

# Efficient Maximum Margin Clustering via Cutting Plane Algorithm

Bin Zhao\*

Fei Wang

Changshui Zhang

## Abstract

*Maximum margin clustering (MMC)* is a recently proposed clustering method, which extends the theory of *support vector machine* to the unsupervised scenario and aims at finding the maximum margin hyperplane which separates the data from different classes. Traditionally, *MMC* is formulated as a *non-convex integer programming* problem and is thus difficult to solve. Several methods have been proposed in the literature to solve the *MMC* problem based on either *semi-definite programming* or *alternative optimization*. However, these methods are time demanding while handling large scale datasets and therefore unsuitable for real world applications. In this paper, we propose the *cutting plane maximum margin clustering (CPMMC)* algorithm, to solve the *MMC* problem. Specifically, we construct a nested sequence of successively tighter relaxations of the original *MMC* problem, and each optimization problem in this sequence could be efficiently solved using the *constrained concave-convex procedure (CCCP)*. Moreover, we prove theoretically that the *CPMMC* algorithm takes time  $O(sn)$  to converge with guaranteed accuracy, where  $n$  is the total number of samples in the dataset and  $s$  is the average number of non-zero features, i.e. the sparsity. Experimental evaluations on several real world datasets show that *CPMMC* performs better than existing *MMC* methods, both in efficiency and accuracy.

## 1 Introduction

Clustering [8] is one of the most fundamental research topics in both data mining and machine learning communities. It aims at dividing data into groups of similar objects, i.e. clusters. From a machine learning perspective, what clustering does is to learn the hidden patterns of the dataset in an unsupervised way, and these patterns are usually referred to as data concepts. Clustering plays an important role in many real world data mining applications such as scientific information retrieval and text mining, web analysis, marketing, computational biology, and many others [7].

Many clustering methods have been proposed in the literature over the decades, including *k-means clustering*

[6], *mixture models*[6] and *spectral clustering* [15, 2, 5]. More recently, *Xu et al.* [19] proposed *maximum margin clustering (MMC)*, which borrows the idea from the *support vector machine* theory and aims at finding the maximum margin hyperplane which can separate the data from different classes in an unsupervised way. Their experimental results showed that the *MMC* technique often obtains more accurate results than conventional clustering methods [19].

Technically, that *MMC* does is just to find a way to label the samples by running an *SVM* implicitly, and the *SVM* margin obtained would be maximized over all possible labelings [19]. However, unlike supervised large margin methods which are usually formulated as *convex optimization* problems, *maximum margin clustering* is a *non-convex integer optimization* problem, which is much more difficult to solve. [19] and [18] made several relaxations to the original *MMC* problem and reformulate it as a *semi-definite programming (SDP)* problem. Then *maximum margin clustering* solution could be obtained using standard *SDP* solvers such as *SDPA* and *SeDuMi*.

However, even with the recent advances in interior point methods, solving *SDPs* is still computationally very expensive [21]. Consequently, the algorithms proposed in [19] and [18] can only handle very small datasets containing several hundreds of samples. In real world applications such as scientific information retrieval and text mining, web analysis, and computational biology, the dataset usually contains a large amount of data samples. Therefore, how to efficiently solve the *MMC* problem to make it capable of clustering large scale dataset is a very challenging research topic. Very recently, *Zhang et al.* utilized alternating optimization techniques to solve the *MMC* problem [21], in which the *maximum margin clustering* result is obtained by solving a series of *SVM* training problems. However, there is no guarantee on how fast it can converge and the algorithm is still likely to be time demanding on large scale datasets.

In this paper, we propose a *cutting plane maximum margin clustering* algorithm *CPMMC*. Specifically, we construct a nested sequence of successively tighter relaxations of the original *MMC* problem, and each optimization problem in this sequence could be efficiently

\*State Key Laboratory on Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Automation, Tsinghua University, Beijing, China.

solved using the *constrained concave-convex procedure* (CCCP). Moreover, we prove theoretically that the CPMMC algorithm takes time  $O(sn)$  to converge with guaranteed accuracy, where  $n$  is the total number of samples in the dataset and  $s$  is the average number of non-zero features, i.e. the sparsity. Our experimental evaluations on several real world datasets show that CPMMC performs better than existing MMC methods, both in efficiency and accuracy.

The rest of this paper is organized as follows. In section 2, we introduce some works related to this paper. The CPMMC algorithm is presented in detail in section 3. In section 4, we provide a theoretical analysis on the accuracy and time complexity of the CPMMC algorithm. Experimental results on several real-world datasets are provided in section 5, followed by the conclusions in section 6.

## 2 Notations and Related Works

In this section we will briefly review some related works of this paper and establish the notations we will use.

**2.1 Maximum Margin Clustering** *Maximum margin clustering* (MMC) extends the theory of *support vector machine* (SVM) to the unsupervised scenario, where instead of finding a large margin classifier given labels on the data as in SVM, the target is to find a labeling that would result in a large margin classifier [19]. That is to say, if we subsequently run an SVM with the labeling obtained from MMC, the margin obtained would be maximal among all possible labelings.

Given a point set  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  and their labels  $\mathbf{y} = (y_1, \dots, y_n) \in \{-1, +1\}^n$ , SVM finds a hyperplane  $f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$  by solving the following optimization problem

$$(2.1) \quad \min_{\mathbf{w}, b, \xi_i} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^n \xi_i$$

$$s.t. \quad y_i (\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0 \quad i = 1, \dots, n$$

where the data samples  $\mathcal{X}$  are mapped into a high (possibly infinite) dimensional feature space using a possibly nonlinear function  $\phi(\mathbf{x})$ , and by using the kernel trick, this mapping could be done implicitly. Specifically, we define the kernel matrix  $K$  formed from the inner products of feature vectors, such that  $K_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ , and transform the problem such that  $\phi(\mathbf{x})$  only appears in the inner product. However, in those cases where kernel trick cannot be applied, if we still want to use a nonlinear kernel, it is possible to compute the coordinates of each sample in the *kernel*

*PCA basis* [14] according to the kernel  $K$ . More directly, as stated in [3], one can also compute the Cholesky decomposition of the kernel matrix  $K = \tilde{\mathbf{X}} \tilde{\mathbf{X}}^T$ , and set  $\phi(\mathbf{x}_i) = (\tilde{\mathbf{X}}_{i,1}, \dots, \tilde{\mathbf{X}}_{i,n})^T$ . Therefore, throughout the rest of this paper, we use  $\phi(\mathbf{x}_i)$  to denote the sample mapped by the kernel function  $\phi(\cdot)$ .

Different from SVM, where the class labels are given and the only variables are the hyperplane parameters  $(\mathbf{w}, b)$ , MMC targets to find not only the optimal hyperplane  $(\mathbf{w}^*, b^*)$ , but also the optimal labeling vector  $\mathbf{y}^*$  [19]

$$(2.2) \quad \min_{\mathbf{y} \in \{-1, +1\}^n} \min_{\mathbf{w}, b, \xi_i} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^n \xi_i$$

$$s.t. \quad y_i (\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0 \quad i = 1, \dots, n$$

However, the above optimization problem has a trivially “optimal” solution, which is to assign all patterns to the same class, and the resultant margin will be infinite. Moreover, another unwanted solution is to separate a single outlier or a very small group of samples from the rest of the data. To alleviate these trivial solutions, Xu *et al.* [19] introduced a class balance constraint on  $\mathbf{y}$

$$(2.3) \quad -l \leq \mathbf{e}^T \mathbf{y} \leq l$$

where  $l \geq 0$  is a constant controlling the class imbalance and  $\mathbf{e}$  is the all-one vector.

Similar as in SVM, [19] solves problem (2.2) in its dual form which is a non-convex integer optimization problem, and to get the MMC solution in reasonable time, several relaxations are made. The first one relaxes the labeling vector  $\mathbf{y}$  to take continuous values. The second one relaxes  $\mathbf{y}\mathbf{y}^T$  to a positive semi-definite matrix  $M$  with diagonal elements all set to 1. The final relaxation is to set the bias term  $b$  in the classifying hyperplane to 0. After making these relaxations, the dual problem is simplified into a *semi-definite programming* (SDP) problem. To ensure  $M = \mathbf{y}\mathbf{y}^T$ , a few more constraints are added into the SDP problem [19].

However, the number of parameters in the SDP problem scales quadratically with the number of samples in  $\mathcal{X}$  [19], and by setting the bias term  $b$  to zero restrains the classifying hyperplane to cross the origin of the feature space. To alleviate these deficiencies, Valizadegan and Jin proposed the *generalized maximum margin clustering* (GMMC) algorithm [18], in which the number of parameters is reduced from  $n^2$  to  $n$  and the bias term  $b$  could take non-zero values.

