SPARSE KERNEL MAXIMUM MARGIN CLUSTERING

Ji Wu, Xiao-Lei Zhang[*]

Abstract: Recently, a new clustering method called maximum margin clustering (MMC) was proposed. It extended the support vector machine (SVM) thoughts to unsupervised scenarios and had shown promising performances. Traditionally, it was formulated as a non-convex integer optimization problem which was difficult to solve. In order to alleviate the computational burden, the efficient cutting-plane MMC (CPMMC) [22] was proposed which solved the MMC problem in its primal. However, the CPMMC is restricted to linear kernel. In this paper, we extend the CPMMC algorithm to the nonlinear kernel scenarios, which is the proposed sparse kernel MMC (SKMMC). Specifically, we propose to solve an adaptive threshold version of CPMMC in its dual and alleviate its computational complexity by employing the cutting plane subspace pursuit (CPSP) algorithm [7]. Eventually, the SKMMC algorithm could work with nonlinear kernels at a linear computational complexity and a linear storage complexity. Our experimental results on several real-world data sets show that the SKMMC has higher accuracies than existing MMC methods, and takes less time and storage demands than existing kernel MMC methods.

Key words: *Large scale data set, maximum margin clustering (MMC), nonlinear kernel*

Received: June 3, 2011
Revised and accepted: December 19, 2011

1. Introduction

Clustering finds a structure in a collection of unlabeled data and has been identified as a significant technique for many applications. Since the early work in k-means clustering [11, 5], data clustering has been studied for years and many algorithms have been developed, such as mixture model [12], fuzzy clustering [23, 3] and spectral clustering [16, 13, 8].

Recently, the maximum margin clustering (MMC) technique has attracted much attention [24]. It borrows the idea from the large margin thoughts in support vector

[*]Ji Wu, Xiao-Lei Zhang
Multimedia Signal and Intelligent Information Processing Laboratory, Tsinghua National Laboratory for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing, China, wuji_ee@tsinghua.edu.cn, huoshan6@126.com

machine (SVM), and aims at finding not only the maximum margin hyperplane in the feature space but also the optimal labeling vector which makes the margin maximized among all possible labeling vectors. However, the MMC is a nonconvex integer optimization problem, which is difficult to solve. The early works resolved this problem as a convex semi-definite programming (SDP) problem [24, 21], which made them computationally intolerable (above $\mathcal{O}(n^3)$) when the datasets contained over thousands of samples.

In order to solve the MMC problem efficiently, several important works based on quadratic programming (QP) have been done [26, 10, 4, 22]. Among them, the most efficient one was the cutting-plane MMC algorithm [22] which employed the constrained concave-convex procedure (CCCP) [25, 17] to decompose the MMC problem into a serial sub-SVM problem and employed the efficient cutting plane algorithm [9] to solve each sub-SVM problem approximately. Although each sub-SVM problem could be solved in a linear time $\mathcal{O}(sn)$, the CPMMC is restricted to linear kernel, where s denotes the sparsity of the data set. If we want to use CPMMC with nonlinear kernel for better clustering performance, we have to compute the coordinates of each sample in the kernel principle components analysis (KPCA) basis [14] according to the kernel matrix $\mathbf{K}$. It spent $\mathcal{O}(n^2)$ to do the KPCA [15] and about $\mathcal{O}(n^2 D)$ to get the coordinates, where D denotes the first D largest eigenvalues of $\mathbf{K}$. This pre-processing demand is too computational expensive and suffers terrible information loss when D is set to a small integer. Another way to use the nonlinear kernel is to do the Cholesky decomposition of $\mathbf{K}$ [1]. It has a computational complexity of $\mathcal{O}(\frac{1}{3}n^3)$ [18] and a positive-definite demand of $\mathbf{K}$.

In this paper, we propose a new kernel MMC algorithm called sparse kernel MMC (SKMMC) which extends the CPMMC algorithm to nonlinear kernel scenarios. More specifically, we eliminate the computational expensive kernel decomposition of the CPMMC by solving the inner cutting plane algorithm in its dual, which is the proposed kernel CPMMC algorithm. However, the computational complexity of kernel CPMMC is $\mathcal{O}(n^2)$. To remove the $\mathcal{O}(n^2)$ scaling behavior, we employ the recently proposed cutting-plane subspace pursuit (CPSP) algorithm [7] which constructs a small set of basis vectors from the cutting-plane model and makes a sparse approximation of the kernel matrix of the kernel CPMMC. Finally, the computational complexity and the storage complexity of the SKMMC are both linear with the sample size, which is more efficient than existing kernel MMC algorithm. The rest of the paper is organized as follows. In Section 2, we revisit the original definition of the MMC problem and the efficient CPMMC algorithm. In Section 3, we derive the proposed SKMMC algorithm in detail. In Section 4, we analyze the complexity of SKMMC theoretically. Several experiments are conducted on a wide range of real-world data sets and the results are shown in Section 5. In Section 6, some concluding remarks are drawn.

2. Related Works

2.1 Maximum margin clustering

The maximum margin clustering is to extend the theory of supervised support vector machine (SVM) to unsupervised learning scenario. Given the unlabeled samples $\bar{\mathbf{x}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ with $\mathbf{x}_i \subseteq \mathcal{R}^N$, MMC aims at finding the best label combination $\bar{\mathbf{y}} = \{y_1, \ldots, y_n\}$ with $y_i \in \{-1, +1\} \triangleq \mathcal{Y}$, such that an SVM trained on $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ will yield the largest margin. It could be formed as the following computational optimization problem

$$\begin{aligned} &\min_{\bar{y} \in \{\pm 1\}^n} \min_{\mathbf{w}, b, \xi_i \geq 0} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i \\ &\qquad s.t.\ \forall i \in \{1, \ldots, n\} : y_i \left(\mathbf{w}^T \phi(\mathbf{x}_i) + b\right) \geq 1 - \xi_i, \end{aligned} \tag{1}$$

where $\phi(\cdot)$ is the mapping function used to map $\mathbf{x}_i$ into a possibly high-dimensional kernel space.

One problem of MMC is that it is possible to classify all samples to only one class with a very large margin. In order to avoid this, Xu et al. [24] used the following balance constraint to control the class balance

$$-l \leq \sum_{i=1}^{n} y_i \leq l \tag{2}$$

Wang [22] further revised the class balance constraint as

$$-l \leq \sum_{i=1}^{n} \left(\mathbf{w}^T \phi(\mathbf{x}_i) + b\right) \leq l, \tag{3}$$

where $l \geq 0$ is a constant.

2.2 Cutting-plane maximum margin clustering

Wang proposed CPMMC algorithm [22] to solve (1) efficiently. It firstly reformulated the n-slacks problem in (1) as the following 1-slack problem

$$\begin{aligned} &\min_{\mathbf{w}, b, \xi \geq 0} \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \\ &\qquad s.t.\ \forall \mathbf{c} \in \{0,1\}^n : \frac{1}{n}\sum_{i=1}^{n} c_i - \xi - \frac{1}{n}\sum_{i=1}^{n} c_i \cdot \left|\mathbf{w}^T \phi(\mathbf{x}_i) + b\right| \leq 0 \end{aligned} \tag{4}$$

Though (4) is non-convex, the first constraints of (4) could be decomposed to a sum of a convex function and a concave function. Thus, the constrained concave-convex procedure (CCCP) [17] is employed to get a saddle point of (4) by solving the following convex quadratic programming (QP) iteratively

$$\begin{aligned} &\min_{\mathbf{w}, b, \xi \geq 0} \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \\ &\qquad s.t.\ \forall \mathbf{c} \in \{0,1\}^n : \frac{1}{n}\sum_{i=1}^{n} c_i - \xi - \frac{1}{n}\sum_{i=1}^{n} c_i \hat{y}_i \left(\mathbf{w}^T \phi(\mathbf{x}_i) + b\right) \leq 0, \end{aligned} \tag{5}$$

where $\hat{\mathbf{y}} = [\hat{y}_1 \ldots \hat{y}_n]^T$ are the predicted results from previous CCCP iteration (the tth iteration) with each element of $\hat{\mathbf{y}}$ defined as

$$\hat{y}_i = \operatorname{sign}\left(\mathbf{w}^T \phi(\mathbf{x}_i) + b\right) \tag{6}$$

However, there are 2^n constraints in (5) which make the problem (5) difficult to solve directly. In order to solve (5) efficiently, the well known cutting-plane algorithm [9, 20, 6] is employed to construct an approximate solution of (5). More precisely, assuming the current working constraint set of the cutting-plane algorithm is Ω with a total constraint number $|\Omega|$, we could get an approximate solution of (5) by iteratively adding the most violated constraint to $\mathbf{\Omega}$ and solving the following optimization problems until no violations of constraints are detected.

$$\begin{aligned} &\min_{\mathbf{w},b,\xi\geq 0} \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \\ &\qquad s.t.\ \forall\ k \in \{1,\dots,|\Omega|\} : \frac{1}{n}\sum_{i=1}^{n} c_{k,i} - \xi - \frac{1}{n}\sum_{i=1}^{n} c_{k,i}\hat{y}_i \left(\mathbf{w}^T\phi(\mathbf{x}_i)+b\right) \leq 0 \end{aligned} \tag{7}$$

The most violated constraint is obtained by [22]

$$c_{|\Omega|+1,i} = \begin{cases} 1, & \text{if } \hat{y}_i \left(\mathbf{w}^T\phi(\mathbf{x}_i)+b\right) \leq 1 \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

However, its efficiency is only reflected in linear kernel. As stated in the Introduction section, $\phi(\mathbf{x}_i)$ could only be obtained by matrix decomposition, which is very time consuming. A common thought to apply the nonlinear kernels is to move to its dual.

