Maximum Margin Clustering Made Practical

Kai Zhang, Ivor W. Tsang, and James T. Kwok

***Abstract*—Motivated by the success of large margin methods in supervised learning, maximum margin clustering (MMC) is a recent approach that aims at extending large margin methods to unsupervised learning. However, its optimization problem is nonconvex and existing MMC methods all rely on reformulating and relaxing the nonconvex optimization problem as semidefinite programs (SDP). Though SDP is convex and standard solvers are available, they are computationally very expensive and only small data sets can be handled. To make MMC more practical, we avoid SDP relaxations and propose in this paper an efficient approach that performs alternating optimization directly on the original nonconvex problem. A key step to avoid premature convergence in the resultant iterative procedure is to change the loss function from the hinge loss to the Laplacian/square loss so that overconfident predictions are penalized. Experiments on a number of synthetic and real-world data sets demonstrate that the proposed approach is more accurate, much faster (hundreds to tens of thousands of times faster), and can handle data sets that are hundreds of times larger than the largest data set reported in the MMC literature.**

***Index Terms*—Large margin methods, maximum margin clustering (MMC), scalability, unsupervised learning.**

I. Introduction

TRADITIONALLY, there are two key learning paradigms in machine learning: supervised learning and unsupervised learning. Supervised learning assumes that the training samples are labeled, while unsupervised learning does not. A particularly successful family of supervised learning methods are the large margin methods [28], [36], with the support vector machine (SVM) being the most prominent. It has outperformed traditional methods even on difficult, high-dimensional problems such as handwritten digit recognition and text categorization. One beauty of the SVM is that it aims at minimizing an upper bound of the generalization error and can be regarded as an approximate implementation of the structural risk minimization principle.

Manuscript received January 13, 2008; revised May 19, 2008 and August 04, 2008; accepted October 31, 2008. First published March 06, 2009; current version published April 03, 2009. This work was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region.

K. Zhang was with the Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. He is now with the Life Science Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 USA (e-mail: kai_zhang@lbl.gov; zk1980@hotmail.com).

I. W. Tsang was with the Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. He is now with the School of Computer Engineering, Nanyang Technological University, Singapore 639798, Singapore (e-mail: ivortsang@ntu.edu.sg; ivor.tsang@gmail.com).

J. T. Kwok is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong (e-mail: jamesk@cse.ust.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2008.2010620

On the other hand, unsupervised learning is more challenging as no labeled data are available. An important task of unsupervised learning is clustering, which aims at grouping the examples into a number of classes, or clusters [2], [14], [18]. Examples belonging to the same cluster should be similar to each other, while those belonging to different clusters should not. Clustering has been extremely valuable for data analysis in practice and is widely used in diverse domains ranging from engineering, medical science, earth science, social science to economics [11], [24], [29], [34], [37].

Over the decades, a battery of clustering approaches have been developed. Examples include the k-means clustering algorithm [16], mixture models [22], and spectral clustering [24], [29]. Motivated by the superiority of large margin methods in supervised learning, there is growing interest in extending them to unsupervised learning. Recently, Xu *et al.* [38] proposed a novel approach called maximum margin clustering (MMC), which performs clustering by simultaneously finding the large margin separating hyperplane between clusters. Experimental results showed that this large margin clustering method (and its variant [35]) have been very successful in many clustering problems. Moreover, it can also be extended to a general framework for both unsupervised and semisupervised learning [39].

However, while large margin supervised learning methods are usually formulated as convex optimization problems [e.g., quadratic programs (QPs) for SVMs], large margin unsupervised learning is much more computationally difficult. As the labels are missing, optimization over all the possible discrete class labelings lead to a hard, nonconvex, combinatorial problem. Existing MMC methods [35], [38], [39] all rely on reformulating and relaxing the nonconvex optimization problem as semidefinite programs (SDPs) [6], which can then be solved by standard SDP solvers such as SDPT3 [33] and SeDuMi [31]. In particular, the generalized maximum margin clustering (GMMC) method [35] reduces the number of parameters in the SDP formulation from n^2 in [38] to n, where n is the number of samples. This leads to significant computational savings.

However, even with the recent advances in interior point methods of mathematical programming [17], [23], solving SDPs is still computationally very expensive. While the worst-case complexity of interior point SDP solvers is provably polynomial [21], [23], the exponent can be quite high. Thus, MMC is often not viable in practice and the data sets that can be handled are very small (the largest data set reported in the literature has only 360 examples [35]). On the other hand, there can be an enormous amount of data available in many real-world learning problems. For example, by casting image segmentation as a clustering problem, a small 200×200 image already has 40 000 pixels to be clustered. In web and data mining, even medium-sized data sets have at least tens/hundreds of thousands of patterns. How to scale up the

clustering methods to cater large scale problems and turn them into practical tools is thus a very challenging research topic.

In this paper, we perform MMC by avoiding the use of expensive SDP relaxations. Instead, we revisit a natural approach that was considered ineffective [38], namely, by performing alternating optimization [3] directly on the original nonconvex problem. However, a straightforward implementation still easily gets stuck in poor locally optimal solutions. Our key modification is to replace the SVM by support vector regression (SVR) with the Laplacian loss. As will be seen, this discourages premature convergence and, computationally, the proposed procedure involves only a sequence of QPs used in SVR. The resultant implementation is fast and scales well compared to existing approaches.

The rest of this paper is organized as follows. Section II gives a brief review on MMC. Section III then describes our proposed approach based on iterative kernel regression procedures. Experimental results are presented in Section IV, where we compare our approach with a number of related approaches on the tasks of clustering and image segmentation. The last section gives concluding remarks.

In the sequel, $\mathbf{A} \succ 0$ (resp., $\mathbf{A} \succeq 0$) means that the matrix $\mathbf{A}$ is symmetric and positive definite (pd) [resp., positive semidefinite (psd)]. Moreover, the transpose of vector/matrix (in both the input and feature spaces) will be denoted by the superscript $^\top$, and $\mathbf{0}, \mathbf{e} \in \mathbb{R}^n$ denote the zero vector and the vector of all ones, respectively. The inequality $\mathbf{v} = [v_1, \ldots, v_k]^\top \geq \mathbf{0}$ means that $v_i \geq 0$ for $i = 1, \ldots, k$. In addition, $\mathbb{R}^k_+$ denotes the set of nonnegative vectors in $\mathbb{R}^k$. Some preliminary results have been reported in [43].

II. Maximum Margin Clustering

Large margin methods, notably the SVMs, have been highly successful in supervised learning. Given a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathcal{X}$ is the input and $y_i \in \{\pm 1\}$ is the output, the SVM finds a large margin hyperplane $f(\mathbf{x}) = \mathbf{w}^\top \varphi(\mathbf{x}) + b$ separating patterns of opposite classes. Here, φ is the mapping induced by the kernel function k. Computationally, this leads to the following optimization problem:

$$\begin{array}{ll} \min_{\mathbf{w},b,\boldsymbol{\xi}} & \|\mathbf{w}\|^2 + 2C\boldsymbol{\xi}^\top \mathbf{e} \\ \text{s.t.} & y_i \left(\mathbf{w}^\top \varphi(\mathbf{x}_i) + b\right) \geq 1 - \xi_i \\ & \xi_i \geq 0, \qquad i = 1, \ldots, n \end{array}$$

where $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_n]^\top$ is the vector of slack variables for the errors, and $C > 0$ is a tradeoff parameter between the smoothness ($\|\mathbf{w}\|^2$) and fitness ($\boldsymbol{\xi}^\top \mathbf{e}$) of the decision function $f(\mathbf{x})$.

