Efficient MultiClass Maximum Margin Clustering

Bin Zhao ZHAOBINHERE@HOTMAIL.COM
Fei Wang FEIWANG03@MAILS.TSINGHUA.EDU.CN
Changshui Zhang ZCS@MAIL.TSINGHUA.EDU.CN
State Key Laboratory of Intelligent Technologies and Systems, Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Automation, Tsinghua University, Beijing 100084, China

Abstract

This paper presents a *cutting plane* algorithm for multiclass *maximum margin clustering* (*MMC*). The proposed algorithm constructs 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*). Experimental evaluations on several real world datasets show that our algorithm converges much faster than existing *MMC* methods with guaranteed accuracy, and can thus handle much larger datasets efficiently.

1. Introduction

Clustering (Duda et al., 2001; Shi & Malik, 2000; Ding et al., 2001) aims at dividing data into groups of similar objects, *i.e.* clusters. Recently, motivated by the success of large margin methods in supervised learning, (Xu et al., 2004) 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.

Technically, what *MMC* does is 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 (Xu et al., 2004). 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.

Several attempts have been made to solve the *maximum margin clustering* problem in polynomial time. (Xu et al., 2004) and (Valizadegan & Jin, 2007) made several relaxations to the original *MMC* problem and reformulated it as a *semi-definite programming* (*SDP*) problem. However, even with the recent advances in interior point methods, solving *SDP*s is still computationally very expensive. Consequently, the algorithms can only handle very small datasets containing several hundreds of samples. More recently, *Zhang et al.* utilized alternating optimization techniques to solve the *MMC* problem (Zhang et al., 2007), 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 time demanding on large scale datasets. Moreover, the methods described above can only handle binary clustering problems (Zhao et al., 2008), and there are significant complications to deriving an effective *maximum margin clustering* approach for the multiclass scenario[1]. Therefore, how to efficiently solve the *multiclass MMC* problem to make it capable of clustering large scale dataset is a very challenging research topic.

In this paper, we propose a *cutting plane multiclass maximum margin clustering* algorithm *CPM3C*. Specifically, the algorithm constructs a nested sequence of successively tighter relaxations of the original *multiclass MMC* problem, and each optimization problem in this sequence could be efficiently solved using the *constrained concave-convex procedure* (*CCCP*). Moreover, we show that the computational time of *CPM3C* scales roughly linearly with the dataset size. Our experimental evaluations on several real world datasets show that *CPM3C* performs better than existing *MMC* methods, both in efficiency and accuracy.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

[1] It should be noted that (Xu & Schuurmans, 2005) proposed a multiclass extension for *MMC*, however, their algorithm has a time complexity of $O(n^7)$, which renders it impractical for real world datasets.

The rest of this paper is organized as follows. Section 2 will show the *CPM3C* algorithm in detail, and the time complexity analysis of *CPM3C* will be presented in section 3. Section 4 presents the experimental results on several real world datasets, followed by the conclusions in section 5.

2. Cutting Plane Multiclass Maximum Margin Clustering

We will formally present the *cutting plane multiclass maximum margin clustering* (*CPM3C*) algorithm in this section.

2.1. Multiclass Maximum Margin Clustering

Maximum margin clustering (*MMC*) extends the theory of *support vector machine* (*SVM*) to the unsupervised scenario. Specifically, given a point set $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ and their labels $\mathbf{y} = (y_1, \ldots, y_n) \in \{1, \ldots, k\}^n$, *SVM* defines a weight vector $\mathbf{w}_p$ for each class $p \in \{1, \ldots, k\}$ and classifies sample $\mathbf{x}$ by $y^* = \arg\max_{y \in \{1,\ldots,k\}} \mathbf{w}_y^T \mathbf{x}$ with the weight vectors obtained as follows [2](Crammer & Singer, 2001)

$$\begin{aligned} \min_{\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi} \quad & \frac{1}{2}\beta\sum_{p=1}^{k}||\mathbf{w}_p||^2+\sum_{i=1}^{n}\xi_i \\ s.t. \quad & \forall i=1,\ldots,n, r=1,\ldots,k \\ & \mathbf{w}_{y_i}^T\mathbf{x}_i+\delta_{y_i,r}-\mathbf{w}_r^T\mathbf{x}_i \geq 1-\xi_i \end{aligned} \tag{1}$$

where the data samples in $\mathcal{X}$ are mapped into a high (possibly infinite) dimensional feature space, and by using the kernel trick, this mapping could be done implicitly. However, in those cases where kernel trick cannot be applied, it is possible to compute the coordinates of each sample in the *kernel PCA basis* (Schölkopf et al., 1999) according to kernel K. Therefore, throughout the rest of this paper, we use $\mathbf{x}_i$ to denote the sample mapped by the kernel function.

Instead of finding a large margin classifier given labels on the data as in *SVM*, *MMC* targets to find a labeling that would result in a large margin classifier (Xu et al., 2004). That is to say, if we subsequently run an *SVM* with the labeling obtained from *MMC*, the margin would be maximal among all possible labelings. *multiclass MMC* could be formulated as follows:

$$\begin{aligned} \min_{\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi,\mathbf{y}} \quad & \frac{1}{2}\beta\sum_{p=1}^{k}||\mathbf{w}_p||^2+\frac{1}{n}\sum_{i=1}^{n}\xi_i \\ s.t. \quad & \forall i=1,\ldots,n, r=1,\ldots,k \\ & \mathbf{w}_{y_i}^T\mathbf{x}_i+\delta_{y_i,r}-\mathbf{w}_r^T\mathbf{x}_i \geq 1-\xi_i \end{aligned} \tag{2}$$

[2]Although we focus on the *multiclass SVM* formulation of (Crammer & Singer, 2001), our method can be directly applied to other *multiclass SVM* formulations.

where $\sum_{i=1}^{n}\xi_i$ is divided by n to better capture how the regularization parameter β scales with the dataset size. Different from *SVM*, where the class labels are given and the only variables are $(\mathbf{w}_1, \ldots, \mathbf{w}_k)$, *MMC* targets to find not only the optimal weight vectors, but also the optimal labeling vector $\mathbf{y}^*$.