3. MMC with Nonlinear Kernels

3.1 Adaptive threshold CPMMC

In fact, the CPMMC algorithm in [22] solves the following n-slack optimization problem (9) iteratively, which is reformulated to an equivalent 1-slack problem (4) and is solved by the cutting-plane algorithm.

$$\begin{aligned} &\min_{\mathbf{w},b,\xi_i\geq 0} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\ &\qquad s.t.\ \forall i \in \{1,\dots,n\} : \hat{y}_i \left(\mathbf{w}^T\phi(\mathbf{x}_i)+b\right) \geq 1-\xi_i \end{aligned} \tag{9}$$

However, in most cases, $\left\{\mathbf{w}^T\phi(\mathbf{x}_i)+b\right\}_{i=1}^{n}$ is very small. If we still use the constant threshold 1 as in (9) in this situation, the inner cutting plane iterations would only add the all 1 constant constraint vector $\mathbf{c}$ time and time again ($\mathbf{c}_k = \{c_{k,i}\}_{i=1}^{n} = \mathbf{1}_{n\times 1}, k = 1,\dots,|\Omega|$). More directly, no new constraint vector is generated, and the cutting-plane algorithm fails.

For the robustness of the MMC, we consider the adaptive threshold scheme by adding the parameter ρ into the objective and reformulate it as

$$\begin{aligned} &\min_{\mathbf{w},b,\rho,\xi_i\geq 0} \frac{1}{2}\|\mathbf{w}\|^2 - \rho + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\ &\qquad s.t.\ \forall i \in \{1,\dots,n\} : \hat{y}_i \left(\mathbf{w}^T\phi(\mathbf{x}_i)+b\right) \geq \rho-\xi_i \end{aligned} \tag{10}$$

After a similar derivation with the CPMMC, we could obtain the objective of the inner cutting plane as:

$$\begin{aligned}\min_{\mathbf{w},b,\rho\geq 0,\xi\geq 0}\ & \frac{1}{2}\|\mathbf{w}\|^2-\rho+C\xi\\ s.t.\ & \forall\ k\in\{1,\ldots,|\Omega|\}:\\ & \frac{\rho}{n}\sum_{i=1}^{n}c_{k,i}-\xi-\frac{1}{n}\sum_{i=1}^{n}c_{k,i}\hat{y}_i\left(\mathbf{w}^T\phi(\mathbf{x}_i)+b\right)\leq 0\end{aligned}\tag{11}$$

with the most violated constraint adapted as

$$c_{|\Omega|+1,i}=\begin{cases}1, & \text{if } \hat{y}_i\left(\mathbf{w}^T\phi(\mathbf{x}_i)+b\right)\leq\rho\\ 0, & \text{otherwise}\end{cases}\tag{12}$$

and the convergence condition of the inner cutting-plane iteration revised as

$$\frac{\rho}{n}\sum_{i=1}^{n}c_{|\Omega|+1,i}-\frac{1}{n}\sum_{i=1}^{n}c_{|\Omega|+1,i}y_i\left(\mathbf{w}^T\phi(\mathbf{x}_i)+b\right)\leq\xi+\rho\eta,\tag{13}$$

where η is a user defined constant balancing the cutting-plane solution precision and the training time of the inner cutting-plane algorithm.

Because ρ is regarded as a constant 1 in the outer CCCP iteration, when we quit the inner cutting-plane iteration, we would have to remove the influence of the parameter ρ. Suppose that the optimized parameter set of the inner cutting-plane algorithm is $(\mathbf{w}^*,b^*,\rho^*,\xi^*)$ with the objective value of (11) as $\mathcal{J}_{CP}$, we remove the scaling influence of ρ^* by defining the output of the inner cutting-plane algorithm as:

$$\begin{aligned}&\mathbf{w}_o=\frac{\mathbf{w}^*}{\rho^*},\ b_o=\frac{b^*}{\rho^*},\ \xi_o=\frac{\xi^*}{\rho^*}\\ &\mathcal{J}_o=\frac{1}{2}\|\mathbf{w}_o\|^2+C\xi_o\end{aligned}\tag{14}$$

From (14), it is clear that the objective value $\mathcal{J}_o$ is a normalized one, which is different from $\mathcal{J}_{CP}$. We could also know that the definition of (10) is just used to derive (11). $\mathcal{J}_o$ is used in the same way as [22]. We present the usage of $\mathcal{J}_o$ shortly as follows.

The convergence condition of the CCCP is defined as follows. Given the normalized output value of the objective (11) at present CCCP iteration $\mathcal{J}_o$ and that at previous iteration $\mathcal{J}_o^{pre}$, the convergence condition of the CCCP is defined as:

$$\left|\frac{\mathcal{J}_o}{\mathcal{J}_o^{pre}}-1\right|<\epsilon,\tag{15}$$

where ϵ is the user defined CCCP solution precision balancing the solution precision and the clustering time.

We summarize the adaptive threshold based CPMMC in Algorithm 1. Note that Algorithm 1 is not sensitive to parameter C, which enhances the robustness of CPMMC.

Algorithm 1: Adaptive Threshold Cutting Plane MMC.

```
     (CCCP iteration)
1:   repeat: solve (5) under constraint (3) by using inner cutting plane iterations, and
     get the pseudo labels $\hat{\mathbf{y}}$.
       (Adaptive threshold based cutting-plane iteration)
2:     initialization $\boldsymbol{\Omega} \leftarrow \emptyset$, $t \leftarrow 0$, input $\hat{\mathbf{y}}$.
3:     repeat:
4:       $t \leftarrow t + 1$.
5:       Solve problem (11) and get the solution $(\mathbf{w}_t, b_t, \rho, \xi)$.
6:       Calculate the most violated constraint $\mathbf{c}_t$ from (12).
7:       Renew $\Omega$: $\Omega \leftarrow \Omega \cup \mathbf{c}_t$.
8:     until (13) is satisfied.
9:     return $\mathcal{J}_o$ from (14), $\hat{\mathbf{y}}$ from (6).
10:  until (15) is satisfied.
```

3.2 Kernel CPMMC

As mentioned in Section 2, CPMMC could be regarded as a serial sub-linear SVM problems which are solved in their primal forms (5), if we want to extend it to the non-linear case with kernels, we need to move to its dual representation. In this section, we focus on the inner cutting-plane SVM problem only.

By using the Karush-Kuhn-Tucker (KKT) conditions [15], we could write the Lagrangian of (11) under class balance constraint (3) as

$$\begin{aligned}
&\mathcal{L}(\mathbf{w}, b, \rho, \xi) \\
=&\frac{1}{2}\mathbf{w}^T\mathbf{w} - \rho + C\xi + \sum_{k=1}^{|\Omega|}\lambda_k\left\{\frac{\rho}{n}\sum_{i=1}^{n}c_{k,i} - \frac{1}{n}\sum_{i=1}^{n}c_{k,i}\cdot\hat{y}_i\left[\mathbf{w}^T\phi(\mathbf{x}_i)+b\right] - \xi\right\} \\
&+\frac{1}{n}\mu_1\left\{\sum_{i=1}^{n}\left[\mathbf{w}^T\phi(\mathbf{x}_i)+b\right] - l\right\} + \frac{1}{n}\mu_2\left\{-\sum_{i=1}^{n}\left[\mathbf{w}^T\phi(\mathbf{x}_i)+b\right] - l\right\} \\
&-\varsigma\xi - \nu\rho,
\end{aligned} \tag{16}$$

where $\boldsymbol{\lambda}$, μ_1, μ_2, ς, ν are non-negative Lagrangian variables. Calculating the partial derivatives with respect to the primal variables,

$$\frac{\partial\mathcal{L}}{\partial\mathbf{w}} = 0, \quad \frac{\partial\mathcal{L}}{\partial b} = 0, \quad \frac{\partial\mathcal{L}}{\partial\xi} = 0, \quad \frac{\partial\mathcal{L}}{\partial\rho} = 0 \tag{17}$$

we could get

$$\mathbf{w} = \frac{1}{n}\sum_{k=1}^{|\Omega|}\lambda_k\sum_{i=1}^{n}c_{k,i}\hat{y}_i\phi(x_i) - \frac{1}{n}(\mu_1-\mu_2)\sum_{i=1}^{n}\phi(x_i) \tag{18}$$

$$\mu_1 - \mu_2 = \frac{1}{n}\sum_{k=1}^{|\Omega|}\lambda_k\sum_{i=1}^{n}c_{k,i}\hat{y}_i \tag{19}$$

$$\frac{1}{n}\sum_{k=1}^{|\Omega|}\sum_{i=1}^{n} c_{k,i} - 1 - \nu = 0 \tag{20}$$

$$C - \sum_{k=1}^{|\Omega|}\lambda_k - \varsigma = 0 \tag{21}$$

After denoting $\mathbf{K} \triangleq \mathbf{K}(\bar{\mathbf{x}}, \bar{\mathbf{x}})$ and $\bar{\mathbf{c}} = [\mathbf{c}_1, \ldots, \mathbf{c}_{|\Omega|}] = \begin{bmatrix} c_{1,1} \cdots c_{|\Omega|,1} \\ \vdots \ \ddots \ \vdots \\ c_{1,n} \cdots c_{|\Omega|,n} \end{bmatrix}$, we could get the Lagrangian dual of (7) as the following matrix form by substituting (18)–(21) to (16)

$$\begin{aligned}
\max_{\boldsymbol{\lambda},\mu_1,\mu_2} & -\frac{1}{2n^2}\boldsymbol{\lambda}^T\bar{\mathbf{c}}^T\mathrm{diag}(\hat{\mathbf{y}})\mathbf{K}\mathrm{diag}(\hat{\mathbf{y}})\bar{\mathbf{c}}\boldsymbol{\lambda} + \frac{1}{n^2}(\mu_1-\mu_2)\boldsymbol{\lambda}^T\bar{\mathbf{c}}^T\mathrm{diag}(\hat{\mathbf{y}})\mathbf{K}\mathbf{1}_{n\times 1} \\
& -\frac{1}{2n^2}(\mu_1-\mu_2)^2\mathbf{1}_{1\times n}\mathbf{K}\mathbf{1}_{n\times 1} - \frac{1}{n}(\mu_1+\mu_2)l \\
s.t.\ & \mu_1 - \mu_2 = \frac{1}{n}\lambda^T\bar{\mathbf{c}}^T\hat{\mathbf{y}},\ \frac{1}{n}\boldsymbol{\lambda}^T\bar{\mathbf{c}}^T\mathbf{1}_{n\times 1} \geq 1 \\
& C - \sum_{k=1}^{|\Omega|}\lambda_k \geq 0,\ \mu_1 \geq 0,\ \mu_2 \geq 0,\ \boldsymbol{\lambda} \geq 0
\end{aligned} \tag{22}$$

with $\mathbf{w}$ calculated from (18), where $\mathbf{1}_{x\times y}$ denotes an $x \times y$ size matrix with all entries equaling to 1.