Motivated by its success in supervised learning, MMC [38] aims at extending large margin methods to unsupervised learning by finding a large margin hyperplane separating patterns of opposite clusters. Here, we consider the case when there are only two clusters. As the class (cluster) labels $\mathbf{y} = [y_1, \ldots, y_n]^\top$ are unknown in the unsupervised setting, one can obtain a trivially "optimal" solution with infinite margin by assigning all patterns to a single cluster. To prevent such a useless solution, Xu *et al.* [38] introduced a class balance constraint that requires $\mathbf{y}$ to satisfy

$$-\ell \leq \mathbf{e}^\top \mathbf{y} \leq \ell \tag{1}$$

where $\ell \geq 0$ is a user-defined constant controlling the class imbalance. Then, the margin is maximized with respect to (w.r.t.) both the unknown $\mathbf{y}$ and the unknown SVM parameter $(\mathbf{w}, b)$ as

$$\begin{array}{ll} \min_{\mathbf{y}} \min_{\mathbf{w},b,\boldsymbol{\xi}} & \|\mathbf{w}\|^2 + 2C\boldsymbol{\xi}^\top \mathbf{e} \\ \text{s.t.} & y_i \left(\mathbf{w}^\top \varphi(\mathbf{x}_i) + b\right) \geq 1 - \xi_i \\ & \xi_i \geq 0, \qquad y_i \in \{\pm 1\}, \qquad i = 1, \ldots, n \\ & -\ell \leq \mathbf{e}^\top \mathbf{y} \leq \ell. \end{array} \tag{2}$$

Recall that the SVM is usually solved in its dual form

$$\begin{array}{ll} \max_{\boldsymbol{\lambda}} & 2\boldsymbol{\lambda}^\top \mathbf{e} - \boldsymbol{\lambda}^\top (\mathbf{K} \circ \mathbf{y}\mathbf{y}^\top)\boldsymbol{\lambda} \\ \text{s.t.} & \boldsymbol{\lambda}^\top \mathbf{y} = 0 \\ & C\mathbf{e} \geq \boldsymbol{\lambda} \geq \mathbf{0} \end{array} \tag{3}$$

where $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_n]^\top$ is the vector of Lagrangian multipliers, $\mathbf{K} = [k(\mathbf{x}_i, \mathbf{x}_j)]_{n \times n}$ is the kernel matrix with $k(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^\top \varphi(\mathbf{x}_j)$, and $\circ$ denotes the elementwise product between matrices. Hence, (2) can also be written as a minimax saddle-point problem

$$\begin{array}{ll} \min_{\mathbf{y}} \max_{\boldsymbol{\lambda}} & 2\boldsymbol{\lambda}^\top \mathbf{e} - \boldsymbol{\lambda}^\top (\mathbf{K} \circ \mathbf{y}\mathbf{y}^\top)\boldsymbol{\lambda} \\ \text{s.t.} & \boldsymbol{\lambda}^\top \mathbf{y} = 0, \qquad C\mathbf{e} \geq \boldsymbol{\lambda} \geq \mathbf{0} \\ & -\ell \leq \mathbf{e}^\top \mathbf{y} \leq \ell \\ & y_i \in \{\pm 1\}, \qquad i = 1, \ldots, n. \end{array} \tag{4}$$

Note that the constraint $y_i \in \{\pm 1\} \Leftrightarrow y_i^2 - 1 = 0$ is nonconvex. Thus, (4) is a nonconvex optimization problem that is difficult to solve. Following [20], the above optimization problem can be simplified to

$$\begin{array}{ll} \min\limits_{\mathbf{y},\delta,\boldsymbol{\mu},\boldsymbol{\nu},\bar{b}} & \delta \\ \text{s.t.} & \begin{bmatrix} (\mathbf{y}\mathbf{y}^\top) \circ \mathbf{K} & \mathbf{e} + \boldsymbol{\mu} - \boldsymbol{\nu} - \bar{b}\mathbf{y} \\ (\mathbf{e} + \boldsymbol{\mu} - \boldsymbol{\nu} - \bar{b}\mathbf{y})^\top & \delta - 2C\boldsymbol{\nu}^\top \mathbf{e} \end{bmatrix} \succeq 0 \\ & \boldsymbol{\mu} \geq \mathbf{0}, \qquad \boldsymbol{\nu} \geq \mathbf{0}, \qquad -\ell \leq \mathbf{e}^\top \mathbf{y} \leq \ell \\ & y_i \in \{\pm 1\}, \qquad i = 1, \ldots, n \end{array} \tag{5}$$

where $\bar{b} \in \mathbb{R}$ is the dual variable for the equality constraint $\boldsymbol{\lambda}^\top \mathbf{y} = 0$, and $\boldsymbol{\mu}, \boldsymbol{\nu} \in \mathbb{R}^n_+$ are nonnegative vectors of the dual variables for the box constraints $C\mathbf{e} \geq \boldsymbol{\lambda} \geq \mathbf{0}$. Instead of working with the label vector $\mathbf{y}$ directly, problem (5) is further reduced in [38] to a convex problem by using two relaxations. First, the label relation matrix $\mathbf{y}\mathbf{y}^\top \in \{\pm 1\}^{n \times n}$ in (5) is replaced by a $n \times n$ real-valued psd matrix $\mathbf{M} \succeq 0$. As the desired matrix $\mathbf{M}$ is equal to $\mathbf{y}\mathbf{y}^\top$, the diagonal entries of $\mathbf{M}$ are set to 1, i.e., $\mathrm{diag}(\mathbf{M}) = \mathbf{e}$. Since $y_i \in \{\pm 1\}$, the balance constraint in (1) becomes $-\ell\mathbf{e} \leq \mathbf{M}\mathbf{e} \leq \ell\mathbf{e}$. Moreover, in order to avoid $\bar{b}$ and $\mathbf{y}$ being bilinear in the psd constraint (6), the second relaxation sets the variable $\bar{b}$ to zero, which is equivalent to assuming that the decision hyperplane $f(\mathbf{x})$ passes through the origin (i.e.,

$b = 0$). By doing this, it can be shown that (5) can be relaxed to the following SDP:

$$\begin{aligned} \min_{\mathbf{M},\delta,\boldsymbol{\mu},\boldsymbol{\nu}} \quad & \delta \\ \text{s.t.} \quad & \begin{bmatrix} \mathbf{M} \circ \mathbf{K} & \mathbf{e} + \boldsymbol{\mu} - \boldsymbol{\nu} \\ (\mathbf{e} + \boldsymbol{\mu} - \boldsymbol{\nu})^\top & \delta - 2C\boldsymbol{\nu}^\top \mathbf{e} \end{bmatrix} \succeq 0 \\ & \boldsymbol{\mu} \geq \mathbf{0}, \boldsymbol{\nu} \geq \mathbf{0}, \qquad \mathbf{M} \succeq 0 \\ & -\ell\mathbf{e} \leq \mathbf{M}\mathbf{e} \leq \ell\mathbf{e}, \qquad \operatorname{diag}(\mathbf{M}) = \mathbf{e}. \end{aligned} \tag{6}$$

After obtaining the optimal matrix $\mathbf{M}^*$ from (6), the soft cluster labels $\mathbf{y}$ can be recovered as $\mathbf{y} = \sqrt{\mu_1}\mathbf{v}_1$, where $\mu_1, \mathbf{v}_1$ are the leading eigenvalue and eigenvector of $\mathbf{M}^*$, respectively. While initially developing the two-cluster case, MMC is also extended to the multicluster formulation [39], which again leads to a SDP.