2.2. Cutting Plane Algorithm

In this section, we will reformulate problem (2) to reduce the number of variables. Specifically,

Theorem 1 *Problem (2) is equivalent to*

$$\begin{aligned} \min_{\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi} \quad & \frac{1}{2}\beta\sum_{p=1}^{k}||\mathbf{w}_p||^2+\frac{1}{n}\sum_{i=1}^{n}\xi_i \\ s.t. \quad & \forall i=1,\ldots,n, r=1,\ldots,k \\ & \sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i\prod_{q=1,q\neq p}^{k}I_{(\mathbf{w}_p^T\mathbf{x}_i>\mathbf{w}_q^T\mathbf{x}_i)} \\ & +\prod_{q=1,q\neq r}^{k}I_{(\mathbf{w}_r^T\mathbf{x}_i>\mathbf{w}_q^T\mathbf{x}_i)}-\mathbf{w}_r^T\mathbf{x}_i\geq 1-\xi_i \end{aligned} \tag{3}$$

where $I(\cdot)$ is the indicator function *and the label for sample $\mathbf{x}_i$ is determined as*

$$y_i=\sum_{p=1}^{k}p\prod_{q=1,q\neq p}^{k}I_{(\mathbf{w}_p^T\mathbf{x}_i>\mathbf{w}_q^T\mathbf{x}_i)} \tag{4}$$

Proof. We will show that for every $(\mathbf{w}_1, \ldots, \mathbf{w}_k)$ the smallest feasible $\sum_{i=1}^{n}\xi_i$ are identical for problem (2) and problem (3), and their corresponding labeling vectors are the same. For a given $(\mathbf{w}_1, \ldots, \mathbf{w}_k)$, the ξ_i in problem (2) can be optimized individually. According to the constraint in problem (2),

$$\xi_i \geq 1-(\mathbf{w}_{y_i}^T\mathbf{x}_i+\delta_{y_i,r}-\mathbf{w}_r^T\mathbf{x}_i),\ \forall r=1,\ldots,k \tag{5}$$

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

$$\xi_i^{(1)} = \min_{y_i=1,\ldots,k}\max_{r=1,\ldots,k}\{1-(\mathbf{w}_{y_i}^T\mathbf{x}_i+\delta_{y_i,r}-\mathbf{w}_r^T\mathbf{x}_i)\} \tag{6}$$

and we denote the corresponding class label by $y_i^{(1)}$. Without loss of generality, we assume the following relationship

$$\mathbf{w}_{i_1}^T\mathbf{x}_i \geq \mathbf{w}_{i_2}^T\mathbf{x}_i \geq \ldots \geq \mathbf{w}_{i_k}^T\mathbf{x}_i \tag{7}$$

where $(i_1, i_2, \ldots, i_k)$ is a permutation of $(1, 2, \ldots, k)$. For $y_i \neq i_1$, $\max_{r=1,\ldots,k}\{1-(\mathbf{w}_{y_i}^T\mathbf{x}_i+\delta_{y_i,r}-\mathbf{w}_r^T\mathbf{x}_i)\} \geq 1$, while for $y_i = i_1$, $\max_{r=1,\ldots,k}\{1-(\mathbf{w}_{y_i}^T\mathbf{x}_i+\delta_{y_i,r}-\mathbf{w}_r^T\mathbf{x}_i)\} \leq 1$, therefore, $y_i^{(1)} = i_1$ and

$$\begin{aligned} \xi_i^{(1)} &= \max_{r=1,\ldots,k}\{1-(\mathbf{w}_{i_1}^T\mathbf{x}_i+\delta_{i_1,r}-\mathbf{w}_r^T\mathbf{x}_i)\} \\ &= \max\{0, 1-(\mathbf{w}_{i_1}^T\mathbf{x}_i-\mathbf{w}_{i_2}^T\mathbf{x}_i)\} \end{aligned} \tag{8}$$

Similarly, for problem (3), the optimal value for ξ_i is

$$\begin{aligned}\xi_i^{(2)} &= \max_{r=1,\ldots,k}\left\{1-\left[\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i\prod_{q=1,q\neq p}^{k}I_{(\mathbf{w}_p^T\mathbf{x}_i>\mathbf{w}_q^T\mathbf{x}_i)}\right.\right.\\ &\quad\left.\left.+\prod_{q=1,q\neq r}^{k}I_{(\mathbf{w}_r^T\mathbf{x}_i>\mathbf{w}_q^T\mathbf{x}_i)}-\mathbf{w}_r^T\mathbf{x}_i\right]\right\} \qquad (9)\\ &= \max_{r=1,\ldots,k}\{1-(\mathbf{w}_{i_1}^T\mathbf{x}_i+\delta_{i_1,r}-\mathbf{w}_r^T\mathbf{x}_i)\}\\ &= \max\{0,1-(\mathbf{w}_{i_1}^T\mathbf{x}_i-\mathbf{w}_{i_2}^T\mathbf{x}_i)\}\end{aligned}$$

and the class label could be calculated as

$$y_i^{(2)}=\sum_{p=1}^{k}p\prod_{q=1,q\neq p}^{k}I_{(\mathbf{w}_p^T\mathbf{x}_i>\mathbf{w}_q^T\mathbf{x}_i)}=i_1 \qquad (10)$$

Therefore, the objective functions of both optimization problems are equivalent for any $(\mathbf{w}_1,\ldots,\mathbf{w}_k)$ with the same optimal ξ_i, and consequently so are their optima. Moreover, their corresponding labeling vectors $\mathbf{y}$ are the same. Hence, we proved that problem (2) is equivalent to problem (3). □