However, the bias term b cannot be derived directly from (22), which is different from the n-slack supervised SVM problem. We developed another simple calculation method of b from the class balance constraint (3). If $n \gg l$, we could use *squeeze rule* to get b approximately as

$$b \approx -\frac{1}{n}\sum_{i=1}^{n}\mathbf{w}^T\phi(\mathbf{x}_i) \tag{23}$$

Though we could get b into consideration by adding a constant feature to each sample which is commonly used in supervised SVM [6, 2], empirically, it is inferior to the proposed calculation method in the MMC problems.

Until now we have obtained the parameters $(\mathbf{w}, b)$ of the maximum margin hyperplane at the current cutting-plane working constraint set Ω.

From the fact that the QPs have zero duality gap for strong convex problem, and from the implicit KKT conditions, we could derive ρ as (24) and ξ as (23).

$$\rho = \frac{n\left(\sum_{k=1}^{|\Omega|}\lambda_k\right)\left\{\|\mathbf{w}\|^2 + \frac{1}{n}(\mu_1+\mu_2)l\right\} - C\sum_{k=1}^{|\Omega|}\lambda_k\sum_{i=1}^{n}c_{k,i}\hat{y}_i\left(\mathbf{w}^T\phi(\mathbf{x}_i)+b\right)}{n\sum_{k=1}^{|\Omega|}\lambda_k - C\sum_{k=1}^{|\Omega|}\lambda_k\sum_{i=1}^{n}c_{k,i}} \tag{24}$$

$$\xi = \frac{\rho\sum_{k=1}^{|\Omega|}\lambda_k\sum_{i=1}^{n}c_{k,i} - \sum_{k=1}^{|\Omega|}\lambda_k\sum_{i=1}^{n}c_{k,i}\hat{y}_i\left(\mathbf{w}^T\phi(\mathbf{x}_i)+b\right)}{n\sum_{k=1}^{|\Omega|}\lambda_k} \tag{23}$$

Note that **the parameter C could not be set to 1 in practice, because it might make the denominator of (24) to be zero.**

Until now all parameters have been calculated. And then, the most violated constraint of the inner cutting-plane algorithm is obtained by (12). The convergence condition of the cutting-plane algorithm is the same as (13).

The outer CCCP algorithm is the same as Algorithm 1. We summarize the kernel CPMMC algorithm in Algorithm 2.

Algorithm 2: Kernel Cutting Plane MMC.

```
    **initialization.**
1:  Input: $\bar{\mathbf{x}}$, CCCP solution precision $\epsilon$, cutting-plane solution precision $\eta$, kernel parameters.
    **(CCCP iteration)**
2:  **repeat**: solve (5) and get the pseudo labels $\hat{\mathbf{y}}$.
      **(Adaptive threshold based cutting-plane iteration)**
3:    **initialization** $t \leftarrow 1$, $\hat{\mathbf{y}}$, $\mathbf{c}_1 \leftarrow \mathbf{1}_{n\times 1}$, $\mathbf{\Omega} \leftarrow \emptyset$
4:    **repeat**:
5:      Renew $\Omega$: $\Omega \leftarrow \Omega \cup \mathbf{c}^{(t)}$.
6:      $t \leftarrow t+1$.
7:      Solve problem (22) and get the solution $(\boldsymbol{\lambda}, \mu_1, \mu_2)$.
        (Note: $\mathbf{w}_t$ is obtained implicitly from (18))
8:      Calculate $b_t$ from (23)
9:      $\hat{y}_i \leftarrow \mathrm{sign}(\mathbf{w}^T\phi(\mathbf{x}_i)+b), \quad i=1,\dots,n$
10:     Calculate $\rho$ from (24), $\xi$ from (23)
11:     Calculate $\mathbf{c}^{(t)}$ from (12).
12:   **until** (13) is satisfied.
13:   **return** $\mathcal{J}_o$ from (14), $\hat{\mathbf{y}}$ from (6)
14: **until** (15) is satisfied.
```

Any inner cutting plane iteration takes at most $\mathcal{O}\left(|\Omega|^3\right)$ for solving the QP, $\mathcal{O}(n)$ for the most violated constraints $\mathbf{c}$ and for predicting labels $\hat{\mathbf{y}}$ respectively. But it takes $\mathcal{O}((|\Omega|+1)n^2 + 2|\Omega|n)$ for the objective function (22)[1,2] and $\mathcal{O}(n^2 + |\Omega|n)$ for $\left\{\mathbf{w}^T\phi(\mathbf{x}_i)+b\right\}_{i=1}^{n}$. Therefore, the overall complexity of the kernel CP-MMC is scaled with $\mathcal{O}\left(Ttn^2\right)$, where t and T are the average iteration numbers of the inner cutting plane algorithm and the outer CCCP algorithm, respectively. Because the kernel matrix has to be stored, the storage complexity of the kernel CPMMC is scaled with $\mathcal{O}(n^2)$. Hence, the kernel CPMMC algorithm is hard to deal with large scale data sets.

[1]Though the third item has $\mathcal{O}(n^2)$, we do not take it into account since it could be calculated once.

[2]A single kernel calculation is evaluated as $\mathcal{O}(1)$.

3.3 Sparse Kernel MMC

The overall computational complexity $\mathcal{O}(n^2)$ of the kernel CPMMC is mainly caused by the following operator

$$\mathbf{w} = \frac{1}{n}\sum_{k=1}^{|\Omega|}\lambda_k\mathbf{\Psi}_k - \frac{1}{n}(\mu_1-\mu_2)\mathbf{\Psi}_0 \tag{24}$$

which has a computational load of $\mathcal{O}(n)$, where Ψ is defined as

$$\mathbf{\Psi}_k=\frac{1}{n}\sum_{i=1}^{n}c_{k,i}\hat{y}_i\phi(x_i), \quad k=1,\ldots,|\Omega| \tag{25}$$

$$\mathbf{\Psi}_0=\frac{1}{n}\sum_{i=1}^{n}\phi(x_i) \tag{26}$$

Could we remove the expensive $\mathcal{O}(n)$ scaling behavior? Recently, Joachims has proposed a sparse solution of the normal vector $\mathbf{w}$ of the maximum margin hyperplane, called cutting-plane subspace pursuit (CPSP) [7], to eliminate the $\mathcal{O}(n)$ for the supervised structural SVM problem. In this paper, we employ it to accelerate the kernel CPMMC algorithm, which is the proposed sparse kernel MMC (SKMMC) algorithm.

The core idea of the SKMMC algorithm is to find a small set of basis vectors $\bar{\mathbf{b}}_x = \{\mathbf{b}_{x,i}\}_{i=1}^{p_x}$ for each $\mathbf{\Psi}_x$, such that $\mathbf{\Psi}_x$ could be approximated as

$$\hat{\mathbf{\Psi}}_x=\sum_{i=1}^{p_x}\beta_{x,i}\phi(\mathbf{b}_{x,i}), \quad x=0,1,\ldots,|\Omega|, \tag{27}$$

where $\boldsymbol{\beta}_x=[\beta_{x,1},\ldots,\beta_{x,p_x}]^T$ are the coefficients of the basis vectors in the high-dimension kernel space, and should be estimated as well. Therefore, the approximation of $\mathbf{w}$ is formulated as

$$\hat{\mathbf{w}} = \sum_{k=1}^{|\Omega|}\lambda_k\hat{\mathbf{\Psi}}_k - (\mu_1-\mu_2)\hat{\mathbf{\Psi}}_0 \tag{28}$$

Substituting $\hat{\mathbf{w}}$ back to Algorithm 2 could get the approximation of the objective function (22) as the following matrix form

$$\begin{aligned}\max_{\boldsymbol{\lambda},\mu_1\geq 0,\mu_2\geq 0} & -\frac{1}{2}\boldsymbol{\lambda}^T\bar{\boldsymbol{\beta}}^T\mathbf{K}(\bar{\bar{\mathbf{b}}},\bar{\bar{\mathbf{b}}})\bar{\boldsymbol{\beta}}\boldsymbol{\lambda}+(\mu_1-\mu_2)\boldsymbol{\lambda}^T\bar{\boldsymbol{\beta}}^T\mathbf{K}(\bar{\bar{\mathbf{b}}},\bar{\mathbf{b}}_0)\boldsymbol{\beta}_0\\ & -\frac{1}{2}(\mu_1-\mu_2)^2\boldsymbol{\beta}_0^T\mathbf{K}(\bar{\mathbf{b}}_0,\bar{\mathbf{b}}_0)\boldsymbol{\beta}_0-\frac{1}{n}(\mu_1+\mu_2)l\\ s.t.\ \mu_1-\mu_2 = & \frac{1}{n}\lambda^T\bar{\mathbf{c}}^T\hat{\mathbf{y}},\ \frac{1}{n}\boldsymbol{\lambda}^T\bar{\mathbf{c}}^T\mathbf{1}_{n\times 1}\geq 1\\ C-\sum_{k=1}^{|\Omega|}\lambda_k\geq 0,\ & \mu_1\geq 0,\ \mu_2\geq 0,\ \boldsymbol{\lambda}\geq 0,\end{aligned} \tag{29}$$

where $\bar{\bar{\mathbf{b}}} = \{\bar{\mathbf{b}}_1, \ldots, \bar{\mathbf{b}}_{|\Omega|}\} = \{\mathbf{b}_{1,1}, \ldots, \mathbf{b}_{1,p_1}, \ldots, \mathbf{b}_{1,p_{|\Omega|}}, \ldots, \mathbf{b}_{|\Omega|,p_{|\Omega|}}\}^3$, and $\bar{\boldsymbol{\beta}} = \begin{bmatrix} \boldsymbol{\beta}_1 & & \\ & \ddots & \\ & & \boldsymbol{\beta}_{|\Omega|} \end{bmatrix}$.