Note that as $\mathbf{M} \in \mathbb{R}^{n \times n}$, the number of parameters in (6) scales quadratically with n, the number of examples. Moreover, as mentioned above, the decision hyperplane is assumed to pass through the origin. Recently, these two problems are alleviated by the GMMC algorithm [35], which leads to the following SDP:

$$\begin{aligned} \max_{\boldsymbol{\gamma}} \quad & \boldsymbol{\gamma}^\top \mathbf{e} \\ \text{s.t.} \quad & \mathbf{P}^\top \mathbf{K}^{-1} \mathbf{P} + C_e \mathbf{e}_0 \mathbf{e}_0^\top - \sum_{i=1}^{n} \gamma_i \mathbf{I}_i^{(n+1)} \succeq 0 \\ & \mathbf{0} \leq \boldsymbol{\gamma} \leq C\mathbf{e} \end{aligned} \tag{7}$$

where $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_n]^\top$, $\mathbf{P} = [\mathbf{I}^{(n)}, \mathbf{e}]$ with $\mathbf{I}^{(n)}$ being the $n \times n$ identity matrix, $\mathbf{e}_0 = [\mathbf{e}^\top \; 0]^\top$, $\mathbf{I}_i^{(n+1)}$ is the $(n+1) \times (n+1)$ matrix with all zeroes except that the ith diagonal element is 1, and C_e is another user-defined parameter. The number of parameters in this SDP is reduced from n^2 in MMC to n, and thus leads to significant computational savings.

III. Proposed Procedure

In Section III-A, we first revisit the use of alternating optimization for the original nonconvex MMC problem in (2). Computationally, this allows the problem to be formulated as a sequence of QPs for which many efficient and scalable QP solvers are available [5]. However, empirically, it suffers from premature convergence and easily gets stuck in poor local optima. Our key proposal is to replace the SVM by either SVR with the Laplacian loss (Section III-B) or the least squares SVM (LS-SVM), which uses the square loss (Section III-C). An efficient procedure for enforcing the class balance constraint is discussed in Section III-D. Finally, complexity analysis is presented in Section III-E, which is then concluded by some discussion in Section III-F.

A. Iterative SVM

A natural way to solve (2) is to use a simple iterative approach based on alternating optimization [3]. This is also similar to the one proposed in [40], and the complete algorithm is summarized in Algorithm 1. First, we fix $\mathbf{y}$ and maximize (3) w.r.t., $\boldsymbol{\lambda}$, which is just a standard SVM training. Then, we fix $\boldsymbol{\lambda}$ and minimize (2) w.r.t. $\mathbf{y}$, which reduces to

TABLE I
Clustering Errors on Digits "3" and "9" Using the iterSVM and the iterSVR

C	0.00001	0.0001	0.001	0.01	0.1	1	100
iterSVM	31.13%	21.27%	19.7%	19.2%	19.7%	19.46%	19.46%
iterSVR	14.01%	2.98%	2.85%	3.24%	3.76%	3.5%	3.5%

$$\begin{aligned} \min_{\mathbf{y},\xi_i} \quad & \boldsymbol{\xi}^\top \mathbf{e} \\ \text{s.t.} \quad & y_i \left(\mathbf{w}^\top \varphi(\mathbf{x}_i) + b \right) \geq 1 - \xi_i \\ & \xi_i \geq 0, \qquad i = 1, \ldots, n \\ & y_i \in \{\pm 1\}, \qquad i = 1, \ldots, n \\ & -\ell \leq \mathbf{e}^\top \mathbf{y} \leq \ell. \end{aligned} \tag{8}$$

As shown by the following proposition, (8) can be solved without the use of any optimization solver.

Proposition 1: For a fixed b, the optimal strategy to determine the y_i's in (8) is to assign all y_i's as -1 for those with $\mathbf{w}^\top \varphi(\mathbf{x}_i) + b \leq 0$, and assign y_i's as 1 for those with $\mathbf{w}^\top \varphi(\mathbf{x}_i) + b > 0$.

Proof: Assume, to the contrary, that the optimal assignment of $\mathbf{y}$ is not according to the sorted order of $\mathbf{w}^\top \varphi(\mathbf{x}_j)$'s. So there are two points $\mathbf{x}_j, \mathbf{x}_k$, with $f(\mathbf{x}_j) = \mathbf{w}^\top \varphi(\mathbf{x}_j) + b > 0$ and $f(\mathbf{x}_k) = \mathbf{w}^\top \varphi(\mathbf{x}_k) + b < 0$, but the optimal predictions are $y_j = -1$ and $y_k = 1$. The objective in (8) can be written as

$$\begin{aligned} & \sum_{i=1}^{n} \max(0, 1 - y_i f(\mathbf{x}_i)) \\ & = \left(\sum_{i \neq j,k} \max(0, 1 - y_i f(\mathbf{x}_i)) \right) \\ & \quad + \max(0, 1 - (-1) f(\mathbf{x}_j)) + \max(0, 1 - f(\mathbf{x}_k)) \\ & > \left(\sum_{i \neq j,k} \max(0, 1 - y_i f(\mathbf{x}_i)) \right) \\ & \quad + \max(0, 1 - f(\mathbf{x}_j)) + \max(0, 1 + f(\mathbf{x}_k)) \end{aligned}$$

which is thus nonoptimal, a contradiction. □

With Proposition 1, we then proceed to determine b as follows. First, we sort the $\mathbf{w}^\top \varphi(\mathbf{x}_j)$'s and use the set of midpoints between any two consecutive sorted values as the candidates of b. From these sorted b's, the first $(n-\ell)/2$ and the last $(n-\ell)/2$ of them will not satisfy the class balance constraint in (1) and so can be dropped. For each remaining candidate, we determine the y_i's according to Proposition 1 and compute the corresponding objective value in (9). Finally, we choose the b that has the smallest objective. With this learned b, the optimal strategy for determining $\mathbf{y}$ in Proposition 1 is obviously the same as SVM's decision rule $y_i = \operatorname{sign}(\mathbf{w}^\top \varphi(\mathbf{x}_i) + b)$. Therefore, the whole procedure simply alternates between standard SVM training and testing (on the given points) until convergence.

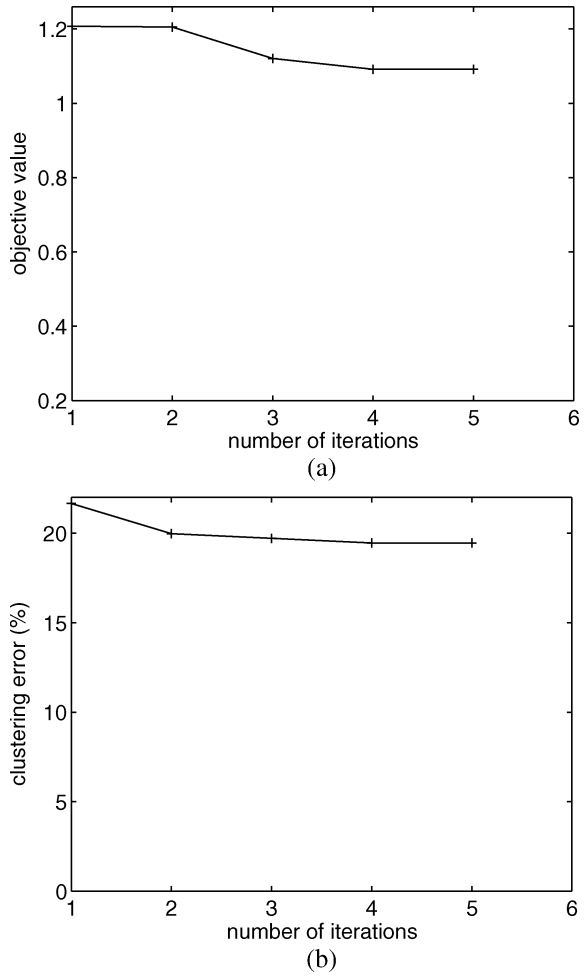


Fig. 1. Poor performance of the iterative SVM procedure (with $C = 1$). (a) Objective value. (b) Clustering error (in percent).