Nevertheless, MMC and GMMC both solve the maximum margin clustering problem via SDP, which could be quite time demanding when handling large

datasets. Therefore, *Zhang et al.* [21] proposed a simple alternating optimization technique which alternates between maximizing the margin w.r.t. the class label vector  $\mathbf{y}$  with the classifying hyperplane fixed and maximizing the margin w.r.t. the hyperplane parameter  $(\mathbf{w}, b)$  with  $\mathbf{y}$  fixed, until convergence.

**2.2 Concave-Convex Procedure** The *concave-convex procedure (CCCP)* [20] is a method for solving non-convex optimization problem whose objective function could be expressed as a difference of convex functions. It can be viewed as a special case of variational bounding [10] and related techniques including lower(upper) bound maximization(minimization) [13], surrogate functions and majorization [12]. While in [20] the authors only considered linear constraints, Smola et al. [16] proposed a generalization, the *constrained concave-convex procedure (CCCP)*, for problems with a concave-convex objective function under concave-convex constraints.

Assume we are solving the following optimization problem [16]

$$(2.4) \quad \min_{\mathbf{z}} \quad f_0(\mathbf{z}) - g_0(\mathbf{z})$$

$$s.t. \quad f_i(\mathbf{z}) - g_i(\mathbf{z}) \leq c_i \quad i = 1, \dots, n$$

where  $f_i$  and  $g_i$  are real-valued convex functions on a vector space  $\mathcal{Z}$  and  $c_i \in \mathcal{R}$  for all  $i = 1, \dots, n$ . Denote by  $T_1\{f, \mathbf{z}\}(\mathbf{z}')$  the first order *Taylor expansion* of  $f$  at location  $\mathbf{z}$ , that is  $T_1\{f, \mathbf{z}\}(\mathbf{z}') = f(\mathbf{z}) + \partial_{\mathbf{z}}f(\mathbf{z})(\mathbf{z}' - \mathbf{z})$ , where  $\partial_{\mathbf{z}}f(\mathbf{z})$  is the gradient of the function  $f$  at  $\mathbf{z}$ . For non-smooth functions, it can be easily shown that the gradient  $\partial_{\mathbf{z}}f(\mathbf{z})$  would be replaced by the subgradient [4]. Given an initial point  $\mathbf{z}_0$ , the *CCCP* computes  $\mathbf{z}_{t+1}$  from  $\mathbf{z}_t$  by replacing  $g_i(\mathbf{z})$  with its first-order Taylor expansion at  $\mathbf{z}_t$ , i.e.  $T_1\{g_i, \mathbf{z}_t\}(\mathbf{z})$ , and setting  $\mathbf{z}_{t+1}$  to the solution to the following relaxed optimization problem

$$(2.5) \quad \min_{\mathbf{z}} \quad f_0(\mathbf{z}) - T_1\{g_0, \mathbf{z}_t\}(\mathbf{z})$$

$$s.t. \quad f_i(\mathbf{z}) - T_1\{g_i, \mathbf{z}_t\}(\mathbf{z}) \leq c_i \quad i = 1, \dots, n$$

The above procedure continues until  $\mathbf{z}_t$  converges, and *Smola et al.* [16] proved that the *CCCP* is guaranteed to converge.

**2.3 Notations** We denote the number of samples in  $\mathcal{X}$  by  $n$  and the number of features for each sample by  $N$ . For the high-dimensional sparse data commonly encountered in applications like text mining, web log analysis and bioinformatics, we assume each sample has only  $s \ll N$  non-zero features, i.e.,  $s$  implies the sparsity.

### 3 Maximum Margin Clustering via Cutting Plane Algorithm

In this section, we first introduce a slightly different formulation of the *maximum margin clustering* problem that will be used throughout this paper and show that it is equivalent to the conventional *MMC* formulation. Then we present the main procedure of the *cutting plane maximum margin clustering (CPMMC)* algorithm.

**3.1 Cutting Plane Algorithm** Conventionally, *maximum margin clustering* could be formulated as the following optimization problem

$$(3.6) \quad \min_{\mathbf{y} \in \{-1, +1\}^n} \min_{\mathbf{w}, b, \xi_i} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + \frac{C}{n} \sum_{i=1}^n \xi_i$$

$$s.t. \quad y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0 \quad i = 1, \dots, n$$

where  $\sum_{i=1}^n \xi_i$  is divided by  $n$  to better capture how  $C$  scales with the dataset size. Existing *MMC* algorithms solve either problem (3.6) directly [21] or its dual form [19, 18]. The difficulty with problem (3.6) lies in the fact that we have to minimize the objective function w.r.t. the labeling vector  $\mathbf{y}$ , in addition to  $\mathbf{w}$ ,  $b$  and  $\xi_i$ . To reduce the variables involved, we could formulate the *MMC* problem as

$$(3.7) \quad \min_{\mathbf{w}, b, \xi_i} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + \frac{C}{n} \sum_{i=1}^n \xi_i$$

$$s.t. \quad |\mathbf{w}^T \phi(\mathbf{x}_i) + b| \geq 1 - \xi_i$$

$$\xi_i \geq 0 \quad i = 1, \dots, n$$

where the labeling vector  $\mathbf{y}$  is calculated as  $y_i = \text{sign}(\mathbf{w}^T \phi(\mathbf{x}_i) + b)$ . Moreover, we have the following theorem.

**THEOREM 3.1.** *Any solution to problem (3.7) is also a solution to problem (3.6) (and vice versa).*

*Proof.* We will show that for every  $(\mathbf{w}, b)$  the smallest feasible  $\sum_{i=1}^n \xi_i$  are identical for problem (3.6) and problem (3.7), and their corresponding labeling vectors are the same. For a given  $(\mathbf{w}, b)$ , the  $\xi_i$  in problem (3.6) can be optimized individually. According to the constraint in problem (3.6),

$$(3.8) \quad \xi_i \geq 1 - y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b)$$

As the objective is to minimize  $\frac{C}{n} \sum_{i=1}^n \xi_i$ , the optimal value for  $\xi_i$  is

$$\xi_i^{(1)} = \min_{y_i} \max\{0, 1 - y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b)\}$$

$$= \min\{\max\{0, 1 - \mathbf{w}^T \phi(\mathbf{x}_i) - b\}, \max\{0, 1 + \mathbf{w}^T \phi(\mathbf{x}_i) + b\}\}$$

and we denote the corresponding class label by  $y_i^{(1)}$ . Similarly, for problem (3.7), the optimal value for  $\xi_i$  is

$$\begin{aligned}\xi_i^{(2)} &= \max\{0, 1 - |\mathbf{w}^T \phi(\mathbf{x}_i) + b|\} \\ &= \max\{0, \min\{1 - \mathbf{w}^T \phi(\mathbf{x}_i) - b, 1 + \mathbf{w}^T \phi(\mathbf{x}_i) + b\}\}\end{aligned}$$

and the class label is calculated as  $y_i^{(2)} = \text{sign}(w^T \phi(x_i) + b)$ .  $\xi_i^{(1)}, \xi_i^{(2)}$  and  $y_i^{(1)}, y_i^{(2)}$  are determined by the value of  $\mathbf{w}^T \phi(\mathbf{x}_i) + b$ . If we arrange  $(-1, 0, 1, \mathbf{w}^T \phi(\mathbf{x}_i) + b)$  in ascending order, there are four possible sequences in total, and the corresponding values for  $\xi_i^{(1)}, \xi_i^{(2)}$  and  $y_i^{(1)}, y_i^{(2)}$  are shown in table 1, from which we see  $\xi_i^{(1)} = \xi_i^{(2)}$  and  $y_i^{(1)} = y_i^{(2)}$  always hold. For simplicity, define  $z_1 = 1 - \mathbf{w}^T \phi(\mathbf{x}_i) - b$  and  $z_2 = 1 + \mathbf{w}^T \phi(\mathbf{x}_i) + b$ ,

Relation	$\xi_i^{(1)}$	$\xi_i^{(2)}$	$y_i^{(1)}$	$y_i^{(2)}$
$\mathbf{w}^T \phi(\mathbf{x}_i) + b \leq -1$	0	0	-1	-1
$-1 < \mathbf{w}^T \phi(\mathbf{x}_i) + b \leq 0$	$z_2$	$z_2$	-1	-1
$0 < \mathbf{w}^T \phi(\mathbf{x}_i) + b \leq 1$	$z_1$	$z_1$	1	1
$\mathbf{w}^T \phi(\mathbf{x}_i) + b \geq 1$	0	0	1	1

Table 1: Proof of  $\xi_i^{(1)} = \xi_i^{(2)}$  and  $y_i^{(1)} = y_i^{(2)}$

Therefore, the objective functions of both optimization problems are equivalent for any  $(\mathbf{w}, b)$  with the same optimal  $\xi_i$ , and consequently so are their optima. Moreover, their corresponding labeling vectors  $\mathbf{y}$  are the same. Hence, we proved that problem (3.6) is equivalent to problem (3.7).  $\square$

By reformulating problem (3.6) as problem (3.7), the number of variables involved in the *MMC* problem is reduced by  $n$ , but there are still  $n$  slack variables  $\xi_i$  in problem (3.7). Next we will further reduce the number of variables by reformulating problem (3.7) into the following optimization problem

$$\begin{aligned}(3.9) \quad & \min_{\mathbf{w}, b, \xi \geq 0} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C\xi \\ & s.t. \quad \forall \mathbf{c} \in \{0, 1\}^n : \\ & \quad \frac{1}{n} \sum_{i=1}^n c_i |\mathbf{w}^T \phi(\mathbf{x}_i) + b| \geq \frac{1}{n} \sum_{i=1}^n c_i - \xi\end{aligned}$$

Although problem (3.9) has  $2^n$  constraints, one for each possible vector  $\mathbf{c} = (c_1, \dots, c_n) \in \{0, 1\}^n$ , it has only one slack variable  $\xi$  that is shared across all constraints, thus, the number of variables is further reduced by  $n - 1$ . Each constraint in this formulation corresponds to the sum of a subset of constraints from problem (3.7), and the vector  $\mathbf{c}$  selects the subset. Problem (3.7) and problem (3.9) are equivalent in the following sense.