For calculation convenience, (29) could be rewritten in a standard QP form as

$$\max_{\boldsymbol{\gamma}} -\frac{1}{2}\boldsymbol{\gamma}^T \mathbf{H}\boldsymbol{\gamma} - \boldsymbol{\gamma}^T \boldsymbol{\zeta}$$
$$s.t.\ \mathbf{A}_{eq}\boldsymbol{\gamma} = 0;\ \mathbf{A}\boldsymbol{\gamma} \le \mathbf{d};\ \boldsymbol{\gamma} \ge 0, \tag{30}$$

where the following notations are used to formulate the objective: $\boldsymbol{\gamma} = [\boldsymbol{\lambda}, \mu_1, \mu_2]^T$, $\mathbf{A} = \begin{bmatrix} -\frac{1}{n}\bar{\mathbf{c}}^T \mathbf{1}_{n\times 1} & 0 & 0 \\ \mathbf{1}_{1\times|\Omega|} & 0 & 0 \end{bmatrix}$, $\mathbf{d} = [-1\ C]^T$, $\mathbf{A}_{eq} = \begin{bmatrix} \frac{1}{n}\hat{\mathbf{y}}^T\bar{\mathbf{c}} & -1 & 1 \end{bmatrix}$, $\boldsymbol{\zeta} = \begin{bmatrix} -\frac{1}{n}\bar{\mathbf{c}}^T \mathbf{1}_{n\times 1} & \frac{l}{n} & \frac{l}{n} \end{bmatrix}^T$, and

$$\bar{\bar{\boldsymbol{\beta}}} = \begin{bmatrix} \bar{\boldsymbol{\beta}} & & \\ & \boldsymbol{\beta}_0 & \\ & & \boldsymbol{\beta}_0 \end{bmatrix} \qquad \mathbf{H} = \bar{\bar{\boldsymbol{\beta}}}^T \begin{bmatrix} \mathbf{K}(\bar{\bar{\mathbf{b}}}, \bar{\bar{\mathbf{b}}}) & -\mathbf{K}(\bar{\bar{\mathbf{b}}}, \bar{\mathbf{b}}_0) & \mathbf{K}(\bar{\bar{\mathbf{b}}}, \bar{\mathbf{b}}_0) \\ -\mathbf{K}(\bar{\mathbf{b}}_0, \bar{\bar{\mathbf{b}}}) & \mathbf{K}(\bar{\mathbf{b}}_0, \bar{\mathbf{b}}_0) & -\mathbf{K}(\bar{\mathbf{b}}_0, \bar{\mathbf{b}}_0) \\ \mathbf{K}(\bar{\mathbf{b}}_0, \bar{\bar{\mathbf{b}}}) & -\mathbf{K}(\bar{\mathbf{b}}_0, \bar{\mathbf{b}}_0) & \mathbf{K}(\bar{\mathbf{b}}_0, \bar{\mathbf{b}}_0) \end{bmatrix} \bar{\bar{\boldsymbol{\beta}}}$$

All other parts of the SKMMC algorithm are the same as the kernel CPMMC algorithm except that $\mathbf{w}$ should be replaced by $\hat{\mathbf{w}}$. As a conclusion, the SKMMC algorithm is summarized in Algorithm 3.

Note that recomputing the basis vector set related to the entire constraint set Ω in each iteration is costly and unnecessary. In each CCCP subproblem, only $\boldsymbol{\Psi}_{|\Omega|}$ is new and all other $\boldsymbol{\Psi}$s are already well approximated by the set of basis vectors from previous cutting-plane iteration.

The basis vector estimation algorithm is the same as that used in CPSP algorithm [7]. Only the RBF kernel is applicable at present. Only one basis vector is used to estimate a single $\boldsymbol{\Psi}$. For completeness of this paper, we append the basis estimation algorithm in C.

4. Theoretical Analysis

4.1 Computational complexity

Suppose the basis vector number for each $\hat{\boldsymbol{\Psi}}_x$ is 1, where x is any of $\{0, 1, \ldots, |\Omega|\}$. For each CCCP iteration, the SKMMC takes at most $\mathcal{O}(|\Omega|^3)$ for the QP, $\mathcal{O}(n)$ for the most violated constraints $\mathbf{c}$, $\mathcal{O}\left((|\Omega|+1)n\right)$ for $\left\{\hat{\mathbf{w}}^T\phi(\mathbf{x}_i) + \hat{b}\right\}_{i=1}^n$. For formulating the objective (30), it takes $\mathcal{O}\left(|\Omega|(|\Omega|+1)^2\right)$ for $\mathbf{H}$, $\mathcal{O}(|\Omega|n)$ for $\mathbf{A}_{eq}$, therefore, the total computational complexity for each CCCP iteration of Algorithm 3 is $\mathcal{O}(2t(|\Omega|+1)n)$, where t is the average cutting-plane iteration number for each CCCP sub-problem.

Additionally, it takes about $\mathcal{O}(udsn)$ for each basis vector estimation, where u is the average iteration number for the convergence of BEF algorithm (Algorithm 4), d is the dimension of the sample, s is the sparsity of the data set.

[3] $\bar{\bar{\mathbf{b}}}$ is arranged in order.

Algorithm 3: Sparse Kernel MMC.

```
     initialization.
1:   Input: $\bar{\mathbf{x}}$, CCCP solution precision $\epsilon$, cutting-plane solution precision $\eta$, kernel parameters.
2:   $\hat{\mathbf{\Psi}}_0 \leftarrow estimate\_basis(\mathbf{\Psi}_0)$
     (CCCP iteration)
3:   repeat: solve (5) and get the pseudo labels $\hat{\mathbf{y}}$.
       (Adaptive threshold based cutting-plane iteration)
4:     initialization $t \leftarrow 1$, input $\hat{\mathbf{y}}$ and $\hat{\mathbf{\Psi}}_0$, random constraint vector $\mathbf{c}_1$, $\mathbf{\Omega} \leftarrow \emptyset$
5:     repeat:
6:       $\hat{\mathbf{\Psi}}_t \leftarrow estimate\_basis(\mathbf{\Psi}_t)$
7:       Renew $\Omega$: $\Omega \leftarrow \Omega \cup \mathbf{c}^{(t)}$.
8:       $t \leftarrow t + 1$.
9:       Solve problem (30) and get the solution $\boldsymbol{\gamma}$.
         (Note: $\hat{\mathbf{w}}_t$ is obtained implicitly from (28))
10:      Calculate $\hat{b}_t$ from (23)
11:      $\hat{y}_i \leftarrow \mathrm{sign}(\hat{\mathbf{w}}_t^T \phi(\mathbf{x}_i) + \hat{b}_t), \quad i = 1, \ldots, n$
12:      Calculate $\rho$ from (24), $\xi$ from (23)
13:      Calculate $\mathbf{c}^{(t)}$ from (12).
14:    until (13) is satisfied.
15:    return $\mathcal{J}_o$ from (14), $\hat{\mathbf{y}}$ from (6)
16:  until (15) is satisfied.
```

Suppose the SKMMC algorithm needs T iterations to converge to a local minimum, **the overall computational complexity for large-scale data set is scaled with** $\mathcal{O}\left((uds(T+1) + 2tT(|\Omega|+1))n\right)$, which is more efficient than existing kernel MMC algorithms.

4.2 Storage complexity

The whole data set requires $\mathcal{O}(sdn)$ space, all basis vectors need $\mathcal{O}((|\Omega|+1)d)$ space, the $\mathbf{\Omega}$ needs $\mathcal{O}(|\Omega|n)$ space. Because $|\Omega|$ is usually very small, the memory used to store $\mathbf{H}$ in Algorithm 3 could be omitted, this is a very significant merit of the SKMMC since all existing kernel MMC algorithms have to store the gram matrix $\mathbf{K}$ of the whole data set which requires $\mathcal{O}(n^2)$ space. As a conclusion, **the overall storage complexity is about** $\mathcal{O}\left((sd + |\Omega|)n\right)$, which has a linear relationship with the data set size.

5. Experimental Analysis

In this section, we will firstly compare our SKMMC algorithm with several existing clustering methods on various real-world small scale data sets at first. And then, we will illustrate the scaling behavior of the SKMMC on several large scale data sets. At last, we will discuss in detail how the parameter ϵ affect the performance of the

SKMMC algorithm. All experiments are conducted with Matlab 7.8 on a 2.4 GHZ Iter(R) Core(TM)2 Duo PC running Windows XP with 4 GB main memory.

5.1 Comparison schemes and experimental settings

To examine the effectiveness of the proposed SKMMC algorithm, we compare it with the following existing data clustering methods.[4]

- K-Means (KM) [5]. All performances reported are averaged over 20 independent runs.[5]
- Normalized Cut (NC) [16].
- Iterative Support Vector Regression (IterSVR) [27].[6] RBF kernel is adopted. Parameter settings are exactly the same as [27] did. All performances reported are averaged over 20 independent runs.
- Cutting Plane Maximum Margin Clustering (CPMMC) [22].[7] According to [22], only the linear kernel is used in all experiments. Parameter settings are exactly the same as [22] did.
- Label-Generating Maximum Margin Clustering (LG-MMC) [10].[8] According to [10], if the data sets are small scale ($n < 10000$), the Gaussian kernel is used; otherwise, the linear kernel is used. All parameters are set exactly the same as Li did [10] in his experiments.