By reformulating problem (2) as problem (3), the number of variables involved is reduced by n, but there are still n slack variables ξ_i in problem (3). Define $\mathbf{e}_p$ as the $k\times 1$ vector with only the p-th element being 1 and others 0, $\mathbf{e}_0$ as the $k\times 1$ zero vector and $\mathbf{e}$ as the all one vector. To further reduce the number of variables involved in the optimization problem, we have the following theorem

Theorem 2 *Problem (3) can be equivalently formulated as problem (11), with* $\xi^*=\frac{1}{n}\sum_{i=1}^{n}\xi_i^*$.

$$\begin{aligned}\min_{\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi\geq 0}\quad & \frac{1}{2}\beta\sum_{p=1}^{k}||\mathbf{w}_p||^2+\xi \qquad (11)\\ s.t.\quad & \forall\mathbf{c}_i\in\{\mathbf{e}_0,\mathbf{e}_1,\ldots,\mathbf{e}_k\},\ i=1,\ldots,n\\ & \frac{1}{n}\sum_{i=1}^{n}\left\{\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_iz_{ip}+\sum_{p=1}^{k}c_{ip}(z_{ip}-\mathbf{w}_p^T\mathbf{x}_i)\right\}\\ & \geq\frac{1}{n}\sum_{i=1}^{n}\mathbf{c}_i^T\mathbf{e}-\xi\end{aligned}$$

where $z_{ip}=\prod_{q=1,q\neq p}^{k}I_{(\mathbf{w}_p^T\mathbf{x}_i>\mathbf{w}_q^T\mathbf{x}_i)}\ \forall i=1,\ldots,n;p=1\ldots,k$ *and each constraint* $\mathbf{c}$ *is represented as a* $k\times n$ *matrix* $\mathbf{c}=(\mathbf{c}_1,\ldots,\mathbf{c}_n)$.

Proof. To justify the above theorem, we will show that problem (3) and problem (11) have the same objective value and an equivalent set of constraints. Specifically, we will prove that for every $(\mathbf{w}_1,\ldots,\mathbf{w}_k)$, the smallest feasible ξ and $\sum_{i=1}^{n}\xi_i$ are related by $\xi=\frac{1}{n}\sum_{i=1}^{n}\xi_i$. This means, with $(\mathbf{w}_1,\ldots,\mathbf{w}_k)$ fixed, $(\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi)$ and $(\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi_i)$ are optimal solutions to problem (3) and (11) respectively, and they result in the same objective function value.

For any given $(\mathbf{w}_1,\ldots,\mathbf{w}_k)$, the ξ_i in problem (3) can be optimized individually and the optimum is achieved as

$$\xi_i^{(2)}=\max\{0,1-(\mathbf{w}_{i_1}^T\mathbf{x}_i-\mathbf{w}_{i_2}^T\mathbf{x}_i)\} \qquad (12)$$

where we assume the relation in (7) holds.

Similarly for problem (11), the optimal ξ is

$$\begin{aligned}\xi^{(3)}=\max_{\mathbf{c}_1,\ldots,\mathbf{c}_n\in\{\mathbf{e}_0,\ldots,\mathbf{e}_k\}}&\left\{\frac{1}{n}\sum_{i=1}^{n}\mathbf{c}_i^T\mathbf{e}-\frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_iz_{ip}\right.\right. \qquad (13)\\ &\left.\left.+\sum_{p=1}^{k}c_{ip}(z_{ip}-\mathbf{w}_p^T\mathbf{x}_i)\right]\right\}\end{aligned}$$

Since each $\mathbf{c}_i$ are independent in Eq.(13), they can be optimized individually. Therefore,

$$\begin{aligned}\xi^{(3)}&=\frac{1}{n}\sum_{i=1}^{n}\max_{\mathbf{c}_i}\{\mathbf{c}_i^T\mathbf{e}-\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_iz_{ip}-\sum_{p=1}^{k}c_{ip}(z_{ip}-\mathbf{w}_p^T\mathbf{x}_i)\}\\ &=\frac{1}{n}\sum_{i=1}^{n}\max\left\{0,\max_{p=1,\ldots,k}[1-(\mathbf{w}_{i_1}^T\mathbf{x}_i+\delta_{i_1,p}-\mathbf{w}_p^T\mathbf{x}_i)]\right\}\\ &=\frac{1}{n}\sum_{i=1}^{n}\max\left\{0,\max[0,1-(\mathbf{w}_{i_1}^T\mathbf{x}_i-\mathbf{w}_{i_2}^T\mathbf{x}_i)]\right\}\\ &=\frac{1}{n}\sum_{i=1}^{n}\max\{0,1-(\mathbf{w}_{i_1}^T\mathbf{x}_i-\mathbf{w}_{i_2}^T\mathbf{x}_i)\}=\frac{1}{n}\sum_{i=1}^{n}\xi_i^{(2)}\end{aligned}$$

Hence, for any $(\mathbf{w}_1,\ldots,\mathbf{w}_k)$, the objective functions for problem (3) and problem (11) have the same value given the optimal ξ and ξ_i. Therefore, the optima of the two optimization problems are the same. □

Putting theorems 1 and 2 together, we could therefore solve problem (11) instead to find the same *maximum margin clustering* solution, with the number of variables reduced by $2n-1$. Although the number of variables in problem (11) is greatly reduced, the number of constraints increases from nk to $(k+1)^n$. The algorithm we propose in this paper targets to find a small subset of constraints from the whole set of constraints in problem (11) that ensures a sufficiently accurate solution. Specifically, we employ an adaptation of the *cutting plane* algorithm (Kelley, 1960) to solve problem (11), where we construct a nested sequence of successively tighter relaxations of problem (11). Moreover, we can prove theoretically (see section 3) that we can always find a polynomially sized subset of constraints, with which the solution of the relaxed problem fulfills all constraints from problem (11) up to a precision of ϵ. That is to say, the remaining exponential number of constraints are guaranteed to be violated by no more than ϵ, without the need for explicitly adding them to the optimization problem (Tsochantaridis et al., 2005). Specifically, the *CPM3C* algorithm keeps a subset Ω of working constraints and

computes the optimal solution to problem (11) subject to the constraints in Ω. The algorithm then adds the most violated constraint in problem (11) into Ω. In this way, a successively strengthening approximation of the original *MMC* problem is constructed by a cutting plane that cuts off the current optimal solution from the feasible set (Kelley, 1960). The algorithm stops when no constraint in (11) is violated by more than ϵ. Here, the feasibility of a constraint is measured by the corresponding value of ξ, therefore, the most violated constraint is the one that results in the largest ξ. Since each constraint in problem (11) is represented by a $k \times n$ matrix $\mathbf{c}$, then we have