Algorithm 1: Iterative SVM (iterSVM) procedure

1: Initialize the labels $\mathbf{y}$ (e.g., by using a simple clustering algorithm such as k-means).
2: Fix $\mathbf{y}$ and perform standard SVM training.
3: Compute $\mathbf{w}$ from the Karush–Kuhn–Tucker (KKT) conditions.
4: Compute the bias b as described in Section III-A.
5: Assign the labels as $y_i = \text{sign}(\mathbf{w}^\top \varphi(\mathbf{x}_i) + b)$.
6: Repeat steps 2–5 until convergence.

Recall that clustering becomes transductive learning if some labeled patterns are available. Recently, Collobert *et al.* [9], [10] showed that the constrained concave-convex procedure (CCCP) [42] can be used to efficiently solve the SVM in such a transductive setting. Interestingly, if we discard the balance constraint, then applying the CCCP in our MMC setting can lead to an algorithm similar to Algorithm 1. Details are shown in the Appendix.

1) Poor Empirical Performance: In practice, the performance of this iterative SVM is not satisfactory and its improvement over the initial labeling is usually insignificant. An example is shown in Fig. 1. Here, the task is to separate digits 3 and 9 [from the University of California at Irvine (UCI) optdigits data set], where each digit has about 390 samples. The linear kernel is used. Initialization[1] (step 1) is performed with the normalized cut algorithm [29], which yields an error rate of 21.9%. Fig. 1 shows that both the objective value and the clustering error[2] change little during the iterations. The same problem is observed when different C's are used (Table I).

To understand this poor performance, consider the hinge loss [Fig. 2(a)] used by the SVM

$$L_h = \begin{cases} 0, & \text{if } y_i f_i \geq 1 \\ 1 - y_i f_i, & \text{otherwise} \end{cases}$$

where $f_i = f(\mathbf{x}_i)$ and $y_i f_i$ defines the margin for patterns $\mathbf{x}_i$. Because of this loss, the SVM tries to push the $y_i f_i$'s to the right of the elbow (i.e., the point where $y_i f_i = 1$). As evidenced from the empirical margin distribution of $y_i f_i$'s [Fig. 2(b)], most of the patterns have margins $y_i f_i \gg 1$ (even with $C = 0.0001$). If we want to flip the label of a pattern, the loss changes from the solid line to the dashed line in Fig. 3. The resultant increase in the loss will be very large as most of the patterns are far from the elbow (e.g., point "b"). Therefore, the SVM is unwilling to flip the class labels. In other words, the procedure overcommits to the initial label estimates, and thus easily gets stuck in a local optimum.

B. Iterative SVR

1) Discouraging Premature Convergence: To prevent overconfident predictions (i.e., $y_i f_i \gg 1$) and thus premature convergence of the iterSVM, our approach aims at changing the loss function so as to discourage $y_i f_i$'s from lying on the far right of the elbow. This can be done by penalizing predictions with $y_i f_i > 1$. Combining with the original hinge loss that penalizes predictions with $y_i f_i < 1$, we obtain the Laplacian loss [Fig. 4(a)]

$$L_s = |f_i - y_i|.$$

This can also be seen as a special case of the ε-insensitive loss (with $\varepsilon = 0$) commonly used in SVR. The Laplacian loss can be equivalently expressed in terms of $y_i f_i$ as

$$L_s = \begin{cases} 1 - y_i f_i, & y_i f_i < 1 \\ y_i f_i - 1, & y_i f_i \geq 1. \end{cases}$$

Thus, points with $y_i f_i < 1$ receive the same loss value as the hinge loss, while points with $y_i f_i > 1$ are penalized as desired. Moreover, as L_s is symmetric around the point where $y_i f_i = 1$, the resulting $y_i f_i$'s will tend to lie around this point at the solution. As an example, Fig. 4(b) plots the empirical distribution of $y_i f_i$'s obtained with the Laplacian loss on the same digits task previously shown in Fig. 2(b). As can be seen, the $y_i f_i$'s are

[1]The purpose of using the normalized cut algorithm for initialization here is to demonstrate that the iterative SVM has poor performance even when started with a good initial solution. In the sequel (Section IV), we will always use the standard k-means clustering algorithm for initialization, which is computationally much cheaper.

[2]Clustering error: the percentage of examples whose class labels are not correctly assigned using the clustering algorithm.

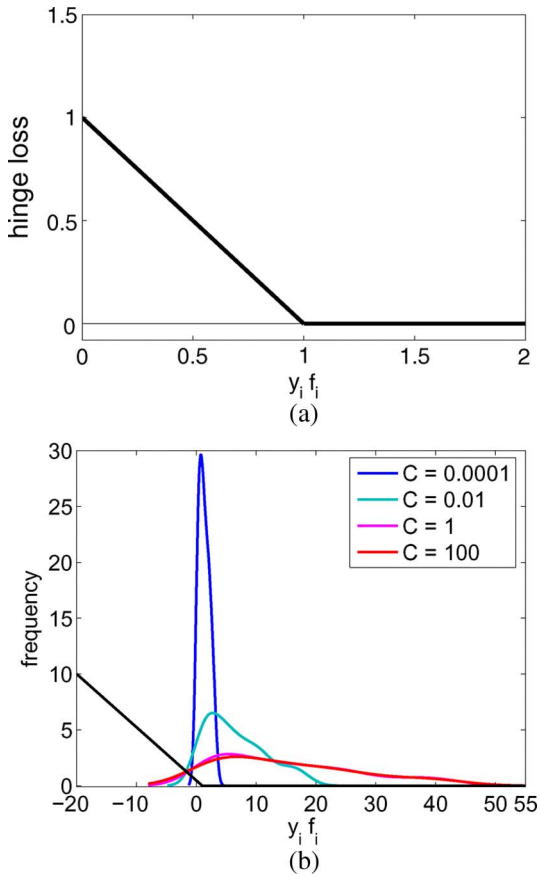


Fig. 2. Loss function and empirical distribution of $y_i f_i$'s on the digits task for iterSVM. (a) Hinge loss. (b) Distribution of $y_i f_i$'s.

now close to 1. Therefore, it is easier to flip the labels if needed, and one can escape from a poor solution at a smaller expense. As expected, this leads to a significant improvement in the clustering performance compared to that of iterSVM (Table I). The complete iterative SVR procedure (also called iterSVR in the sequel) is shown in Algorithm 2.

Algorithm 2: Iterative SVR/LS-SVM

1: Initialize the labels $\mathbf{y}$ by a simple clustering method.
2: Fix $\mathbf{y}$, and perform SVR with Laplacian loss/LS-SVM.
3: Compute $\mathbf{w}$ from the KKT condition.
4: Compute the bias b as described in Section III-D.
5: Assign the labels as $y_i = \text{sign}(\mathbf{w}^\top \varphi(\mathbf{x}_i) + b)$.
6: Repeat steps 2–5 until convergence.

A common objection to the Laplacian loss is that the resultant SVR solution will not be sparse. However, in the context of MMC, typically we are only interested in the clustering solution. Besides, after the clustering solution has been obtained, one can still obtain a sparse classification model by running the SVM if desired.

2) Effect of Nonzero ϵ: Recall that the Laplacian loss is a special case of the ε-insensitive loss with $\varepsilon = 0$. In this section,

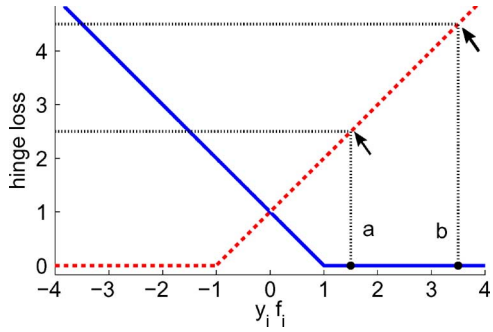


Fig. 3. Flipping the labels when the hinge loss is used.