For the proposed SKMMC algorithm, the following parameter settings are adopted. The CCCP solution precision ϵ is set to 0.1. The inner cutting plane solution precision η is set to 0.01. The width σ of the Gaussian RBF kernel $\exp(-\|\mathbf{x}\|^2/2\sigma^2)$ is searched through $\{0.25\sqrt{\gamma}, 0.5\sqrt{\gamma}, \sqrt{\gamma}, 2\sqrt{\gamma}, 4\sqrt{\gamma}\}$ which is similar to the setup of the RBF kernel based LG-MMC. In order to compare with CPMMC in a similar setting, the parameter C is set to 10, and the class balance constraint, i.e., l is searched through $[0, 20]$ with granularity 1 as well. Only one basis vector is used to approximate each $\mathbf{\Psi}$.

In order to show the effectiveness of ρ, we developed another dual form SKMMC version (SKMMCv2) where ρ is set to a constant 1, and presented it in B. All parameters are set to the same values as SKMMC.

For performance evaluation, the balanced clustering error *B_Err* [26] is adopted. It has been proved to be a more reliable metric than clustering error rate *Err*,

[4]Because the MMC algorithm proposed by Xu [24] and the Generalized MMC (GMMC) algorithm proposed by Valizadegan [21] is based on SDP, and can only be run with at most several hundreds of samples on our PC, hence, we would not compare with them anymore.

[5]The implementation code is in the VOICEBOX developed by Cambridge University for speech processing, it can be downloaded from "http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox /doc/voicebox/kmeans.html".

[6]The implementation code is downloaded from "http://www3.ntu.edu.sg/home/IvorTsang /itMMC_code.zip".

[7]The implementation code is downloaded from "http://sites.google.com/ site/binzhao02".

[8]The implementation code is downloaded from "http://cs.nju.edu.cn/zhouzh/zhouzh.files /publication/annex/LGMMC_v2.rar".

especially when the data set is severely class imbalanced. *B_Err* is defined as

$$B\_Err = \frac{1}{2}(Err^+ + Err^-), \tag{31}$$

where Err^+ and Err^- are the error rates of the positive and negative samples in the full set. Furthermore, the imbalanced degree of the data set is defined as

$$\text{Imbalance}^\circ = \frac{|Num^+ - Num^-|}{Num^+ + Num^-} \times 100\%, \tag{32}$$

where Num^+ and Num^- are the numbers of the positive and negative samples in the full set, respectively.

5.2 Small scale experiments

In this section, we present small scale experiments suitable for comparing our algorithm with other clustering methods. Tab. I lists the small scale UCI data sets we use.[9] For *Satellite* and *Waveform*, they have multiple classes, we use the first two classes of the *Satellite* (C1 versus C2) and the last two classes of the *Waveform* (W1 versus W2).

Data	**# Sample**	**# Feature**	**Imbalance°**	**Sparsity°**
Echocardiog	132	8	34.35	96.95
Spectf	267	44	58.80	100.00
Heart-stalog	270	13	11.11	75.10
Haberman	306	3	47.06	85.19
LiveDisorders	345	6	15.94	99.57
Ionosphere	351	34	28.21	88.09
House-votes	435	16	22.76	100.00
Clean1	476	166	13.03	99.84
Breast	683	9	31.04	99.75
Australian	690	14	11.01	93.90
German	1000	24	40.00	74.95
Satellite1vs2	2236	36	38.23	100.00
Krvskp	3196	36	4.44	100.00
Waveform1vs2	3308	40	0.06	99.70
Spambase	4601	57	21.19	71.51

Tab. I *Small scale experimental data sets and their properties.*

The clustering results of our algorithm and other referenced methods are shown in Tab. II. From the table, the SKMMC algorithm has better performances than SKMMCv2 and other referenced methods in 8 data sets. And the SKMMC algorithm can outperform the SKMMCv2 and CPMMC algorithms in most of the data sets, which proved the effectiveness of the parameter ρ empirically.

[9] All UCI data sets are downloaded from "http://archive.ics.uci.edu/ml".

Data	KM	NC	IterSVR	CPMMC	LG-MMC	SKMMC	SKMMCv2
Echocardiog	16.07	18.42	**8.05**	16.01	19.04	13.69	15.37
Spectf	38.71	41.84	39.31	28.30	32.41	**18.63**	21.46
Heart-stalog	42.10	36.00	38.89	40.00	28.85	**28.25**	37.92
Haberman	48.25	47.83	**35.13**	48.07	41.83	35.16	35.31
LiveDisorders	48.90	45.79	41.86	42.11	36.11	**35.86**	42.79
Ionosphere	29.33	34.92	27.92	28.21	36.81	**21.29**	23.02
House-votes	13.81	49.44	14.48	12.71	14.36	**11.52**	11.55
Clean1	48.95	31.00	44.33	19.18	33.77	**7.01**	28.36
Breast	5.86	8.37	4.43	3.41	23.12	**2.70**	2.89
Australian	49.84	43.22	37.22	37.07	**24.68**	33.49	33.59
German	44.10	42.07	**33.03**	43.67	45.38	38.92	43.71
Satellite1vs2	6.89	8.07	3.80	2.65	13.10	**1.37**	2.41
Krvskp	45.39	49.96	42.29	43.49	**36.17**	37.08	38.04
Waveform1vs2	5.26	35.52	5.50	48.15	39.17	5.02	**4.78**
Spambase	44.18	26.63	32.92	37.97	**13.93**	23.14	23.49

Tab. II *Balanced error rate (B_Err) (%) of different clustering methods.*

Data	KM	NC	IterSVR	CPMMC	LG-MMC	SKMMC	SKMMCv2
Echocardiog	0.01	0.05	0.23 (5.00)	0.06 (1)	0.38 (5)	0.84 (7)	0.44 (6)
Spectf	0.02	0.31	1.29 (3.00)	0.04 (1)	1.39 (5)	0.76 (4)	0.30 (5)
Heart-stalog	0.01	0.17	2.46 (4.00)	0.07 (1)	1.08 (5)	0.15 (2)	0.51 (7)
Haberman	0.00	0.15	0.10 (10.15)	3.16 (1)	1.41 (5)	3.28 (11)	0.08 (4)
LiveDisorders	0.01	0.27	0.25 (5.30)	0.96 (2)	1.66 (5)	0.62 (3)	0.11 (4)
Ionosphere	0.02	0.29	2.97 (3.85)	0.08 (1)	2.05 (5)	1.67 (4)	0.55 (4)
House-votes	0.02	0.49	5.44 (3.20)	0.19 (2)	3.06 (5)	9.13 (14)	1.71 (19)
Clean1	0.20	2.59	16.34 (12.00)	3.97 (6)	5.61 (5)	10.95 (26)	0.77 (4)
Breast	0.01	1.02	9.53 (3.00)	0.09 (1)	8.09 (5)	0.34 (3)	0.21 (4)
Australian	0.04	1.19	1.52 (4.00)	0.08 (1)	9.34 (5)	0.47 (3)	0.20 (5)
German	0.06	2.86	13.07 (3.15)	0.16 (1)	27.42 (5)	2.01 (4)	3.11 (17)
Satellite1vs2	0.13	9.03	16.57 (5.00)	0.63 (2)	79.97 (5)	2.33 (3)	0.30 (3)
Krvskp	0.23	39.12	1180.40 (11.25)	0.49 (1)	634.98 (5)	17.44 (5)	1.63 (3)
Waveform1vs2	0.95	312.03	945.08 (4.00)	3.10 (2)	645.81 (5)	6.47 (4)	5.09 (6)
Spambase	0.88	203.60	460.04 (12.00)	0.58 (1)	1216.92 (5)	7.95 (6)	9.03 (6)

Tab. III *CPU time (in seconds) and iteration numbers (in the brackets) of different clustering methods.*

The CPU time of our algorithm and the competitive MMCs is reported in Tab. III. From the table we could conclude that the SKMMC, the SKMMCv2 and the CPMMC algorithms are the most efficient methods among the SVM-type algorithms, while the runtime of the other two is increasing dramatically with the data set size.

5.3 Large scale experiments

For the sake of simplicity, we set $C = 10$, $l = 0$, $\eta = 1$, $\epsilon = 0.3$, and search σ in the same way as in the small scale experiments. For the competitive methods, all parameters are set to the same values as in the small scale experiments.

5.3.1 Experiments on MNIST

In the first large scale experiment, we conduct experiments on the MNIST hand-written digit data set, as a 2-class clustering problem. For the digits of the MNIST data, we follow [10, 27] and focus on the pairs that are difficult to differentiate. The details of the selected digital pairs are listed in Tab. IV.

Data	**# Sample**	**# Feature**	**Imbalance$^\circ$**	**Sparsity$^\circ$**
1vs7	15170	784	3.85	13.75
2vs7	14283	784	2.12	19.10
3vs8	13966	784	2.26	21.48
8vs9	13783	784	0.96	20.20

Tab. IV *Selected MNIST digital pairs and their properties.*

The clustering results and the CPU time of the proposed SKMMC algorithm and three other competitive algorithms are shown in Tab. V and Tab. VI, respectively.