Theorem 3 *Define* $p^* = \arg\max_p(\mathbf{w}_p^T\mathbf{x}_i)$ *and* $r^* = \arg\max_{r\neq p^*}(\mathbf{w}_r^T\mathbf{x}_i)$ *for* $i = 1, \ldots, n$, *the most violated constraint could be calculated as follows*

$$\mathbf{c}_i = \begin{cases} \mathbf{e}_{r^*} & if\ (\mathbf{w}_{p^*}^T\mathbf{x}_i - \mathbf{w}_{r^*}^T\mathbf{x}_i) < 1 \\ \mathbf{0} & otherwise \end{cases},\ i = 1, \ldots, n \qquad (14)$$

Proof. The most violated constraint is the one that would result in the largest ξ. As each $\mathbf{c}_i$ in the constraint is independent, in order to fulfill all constraints in problem (11), the value of ξ is as follows

$$\begin{aligned}\xi^* &= \frac{1}{n}\sum_{i=1}^{n}\max_{\mathbf{c}_i}\{\mathbf{c}_i^T\mathbf{e} - \mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i z_{ip} - \sum_{p=1}^{k}c_{ip}(z_{ip} - \mathbf{w}_p^T\mathbf{x}_i)\} \\ &= \frac{1}{n}\sum_{i=1}^{n}\max_{\mathbf{c}_i}\{\mathbf{c}_i^T[\mathbf{e} - \mathbf{w}_{p^*}^T\mathbf{x}_i\mathbf{e} - \mathbf{z}_i + \mathbf{t}_i]\}\end{aligned}$$

where $\mathbf{t}_i = (\mathbf{w}_1^T\mathbf{x}_i, \ldots, \mathbf{w}_k^T\mathbf{x}_i)^T$. Since $\mathbf{c}_i \in \{\mathbf{e}_0, \ldots, \mathbf{e}_k\}$, $\mathbf{c}_i$ selects the largest element of the vector $\mathbf{e} - \mathbf{w}_{p^*}^T\mathbf{x}_i\mathbf{e} - \mathbf{z}_i + \mathbf{t}_i$, which could be calculated as $1 - (\mathbf{w}_{p^*}^T\mathbf{x}_i - \mathbf{w}_{r^*}^T\mathbf{x}_i)$. Therefore, the most violated constraint $\mathbf{c}$ that results in the largest ξ^* could be calculated as in Eq.(14). □

The *CPM3C* algorithm iteratively selects the most violated constraint under the current weight vectors and adds it into the working constraint set Ω until no violation of constraints is detected. Moreover, if a point $(\mathbf{w}_1, \ldots, \mathbf{w}_k, \xi)$ fulfills all constraints up to precision ϵ

$$\begin{aligned}&\forall \mathbf{c}_i \in \{\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_k\}^n,\ i = 1, \ldots, n \qquad (15)\\ &\frac{1}{n}\sum_{i=1}^{n}\left\{\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i z_{ip} + \sum_{p=1}^{k}c_{ip}(z_{ip} - \mathbf{w}_p^T\mathbf{x}_i)\right\} \geq \frac{1}{n}\sum_{i=1}^{n}\mathbf{c}_i^T\mathbf{e} - \xi - \epsilon\end{aligned}$$

then the point $(\mathbf{w}_1, \ldots, \mathbf{w}_k, \xi + \epsilon)$ is feasible. Furthermore, as in the objective function of problem (11), there is a single slack variable ξ that measures the clustering loss. Hence, we could simply select the stopping criterion as all samples satisfying the inequality (15). Then, the approximation accuracy ϵ of this approximate solution is directly related to the training loss.

2.3. Enforcing the Class Balance Constraint

In 2-class *maximum margin clustering*, a trivially "optimal" solution is to assign all patterns to the same class, and the resultant margin will be infinite (Xu et al., 2004). Similarly, for the multiclass scenario, a large margin can always be achieved by eliminating classes (Xu & Schuurmans, 2005). Therefore, we add the following class balance constraints to avoid the trivially "optimal" solutions

$$-l \leq \sum_{i=1}^{n}\mathbf{w}_p^T\mathbf{x}_i - \sum_{i=1}^{n}\mathbf{w}_q^T\mathbf{x}_i \leq l,\ \forall p, q = 1, \ldots, k \qquad (16)$$

where $l \geq 0$ controls the class imbalance. Therefore, *multiclass maximum margin clustering* with working constraint set Ω could be formulated as follows

$$\begin{aligned}\min_{\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi\geq 0}\quad & \frac{1}{2}\beta\sum_{p=1}^{k}||\mathbf{w}_p||^2 + \xi \qquad (17)\\ s.t.\quad & \frac{1}{n}\sum_{i=1}^{n}\left\{\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i z_{ip} + \sum_{p=1}^{k}c_{ip}(z_{ip} - \mathbf{w}_p^T\mathbf{x}_i)\right\} \\ & \geq \frac{1}{n}\sum_{i=1}^{n}\mathbf{c}_i^T\mathbf{e} - \xi,\ \forall[\mathbf{c}_1, \ldots, \mathbf{c}_n] \in \Omega \\ & -l \leq \sum_{i=1}^{n}\mathbf{w}_p^T\mathbf{x}_i - \sum_{i=1}^{n}\mathbf{w}_q^T\mathbf{x}_i \leq l,\ \forall p, q = 1, \ldots, k\end{aligned}$$

Before getting into details of solving problem (17), we first present the *CPM3C* approach in Algorithm 1.