**THEOREM 3.2.** *Any solution  $(\mathbf{w}^*, b^*)$  to problem (3.9) is also a solution to problem (3.7) (and vice versa), with  $\xi^* = \frac{1}{n} \sum_{i=1}^n \xi_i^*$ .*

*Proof.* Similar with the proof for theorem 3.1, we will show that problem (3.7) and problem (3.9) have the same objective value and an equivalent set of constraints. Specifically, we will prove that for every  $(\mathbf{w}, b)$ , the smallest feasible  $\xi$  and  $\sum_{i=1}^n \xi_i$  are related by  $\xi = \frac{1}{n} \sum_{i=1}^n \xi_i$ . This means, with  $(\mathbf{w}, b)$  fixed,  $(\mathbf{w}, b, \xi)$  and  $(\mathbf{w}, b, \xi_i)$  are optimal solutions to problem (3.7) and (3.9) respectively, and they result in the same objective function value.

For any given  $(\mathbf{w}, b)$ , the  $\xi_i$  in problem (3.7) can be optimized individually and the optimum is achieved as

$$(3.10) \quad \xi_i^{(2)} = \max\{0, 1 - |\mathbf{w}^T \phi(\mathbf{x}_i) + b|\}$$

Similarly for problem (3.9), the optimal  $\xi$  is

$$(3.11) \quad \xi^{(3)} = \max_{\mathbf{c} \in \{0, 1\}^n} \left\{ \frac{1}{n} \sum_{i=1}^n c_i - \frac{1}{n} \sum_{i=1}^n c_i |\mathbf{w}^T \phi(\mathbf{x}_i) + b| \right\}$$

Since each  $c_i$  are independent in Eq.(3.11), they can be optimized individually. Therefore,

$$\begin{aligned}\xi^{(3)} &= \sum_{i=1}^n \max_{c_i \in \{0, 1\}} \left\{ \frac{1}{n} c_i - \frac{1}{n} c_i |\mathbf{w}^T \phi(\mathbf{x}_i) + b| \right\} \\ &= \sum_{i=1}^n \max\{0, \frac{1}{n} (1 - |\mathbf{w}^T \phi(\mathbf{x}_i) + b|)\} \\ &= \frac{1}{n} \sum_{i=1}^n \max\{0, 1 - |\mathbf{w}^T \phi(\mathbf{x}_i) + b|\} = \frac{1}{n} \sum_{i=1}^n \xi_i^{(2)}\end{aligned}$$

Hence, for any  $(\mathbf{w}, b)$ , the objective functions for problem (3.7) and problem (3.9) have the same value given the optimal  $\xi$  and  $\xi_i$ . Therefore, the optima of the two optimization problems are the same.  $\square$

The above theorem shows that it is possible to solve problem (3.9) instead to get the optimal solution to problem (3.7), i.e. to get the same soft-margin classifying hyperplane. Putting theorem 3.1 and 3.2 together, we could therefore solve problem (3.9) instead of problem (3.6) to find the same *maximum margin clustering* solution, with the number of variables reduced by  $2n - 1$ .

Although the number of variables in problem (3.9) is greatly reduced, the number of constraints is increased from  $n$  to  $2^n$ . The algorithm we propose in this paper targets to find a small subset of constraints from the whole set of constraints in problem (3.9) that ensures a sufficiently accurate solution. Specifically, we employ an adaptation of the *cutting plane* algorithm[11], recently proposed in [17] for *structural SVM* training, to solve the *maximum margin clustering* problem, where we construct a nested sequence of successively tighter relaxations of problem (3.9). Moreover, we will prove

theoretically in section 4 that we can always find a polynomially sized subset of constraints, with which the solution of the relaxed problem fulfills all constraints from problem (3.9) up to a precision of  $\epsilon$ . That is to say, the remaining exponential number of constraints are guaranteed to be violated by no more than  $\epsilon$ , without the need for explicitly adding them to the optimization problem [17]. The *CPMMC* algorithm starts with an empty constraint subset  $\Omega$ , and it computes the optimal solution to problem (3.9) subject to the constraints in  $\Omega$ . The algorithm then finds the most violated constraint in problem (3.9) and adds it into the subset  $\Omega$ . In this way, we construct a successive strengthening approximation of the original *MMC* problem by a cutting plane that cuts off the current optimal solution from the feasible set [11]. The algorithm stops when no constraint in (3.9) is violated by more than  $\epsilon$ .

Here, the feasibility of a constraint is measured by the corresponding value of  $\xi$ , therefore, the most violated constraint is the one that would result in the largest  $\xi$ . Since each constraint in problem (3.9) is represented by a vector  $\mathbf{c}$ , then we have

**THEOREM 3.3.** *The most violated constraint could be computed as follows*

$$(3.12) \quad c_i = \begin{cases} 1 & \text{if } |\mathbf{w}^T \phi(\mathbf{x}_i) + b| < 1 \\ 0 & \text{otherwise} \end{cases}$$

*Proof.* As stated above, the most violated constraint is the one that would result in the largest  $\xi$ . In order to fulfill all constraints in problem (3.9), the minimum value of  $\xi$  is as follows

$$(3.13) \xi^* = \max_{\mathbf{c} \in \{0,1\}^n} \left\{ \frac{1}{n} \sum_{i=1}^n c_i - \frac{1}{n} \sum_{i=1}^n c_i |\mathbf{w}^T \phi(\mathbf{x}_i) + b| \right\} \\ = \sum_{i=1}^n \max_{c_i \in \{0,1\}} \left\{ \frac{1}{n} c_i - \frac{1}{n} c_i |\mathbf{w}^T \phi(\mathbf{x}_i) + b| \right\} \\ = \frac{1}{n} \sum_{i=1}^n \max_{c_i \in \{0,1\}} \{c_i (1 - |\mathbf{w}^T \phi(\mathbf{x}_i) + b|)\}$$

Therefore, the most violated constraint  $\mathbf{c}$  that results in the largest  $\xi^*$  could be calculated as in Eq.(3.12).  $\square$

The *CPMMC* algorithm iteratively selects the most violated constraint under the current hyperplane parameter and adds it into the working constraint set  $\Omega$  until no violation of constraint is detected. Moreover, since in problem (3.9), there is a direct correspondence between  $\xi$  and the feasibility of the set of constraints. If a point  $(\mathbf{w}, b, \xi)$  fulfills all constraints up to precision  $\epsilon$ , i.e.

$$(3.14) \quad \forall \mathbf{c} \in \{0,1\}^n : \\ \frac{1}{n} \sum_{i=1}^n c_i |\mathbf{w}^T \phi(\mathbf{x}_i) + b| \geq \frac{1}{n} \sum_{i=1}^n c_i - (\xi + \epsilon)$$

then the point  $(\mathbf{w}, b, \xi + \epsilon)$  is feasible. Furthermore, as in the objective function of problem (3.9), there is a single slack variable  $\xi$  that measures the clustering loss. Hence, we could simply select the stopping criterion as all samples satisfying the inequality (3.14). Then, the approximation accuracy  $\epsilon$  of this approximate solution is directly related to the training loss.

**3.2 Enforcing the Class Balance Constraint** As discussed in section 2, one has to enforce the class balance constraint to avoid trivially “optimal” solutions. However, in our algorithm, the optimal labeling vector  $\mathbf{y}$  is calculated as  $y_i = \text{sign}(\mathbf{w}^T \phi(\mathbf{x}_i) + b)$ . The non-linear function ‘*sign*’ greatly complicates the situation. Since the intention of restraining class balance is to exclude assigning all data samples into the same class or separating a very small group of outliers from the rest of the dataset. We can approximate the class balance constraint (2.3) originally proposed in [19] with the following similar but slightly relaxed constraint, which excludes the above stated two trivial solutions

$$(3.15) \quad -l \leq \sum_{i=1}^n (\mathbf{w}^T \phi(\mathbf{x}_i) + b) \leq l$$

where  $l \geq 0$  is a constant controlling the class imbalance. This approximation could also be viewed as an analogy to the treatment of the min-cut problem in spectral clustering [15].