Data	**KM**	**CPMMC**	**LG-MMC**	**SKMMC**
1vs7	4.23	2.85	3.90	**2.81**
2vs7	4.44	**3.65**	11.24	4.15
3vs8	20.06	19.50	13.16	**13.07**
8vs9	18.68	6.41	16.00	**5.87**

Tab. V *Balanced error rate (B_Err) of different clustering methods (%) on MNIST.*

Data	**KM**	**CPMMC**	**LG-MMC**	**SKMMC**
1vs7	47.24	7.27 (2)	19292.94 (20)	25.88 (2)
2vs7	49.85	9.30 (2)	22111.89 (20)	44.23 (2)
3vs8	95.38	10.98 (2)	11060.67 (13)	57.46 (2)
8vs9	114.80	9.76 (2)	51585.47 (20)	63.92 (2)

Tab. VI *CPU time (in seconds) and iteration numbers (in brackets) of different clustering methods on MNIST.*

From the tables, we could see clearly that the SKMMC could achieve lower *B_Err* than the LG-MMC algorithm and is hundreds of times faster than LG-MMC, which proves the superiority of the SKMMC algorithm to the LG-MMC algorithm. Although the SKMMC is several times slower than the linear kernel based CPMMC algorithm, it has better clustering results than the CPMMC algorithm. Moreover, the SKMMC is even faster than the KM algorithm.

5.3.2 Experiments on UCI adult data set

In the second large scale experiment, we use the UCI adult data set in a similar way as [26]. More precisely, a serial subsets of adult data set is formed[10], ranging in size [1605, 2265, 3185, 4781, 6414, 11220, 16100, 22696, 32561]. The imbalanced degree of the adult data set is 51.84% and the sparsity is 11.28%.

Experiments are shown in Fig. 1. From the figure, we could see that the SKMMC could achieve the best *B_Err* over all referenced methods.

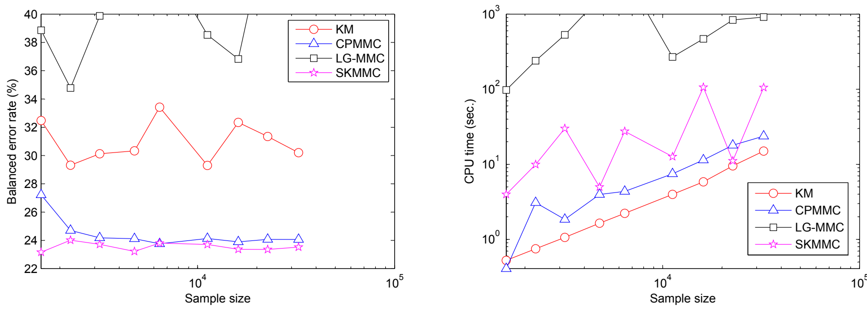

Fig. 1 *Comparison of different clustering methods on UCI adult data set. The balanced error rates (%) of different methods are in the left figure. The CPU time (in seconds) of different methods is in the right figure. Note that for LG-MMC, if the sample size is smaller than 10000, the RBF kernel is used; otherwise, the linear kernel is used.*

5.3.3 Experiments on extended USPS digits data set

The extended USPS digits data set[11] is developed from the digital classes "zero and one" of the USPS set for the purpose of studying the scaling behavior of the CVM [19]. It has a training set size of 266079 and a test set size of 75383 with 676 attributes. It also has a serial predefined subsets, ranging in size of [1000, 3000, 10000, 30000, 100000]. Its imbalanced degree is 8.63% and its sparsity is 14.95%. In our experiments, we use the subsets and the training set to study the scaling behavior of the proposed SKMMC algorithm and its competitive methods.

Experimental results are shown in Fig. 2. From the figure we could see that the proposed SKMMC algorithm yields much better *B_Err* curve than the other clustering methods while consuming comparable CPU time with the KM algorithm.

As a conclusion, although we sacrifice the solution precision by setting η to a relatively large value, we could still reach a better performance than existing clustering methods while keeping a low computational complexity.

[10]http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
[11]http://www.cs.ust.hk/~ivor/cvm.html

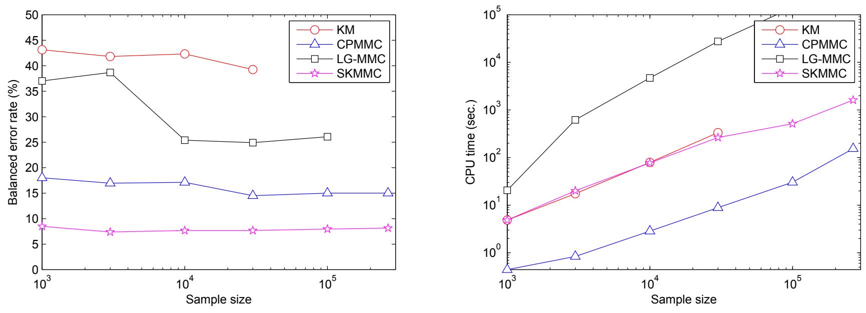

Fig. 2 *Comparison of different clustering methods on extended USPS data set. The balanced error rates (%) of different methods are in the left figure. The CPU time (in seconds) of different methods is in the right figure. Note that for LG-MMC, if the sample size is smaller than 10000, the RBF kernel is used; otherwise, the linear kernel is used.*

5.4 How does the cutting plane solution precision η affect the performance?

The parameter η is very important to balance the clustering accuracy and the runtime of the cutting-plane algorithm. To investigate the optimal working region of η, η is searched through $(0, 1]$ on the small-scale data sets, the CCCP solution precision ϵ is set to 0.1, and all other parameters are the same as in the small scale data set experiments.

The experimental results are shown from Fig. 3 to Fig. 6. From the figures, **firstly**, we could see that the cutting plane solution precision has small effects on most of the data sets. Only one or two cuts per CCCP iteration are sufficient to offer a meaningful final solution precision. Therefore, in practice, if the data sets are not highly sensitive to η, we could set η to a relatively large value and sacrifice some clustering precision for the efficiency of SKMMC as we have done in the large scale experiments. **Secondly**, η has small effects on the outer CCCP iteration number.

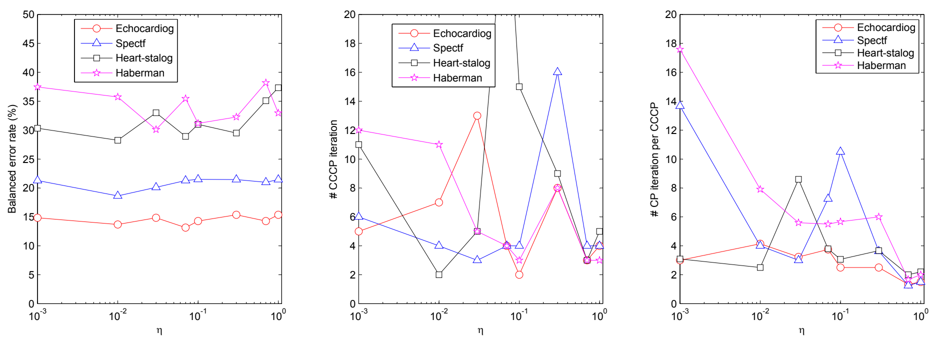

Fig. 3 *Effects of the cutting-plane (CP) solution precision η on the Echocardiog, Spectf, Heart-stalog and Haberman data sets.*

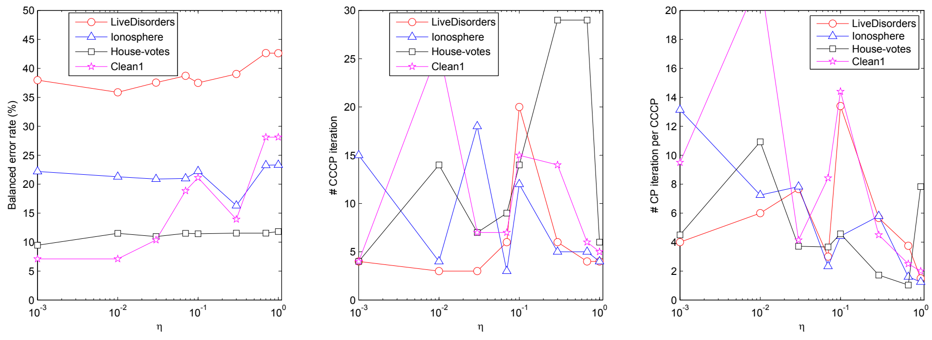

Fig. 4 *Effects of the cutting-plane (CP) solution precision η on the LiveDisorders, Ionosphere, House-votes and Clean1 data sets.*

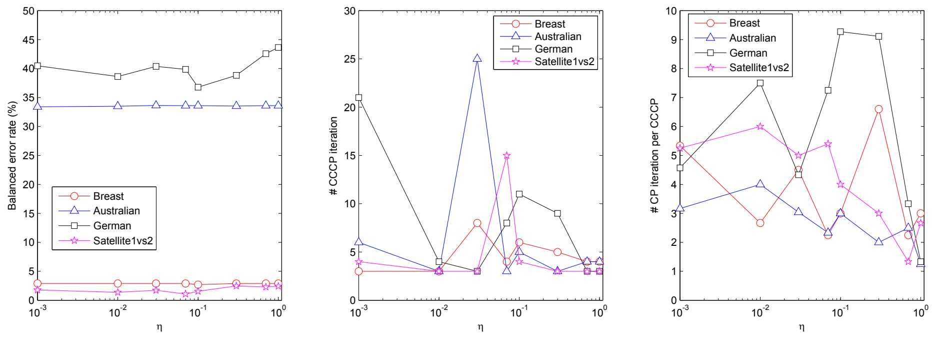

Fig. 5 *Effects of the cutting-plane (CP) solution precision η on the Breast, Australian, German, and Satellite1vs2 data sets.*

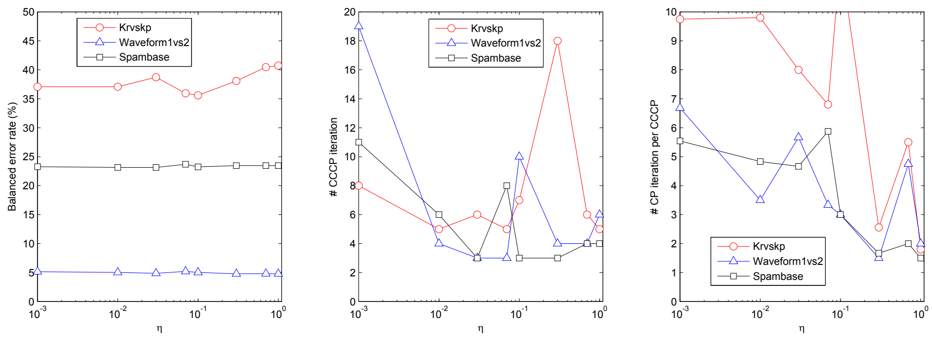

Fig. 6 *Effects of the cutting-plane (CP) solution precision η on the Krvskp, Waveform1vs2, and Spambase data sets.*

6. Conclusions

In this paper, we have proposed the sparse kernel MMC algorithm. The SKMMC algorithm used the CCCP algorithm to deal with the non-convex of the MMC problem, and uses cutting-plane algorithm to deal with the exponential constraints of each sub-CCCP problem. We proposed to solve each CCCP subproblem in its dual with an adapted threshold, named as ρ based kernel CPMMC. Then we accelerated the kernel CPMMC by employing the CPSP algorithm. Eventually, the SKMMC could work with nonlinear kernel at a linear computational complexity and storage complexity, which was the most efficient existing kernel MMC algorithm. Our experiments on a large amount of real-world data sets proved the effectiveness and efficiency of the proposed algorithm.