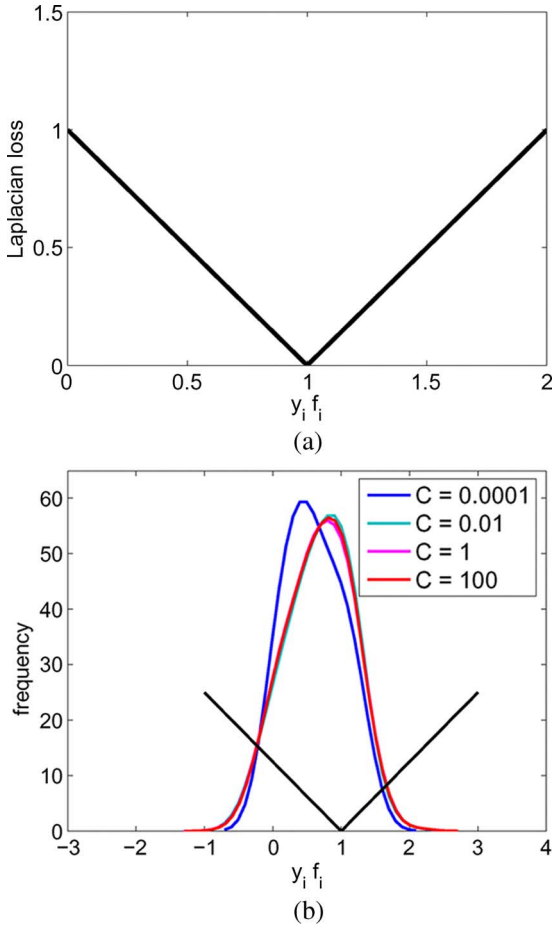


Fig. 4. Loss function and empirical distribution of $y_i f_i$'s on the digits task for SVR with the Laplacian loss. (a) Laplacian loss. (b) Distribution of $y_i f_i$'s.

we investigate the effectiveness of a nonzero ε. Using the same digit task in Section III-A1 as an example, we study the objective value[3] and clustering error of the iterative SVM procedure, as well as those of the iterative SVR procedure using $\varepsilon = 0, 0.5$, and 0.9. As can be seen from Fig. 5, iterSVR using the Laplacian loss leads to better objective values and clustering performance, while iterSVR using a large ε has little improvement on both criteria. Indeed, as ε is increased, the advantage of iterSVR gradually diminishes. This can be explained by examining the

[3]Here, we report the value of the SVM objective in (3). For iterSVR, this objective is computed by fixing the obtained value of $\mathbf{y}$ in (3) and then perform standard SVM training.

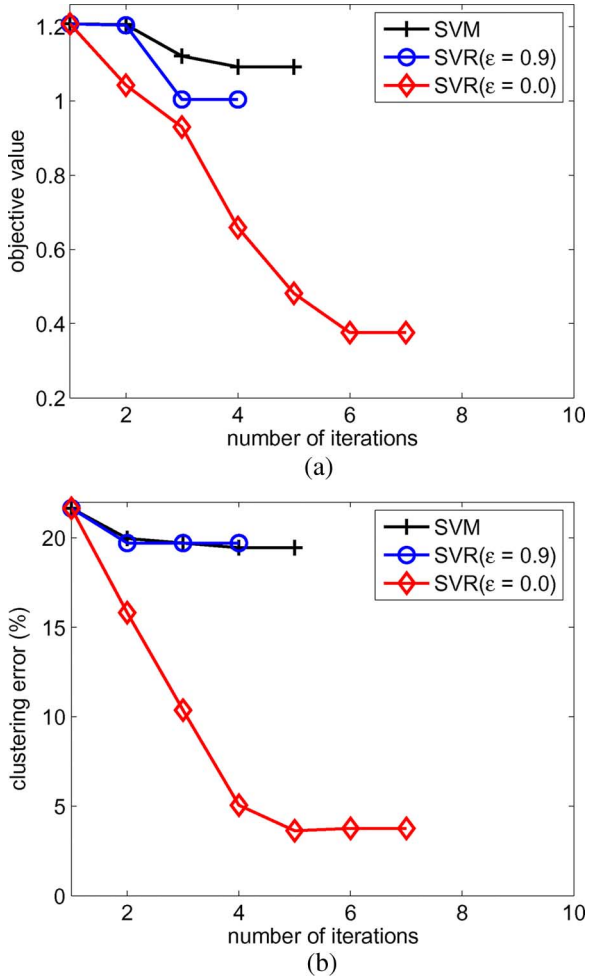


Fig. 5. Comparison of the objective values and clustering errors obtained by the iterative SVM and SVR procedures. (a) Objective value. (b) Clustering error (in percent).

empirical distribution of the resultant $y_i f_i$'s (Fig. 6). As can be seen, when ε is large, most of the $y_i f_i$'s are distributed in the valley where the loss is 0, and thus the right elbow of the loss function becomes ineffective.

C. Iterative Least Squares SVM

Instead of using the Laplacian loss in Section III-B, one can also penalize overconfident predictions by using the square loss

$$L_q = (f_i - y_i)^2 = (y_i f_i - 1)^2.$$

As both the Laplacian and square losses are symmetric around $y_i f_i = 1$ and have similar trends, we expect both to have similar performance, as will be verified experimentally in Section IV. Note that with the use of the square loss, step 2 of Algorithm 1 now involves the training of an LS-SVM [32], which can be obtained efficiently by solving a simple linear system.

D. Enforcing the Class Balance Constraint

As discussed in Section II, in order to avoid trivially "optimal" clustering solutions, one has to enforce a class balance

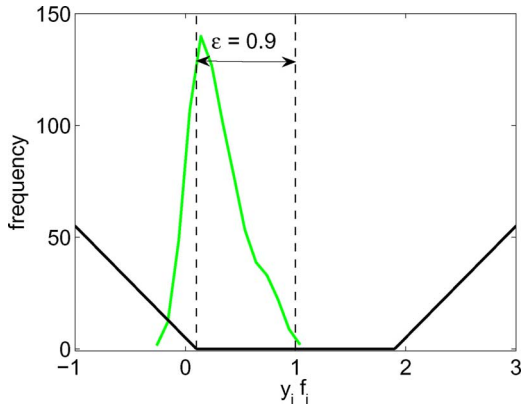


Fig. 6. Loss function and empirical distribution of $y_i f_i$'s on the digits task when $\varepsilon = 0.9$.

constraint such as the one in (1). Here, we require each of the $\mathbf{y}$'s obtained throughout the proposed iterative process to satisfy (1). To guarantee this, we cannot compute the bias b from the KKT conditions as usual.

Consider the iterSVR procedure in Section III-B. Recall that the primal problem of SVR with the Laplacian loss is

$$\begin{aligned} \min_{\mathbf{w},b,\xi_i,\xi_i^*} \quad & \|\mathbf{w}\|^2 + 2C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \\ \text{s.t.} \quad & y_i - \left(\mathbf{w}^\top \varphi(\mathbf{x}_i) + b\right) \le \xi_i \\ & -y_i + \left(\mathbf{w}^\top \varphi(\mathbf{x}_i) + b\right) \le \xi_i^* \\ & \xi_i \ge 0, \qquad \xi_i^* \ge 0 \end{aligned}$$

for $i = 1, \ldots, n$, where ξ_i and ξ_i^* are slack variables. With the labels being unknown, the complete MMC problem becomes

$$\begin{aligned} \min_{\mathbf{y},\mathbf{w},b,\xi_i,\xi_i^*} \quad & \|\mathbf{w}\|^2 + 2C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \\ \text{s.t.} \quad & y_i - \left(\mathbf{w}^\top \varphi(\mathbf{x}_i) + b\right) \le \xi_i \\ & -y_i + \left(\mathbf{w}^\top \varphi(\mathbf{x}_i) + b\right) \le \xi_i^* \\ & \xi_i \ge 0, \qquad \xi_i^* \ge 0 \\ & y_i \in \{\pm 1\}, \qquad i = 1, \ldots, n \\ & -\ell \le \mathbf{e}^\top \mathbf{y} \le \ell. \end{aligned}$$