Assume the current working constraint set is  $\Omega$ , *maximum margin clustering* with the class balance constraint could be formulated as the following optimization problem

$$(3.16) \min_{\mathbf{w}, b, \xi \geq 0} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C\xi \\ s.t. \forall \mathbf{c} \in \Omega: \frac{1}{n} \sum_{i=1}^n c_i |\mathbf{w}^T \phi(\mathbf{x}_i) + b| \geq \frac{1}{n} \sum_{i=1}^n c_i - \xi \\ -l \leq \sum_{i=1}^n (\mathbf{w}^T \phi(\mathbf{x}_i) + b) \leq l$$

Before getting into details of solving problem (3.16), we first present the outline of our *CPMMC* algorithm for *maximum margin clustering* in table 2.

**3.3 Optimization via the CCCP** In each iteration of the *CPMMC* algorithm, we need to solve problem (3.16) to obtain the optimal classifying hyperplane under the current working constraint set  $\Omega$ . Although the objective function in (3.16) is convex, the constraints are not, and this makes problem (3.16) difficult to solve. As stated in section 2, the *constrained concave-convex procedure* is designed to solve those optimization problems with a concave-convex objective function under

*CPMMC for maximum margin clustering*

1. Initialization. Set the values for  $C$  and  $\epsilon$ , and set  $\Omega = \phi$ ;
2. Solve optimization problem (3.16) under the current working constraint set  $\Omega$ ;
3. Select the most violated constraint  $\mathbf{c}$  under the current classifying hyperplane with Eq.(3.12).
4. If the selected constraint  $\mathbf{c}$  satisfies the following inequality  
 $\frac{1}{n} \sum_{i=1}^n c_i - \frac{1}{n} \sum_{i=1}^n c_i |\mathbf{w}^T \phi(\mathbf{x}_i) + b| \leq \xi + \epsilon$   
 goto step 5; otherwise  $\Omega = \Omega \cup \{\mathbf{c}\}$ , go to step 2.
5. Output. Return the labeling vector  $\mathbf{y}$  as  $y_i = \text{sign}(\mathbf{w}^T \phi(\mathbf{x}_i) + b)$

Table 2: Outline of the *CPMMC* algorithm.

concave-convex constraints [16]. In the following, we will show how to utilize *CCCP* to solve problem (3.16).

By rearranging the constraints in problem (3.16), we can reformulate it as

$$(3.17) \min_{\mathbf{w}, b, \xi} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C\xi$$

$$s.t. \xi \geq 0$$

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

$$\forall \mathbf{c} \in \Omega: \frac{1}{n} \sum_{i=1}^n c_i |\mathbf{w}^T \phi(\mathbf{x}_i) + b| \geq \frac{1}{n} \sum_{i=1}^n c_i - \xi$$

The objective function in (3.17) is quadratic and the first two constraints are linear. Moreover, the third constraint is, though non-convex, a difference between two convex functions. Hence, we can solve problem (3.17) with the *constrained concave-convex procedure*.

Notice that while  $\frac{1}{n} \sum_{i=1}^n c_i |\mathbf{w}^T \phi(\mathbf{x}_i) + b|$  is convex, it is a non-smooth function of  $(\mathbf{w}, b)$ . To use the *CCCP*, we need to replace the gradients by the *subgradients* [4]:

$$(3.18) \quad \left. \partial_{\mathbf{w}} \left[ \frac{1}{n} \sum_{i=1}^n c_i |\mathbf{w}^T \phi(\mathbf{x}_i) + b| \right] \right|_{\mathbf{w}=\mathbf{w}_t}$$

$$= \frac{1}{n} \sum_{i=1}^n c_i \text{sign}(\mathbf{w}_t^T \phi(\mathbf{x}_i) + b_t) \phi(\mathbf{x}_i)$$

$$(3.19) \quad \left. \partial_b \left[ \frac{1}{n} \sum_{i=1}^n c_i |\mathbf{w}^T \phi(\mathbf{x}_i) + b| \right] \right|_{b=b_t}$$

$$= \frac{1}{n} \sum_{i=1}^n c_i \text{sign}(\mathbf{w}_t^T \phi(\mathbf{x}_i) + b_t)$$

Given an initial point  $(\mathbf{w}_0, b_0)$ , the *CCCP* computes  $(\mathbf{w}_{t+1}, b_{t+1})$  from  $(\mathbf{w}_t, b_t)$  by replacing  $\frac{1}{n} \sum_{i=1}^n c_i |\mathbf{w}^T \phi(\mathbf{x}_i) + b|$  in the constraint with its first-

order Taylor expansion at  $(\mathbf{w}_t, b_t)$ , i.e.

$$(3.20) \quad \frac{1}{n} \sum_{i=1}^n c_i |\mathbf{w}_t^T \phi(\mathbf{x}_i) + b_t| + \frac{1}{n} \sum_{i=1}^n c_i \text{sign}(\mathbf{w}_t^T \phi(\mathbf{x}_i) + b_t) \cdot [\phi(\mathbf{x}_i)^T (\mathbf{w} - \mathbf{w}_t) + (b - b_t)]$$

$$= \frac{1}{n} \sum_{i=1}^n c_i |\mathbf{w}_t^T \phi(\mathbf{x}_i) + b_t| - \frac{1}{n} \sum_{i=1}^n c_i |\mathbf{w}_t^T \phi(\mathbf{x}_i) + b_t|$$

$$+ \frac{1}{n} \sum_{i=1}^n c_i \text{sign}(\mathbf{w}_t^T \phi(\mathbf{x}_i) + b_t) [\mathbf{w}^T \phi(\mathbf{x}_i) + b]$$

$$= \frac{1}{n} \sum_{i=1}^n c_i \text{sign}(\mathbf{w}_t^T \phi(\mathbf{x}_i) + b_t) [\mathbf{w}^T \phi(\mathbf{x}_i) + b]$$

By substituting the above first-order Taylor expansion (3.20) into problem (3.17), we obtain the following *quadratic programming (QP)* problem:

$$(3.21) \min_{\mathbf{w}, b, \xi} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C\xi$$

$$s.t. \xi \geq 0$$

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

$$\forall \mathbf{c} \in \Omega: \frac{1}{n} \sum_{i=1}^n c_i - \xi - \frac{1}{n} \sum_{i=1}^n c_i \cdot \text{sign}(\mathbf{w}_t^T \phi(\mathbf{x}_i) + b_t) [\mathbf{w}^T \phi(\mathbf{x}_i) + b] \leq 0$$

and the above *QP* problem could be solved in polynomial time. Following the *CCCP*, the obtained solution  $(\mathbf{w}, b)$  from this *QP* problem is then used as  $(\mathbf{w}_{t+1}, b_{t+1})$  and the iteration continues until convergence.

Moreover, we will show in the theoretical analysis section that the Wolfe dual of problem (3.21) has desirable sparseness properties. As it will be referred to later, we present the Wolfe dual here. For simplicity, we define the following three variables

$$\|\mathbf{c}_k\|_1 = \frac{1}{n} \sum_{i=1}^n c_{ki}, \quad k=1, \dots, |\Omega|$$

$$\mathbf{z}_k = \frac{1}{n} \sum_{i=1}^n c_{ki} \text{sign}(\mathbf{w}_t^T \phi(\mathbf{x}_i) + b_t) \phi(\mathbf{x}_i), \quad k=1, \dots, |\Omega|$$

$$\hat{\mathbf{x}} = \sum_{i=1}^n \phi(\mathbf{x}_i)$$

The Wolfe dual of problem (3.21) is

$$(3.22) \max_{\lambda \geq 0, \mu \geq 0} -\frac{1}{2} \sum_{k=1}^{|\Omega|} \sum_{l=1}^{|\Omega|} \lambda_k \lambda_l \mathbf{z}_k^T \mathbf{z}_l + (\mu_1 - \mu_2) \sum_{k=1}^{|\Omega|} \lambda_k \hat{\mathbf{x}}^T \mathbf{z}_k$$

$$- \frac{1}{2} (\mu_1 - \mu_2)^2 \hat{\mathbf{x}}^T \hat{\mathbf{x}} - (\mu_1 + \mu_2) l + \sum_{k=1}^{|\Omega|} \lambda_k \|\mathbf{c}_k\|_1$$

$$s.t. \sum_{k=1}^{|\Omega|} \lambda_k \leq C$$

$$(\mu_1 - \mu_2)n - \sum_{k=1}^{|\Omega|} \frac{\lambda_k}{n} \sum_{i=1}^n c_{ki} \text{sign}(\mathbf{w}_t^T \phi(\mathbf{x}_i) + b_t) = 0$$

Problem (3.22) is a *QP* problem with  $|\Omega| + 2$  variables, where  $|\Omega|$  denotes the total number of constraints in the subset  $\Omega$ .