Acknowledgements

This work was supported by The National High-Tech. R&D Program of China (863 Program) under Grant 2006AA010104.

A Derivation of ρ and ξ

For the convergence condition of the inner cutting-plane iteration, we should derive the 1-slack variable ξ at first. From the KKT implicit conditions of (16), we have

$$\forall k \le |\Omega| : \lambda_k \left\{ \rho \sum_{i}^{n} c_{k,i} - n\xi - \sum_{i=1}^{n} c_{k,i}\hat{y}_i \left(\mathbf{w}^T\phi(x_i) + b\right) \right\} = 0 \tag{33}$$

Summing above equations over k could get ξ as:

$$\xi = \frac{\rho \sum_{k=1}^{|\Omega|} \lambda_k \sum_{i=1}^{n} c_{k,i} - \sum_{k=1}^{|\Omega|} \lambda_k \sum_{i=1}^{n} c_{k,i}\hat{y}_i \left(\mathbf{w}^T\phi(\mathbf{x}_i) + b\right)}{n \sum_{k=1}^{|\Omega|} \lambda_k} \tag{34}$$

However, ρ is unknown.

In fact, the objective (22) could be depicted as

$$\begin{aligned} &\max -\frac{1}{2}\|\mathbf{w}\|^2 - \frac{1}{n}(\mu_1+\mu_2)l \\ &s.t.\ \mu_1 - \mu_2 = \frac{1}{n}\lambda^T\bar{\mathbf{c}}^T\hat{\mathbf{y}},\ \frac{1}{n}\boldsymbol{\lambda}^T\bar{\mathbf{c}}^T\mathbf{1}_{n\times 1} \ge 1 \\ &C - \sum_{k=1}^{|\Omega|}\lambda_k \ge 0,\ \mu_1 \ge 0,\ \mu_2 \ge 0 \end{aligned} \tag{35}$$

since $\mathbf{w}$ is defined as (18).

From the fact that the QPs have zero duality gap, we have

$$\frac{1}{2}\|\mathbf{w}\|^2 - \rho + C\xi = -\frac{1}{2}\|\mathbf{w}\|^2 - \frac{1}{n}(\mu_1+\mu_2)l \tag{36}$$

Solving above equation by substituting (34) could derive ρ as (24). Substituting ρ back to (34) could get the parameter ξ.

B Sparse Kernel CPMMC without ρ

In this section, we focus on the kernel CPMMC without ρ only. Its sparse version (SKMMCv2) could be similar to SKMMC.

By using the Karush-Kuhn-Tucker (KKT) conditions [15], we could get the Lagrangian dual of (7) as

$$\begin{aligned}\max_{\boldsymbol{\lambda}\geq 0,\mu_1\geq 0,\mu_2\geq 0} & -\frac{1}{2n^2}\boldsymbol{\lambda}^T\bar{\mathbf{c}}^T\mathrm{diag}(\hat{\mathbf{y}})\mathbf{K}\mathrm{diag}(\hat{\mathbf{y}})\bar{\mathbf{c}}\boldsymbol{\lambda}+\frac{1}{n^2}(\mu_1-\mu_2)\boldsymbol{\lambda}^T\bar{\mathbf{c}}^T\mathrm{diag}(\hat{\mathbf{y}})\mathbf{K}\mathbf{1}_{n\times 1}\\ & -\frac{1}{2n^2}(\mu_1-\mu_2)^2\mathbf{1}_{1\times n}\mathbf{K}\mathbf{1}_{n\times 1}+\frac{1}{n}\boldsymbol{\lambda}^T\bar{\mathbf{c}}^T\mathbf{1}_{n\times 1}-\frac{1}{n}(\mu_1+\mu_2)l\\ & s.t.\ \mu_1-\mu_2=\frac{1}{n}\lambda^T\bar{\mathbf{c}}^T\hat{\mathbf{y}},\ C-\sum_{k=1}^{|\Omega|}\lambda_k\geq 0\end{aligned}\tag{37}$$

with $\mathbf{w}$ calculated as

$$\begin{aligned}\mathbf{w}=&\sum_{k=1}^{|\Omega|}\lambda_k\boldsymbol{\Psi}_k-(\mu_1-\mu_2)\boldsymbol{\Psi}_0\\ =&\frac{1}{n}\sum_{k=1}^{|\Omega|}\lambda_k\sum_{i=1}^{n}c_{k,i}\hat{y}_i\phi(\mathbf{x}_i)-\frac{1}{n}(\mu_1-\mu_2)\sum_{i=1}^{n}\phi(\mathbf{x}_i),\end{aligned}\tag{38}$$

where $\mathbf{K}(\bar{\mathbf{x}},\bar{\mathbf{x}})=[K(\mathbf{x}_i,\mathbf{x}_j)]\in\mathcal{R}^{n\times n}$ is the kernel matrix, and $\bar{\mathbf{c}}=[\mathbf{c}_1,\ldots,\mathbf{c}_{|\Omega|}]=\begin{bmatrix}c_{1,1}\cdots c_{|\Omega|,1}\\ \vdots\ \ddots\ \vdots\\ c_{1,n}\cdots c_{|\Omega|,n}\end{bmatrix}$.

For the convergence condition of the inner cutting-plane iteration, we should derive the 1-slack variable ξ at first. From the KKT implicit conditions, we have

$$\begin{aligned}&\forall k\leq|\Omega|:\\ &\lambda_k\left\{\sum_{i}^{n}c_{k,i}-n\xi-\sum_{i=1}^{n}c_{k,i}\hat{y}_i\left(\mathbf{w}^T\phi(\mathbf{x}_i)+b\right)\right\}=0\end{aligned}\tag{39}$$

Summing above equations over k could get ξ as:

$$\xi=\frac{\sum_{k=1}^{|\Omega|}\lambda_k\sum_{i=1}^{n}c_{k,i}-\sum_{k=1}^{|\Omega|}\lambda_k\sum_{i=1}^{n}c_{k,i}\hat{y}_i\left(\mathbf{w}^T\phi(\mathbf{x}_i)+b\right)}{n\sum_{k=1}^{|\Omega|}\lambda_k}\tag{40}$$

Therefore, the convergence condition of the cutting-plane algorithm is defined as:

$$\frac{1}{n}\sum_{i=1}^{n}c_{|\Omega|+1,i}-\frac{1}{n}\sum_{i=1}^{n}c_{|\Omega|+1,i}\hat{y}_i\left(\mathbf{w}^T\phi(\mathbf{x}_i)+b\right)\leq\xi+\eta,\tag{41}$$

where η is a user defined constant balancing the cutting-plane solution precision and the training time of the inner cutting-plane algorithm. The cutting-plane algorithm continues to add the most violated constraint to Ω until the convergence condition is satisfied.

The outer CCCP algorithm is the same as the CPMMC algorithm [22]. More specifically, given the objective value of (37) at present CCCP iteration $\mathcal{J}$ and that at previous iteration $\mathcal{J}^{pre}$, the convergence condition of the CCCP is defined as:

$$\left| \frac{\mathcal{J}}{\mathcal{J}^{pre}} - 1 \right| < \epsilon, \tag{42}$$

where ϵ is the user defined CCCP solution precision.

By employing the CPSP algorithm, we could find a small set of basis vectors $\bar{\mathbf{b}}_x = \{\mathbf{b}_{x,i}\}_{i=1}^{p_x}$ for each $\mathbf{\Psi}_x$, such that $\mathbf{\Psi}_x$ could be approximated, which is the proposed SKMMCv2.

C Estimation of the Basis Vector

We denote $\bar{\mathbf{b}}_x = \{\mathbf{b}_{x,i}\}_{i=1}^{p_x}$, where x is any of $\{\alpha, 0, 1, \ldots, |\Omega|\}$. Our main job in this section is to get the basis vector set $\bar{\mathbf{b}}_x$ for each $\mathbf{\Psi}_x$ such that the minimum square error (MSE) between $\mathbf{\Psi}_x$ and $\hat{\mathbf{\Psi}}_x$ could be minimized:

$$\begin{aligned}\varepsilon_{MSE} =& \|\mathbf{\Psi}_x - \hat{\mathbf{\Psi}}_x\|^2 \\ =& \frac{1}{n^2}\mathbf{z}_x^T\mathbf{K}(\bar{\mathbf{x}}, \bar{\mathbf{x}})\mathbf{z}_x - \frac{2}{n}\boldsymbol{\beta}_x^T\mathbf{K}(\bar{\mathbf{b}}_x, \bar{\mathbf{x}})\mathbf{1}_{n\times 1} + \boldsymbol{\beta}_x^T\mathbf{K}(\bar{\mathbf{b}}_x, \bar{\mathbf{b}}_x)\boldsymbol{\beta}_x\end{aligned} \tag{43}$$

For simplicity, the subscript x is omitted below.

