72.64
DIABETES	KBK-SR	120	3.51	24.74	73.54
	SR-DSA	107	44.06	22.49	70.33
	No sampling	384	0	312.11	74.69
LIVER-DIS	KBK-SR	59	0.78	12.81	75.02
	SR-DSA	53	1.46	11.76	73.63
	No sampling	173	0	51.74	76.28

For each dataset, we split it in half to training and test samples. Table 1 shows the time complexity and testing accuracies of the three methods, which are averaged over 30 random splits of the data to minimizing the possible misleading results. The results show that our KBK-SR dramatically outperforms SR-DSA in terms of the sampling time, and this advantage is more significant for a larger dataset. Also, our method greatly speeds up SVM training with very little loss of test accuracy when compared with SMO without sampling.

5 Conclusion

In this paper, we present a novel clustering approach, Kernel Bisecting *k*-means, and further integrate it with a subsequent sample removal procedure to develop a new sample reduction algorithm named as KBK-SR, which effectively reduces training samples for SVMs. The main quality of Kernel Bisecting *k*-means consists in fast producing balanced clusters of similar sizes in the kernel feature space, which makes KBK-SR efficient and effective for reducing training samples for nonlinear SVMs. In comparison with SR-DSA, our algorithm significantly reduces the sampling time complexity while maintaining high test accuracy.

Acknowledgement

This work is partially supported by the National Natural Science Foundation of China under grant No. 60675007 and the Education Ministry of China under key grant No. 104145.

References

[1] B.E. Boser, I.M. Guyon, V.N. Vapnik. A training algorithm for optimal margin classifiers. In: *Fifth Annual Workshop on Computational Learning Theory*, Pittsburgh, PA: ACM Press, 1992, pp. 144-152.

[2] E. Osuna, R. Freund, F. Girosi. An improved training algorithm for support vector machines. New York: ICNNSP97, 1997, pp. 276-285.

[3] J. Platt. Fast training of support vector machines using sequential minimal optimization. in: B. Schölkopf, C. Burges, A. Smola (Eds.). *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA, 1999, pp. 185-208.

[4] T. Joachims. Making large-scale SVM learning practical, in: B. Schölkopf, C. Burges, A. Smola (Eds.). *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA, 1999, pp. 169-184.

[5] G. Schohn, D. Cohn. Less is More: Active Learning with Support Vector Machines. In: *Proceedings of the 17th International Conference on Machine Learning* (ICML'00), 2000, pp. 839-846.

[6] Y.J. Lee, O.L. Mangasarian. RSVM: Reduced Support Vector Machines. In: *Proceedings of the 1th SIAM International Conference on Data Mining*, Chicago, 2001.

[7] H. Yu, J. Han, K.C. Chang. Classifying Large Datasets Using SVMs with hierarchical clusters. In: *Proceedings of International Conference on Knowledge Discovery and Data Mining* (KDD'03), 2003, pp. 306-315.

[8] D. Wang, L. Shi. Selecting Valuable Training Samples for SVMs via Data Structure Analysis. *Neurocomputing* (2007), doi: 10.1016 /j.neucom. 2007.09.008.

[9] Y. Li, S.M. Chung. Parallel Bisecting K-means with prediction clustering algorithm. *J. Supercomput.*, 39(1): 19-37, 2007.

[10] A. Ruiz, P.E. López-de-Teruel, Nonlinear Kernel-Based Statistical Pattern Analysis. IEEE Trans. on Neural Networks 12 (2001) 16-32.