Similarly, for the iterLS-SVM procedure in Section III-C, the primal problem of LS-SVM is

$$\begin{aligned} \min_{\mathbf{w},b,\xi_i} \quad & \|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i^2 \\ \text{s.t.} \quad & y_i - \left(\mathbf{w}^\top \varphi(\mathbf{x}_i) + b\right) = \xi_i, \qquad i = 1, \ldots, n \end{aligned}$$

and the corresponding MMC problem is

$$\begin{aligned} \min_{\mathbf{y},\mathbf{w},b,\xi_i} \quad & \|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i^2 \\ \text{s.t.} \quad & y_i - \left(\mathbf{w}^\top \varphi(\mathbf{x}_i) + b\right) = \xi_i \\ & y_i \in \{\pm 1\}, \qquad i = 1, \ldots, n \\ & -\ell \le \mathbf{e}^\top \mathbf{y} \le \ell. \end{aligned}$$

After $\mathbf{w}$ is obtained from the optimization problem of SVR/LS-SVM, both problems reduce to the form

$$\begin{aligned} \min_{\mathbf{y},b} \quad & \sum_{i=1}^{n} \left|\mathbf{w}^\top \varphi(\mathbf{x}_i) + b - y_i\right|^p \\ \text{s.t.} \quad & y_i \in \{\pm 1\}, \qquad i = 1, \ldots, n \\ & -\ell \le \mathbf{e}^\top \mathbf{y} \le \ell \end{aligned} \tag{9}$$

where $p = 1$ for the SVR model, and $p = 2$ for the LS-SVM model.

Analogous to the iterSVM in Section III-A, the following proposition shows that (9) can be easily solved without the use of any optimization solver.

Proposition 2: For a fixed b, the optimal strategy to determine the y_i's in (9) is to assign all y_i's as -1 for those with $\mathbf{w}^\top \varphi(\mathbf{x}_i) + b \le 0$, and assign y_i's as 1 for those with $\mathbf{w}^\top \varphi(\mathbf{x}_i) + b > 0$.

Proof: Assume, to the contrary, that the optimal assignment of $\mathbf{y}$ is not according to the sorted order of $\mathbf{w}^\top \varphi(\mathbf{x}_j)$'s. So there are two points $\mathbf{x}_j, \mathbf{x}_k$, with $f(\mathbf{x}_j) = \mathbf{w}^\top \varphi(\mathbf{x}_j) + b > 0$ and $f(\mathbf{x}_k) = \mathbf{w}^\top \varphi(\mathbf{x}_k) + b < 0$, but the optimal predictions are $y_j = -1$ and $y_k = 1$. Then, the objective in (9) is

$$\begin{aligned} & \sum_{i=1}^{n} \left|\mathbf{w}^\top \varphi(\mathbf{x}_i) + b - y_i\right|^p \\ &= \left(\sum_{i \ne j,k} |f(\mathbf{x}_i) - y_i|^p \right) + |f(\mathbf{x}_j) - (-1)|^p + |f(\mathbf{x}_k) - 1|^p \\ &> \left(\sum_{i \ne j,k} |f(\mathbf{x}_i) - y_i|^p \right) + |f(\mathbf{x}_j) - 1|^p + |f(\mathbf{x}_k) + 1|^p \end{aligned}$$

which is thus nonoptimal, a contradiction. □

Therefore, the optimal bias b and label assignment $\mathbf{y}$ can be determined in the same manner as described in Section III-A.

E. Time Complexities

Consider a general SDP problem of the form

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{f}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}_0^i + \sum_{j=1}^{l} x_j \mathbf{A}_j^i \succeq 0, \qquad i = 1, \ldots, m \end{aligned} \tag{10}$$

where $\mathbf{x} = [x_1, \ldots, x_l]^\top \in \mathbb{R}^l$ is the vector of variables and $\mathbf{f} \in \mathbb{R}^l$ and $\mathbf{A}_j^i \in \mathbb{R}^{n_i \times n_i}$'s are symmetric matrices. On using interior-point methods to solve such SDPs, it is known that the time complexity per iteration is $O(l^2 \sum_{i=1}^{m} n_i^2)$ [21], and the number of iterations is usually $O(\sqrt{\sum_{i=1}^{m} n_i})$ [23].

Now, rewrite the SDP formulation of MMC in (6) in the form of (10). Denote $\mathbf{M} = [M_{ij}]_{i,j=1}^{n}$, where n is the number of examples to be clustered. We then have

$$\begin{aligned} \min_{\mathbf{M},\delta,\boldsymbol{\mu},\boldsymbol{\nu}} \quad & \delta \\ \text{s.t.} \quad & \begin{bmatrix} \mathbf{0} & \mathbf{e} \\ \mathbf{e}^\top & -2C\boldsymbol{\nu}^\top \mathbf{e} \end{bmatrix} + \sum_{\substack{i,j=1,\ldots,n \\ i \ne j}} M_{ij} \begin{bmatrix} \mathbf{I}_{ij}^{(n)} \circ \mathbf{K}/2 & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} \\ & + \sum_{i=1}^{n} M_{ii} \begin{bmatrix} \mathbf{I}_i^{(n)} \circ \mathbf{K} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \sum_{i=1}^{n} \mu_i \mathbf{I}_{i,n+1}^{(n+1)} \\ & + \sum_{i=1}^{n} \nu_i \left(-\mathbf{I}_{i,n+1}^{(n+1)} \right) + \delta \mathbf{I}_{n+1}^{(n+1)} \succeq 0 \end{aligned} \tag{11}$$

$$\sum_{\substack{i,j=1,\ldots,n \\ i \ne j}} M_{ij} \mathbf{I}_{ij}^{(n)}/2 + \sum_{i=1}^{n} M_{ii} \mathbf{I}_i^{(n)} \succeq 0 \tag{12}$$

$$\mu_i \ge 0 \quad \nu_i \ge 0, \qquad i = 1, \ldots, n \tag{13}$$

$$\ell + \sum_{j=1}^{n} M_{ij} \ge 0 \quad \ell - \sum_{j=1}^{n} M_{ij} \ge 0, \qquad i = 1, \ldots, n \tag{14}$$

$$-1 + M_{ii} \ge 0 \quad 1 - M_{ii} \ge 0, \qquad i = 1, \ldots, n. \tag{15}$$

Here, $\mathbf{I}_{ij}^{(n)}$'s are $n \times n$ symmetric matrices with all zeros except that the ijth and jith entries are ones, and $\mathbf{I}_i^{(n)}$'s are $n \times n$ diagonal matrices with all zeros except that the ith diagonal entry is one. On re-expressing in terms of the symbols in (10), we have $l = (n^2 + n)/2 + 2n + 1$, $n_i = n + 1$ for (11), and $n_i = n$ for (12), $n_i = 1$ for each inequality constraint in (13)–(15). So the total time complexity is $O(((n^2 + n)/2 + 2n + 1)^2((n + 1)^2 + n^2 + 6n) \cdot \sqrt{8n + 1}) = O(n^{6.5})$.

Similarly, rewrite the SDP of GMMC in (7) to the form of (10) as

$$\begin{aligned} \max_{\boldsymbol{\gamma}} \quad & \boldsymbol{\gamma}^\top \mathbf{e} \\ \text{s.t.} \quad & \left(\mathbf{P}^\top \mathbf{K}^{-1} \mathbf{P} + C_e \mathbf{e}_0 \mathbf{e}_0^\top\right) + \sum_{i=1}^{n} \gamma_i \left(-\mathbf{I}_{n+1}^i\right) \succeq 0 \end{aligned} \tag{16}$$

$$C - \gamma_i \ge 0, \qquad \gamma_i \ge 0, \qquad i = 1, \ldots, n. \tag{17}$$

Then, $l = n$, $n_i = n + 1$ for (16), and $n_i = 1$ for each inequality constraint in (17). So the time complexity is $O(n^2((n + 1)^2 + n) \cdot \sqrt{2n + 1}) = O(n^{4.5})$. Thus, both MMC and GMMC are computationally expensive even on small data sets.