In practice, we add the basis vector one by one for a single $\mathbf{\Psi}$, so that the ε_{MSE} could be lowered gradually. To decide which basis vector to add, we follow [7, 15] and aim at getting the basis vector $\mathbf{b}_{m+1}$ that minimizes the residual error for $\mathbf{\Psi}$.

$$\min_{\beta_{m+1}, \mathbf{b}_{m+1}} \left\| (\mathbf{\Psi} - \hat{\mathbf{\Psi}}) - \beta_{m+1}\phi(\mathbf{b}_{m+1}) \right\|^2, \tag{44}$$

where $\hat{\mathbf{\Psi}} = \sum_{i=1}^m \beta_i \phi(\mathbf{b}_i)$ at present. As stated in [7], the approximate solutions of above optimization problem could be found by using gradient-based method or randomized search. In this paper, we only consider the fixed point iteration approach (in Chapter 18 of [15]) for the RBF kernel. Therefore, the optimal $\mathbf{b}_{m+1}$ satisfies the following partial derivative

$$\frac{\partial < (\Psi - \hat{\Psi}), \phi(\mathbf{b}_{m+1}) >}{\partial \mathbf{b}_{m+1}} = 0 \tag{45}$$

Solving (45) iteratively by greedy algorithm, we could get $\mathbf{b}_{m+1}$ ($m > 1$) as

$$\mathbf{b}_{m+1}^{(t+1)} = \frac{\sum_{i=1}^n z_i K(\mathbf{x}_i, \mathbf{b}_{m+1}^{(t)})\mathbf{x}_i - n\sum_{i=1}^m \beta_i K(\mathbf{b}_i, \mathbf{b}_{m+1}^{(t)})\mathbf{b}_i}{\sum_{i=1}^n z_i K(\mathbf{x}_i, \mathbf{b}_{m+1}^{(t)}) - n\sum_{i=1}^m \beta_i K(\mathbf{b}_i, \mathbf{b}_{m+1}^{(t)})} \tag{46}$$

if $m = 1$, then

$$\mathbf{b}_1^{(t+1)} = \frac{\sum_{i=1}^n z_i K(\mathbf{x}_i, \mathbf{b}_1^{(t)})\mathbf{x}_i}{\sum_{i=1}^n z_i K(\mathbf{x}_i, \mathbf{b}_1^{(t)})} \tag{47}$$

After getting $\mathbf{b}_{m+1}$, the coefficient vector $\boldsymbol{\beta} = [\beta_1, \dots, \beta_{m+1}]^T$ needs to be re-optimized

$$\min_{\boldsymbol{\beta}} \left\| (\boldsymbol{\Psi} - \hat{\boldsymbol{\Psi}}) \right\|^2 \tag{48}$$

solving (48) could get $\boldsymbol{\beta}$ as

$$\boldsymbol{\beta} = \frac{1}{n} \mathbf{K}(\bar{\mathbf{b}}, \bar{\mathbf{b}})^{-1} \mathbf{K}(\bar{\mathbf{b}}, \bar{\mathbf{x}}) \mathbf{z} \tag{49}$$

Though the above algorithm could get a good approximation of $\boldsymbol{\Psi}$, it is too computationally expensive to calculate ε_{MSE} since we have to calculate $\mathbf{K}(\bar{\mathbf{x}}, \bar{\mathbf{x}})$. In practice, we set the number of basis vectors to a constant N without considering ε_{MSE} anymore. In the end, we summarize the basis vector estimation algorithm in Algorithm 4.

We note that setting $N = 1$, $\delta = 0.01$ always works well and the iteration number for estimating each basis vector is independent of n. For the robustness of the algorithm, **firstly**, $z \neq \mathbf{0}$ due to that the denominator of (47) cannot be zero. If $z = \mathbf{0}$, we should restart $\mathbf{z}$ to another random vector. Note that $z = \mathbf{0}$ seldom happens in the proposed BVMMC algorithm. **Secondly**, we initialize $\mathbf{b}_m^{(1)}$ by averaging some samples randomly chosen from $\bar{\mathbf{x}}$.

Algorithm 4: Estimate Basis Function For RBF Kernel.

```
1:  initialization.
2:  Input: $\mathbf{z}$ ($\mathbf{z} \neq \mathbf{0}$), $\bar{\mathbf{x}}$, estimating precision $\delta$ for each basis vector, RBF kernel
    parameter $\sigma$, total basis vector number $N$.
3:  add $\mathbf{b}_m$
4:    initialize $\mathbf{b}_m^{(1)}$ with random value , $t \leftarrow 1$.
5:    repeat
6:      $t \leftarrow t + 1$.
7:      if $m = 1$
8:        Get $\mathbf{b}_1^{(t+1)}$ from (47).
9:      else
10:       Get $\mathbf{b}_m^{(t+1)}$ from (46).
11:   until $\|\mathbf{b}_m^{(t+1)} - \mathbf{b}_m^{(t)}\|^2 \leq \delta$
14:   $\boldsymbol{\beta} = \frac{1}{n}\mathbf{K}(\bar{\mathbf{b}}, \bar{\mathbf{b}})^{-1}\mathbf{K}(\bar{\mathbf{b}}, \bar{\mathbf{x}})\mathbf{z}$.
15: until $m = N$
16: return $(\boldsymbol{\beta}, \bar{\mathbf{b}})$.
```

References

[1] Chapelle O., Zien A.: Semi-supervised classification by low density separation. In: Proc. 10th Int. Workshop Artif. Intell. Statist., 2005.

[2] Franc V., Sonnenburg S.: Optimized cutting plane algorithm for support vector machines. In: Proc. 25th Int. Conf. Mach. Learn., 2008, pp. 320–327.

[3] Gath I., Geva A. B.: Unsupervised optimal fuzzy clustering. IEEE Trans. Pattern Anal. Mach. Intell., **11**, 7, 2002, pp. 773–780.

[4] Gieseke F., Pahikkala T., Kramer O.: Fast evolutionary maximum margin clustering. In: Proc. 26th Int. Conf. Mach. Learn., 2009, pp. 361–368.

[5] Hartigan J. A., Wong M. A.: A k-means clustering algorithm. Appl. Statist., 28, 1979, pp. 100–108.

[6] Joachims T., Finley T., Yu C. N. J.: Cutting-plane training of structural SVMs. Mach. Learn., **77**, 1, 2009, pp. 27–59.

[7] Joachims T., Yu C. N. J.: Sparse kernel SVMs via cutting-plane training. Mach. Learn., **76**, 2, 2009, pp. 179–193.

[8] Kannan R., Vempala S., Vetta A.: On clusterings: Good, bad and spectral. J. ACM, **51**, 3, 2004, pp. 497–515.

[9] Kelley J. E.: The cutting-plane method for solving convex programs. J. Soc. Ind. Appl. Math., **8**, 4, 1960, pp. 703–712.

[10] Li Y. F., Tsang I. W., Kwok J. T., Zhou Z. H.: Tighter and convex maximum margin clustering. In: Proc. 12th Int. Conf. Artif. Intell. Statist., Clearwater Beach, FL, 2009, pp. 344–351.

[11] MacQueen J., et al.: Some methods for classification and analysis of multivariate observations. In: Proc. 5th Berkeley Sym. Math. Statist. Prob., **1**, pp. 281–297.

[12] McLachlan G. J., Peel D.: Finite Mixture Models. Wiley-Interscience, 2000.

[13] Andrew Y. Ng., Jordan M. F., Weiss Y.: On spectral clustering: Analysis and an algorithm. In: Advances in Neural Inform. Process. Systems, 2001, pp. 849–856.

[14] Schölkopf B., Smola A., Müller K.: Kernel principal component analysis. Artif. Neural Networks, 1997, pp. 583–588.

[15] Schölkopf B., Smola A. J.: Learning With Kernels. MIT Press, Cambridge, MA, 2002.

[16] Shi J., Malik J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell., **22**, 8, 2002, pp. 888–905.

[17] Smola A. J., Vishwanathan S. V. N., Hofmann T.: Kernel methods for missing variables. In: Proc. 10th Int. Workshop Artif. Intell. Statist., 2005, pp. 325–332.

[18] Trefethen L. N., Bau D. Numerical Linear Algebra. Society for Industrial Mathematics, 1997.

[19] Tsang I., Kwok J., Cheung P.: Core vector machines: Fast SVM training on very large data sets. J. Mach. Learn. Res., **6**, 1, 2006, pp. 363–392.

[20] Tsochantaridis I., Joachims T., Hofmann T., Altun Y.: Large margin methods for structured and interdependent output variables. J. Mach. Learn. Res., **6**, 2, 2006, pp. 1453–1484.

[21] Valizadegan H., Jin R.: Generalized maximum margin clustering and unsupervised kernel learning. **19**, pp. 1417–1425.

[22] Wang F., Zhao B., Zhang C. S.: Linear time maximum margin clustering. IEEE Trans. Neural Networks, **21**, 2, 2010, pp. 319–332.

[23] Xie X. L., Beni G.: A validity measure for fuzzy clustering. IEEE Trans. Pattern Anal. Mach. Intell., **13**, 8, 2002, pp. 841–847.

[24] Xu L., Neufeld J., Larson B., Schuurmans D.: Maximum margin clustering. In: Advances in Neural Inform. Process. Systems, 2005, **17**, pp. 1537–1544.

[25] Yuille A. L., Rangarajan A.: The concave-convex procedure. Neural Comput., **15**, 4, 2003, pp. 915–936.

[26] Zhang K., Tsang I. W., Kwok J. T.: Maximum margin clustering made practical. In Proc. 24th Int. Conf. Mach. Learn., 2007, pp. 1126–1134.

[27] Zhang K., Tsang I. W., Kwok J. T.: Maximum margin clustering made practical. IEEE Trans. Neural Networks, **20**, 4, 2009, pp. 583–596.