Note that in successive iterations of the *CPMMC* algorithm, the optimization problem (3.16) differs only by a single constraint. Therefore, we can employ the solution in last iteration of the *CPMMC* algorithm as the initial point for the *CCCP*, which greatly reduces the runtime. Putting everything together, according to the formulation of the *CCCP* [16], we solve problem (3.16) with the algorithm presented in table 3. We set the

Solving problem (3.16) using the <i>CCCP</i>	
1.	Initialize $(\mathbf{w}_0, b_0)$ with the output of the last iteration of <i>CPMMC</i> algorithm, if this is the first iteration of <i>CPMMC</i> algorithm, initialize $(\mathbf{w}_0, b_0)$ with random values.
2.	Find $(\mathbf{w}_{t+1}, b_{t+1})$ as the solution to the <i>quadratic programming</i> problem (3.21);
3.	If convergence criterion satisfied, return $(\mathbf{w}_t, b_t)$ as the optimal hyperplane parameter; otherwise $t = t + 1$ , goto step 2.

Table 3: Outline of the *CCCP* algorithm.

stopping criterion in *CCCP* as the difference between two iterations less than  $\alpha\%$  and set  $\alpha\% = 0.01$ , which means the current objective function is larger than  $1 - \alpha\%$  of the objective function in last iteration, since *CCCP* decreases the objective function monotonically.

#### 4 Theoretical Analysis

In this section, we will provide a theoretical analysis of the *CPMMC* algorithm, including its correctness and time complexity.

Specifically, the following theorem characterizes the accuracy of the solution computed by *CPMMC*.

**THEOREM 4.1.** *For any dataset  $\mathcal{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$  and any  $\epsilon > 0$ , if  $(\mathbf{w}^*, b^*, \xi^*)$  is the optimal solution to problem (3.9), then our *CPMMC* algorithm for maximum margin clustering returns a point  $(\mathbf{w}, b, \xi)$  for which  $(\mathbf{w}, b, \xi + \epsilon)$  is feasible in problem (3.9). Moreover, the corresponding objective value is better than the one corresponds to  $(\mathbf{w}^*, b^*, \xi^*)$ .*

*Proof.* In step 3 of our *CPMMC* algorithm, the most violated constraint  $\mathbf{c}$ , which leads to the largest value of

$\xi$ , is selected using Eq.(3.12). According to the outline of the *CPMMC* algorithm, it terminates only when the newly selected constraint satisfies the following inequality

$$\frac{1}{n} \sum_{i=1}^n c_i - \frac{1}{n} \sum_{i=1}^n c_i |\mathbf{w}^T \phi(\mathbf{x}_i) + b| \leq \xi + \epsilon$$

If the above relation holds, since the newly selected constraint is the most violated one, all other constraints will satisfy the above inequality relation. Therefore, if  $(\mathbf{w}, b, \xi)$  is the solution returned by our *CPMMC* algorithm, then  $(\mathbf{w}, b, \xi + \epsilon)$  will be a feasible solution to problem (3.9). Moreover, since the solution  $(\mathbf{w}, b, \xi)$  is calculated with only constraints in the subset  $\Omega$  instead of all constraints in problem (3.9), it holds that  $\frac{1}{2} \mathbf{w}^{*T} \mathbf{w}^* + C\xi^* \geq \frac{1}{2} \mathbf{w}^T \mathbf{w} + C\xi$ .  $\square$

Based on the above theorem,  $\epsilon$  indicates how close one wants to be to the error rate of the best classifying hyperplane and can thus be used as the stopping criterion [9].

We next analyze the time complexity of *CPMMC*. We will mainly focus on the high-dimensional sparse data, where  $s \ll N$ , while for low-dimensional data, by simply setting  $s = N$ , all our theorems still hold. We will first show that each iteration of the *CPMMC* algorithm takes polynomial time. Since the algorithm is iterative, we will next prove that the total number of constraints added into the working set  $\Omega$ , i.e. the total iterations involved in the *CPMMC* algorithm, is upper bounded.

**THEOREM 4.2.** *Each iteration of *CPMMC* takes time  $O(sn)$  for a constant working set size  $|\Omega|$ .*

*Proof.* In steps 3 and 4 of the *CPMMC* algorithm, we need to compute  $n$  inner products between  $\mathbf{w}$  and  $\phi(\mathbf{x}_i)$ . Each inner product takes time  $O(s)$  when using sparse vector algebra, and totally  $n$  inner products will be computed in  $O(sn)$  time. To solve the *CCCP* problem in step 2, we will need to solve a series of *quadratic programming (QP)* problems. Setting up the dual problem (3.22) is dominated by computing the  $|\Omega|^2$  elements  $\mathbf{z}_k^T \mathbf{z}_l$  involved in the sum  $\sum_{k=1}^{|\Omega|} \sum_{l=1}^{|\Omega|} \lambda_k \lambda_l \mathbf{z}_k^T \mathbf{z}_l$ , and this can be done in time  $O(|\Omega|^2 sn)$  after first computing  $\mathbf{z}_k$ , ( $k = 1, \dots, |\Omega|$ ). Since the number of variables involved in the *QP* problem (3.22) is  $|\Omega| + 2$  and (3.22) can be solved in polynomial time, the time required for solving the dual problem is then independent of  $n$  and  $s$ . Therefore, each iteration in the *CCCP* takes time  $O(|\Omega|^2 sn)$ . Moreover, in numerical analyses, we observed in each round of the *CPMMC* algorithm, less than 10 iterations is required for solving the *CCCP*

problem, even for large scale dataset. Moreover, the number of iterations required is independent of  $n$  and  $s$ . Therefore, the time complexity for each iteration of our *CPMMC* algorithm is  $O(sn)$ , which scales linearly with  $n$  and  $s$ .  $\square$

Next we will prove an upper bound on the number of iterations in the *CPMMC* algorithm. For simplicity, we omit the class balance constraint in problem (3.16) and set the bias term  $b = 0$ . The theorem and proof for the problem with class balance constraint and non-zero bias term could be obtained similarly.

**THEOREM 4.3.** *For any  $\epsilon > 0$ ,  $C > 0$ , and any dataset  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ , the *CPMMC* algorithm terminates after adding at most  $\frac{CR}{\epsilon^2}$  constraints, where  $R$  is a constant number independent of  $n$  and  $s$ .*

*Proof.* Note that  $\mathbf{w} = 0$ ,  $\xi = 1$  is a feasible solution to problem (3.9), therefore, the objective function of the solution of (3.9) is upper bounded by  $C$ . We will prove that in each iteration of the *CPMMC* algorithm, by adding the most violated constraint, the increase of the objective function is at least a constant number. Due to the fact that the objective function of the solution is non-negative and has upper bound  $C$ , the total number of iterations will be upper bounded.

To compute the increase brought up by adding one constraint into the working set  $\Omega$ , we will first need to present the dual problem of (3.9). The difficulty involved in obtaining this dual problem comes from the abstracts in the constraints. Therefore, we first need to replace the constraints in (3.9) with the following

$$\begin{aligned} \frac{1}{n} \sum_{i=1}^n c_i t_i &\geq \frac{1}{n} \sum_{i=1}^n c_i - \xi, & \forall \mathbf{c} \in \Omega \\ t_i^2 &\leq \mathbf{w}^T \phi(\mathbf{x}_i) \phi(\mathbf{x}_i)^T \mathbf{w}, & \forall i \in \{1, \dots, n\} \\ t_i &\geq 0, & \forall i \in \{1, \dots, n\} \end{aligned}$$

and we define  $D_i = \phi(\mathbf{x}_i) \phi(\mathbf{x}_i)^T$ . Hence, the *Lagrangian dual function* can be obtained as follows

$$\begin{aligned} (4.23) \quad L(\lambda, \gamma, \mu, \delta) &= \inf_{\mathbf{w}, \xi, t} \left\{ \frac{1}{2} \mathbf{w}^T \mathbf{w} + C\xi + \sum_{k=1}^{|\Omega|} \lambda_k \left[ \frac{1}{n} \sum_{i=1}^n c_i (1-t_i) - \xi \right] \right. \\ &\quad \left. + \sum_{i=1}^n \gamma_i (t_i^2 - \mathbf{w}^T D_i \mathbf{w}) - \mu\xi - \sum_{i=1}^n \delta_i t_i \right\} \\ &= \inf_{\mathbf{w}, \xi, t} \left\{ \frac{1}{2} \mathbf{w}^T \mathbf{w} - \mathbf{w}^T \sum_{i=1}^n \gamma_i D_i \mathbf{w} + C\xi - \sum_{k=1}^{|\Omega|} \lambda_k \xi - \mu\xi \right. \\ &\quad \left. + \sum_{i=1}^n \gamma_i t_i^2 - \sum_{k=1}^{|\Omega|} \lambda_k \frac{1}{n} \sum_{i=1}^n c_{ki} t_i - \sum_{i=1}^n \delta_i t_i + \sum_{k=1}^{|\Omega|} \lambda_k \frac{1}{n} \sum_{i=1}^n c_{ki} \right\} \end{aligned}$$