Algorithm 1 Cutting Plane Multiclass MMC

Initialize $\Omega = \phi$
repeat
  Solve problem (17) for $(\mathbf{w}_1, \ldots, \mathbf{w}_k)$ under the current working constraint set Ω and select the most violated constraint $\mathbf{c}$ with Eq.(14). Set $\Omega = \Omega \cup \{\mathbf{c}\}$.
until $(\mathbf{w}_1, \ldots, \mathbf{w}_k)$ satisfies $\mathbf{c}$ up to precision ϵ

2.4. Optimization via the *CCCP*

In each iteration of the *CPM3C* algorithm, we need to solve problem (17) to obtain the optimal classifying hyperplanes under the current working constraint set Ω. Although the objective function in (17) is convex, the constraints are not, and this makes problem (17) difficult to solve. Fortunately, the *constrained concave-convex procedure (CCCP)* is just designed to solve those optimization problems with a concave-convex objective function under concave-convex constraints (Smola et al., 2005). In the following, we will show how to utilize *CCCP* to solve problem (17).

The objective function in (17) and the second constraint are convex. Moreover, the first constraint is, although non-convex, the difference of two convex functions. Hence, we can solve (17) with *CCCP*. Notice that while $\frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i z_{ip}+\sum_{p=1}^{k}c_{ip}z_{ip}\right]$ is convex, it is a non-smooth function of $(\mathbf{w}_1,\ldots,\mathbf{w}_k)$. To use *CCCP*, we need to calculate the *subgradients*:

$$\partial_{\mathbf{w}_r}\left\{\frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i z_{ip}+\sum_{p=1}^{k}c_{ip}z_{ip}\right]\right\}\Bigg|_{\mathbf{w}=\mathbf{w}^{(t)}} \tag{18}$$
$$= \frac{1}{n}\sum_{i=1}^{n}\mathbf{c}_i^T\mathbf{e}z_{ip}^{(t)}\mathbf{x}_i \qquad \forall r=1,\ldots,k$$

Given an initial point $(\mathbf{w}_1^{(0)},\ldots,\mathbf{w}_k^{(0)})$, *CCCP* computes $(\mathbf{w}_1^{(t+1)},\ldots,\mathbf{w}_k^{(t+1)})$ from $(\mathbf{w}_1^{(t)},\ldots,\mathbf{w}_k^{(t)})$ by replacing $\frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i z_{ip}+\sum_{p=1}^{k}c_{ip}z_{ip}\right]$ in the constraint with its first order Taylor expansion at $(\mathbf{w}_1^{(t)},\ldots,\mathbf{w}_k^{(t)})$, i.e.

$$\frac{1}{n}\sum_{i=1}^{n}\left\{\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}{\mathbf{w}_p^{(t)}}^T\mathbf{x}_i z_{ip}^{(t)}+\sum_{p=1}^{k}c_{ip}z_{ip}^{(t)}\right\} \tag{19}$$
$$+\frac{1}{n}\sum_{i=1}^{n}\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}(\mathbf{w}_p-\mathbf{w}_p^{(t)})^T\mathbf{x}_i z_{ip}^{(t)}$$
$$= \frac{1}{n}\sum_{i=1}^{n}\left\{\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i z_{ip}^{(t)}+\sum_{p=1}^{k}c_{ip}z_{ip}^{(t)}\right\}$$

By substituting the above first-order Taylor expansion into problem (11), we obtain the following *quadratic programming (QP)* problem:

$$\min_{\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi\geq 0}\ \frac{1}{2}\beta\sum_{p=1}^{k}||\mathbf{w}_p||^2+\xi \tag{20}$$
$$s.t.\ \forall[\mathbf{c}_1,\ldots,\mathbf{c}_n]\in\Omega$$
$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{c}_i^T\mathbf{e}-\xi+\frac{1}{n}\sum_{i=1}^{n}\sum_{p=1}^{k}c_{ip}\mathbf{w}_p^T\mathbf{x}_i$$
$$-\frac{1}{n}\sum_{i=1}^{n}\left\{\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i z_{ip}^{(t)}+\sum_{p=1}^{k}c_{ip}z_{ip}^{(t)}\right\}\leq 0$$
$$-l\leq\sum_{i=1}^{n}\mathbf{w}_p^T\mathbf{x}_i-\sum_{i=1}^{n}\mathbf{w}_q^T\mathbf{x}_i\leq l,\ \forall p,q=1,\ldots,k$$

Moreover, the dual problem of (20) is a *QP* problem with $|\Omega|+2$ variables and could be solved in polynomial time, where $|\Omega|$ denotes the total number of constraints in Ω. Putting everything together, according to the formulation of the *CCCP* (Smola et al., 2005), we solve problem (17) with the approach presented in Algorithm 2, where we set the 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.

Algorithm 2 Solve problem (17) with CCCP

Initialize $\mathbf{w}_p=\mathbf{w}_p^0$ for $p=1,\ldots,k$.
repeat
  Find $(\mathbf{w}_1^{t+1},\ldots,\mathbf{w}_k^{t+1})$ as the solution to the *quadratic programming* problem (20).
  Set $\mathbf{w}_p=\mathbf{w}_p^{t+1}, p=1,\ldots,k$
until stopping criterion satisfied.

2.5. Theoretical Analysis

We provide the following theorem regarding the correctness of the *CPM3C* algorithm.

Theorem 4 *For any dataset $\mathcal{X}=(\mathbf{x}_1,\ldots,\mathbf{x}_n)$ and any $\epsilon>0$, if $(\mathbf{w}_1^*,\ldots,\mathbf{w}_k^*,\xi^*)$ is the optimal solution to problem (11) with the class balance constraint, then our* CPM3C *algorithm returns a point $(\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi)$ for which $(\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi+\epsilon)$ is feasible in problem (11) and satisfies the class balance constraint. Moreover, the corresponding objective value is better than the one corresponds to $(\mathbf{w}_1^*,\ldots,\mathbf{w}_k^*,\xi^*)$.*

Based on the above theorem, ϵ indicates how close one wants to be to the error rate of the best classifying hyperplanes and can thus be used as the stopping criterion (Joachims, 2006).