On the other hand, the iterSVR algorithm involves only a sequence of QPs. Modern SVM implementations typically have an empirical time complexity that scales between $O(n)$ and $O(n^{2.3})$ (for solving one such QP) [27], whereas for the iterLS-SVM algorithm, each LS-SVM iteration involves only solving a linear system with n variables and n constraints, and so the time complexity is $O(n^3)$. The number of iterations in iterSVR/iterLS-SVM is usually small (around ten in practice). As for the determination of b described in Section III-D, $\mathbf{w}^\top \varphi(\mathbf{x}_j)$'s can be obtained without any cost from the cached gradients of the SVM solvers, and the sorting of these $\mathbf{w}^\top \varphi(\mathbf{x}_j)$'s takes $O(n \log(n))$ time. After removing those b candidates that do not satisfy the class balance constraint, there are $O(\ell)$ candidates left and each one takes $O(n)$ time to compute the objective. So the determination of b takes a total of only $O(n \log(n) + \ell n)$ time. Hence, both iterSVR and iterLS-SVM are computationally efficient.

F. Discussions

Our iterative algorithm is similar to self-training [41], [44], in which a classifier uses its own predictions to help classify the unlabeled data. A small amount of labeled data is usually available and the labels of some highly confident, but previously unlabeled patterns are assigned by the classifier incrementally. The classifier is then retrained and the procedure reiterates until all

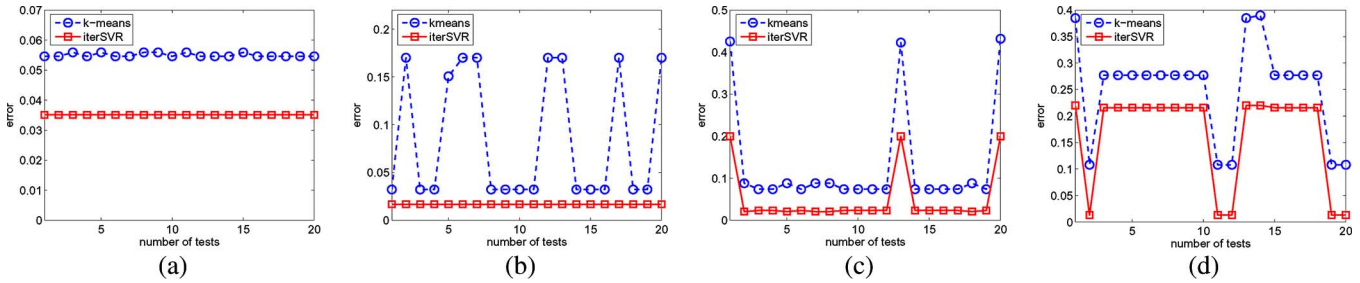


Fig. 7. Clustering errors with different k-means initializations. (a) Digits 4 versus 9. (b) Digits 1 versus 4. (c) Digits 1 versus 9. (d) Digits 1 versus 8.

TABLE II
AVERAGE CLUSTERING ERROR AND CPU TIME ON THE 45 UCI DIGIT CLUSTERING TASKS USING DIFFERENT INITIALIZATION SCHEMES

	clustering error (%)	CPU time (second)
random only	48.21	0.001
random+iterSVR	28.57	12.74
KM only	3.49	0.025
KM+iterSVR	1.92	1.75
NC only	2.43	4.12
NC+iterSVR	0.91	5.57

TABLE III
SUMMARY OF THE DATA SETS

	size	#positive	#negative	#features	balance parameter ℓ
optdigits 3-8 (s)	100	50	50	64	$0.03n$
optdigits 1-7 (s)	100	50	50	64	$0.03n$
optdigits 2-7 (s)	100	50	50	64	$0.03n$
optdigits 8-9 (s)	100	50	50	64	$0.03n$
ionosphere (s)	100	50	50	33	$0.03n$
optdigits 3-8	357	174	183	64	$0.03n$
optdigits 1-7	361	179	182	64	$0.03n$
optdigits 2-7	356	177	179	64	$0.03n$
optdigits 8-9	354	174	180	64	$0.03n$
optdigits 3-9	771	389	382	64	$0.03n$
ionosphere	351	126	225	33	$0.15n$
svmguide1-a (s)	1,000	500	500	4	$0.03n$
ringnorm (s)	1,000	500	500	20	$0.03n$
image	1,010	430	580	18	$0.03n$
letter	1,555	766	789	16	$0.03n$
satellite	2,236	703	1,533	36	$0.15n$
svmguide1-a	3,089	1,089	2,000	4	$0.15n$
ringnorm	7,000	3,457	3,543	20	$0.03n$

unlabeled patterns are labeled. Note that our algorithm does not require any labeled data for initialization and all the unlabeled patterns are assigned labels in the first iteration. Moreover, in self-training, the labels of the unlabeled patterns are typically left unchanged after being assigned. On the other hand, our algorithm keeps updating all these labels. Therefore, it can avoid the situation where labels of some malicious patterns are overconfidently predicted at the early beginning.

The EM algorithm can also be considered as a special case of self-training. Moreover, both the EM algorithm and our approach rely on alternating optimization. However, the main difference is that the expectation–maximization (EM) algorithm is probabilistic and thus requires a good generative model, while ours is not. In particular, each EM iteration computes the expected (soft) cluster label of each pattern, and then updates the model parameters by maximizing the likelihood. On the other hand, our algorithm obtains hard cluster label for each pattern (using SVM testing), and then updates the model parameters by minimizing the structural risk functional (using SVM training). In general, these two approaches can have different clustering results.

IV. EXPERIMENTS

In this section, we demonstrate the superiority of the proposed MMC procedure on a number of real-world data sets. Section IV-A first studies the issue of initialization. Section IV-B then compares the clustering performance of the various MMC methods and standard clustering methods. Finally, experiments on some large data sets are presented in Section IV-C.

A. Initialization

In this section, we study the effect of initialization on iterSVR. The following initialization schemes are included in the experiment:

1) random;
2) standard k-means clustering (KM);
3) normalized cut (NC) [29].

The first two schemes are each repeated three times because of the inherent randomness. NC, on the other hand, is based on eigenvalue decomposition and so does not need multiple repetitions. We use all 45 pairs of the digits 0–9 from the optdigit data set in the UCI machine learning repository [1]. The average performance over these 45 tasks, both immediately after the initialization and at the end of the proposed iterSVR procedure, are reported. The Gaussian kernel is used, and C is set to 500.

As can be seen from Table II, iterSVR with random initialization has poor performance. This can be explained by noting that the random scheme leads to a very poor initialization result (close to 50% error). As in any learning method that relies on local optimization, it cannot recover from a very poor initialization (indeed, though the result obtained by iterSVR is still poor, it has significantly improved the accuracy over the very poor initialization result by about 20%). With better initialization (such as those provided by k-means and normalized cut), the iterSVR procedure is able to obtain a much lower clustering error. However, normalized cut has $O(n^3)$ time and $O(n^2)$ space complexities, and is thus usually slow and memory inefficient. In contrast, the k-means algorithm is easy to implement and the complexity is linear with the sample size n and dimensionality. Moreover, there are also several recent advances on scaling up the k-means algorithm [12], [19], [25]. Therefore, the k-means algorithm is suitable for use in large scale problems. In the sequel, we will always use the k-means algorithm for initialization.