$$= \sum_{i=1}^n \left\{ -\frac{(\sum_{k=1}^{|\Omega|} \lambda_k c_{ki} + n\delta_i)^2}{4n^2 \gamma_i} + \frac{1}{n} \sum_{k=1}^{|\Omega|} \lambda_k c_{ki} \right\}$$

satisfying the following constraints

$$(4.24) \quad I - 2 \sum_{i=1}^n \gamma_i D_i \succeq 0$$

$$(4.25) \quad C - \sum_{k=1}^{|\Omega|} \lambda_k - \mu = 0$$

$$(4.26) \quad t_i = \frac{1}{2n\gamma_i} \sum_{k=1}^{|\Omega|} \lambda_k c_{ki} + \frac{\delta_i}{2\gamma_i}$$

$$(4.27) \quad \lambda, \gamma, \mu, \delta \geq 0$$

The *CPMMC* algorithm selects the most violated constraint  $\mathbf{c}'$  and continues if the following inequality holds

$$(4.28) \quad \frac{1}{n} \sum_{i=1}^n c'_i (1 - t_i^*) \geq \xi + \epsilon$$

Since  $\xi \geq 0$ , the newly added constraint satisfies

$$(4.29) \quad \frac{1}{n} \sum_{i=1}^n c'_i (1 - t_i^*) \geq \epsilon$$

Denote by  $L_{k+1}(\lambda^{(k+1)}, \gamma^{(k+1)}, \mu^{(k+1)}, \delta^{(k+1)})$  the optimal value of the *Lagrangian dual function* subject to  $\Omega_{k+1} = \Omega_k \cup \{\mathbf{c}'\}$ . The addition of a new constraint to the primal problem is equivalent to adding a new variable  $\lambda_{k+1}$  into the dual problem.

$$\begin{aligned} (4.30) \quad L_{k+1}(\lambda^{(k+1)}, \gamma^{(k+1)}, \mu^{(k+1)}, \delta^{(k+1)}) &= \max_{\lambda, \gamma, \mu, \delta} \sum_{i=1}^n \left\{ -\frac{(\sum_{p=1}^k \lambda_p c_{pi} + \lambda_{k+1} c'_i + n\delta_i)^2}{4n^2 \gamma_i} \right. \\ &\quad \left. + \frac{1}{n} \left[ \sum_{p=1}^k \lambda_p c_{pi} + \lambda_{k+1} c'_i \right] \right\} \\ &\geq L_k(\lambda^{(k)}, \gamma^{(k)}, \mu^{(k)}, \delta^{(k)}) + \max_{\lambda_{k+1} \geq 0} \sum_{i=1}^n \left\{ -\frac{\lambda_{k+1} c'_i \sum_{p=1}^k \lambda_p^{(k)} c_{pi}}{2\gamma_i^{(k)} n^2} \right. \\ &\quad \left. - \frac{\lambda_{k+1} c'_i \delta_i^{(k)}}{2\gamma_i^{(k)} n} - \frac{(\lambda_{k+1} c'_i)^2}{4\gamma_i^{(k)} n^2} + \frac{1}{n} \lambda_{k+1} c'_i \right\} \end{aligned}$$

According to inequality (4.29) and the constraint  $\lambda_{k+1} \geq 0$ , we have

$$\sum_{i=1}^n \left[ \frac{\lambda_{k+1} c'_i \sum_{p=1}^k \lambda_p^{(k)} c_{pi}}{2\gamma_i^{(k)} n^2} + \frac{\lambda_{k+1} c'_i \delta_i^{(k)}}{2\gamma_i^{(k)} n} \right] \leq \frac{1}{n} \sum_{i=1}^n \lambda_{k+1} c'_i - \epsilon \lambda_{k+1}$$

Substituting the above inequality into (3.9), we get the lower bound of  $L_{k+1}(\lambda^{(k+1)}, \gamma^{(k+1)}, \mu^{(k+1)}, \delta^{(k+1)})$

as follows

$$\begin{aligned}
(4.31) \quad & L_{k+1}(\lambda^{(k+1)}, \gamma^{(k+1)}, \mu^{(k+1)}, \delta^{(k+1)}) \\
& \geq L_k(\lambda^{(k)}, \gamma^{(k)}, \mu^{(k)}, \delta^{(k)}) + \max_{\lambda_{k+1} \geq 0} \left\{ -\frac{1}{n} \sum_{i=1}^n \lambda_{k+1} c'_i + \epsilon \lambda_{k+1} \right. \\
& \quad \left. - \sum_{i=1}^n \frac{(\lambda_{k+1} c'_i)^2}{4\gamma_i^{(k)} n^2} + \sum_{i=1}^n \frac{1}{n} \lambda_{k+1} c'_i \right\} \\
& = L_k(\lambda^{(k)}, \gamma^{(k)}, \mu^{(k)}, \delta^{(k)}) + \max_{\lambda_{k+1} \geq 0} \left\{ \epsilon \lambda_{k+1} - \sum_{i=1}^n \frac{(\lambda_{k+1} c'_i)^2}{4\gamma_i^{(k)} n^2} \right\} \\
& = L_k(\lambda^{(k)}, \gamma^{(k)}, \mu^{(k)}, \delta^{(k)}) + \frac{\epsilon^2}{\sum_{i=1}^n (c_i'^2 / \gamma_i^{(k)} n^2)}
\end{aligned}$$

where  $\gamma_i^{(k)}$  is the value of  $\gamma_i$  which results in the largest  $L_k(\lambda, \gamma, \mu, \delta)$ . By maximizing the *Lagrangian dual function* shown in Eq.(4.23),  $\gamma^{(k)}$  could be obtained as follows

$$\begin{aligned}
& (\lambda^{(k)}, \gamma^{(k)}, \mu^{(k)}, \delta^{(k)}) \\
& = \arg \max_{\lambda, \gamma, \mu, \delta} \sum_{i=1}^n \left\{ -\frac{(\sum_{p=1}^k \lambda_p c_{pi} + n\delta_i)^2}{4n^2 \gamma_i} + \frac{1}{n} \sum_{p=1}^k \lambda_p c_{pi} \right\} \\
& = \arg \max_{\lambda, \gamma, \mu, \delta} \sum_{i=1}^n (\gamma_i - \delta_i)
\end{aligned}$$

subject to the following equation

$$(4.32) \quad 2n\gamma_i = \sum_{p=1}^k \lambda_p c_{pi} + n\delta_i$$

The only constraint on  $\delta_i$  is  $\delta_i \geq 0$ , therefore, to maximize  $\sum_{i=1}^n (\gamma_i - \delta_i)$ , the optimal value for  $\delta_i$  is 0. Hence, the following equation holds

$$(4.33) \quad 2n\gamma_i^{(k)} = \sum_{p=1}^k \lambda_p^{(k)} c_{pi}$$

Thus,  $n\gamma_i^{(k)}$  is a constant number independent of  $n$ . Moreover,  $\sum_{i=1}^n \frac{(c'_i)^2}{n}$  measures the fraction of non-zero elements in the constraint vector  $\mathbf{c}'$ , and therefore is a constant only related to the newly added constraint, also independent of  $n$ . Hence,  $\sum_{i=1}^n \frac{(c'_i)^2}{\gamma_i^{(k)} n^2}$  is a constant number independent of  $n$  and  $s$ , and we denote it with  $Q_k$ . Moreover, we define  $R = \max_k \{Q_k\}$  as the maximum of  $Q_k$  through the whole *CPMMC* process. Therefore, the increase of the objective function of the *Lagrangian dual problem* after adding the most violated constraint  $\mathbf{c}'$  is at least  $\frac{\epsilon^2}{R}$ . Furthermore, denote with  $G_k$  the value of the objective function in problem (3.9) subject to  $\Omega_k$  after adding  $k$  constraints. Due to the weak duality [1], at the optimal

solution  $L_k(\lambda^{(k)}, \gamma^{(k)}, \mu^{(k)}, \delta^{(k)}) \leq G_k(\mathbf{w}^{(k)}, \xi^{(k)}, t^{(k)}) \leq C$ . Since the *Lagrangian dual function* is upper bounded by  $C$ , the *CPMMC* algorithm terminates after adding at most  $\frac{CR}{\epsilon^2}$  constraints.  $\square$

It is true that the number of constraints can potentially explode for small values of  $\epsilon$ , however, experience with *CPMMC* shows that relatively large values of  $\epsilon$  are sufficient without loss of clustering accuracy. Note that the objective function of problem (3.9) with the scaled  $\frac{C}{n}$  instead of  $C$  is essential for this theorem. Putting everything together, we arrive at the following theorem regarding the time complexity of *CPMMC*.