3. Time Complexity Analysis

In this section, we will provide analysis on the time complexity of *CPM3C*. For the high-dimensional *(say, d-dimensional)* sparse data commonly encountered in applications like text mining and bioinformatics, we assume each sample has only $s \ll d$ non-zero features, *i.e.*, s implies the sparsity, while for non-sparse data, by simply setting $s=d$, all our theorems still hold.

Theorem 5 *Each iteration of* CPM3C *takes time $O(snk)$ for a constant working set size $|\Omega|$.*

Moreover, for the binary clustering scenario, we have the following theorem

Theorem 6 *For any $\epsilon>0$, $\beta>0$, and any dataset $\mathcal{X}=\{\mathbf{x}_1,\ldots,\mathbf{x}_n\}$ with samples belonging to two different classes, the* CPM3C *algorithm terminates after adding at most $\frac{R}{\epsilon^2}$ constraints, where R is a constant number independent of n and s.*

It is true that the number of constraints can potentially explode for small values of ϵ, however, experi-

ence with *CPM3C* shows that relatively large values of ϵ are sufficient without loss of clustering accuracy. Since the number of iterations in *CPM3C* (with $k = 2$) is bounded by $\frac{R}{\epsilon^2}$, a constant independent of n and s, and each iteration of the algorithm takes time $O(snk)$ ($O(sn)$ for the binary clustering scenario), we arrive at the following theorem

Theorem 7 *For any dataset $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ with n samples belonging to 2 classes and sparsity of s, and any fixed value of $\beta > 0$ and $\epsilon > 0$, the* CPM3C *algorithm takes time $O(sn)$ to converge.*

For the multiclass scenario, experimental results shown in section 4 also demonstrate that the computational time of *CPM3C* scales roughly linearly with the dataset size n.

4. Experiments

In this section, we will validate the accuracy and efficiency of the *CPM3C* algorithm on several real world datasets. Moreover, we will also analyze the scaling behavior of *CPM3C* with the dataset size and the sensitivity of *CPM3C* to ϵ, both in accuracy and efficiency. All the experiments are performed with MATLAB 7.0 on a 1.66GHZ Intel Core$^{\text{TM}}$2 Duo PC running Windows XP with 1.5GB main memory.

4.1. Datasets

We use eight datasets in our experiments, which are selected to cover a wide range of properties: **Digits, Letter** and **Satellite** from the **UCI** repository, **MNIST**[3], **20 newsgroup**[4], **WebKB**[5], **Cora** (McCallum et al., 2000) and **RCVI** (Lewis et al., 2004). In order to compare *CPM3C* with other *MMC* algorithms which can only perform binary clustering, we choose the first two classes from **Letter** and **Satellite**. For the **20 newsgroup** dataset, we choose the topic *rec* which contains *autos, motorcycles, baseball* and *hockey* from the version 20-news-18828. For **WebKB**, we select a subset consists of about 6000 web pages from computer science departments of four schools (Cornell, Texas, Washington, and Wisconsin). For **Cora**, we select a subset containing the research paper of subfield data structure (DS), hardware and architecture (HA), machine learning (ML), operating system (OS) and programming language (PL). For **RCVI**, we use the data samples with the highest four topic codes (CCAT, ECAT, GCAT and MCAT) in the "Topic Codes" hierarchy in the training set.

[3]http://yann.lecun.com/exdb/mnist/

[4]http://people.csail.mit.edu/jrennie/20Newsgroups/

[5]http://www.cs.cmu.edu/∼WebKB/

Table 1. Descriptions of the datasets.

Data	Size (n)	Feature (N)	Class	Sparsity
Letter	1555	16	2	98.9%
UCIDig	1797	64	10	51.07%
UCISat	2236	36	2	100%
MNIST	70000	784	10	19.14%
Cora-DS	751	6234	9	0.68%
Cora-HA	400	3989	7	1.1%
Cora-ML	1617	8329	7	0.58%
Cora-OS	1246	6737	4	0.75%
Cora-PL	1575	7949	9	0.56%
WK-CL	827	4134	7	2.32%
WK-TX	814	4029	7	1.97%
WK-WT	1166	4165	7	2.05%
WK-WC	1210	4189	7	2.16%
20-news	3970	8014	4	0.75%
RCVI	21251	47152	4	0.16%

4.2. Comparisons and Clustering Results

Besides our *CPM3C* algorithm, we also implements some other competitive algorithms and present their results for comparison. Specifically, we use **K-Means (KM)** and **Normalized Cut (NC)** as baselines, and also compared with **Maximum Margin Clustering (MMC)** (Xu et al., 2004), **Generalized Maximum Margin Clustering (GMC)** (Valizadegan & Jin, 2007) and **Iterative Support Vector Regression (SVR)** (Zhang et al., 2007) which all aim at clustering data with the maximum margin hyperplane. Technically, for **k-means**, the cluster centers are initialized randomly. For **NC**, the implementation is the same as in (Shi & Malik, 2000), and the width of the Gaussian kernel is set by exhaustive search from the grid $\{0.1\sigma_0, 0.2\sigma_0, \ldots, \sigma_0\}$, where σ_0 is the range of distance between any two data points in the dataset. Moreover, for **MMC** and **GMC**, the implementation is the same as in (Xu et al., 2004; Xu & Schuurmans, 2005) and (Valizadegan & Jin, 2007) respectively. Furthermore, the implementation code for **SVR** is downloaded from http://www.cse.ust.hk/∼twinsen and the initialization is based on *k-means* with randomly selected initial data centers, and the width of the Gaussian kernel is set in the same way as in **NC**.