**THEOREM 4.4.** *For any dataset  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  with  $n$  samples and sparsity of  $s$ , and any fixed value of  $C > 0$  and  $\epsilon > 0$ , the *CPMMC* algorithm takes time  $O(sn)$ .*

*Proof.* Since theorem 4.3 bounds the number of iterations in our *CPMMC* algorithm to a constant  $\frac{CR}{\epsilon^2}$  which is independent of  $n$  and  $s$ . Moreover, each iteration of the algorithm takes time  $O(sn)$ . The *CPMMC* algorithm has time complexity  $O(sn)$ .  $\square$

## 5 Experiments

In this section, we will validate the accuracy and efficiency of the *CPMMC* algorithm on several real world datasets. Specifically, we will analyze the scaling behavior of *CPMMC* with the sample size. Moreover, we will also study the sensitivity of *CPMMC* to  $\epsilon$  and  $C$ , both in accuracy and efficiency. All the experiments are performed with MATLAB 7.0 on a 1.66GHZ Intel Core™2 Duo PC running Windows XP with 1.5GB main memory.

**5.1 Datasets** We use seven datasets in our experiments, selected to cover a wide range of properties. Specifically, experiments are performed on a number of datasets from the UCI repository (**ionosphere**, **digits**, **letter** and **satellite**), MNIST database<sup>1</sup> and 20-newsgroup dataset<sup>2</sup>. For the **digits** data, we follow the experimental setup of [21] and focus on those pairs (3 vs 8, 1 vs 7, 2 vs 7, and 8 vs 9) that are difficult to differentiate. For the **letter** and **satellite** datasets, there are multiple classes and we use their first two classes only [21]. For the **20-newsgroup** dataset, we choose the topic *rec* which contains *autos*, *motorcycles*, *baseball* and *hockey* from the version 20-news-18828. We preprocess the data in the same manner as [22] and obtain 3970 document vectors in a 8014-dimensional space. Similar with the **digits** dataset, we focus on those pairs (*autos* vs. *motorcycles* (*Text-1*), and *baseball* vs. *hockey*

<sup>1</sup><http://yann.lecun.com/exdb/mnist/>

<sup>2</sup><http://people.csail.mit.edu/jrennie/20Newsgroups/>

(Text-2)) that are difficult to differentiate. Furthermore, for UCI **digits** and MNIST datasets, we give a more thorough comparison by considering all 45 pairs of digits 0 – 9.

Data	Size ( $n$ )	Feature ( $N$ )	Sparsity
Ionosphere	351	34	88.1%
Letter	1555	16	98.9%
UCI digits	1797	64	51.07%
Text-1	1980	8014	0.70%
Text-2	1989	8014	0.79%
Satellite	2236	36	100%
MNIST digits	70000	784	19.14%

Table 4: Descriptions of the datasets.

**5.2 Clustering Accuracy** We will first study the clustering accuracy of the *CPMMC* algorithm. We use *k-means clustering* (*KM*) and *normalized cut* (*NC*) as baselines, and also compared with *MMC* [19], *GMMC* [18] and *IterSVR* [21] which all aim at clustering data with the maximum margin hyperplane. For *CPMMC*, a linear kernel is used, while for *IterSVR*, both linear kernel and Gaussian kernel are used and we report the better result. Parameters involved are tuned using *grid search*, unless noted otherwise.

To assess clustering accuracy, we follow the strategy used in [19] where we first remove the labels for all data samples and run the clustering algorithms, then we label each of the resulting clusters with the majority class according to the original training labels, and finally measure the number of misclassifications made by each clustering [19]. The clustering accuracy results are shown in table 5, from which we clearly see that the *CPMMC* algorithm achieves higher accuracy than other methods on most of the datasets.

**5.3 Speed of *CPMMC*** Table 6 compares the CPU-time of *CPMMC* with *k-means clustering*, *normalized cut*, *GMMC* and *IterSVR* on 9 real world clustering problems. According to table 6, *CPMMC* is at least 18 times faster than *IterSVR* and 200 times faster than *GMMC*. As reported in [18], *GMMC* is about 100 times faster than *MMC*. Hence, *CPMMC* is still faster than *MMC* by about four orders of magnitude. Moreover, as the sample size increases, the CPU-time of *CPMMC* grows much slower than that of *IterSVR*, which indicates *CPMMC* has much better scaling property with the sample size than *IterSVR*.

**5.4 Average Number of *CCCP* Iterations** We state in section 4 that in each round of the *CPMMC* algorithm, less than 10 iterations is required for solving the *CCCP* problem, even for large scale datasets. Moreover, the number of iterations required is independent of

Data	KM	NC	GMMC	IterSVR	CPMMC
Digits 3-8	0.51	0.12	276.16	19.72	1.10(5.42)
Digits 1-7	0.54	0.13	289.53	20.49	0.95(6.15)
Digits 2-7	0.50	0.11	304.81	19.69	0.75(7.55)
Digits 8-9	0.49	0.11	277.26	19.41	0.85(3.77)
Ionosphere	0.07	0.12	273.04	18.86	0.78(2.33)
Letter	0.08	2.24	-	2133	0.87(4.00)
Satellite	0.19	5.01	-	6490	4.54(4.08)
Text-1	66.09	6.04	-	5844	19.75(3.80)
Text-2	52.32	5.35	-	6099	16.16(4.00)

Table 6: CPU-time (seconds) on the various datasets. For *CPMMC*, the number inside the bracket is the average number of *CCCP* iterations  $r$ .

$n$  and  $s$ . We validate this statement with experimental results on various datasets in figure 1, where we show how the average number of *CCCP* iterations  $r$  scales with the sample size. Moreover, the last column of table 6 provides the average number of *CCCP* iterations in *CPMMC* until convergence, from which we see regardless of the size of the datasets, in each round of *CPMMC* algorithm, less than 10 *CCCP* iterations are required on average.

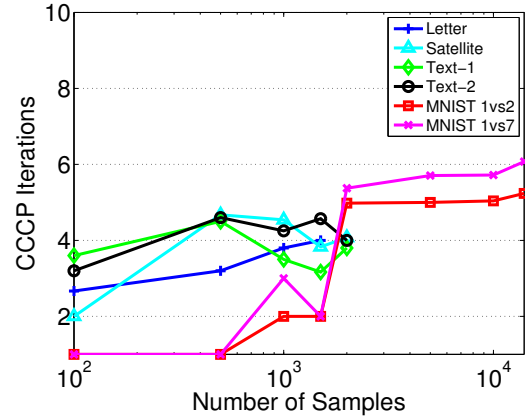


Figure 1: Average number of *CCCP* iterations in *CPMMC* as a function of sample size  $n$ .

**5.5 How does Computational Time Scale with the Number of Samples?** In the theoretical analysis section, we state that the computational time of *CPMMC* scales linearly with the number of samples. We present numerical demonstration for this statement in figure 2, where a log-log plot of how computational time increases with the size of the data set is shown. Specifically, lines in the log-log plot correspond to polynomial growth  $O(n^d)$ , where  $d$  is the slope of the line. Figure 2 shows that the CPU-time of *CPMMC* scales roughly  $O(n)$ , which is consistent with theorem 4.4.

**5.6 How does  $\epsilon$  Affect the Accuracy and Speed of *CPMMC*?** Theorem 4.3 states that the total num-

Data	Size	KM	NC	MMC	GMMC	IterSVR	CPMMC
Digits 3-8	357	5.32± 0	35	10	5.6	3.36± 0	<b>3.08</b>
Digits 1-7	361	0.55± 0	45	31.25	2.2	0.55± 0	<b>0.0</b>
Digits 2-7	356	3.09± 0	34	1.25	0.5	<b>0.0± 0</b>	<b>0.0</b>
Digits 8-9	354	9.32± 0	48	3.75	16.0	3.67± 0	<b>2.26</b>
Ionosphere	351	32± 17.9	25	<b>21.25</b>	23.5	32.3± 16.6	27.64
Letter	1555	17.94± 0	23.2	-	-	7.2± 0	<b>5.53</b>
Satellite	2236	4.07± 0	4.21	-	-	3.18± 0	<b>1.52</b>
Text-1	1980	49.47±0	6.21	-	-	<b>3.18± 0</b>	5.00
Text-2	1989	49.62±0	8.65	-	-	6.01± 1.82	<b>3.72</b>
UCI digits	1797	3.62	2.43	-	-	1.82	<b>0.62</b>
MNIST digits	70000	10.79	10.08	-	-	7.59	<b>4.29</b>

Table 5: Clustering errors(%) on the various datasets.

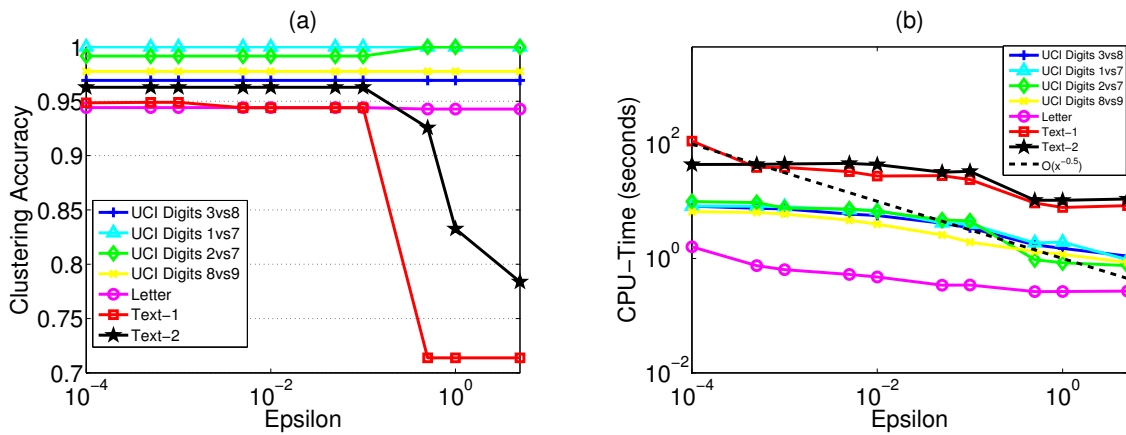


Figure 3: Clustering results of *CPMMC* with various values for  $\epsilon$ . (a) Clustering accuracy of *CPMMC* as a function of  $\epsilon$ ; (b) CPU-time (seconds) of *CPMMC* as a function of  $\epsilon$ .

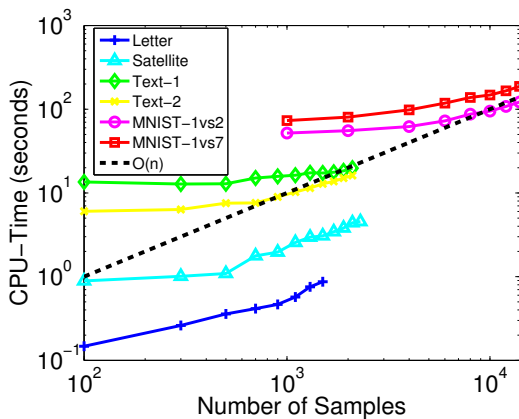


Figure 2: CPU-time (seconds) of *CPMMC* as a function of sample size  $n$ .

number of iterations involved in *CPMMC* is at most  $\frac{CR}{\epsilon^2}$ , and this means with higher  $\epsilon$ , the algorithm might converge fast. However, as  $\epsilon$  is directly related to the training loss in *CPMMC*, we need to determine how small  $\epsilon$  should be to guarantee sufficient accuracy. We present in figure 3 how clustering error and computational time scale with  $\epsilon$ . According to figure 3(a),  $\epsilon = 0.1$  is small enough to guarantee clustering accuracy. The log-log

plot in figure 3(b) verifies that the CPU-time of *CPMMC* decreases as  $\epsilon$  increases. Moreover, the empirical scaling of roughly  $O(\frac{1}{\epsilon^{0.5}})$  is much better than  $O(\frac{1}{\epsilon^2})$  in the bound from theorem 4.3.

**5.7 How dose  $C$  Affect the Accuracy and Speed of *CPMMC*?** Besides  $\epsilon$ ,  $C$  is also a crucial parameter in *CPMMC*, which adjusts the tradeoff between the margin and the clustering loss. We present in figure 4 how clustering accuracy and computational time scale with  $C$ . From this figure we observe that for most of the datasets, the clustering accuracy does not change evidently as long as  $C$  reside in a proper region, and the computational time scales linearly with  $C$ , which coincides with our theoretical analysis in section 4.

## 6 Conclusions

We propose the *cutting plane maximum margin clustering (CPMMC)* algorithm in this paper, to cluster data samples with the maximum margin hyperplane. Detailed theoretical analysis of the algorithm is provided, where we prove that the computational time of *CPMMC* scales linearly with the sample size  $n$  and sparsity  $s$  with guaranteed accuracy. Moreover, experimental evalua-

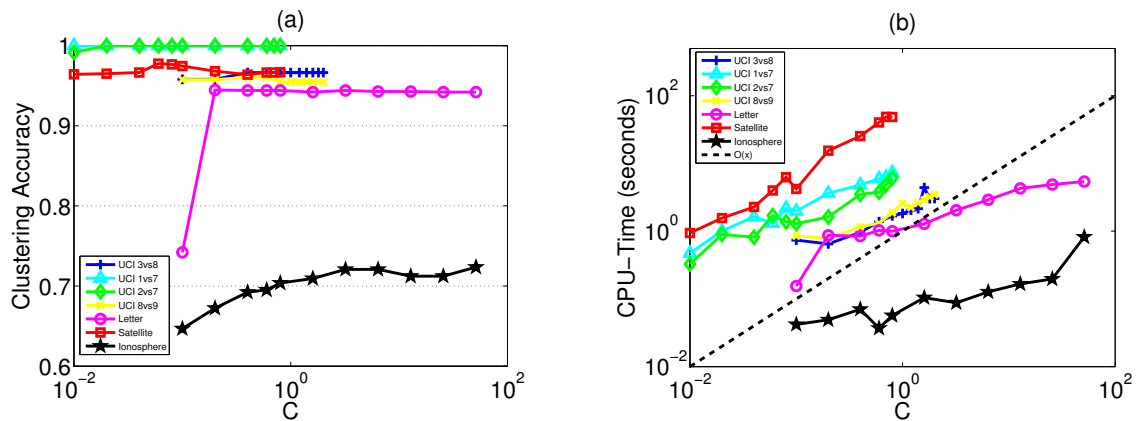


Figure 4: Clustering results of *CPMMC* with various values for  $C$ . (a) Clustering accuracy of *CPMMC* as a function of  $C$ ; (b) CPU-time (seconds) of *CPMMC* as a function of  $C$ .

tions on several real world datasets show that *CPMMC* performs better than existing *MMC* methods, both in efficiency and accuracy.

### Acknowledgements

This work is supported by the project (60675009) of the National Natural Science Foundation of China.

### References

- [1] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.
- [2] P. K. Chan, D. F. Schlag, and J. Y. Zien. Spectral k-way ratio-cut partitioning and clustering. *IEEE Trans. Computer-Aided Design*, 13:1088–1096, 1994.
- [3] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.
- [4] P. M. Cheung and J. T. Kowk. A regularization framework for multiple-instance learning. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.
- [5] C. Ding, X. He, H. Zha, M. Gu, and H. D. Simon. A min-max cut algorithm for graph partitioning and data mining. In *Proceedings of the 1st International Conference on Data Mining*, pages 107–114, 2001.
- [6] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., 2001.
- [7] J. Han and M. Kamber. *Data Mining*. Morgan Kaufmann Publishers, 2001.
- [8] A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Englewood Cliffs, NJ., 1988.
- [9] T. Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006.
- [10] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.
- [11] J. E. Kelley. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial Applied Mathematics*, 8:703–712, 1960.
- [12] K. Lange, D.R. Hunter, and I. Yang. Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics*, 9:1–59, 2000.
- [13] S. P. Luttrell. Partitioned mixture distributions: An adaptive bayesian network for low-level image processing. In *IEEE Proceedings on Vision, Image and Signal Processing*, volume 141, pages 251–260, 1994.
- [14] B. Schölkopf, A. J. Smola, and K. R. Müller. Kernel principal component analysis. *Advances in kernel methods: support vector learning*, pages 327–352, 1999.
- [15] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2000.
- [16] A. J. Smola, S.V.N. Vishwanathan, and T. Hofmann. Kernel methods for missing variables. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.
- [17] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6:1453–1484, 2005.
- [18] H. Valizadegan and R. Jin. Generalized maximum margin clustering and unsupervised kernel learning. In *Advances in Neural Information Processing Systems 19*, pages 1417–1424, 2007.
- [19] L. Xu, J. Neufeld, B. Larson, and D. Schuurmans. Maximum margin clustering. In *Advances in Neural Information Processing Systems*, 2004.
- [20] A. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15:915–936, 2003.
- [21] K. Zhang, I. W. Tsang, and J. T. Kowk. Maximum margin clustering made practical. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.
- [22] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, 2003.