In the experiments, we set the number of clusters equal to the true number of classes k for all the clustering algorithms. To assess clustering accuracy, we follow the strategy used in (Xu et al., 2004) where we first take a set of labeled data, 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 correct classifications made by each clustering. Moreover, we also calculate the *Rand Index* (Rand, 1971) for each clustering result. The *clustering accuracy* and *Rand index* results are summarized in table 2 and table 3 respectively,

Table 2. Clustering accuracy(%) comparisons.

Data	KM	NC	MMC	GMC	SVR	CPM3C
Dig 3-8	94.68	65.00	90.00	94.40	96.64	**96.92**
Dig 1-7	94.45	55.00	68.75	97.8	99.45	**100.0**
Dig 2-7	96.91	66.00	98.75	99.50	**100.0**	**100.0**
Dig 8-9	90.68	52.00	96.25	84.00	96.33	**97.74**
Letter	82.06	76.80	-	-	92.80	**94.47**
UCISat	95.93	95.79	-	-	96.82	**98.48**
Text-1	50.53	93.79	-	-	**96.82**	95.00
Text-2	50.38	91.35	-	-	93.99	**96.28**
UCIDig	96.38	97.57	-	-	98.18	**99.38**
MNIST	89.21	89.92	-	-	92.41	**95.71**
Dig 0689	42.23	93.13	94.83	-	-	**96.63**
Dig 1279	40.42	90.11	91.91	-	-	**94.01**
Cora-DS	28.24	36.88	-	-	-	**43.75**
Cora-HA	34.02	42.00	-	-	-	**59.75**
Cora-ML	27.08	31.05	-	-	-	**45.58**
Cora-OS	23.87	23.03	-	-	-	**58.89**
Cora-PL	33.80	33.97	-	-	-	**46.83**
WK-CL	55.71	61.43	-	-	-	**71.95**
WK-TX	45.05	35.38	-	-	-	**69.29**
WK-WT	53.52	32.85	-	-	-	**77.96**
WK-WC	49.53	33.31	-	-	-	**73.88**
20-news	35.27	41.89	-	-	-	**70.63**
RCVI	27.05	-	-	-	-	**61.97**

where the results for *k-means* and *iterative SVR* are averaged over 50 independent runs and '-' means the corresponding algorithm cannot handle the dataset in reasonable time. Since *GMC* and *iterative SVR* can only handle binary clustering problems, we also provide experiments on several 2-class problems: **Letters**, **Satellite**, **autos vs. motorcycles (Text-1)** and **baseball vs. hockey (Text-2)**. Moreover, for the **UCI-Digits** and **MNIST** datasets, we enumerate all 45 possible class pairs, and report the average clustering results. Furthermore, as the *MMC* and *GMC* algorithms can only handle datasets with no more than a few hundred samples, we perform experiments on **UCI Digits** and focus on those pairs (3 vs 8, 1 vs 7, 2 vs 7, 8 vs 9, 0689 and 1279) that are difficult to differentiate. From the tables we can clearly observe

Table 3. Rand Index comparisons.

Data	KM	NC	MMC	GMC	SVR	CPM3C
Dig 3-8	0.904	0.545	0.823	0.899	0.940	**0.945**
Dig 1-7	0.995	0.504	0.571	0.962	0.995	**1.00**
Dig 2-7	0.940	0.550	0.978	0.994	**1.00**	**1.00**
Dig 8-9	0.835	0.500	0.929	0.733	0.934	**0.956**
Letter	0.706	0.644	-	-	0.867	**0.897**
UCISat	0.922	0.919	-	-	0.939	**0.971**
Text-1	0.500	0.884	-	-	**0.939**	0.905
Text-2	0.500	0.842	-	-	0.887	**0.929**
UCIDig	0.933	0.956	-	-	0.967	**0.989**
MNIST	0.808	0.818	-	-	0.860	**0.921**
Dig 0689	0.696	0.939	0.941	-	-	**0.968**
Dig 1279	0.681	0.909	0.913	-	-	**0.943**
Cora-DS	0.589	**0.744**	-	-	-	0.735
Cora-HA	0.385	0.659	-	-	-	**0.692**
Cora-ML	0.514	0.720	-	-	-	**0.754**
Cora-OS	0.518	0.522	-	-	-	**0.721**
Cora-PL	0.643	0.675	-	-	-	**0.703**
WK-CL	0.603	0.602	-	-	-	**0.728**
WK-TX	0.604	0.514	-	-	-	**0.707**
WK-WT	0.616	0.581	-	-	-	**0.747**
WK-WC	0.581	0.509	-	-	-	**0.752**
20-news	0.581	0.496	-	-	-	**0.782**
RCVI	0.471	-	-	-	-	**0.698**

that our *CPM3C* algorithm can beat other competitive algorithms on almost all the datasets.

4.3. Speed of CPM3C

Table 4 compares the CPU-time of *CPM3C* with other competitive algorithms. According to the table, *CPM3C* is at least 18 times faster than *SVR*, 200 times faster than *GMC*. As reported in (Valizadegan & Jin, 2007), *GMC* is about 100 times faster than *MMC*. Hence, *CPM3C* is still faster than *MMC* by about four orders of magnitude. Moreover, as the sample size increases, the CPU-time of *CPM3C* grows much slower than that of *iterative SVR*, which indicates *CPM3C* has much better scaling property with the sample size than *SVR*. Finally, *CPM3C* also performs much faster than conventional *kmeans*, which is a very appealing result. As for the *Ncut* method, since the calculation of the similarity matrix is very time consuming and usually takes several hours on the text datasets, we do not report the time it spends here.

Table 4. CPU-time (seconds) comparisons.

Data	KM	GMC	SVR	CPM3C
Dig 3-8	0.51	276.16	19.72	1.10
Dig 1-7	0.54	289.53	20.49	0.95
Dig 2-7	0.50	304.81	19.69	0.75
Dig 8-9	0.49	277.26	19.41	0.85
Letter	0.08	-	2133	0.87
UCISat	0.19	-	6490	4.54
Text-1	66.09	-	930.0	19.75
Text-2	52.32	-	913.8	16.16
Dig 0689	34.28	-	-	9.66
Dig 1279	17.78	-	-	17.47
Cora-DS	839.67	-	-	35.31
Cora-HA	204.43	-	-	24.35
Cora-ML	22781	-	-	69.04
Cora-OS	47931	-	-	13.98
Cora-PL	7791.4	-	-	165.0
WK-CL	672.69	-	-	9.534
WK-TX	766.77	-	-	10.53
WK-WT	4135.2	-	-	10.67
WK-WC	1578.2	-	-	9.041
20-news	2387.8	-	-	215.6
RCVI	428770	-	-	587.9

4.4. Dataset size n vs. Speed

In the theoretical analysis section, we state that the computational time of *CPM3C* scales linearly with the number of samples. We present numerical demonstra-

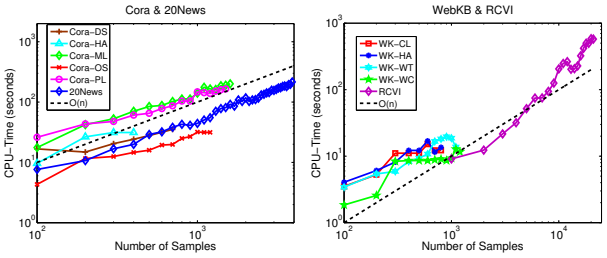

Figure 1. CPU-Time (seconds) of *CPM3C* as a function of dataset size n.

tion for this statement in figure 1, 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 1 shows that the CPU-time of *CPM3C* scales roughly $O(n)$, which is consistent with the statement in section 3.

4.5. ϵ vs. Accuracy & Speed

Theorem 6 states that the total number of iterations involved in *CPM3C* is at most $\frac{R}{\epsilon^2}$, and this means with higher ϵ, the algorithm might converge fast. However, as ϵ is directly related to the training loss in *CPM3C*, we need to determine how small ϵ should be to guarantee sufficient accuracy. We present in figure 2 and figure 3 how clustering accuracy and computational time scale with ϵ. According to figure 2, $\epsilon = 0.01$ is small enough to guarantee clustering accuracy. The log-log plot in figure 3 verifies that the CPU-time of *CPM3C* decreases as ϵ 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 6.

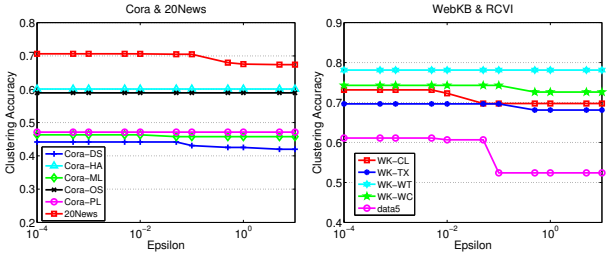

Figure 2. Clustering accuracy of *CPM3C* vs. ϵ.

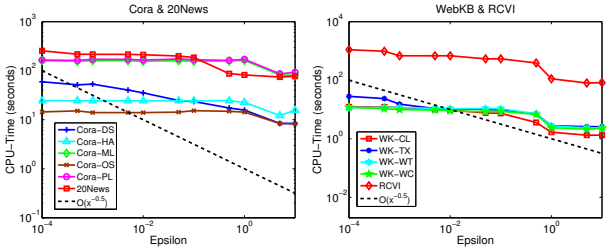

Figure 3. CPU-time (seconds) of *CPM3C* vs. ϵ.

5. Conclusions

We propose the *cutting plane multiclass maximum margin clustering* (*CPM3C*) algorithm in this paper, to cluster data samples with the maximum margin hyperplane. Preliminary theoretical analysis of the algorithm is provided, where we show that the computational time of *CPM3C* scales linearly with the sample size n with guaranteed accuracy. Moreover, experimental evaluations on several real world datasets show that *CPM3C* performs better than existing *MMC* methods, both in efficiency and accuracy.

Acknowledgments

This work is supported by the projects (60721003) and (60675009) of the National Natural Science Foundation of China.

References

Crammer, K., & Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *JMLR*, *2*, 265–292.

Ding, C., He, X., Zha, H., Gu, M., & Simon, H. D. (2001). A min-max cut algorithm for graph partitioning and data mining. *ICDM* (pp. 107–114).

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification*. John Wiley & Sons, Inc.

Joachims, T. (2006). Training linear svms in linear time. *SIGKDD 12*.

Kelley, J. E. (1960). The cutting-plane method for solving convex programs. *Journal of the Society for Industrial Applied Mathematics*, *8*, 703–712.

Lewis, D. D., Yang, Y., Rose, T., & Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *JMLR*, *5*, 361–397.

McCallum, A., Nigam, K., Rennie, J., & Seymore, K. (2000). Automating the contruction of internet portals with machine learning. *Information Retrieval Journal*, *3*, 127–163.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *JASA*, *66*, 846–850.

Schölkopf, B., Smola, A. J., & Müller, K. R. (1999). Kernel principal component analysis. *Advances in kernel methods: support vector learning*, 327–352.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *PAMI*.

Smola, A. J., Vishwanathan, S., & Hofmann, T. (2005). Kernel methods for missing variables. *AISTATS 10*.

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *JMLR*, *6*, 1453–1484.

Valizadegan, H., & Jin, R. (2007). Generalized maximum margin clustering and unsupervised kernel learning. *NIPS 19* (pp. 1417–1424).

Xu, L., Neufeld, J., Larson, B., & Schuurmans, D. (2004). Maximum margin clustering. *NIPS 17*.

Xu, L., & Schuurmans, D. (2005). Unsupervised and semi-supervised multi-class support vector machines. *AAAI*.

Zhang, K., Tsang, I. W., & Kowk, J. T. (2007). Maximum margin clustering made practical. *ICML 24*.

Zhao, B., Wang, F., & Zhang, C. (2008). Efficient maximum margin clustering via cutting plane algorithm. *SDM* (pp. 751–762).