In the following, we further investigate how the k-means initialization may influence the clustering performance. We focus on four digit clustering tasks (1 versus 9, 1 versus 8, 4 versus 9, and 1 versus 4), and initialize the iterSVR procedure with

TABLE IV
CLUSTERING ERRORS (IN PERCENT) ON THE VARIOUS DATA SETS

	KM	KKM	NC	MMC	GMMC	iterSVM	iterLS-SVM	iterSVR
optdigits 3-8 (s)	**0**±0	**0**±0	**0**	2.0	**0**	**0**±0	**0**±0	**0**±0
optdigits 1-7 (s)	8.0±0	5.6±3.1	3.0	**0**	**0**	1.0±0	**0**±0	**0**±0
optdigits 2-7 (s)	9.0±0	9.0±0	6.0	15.0	**2.0**	3.0±0	3.0±0	3.0±0
optdigits 8-9 (s)	11.0±0	11.0± 0	9.0	3.0	7.0	9.0±0	**0**±0	3.0±0
ionosphere (s)	20.0±0	**10.0**±0	23.0	25.0	14.0	23.0±0	19.0±0	25.0±0
optdigits 3-8	5.3 ± 0	5.5±0	6.4	-	4.2	4.5 ± 0	4.2 ± 0	$\mathbf{3.4 \pm 0}$
optdigits 1-7	$\mathbf{0} \pm 0$	$\mathbf{0} \pm 0$	**0**	-	**0**	$\mathbf{0} \pm 0$	$\mathbf{0} \pm 0$	$\mathbf{0} \pm 0$
optdigits 2-7	3.1 ± 0	2.8±0	0.3	-	**0**	3.1 ± 0	0.6 ± 0	$\mathbf{0} \pm 0$
optdigits 8-9	9.3 ± 0	9.0±0	10.0	-	5.9	4.8 ± 0	4.2 ± 0	$\mathbf{3.7} \pm 0$
optdigits 3-9	20.9 ± 14.5	19.3±15.1	21.9	-	**8.3**	18.8 ± 10.9	15.6 ± 12.0	9.2 ± 7.8
ionosphere	28.8±0	28.2±0	25.0	-	**24.2**	28.5±0	24.5±0	28.2±0
svmguide1-a (s)	23.5 ± 0	9.0±0	12.5	-	10.0	6.6 ± 0	$\mathbf{6.0} \pm 0$	6.8 ± 0
ringnorm (s)	24.0±0.5	3.6±0	22.3	-	**1.4**	23.8 ± 0.5	15.5 ± 0.7	3.2 ± 0.01
image	43.5 ± 0	39.2±0.2	41.2	-	41.2	38.7 ± 0	$\mathbf{28.2} \pm 0$	31.6 ± 0
letter	17.9 ± 0	**6.8**±0	23.2	-	7.0	7.4 ± 0	7.4 ± 0	7.2 ± 0
satellite	4.1 ± 0	2.9±0	4.2	-	**2.1**	3.8 ± 0	4.4 ± 0	3.6±0
svmguide1-a	35.8±0	26.8±0	23.9	-	-	29.8±0	**6.6**±0	16.7±0
ringnorm	23.4±0.08	5.2±0.01	6.9±0.4	-	-	23.4±0.09	7.2±0.04	**2.5**±0.02
#times best	1	3	2	1	9	1	7	7

k-means (using random seeds). For each task, the experiment is repeated 20 times. Fig. 7 shows the initial and final clustering errors. As can be seen, different k-means initializations sometimes do not influence the final clustering performance [Fig. 7(a) and (b)]. In some other cases [Fig. 7(c) and (d)], a better k-means initialization leads to better clustering performance, as is typical of algorithms based on local optimization. Nevertheless, as discussed above, the iterSVR procedure can always improve the initial clustering result. Moreover, in practice, one can often avoid a bad k-means solution by a more careful selection of the k-means seeds. For example, in Section IV-B, this is achieved by ensuring that the seeds are sufficiently far away. As will be seen, the clustering performance of iterSVR (averaged over a number of k-means initializations) is very competitive with those of the other clustering algorithms.

B. Clustering Performance

In this section, experiments are performed on a number of data sets from the UCI machine learning repository [1] (optdigits,[4] ionosphere, letter, and satellite), the LIBSVM data[5] (adult and svmguide1 − a) and another benchmark repository[6] (image and ringnorm). For the optdigits data, we follow [35] and focus on those pairs that are difficult to differentiate (namely, 3 versus 8, 1 versus 7, 2 versus 7, 8 versus 9, and 3 versus 9). For letter and satellite, they have multiple classes and we use their first two classes only (A versus B, and C1 versus C2, respectively). The class balance parameter is always set to $\ell = 0.03n$ for the balanced data sets, and $\ell = 0.15n$ for the imbalanced ones. The other settings are the same as in [38].

Note that some of the data sets here are quite large. As mentioned in Section I, the SDP formulation for the original MMC algorithm [38] is very expensive in terms of both time and memory, and can only be run with at most 100 patterns on our PC. Hence, in order to compare with the original MMC algorithm on these data sets, downsampling is performed. For the optdigits and ionosphere data, we create subsets [labeled with the suffix "(s)" in Table III] by sampling 50 samples from each class. Similarly, for svmguide1 − a and ringnorm, we create subsets with 500 samples randomly selected from both the positive and negative classes. A summary of all the data sets is in Table III.

The following clustering methods are compared:

1) standard k-means clustering (KM), with $k = 2$;
2) kernel k-means clustering (KKM), with $k = 2$;
3) normalized cut (NC) [29], using the Gaussian affinity function and with all points included in the neighborhood;
4) MMC [38];
5) GMMC [35];
6) iterSVM (Section III-A);
7) iterSVR (Section III-B);
8) iterLS-SVM (Section III-C).

Recall that the last three methods follow basically the same procedure and only differ on the choice of loss function: iterSVM uses the hinge loss, iterSVR uses the Laplacian loss (which corresponds to $\epsilon = 0$ of the ϵ-insensitive loss), while iterLS-SVM uses the square loss. Preliminary results show that the performance of iterSVR is not sensitive to ϵ when ϵ is very small. Hence, we use $\epsilon = 0.05$ in the sequel, as this is experimentally more efficient than setting $\epsilon = 0$. The Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2/\sigma^2)$ is used for all the kernel methods in the experiment. For each method and each data set, we report the result with its best σ value chosen from a set of candidates. As for the regularization parameter C, the results for the existing MMC methods (MMC and GMMC) are based on the best C value chosen from a set of candidates $(\{1, 10, 100, 500\})$ for each data set. For the proposed iterative procedures, the value of C is usually less crucial and we simply fix it to 10 for iterSVM, 500 for iterSVR, and 100 for iterLS-SVM on all the data sets.

[4]The test set of optdigits is used here.

[5]http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

[6]http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm

TABLE V
CPU TIME (IN SECONDS) TAKEN BY THE VARIOUS ALGORITHMS

	KM	KKM	NC	MMC	GMMC	iterSVM	iterLS-SVM	iterSVR
optdigits 3-8 (s)	0.004±0.007	0.04±0.04	0.02	859.8	2.04	0.02±0.01	0.10±0.01	0.05±0.02
optdigits 1-7 (s)	0.006±0.01	0.09±0.01	0.02	679.4	1.78	0.06±0.01	0.09±0.01	0.08±0.03
optdigits 2-7 (s)	0.005±0.007	0.03±0.01	0.08	1078.0	1.89	0.08±0.02	0.12±0.02	0.12±0.01
optdigits 8-9 (s)	0.004±0.007	0.03±0.01	0.09	957.0	1.69	0.03±0.01	0.11±0.02	0.08±0.01
ionosphere (s)	0.006±0.008	0.02±0.01	0.11	764.0	1.97	0.02±0.01	0.15±0.02	0.06±0.02
optdigits 3-8	0.02±0.01	0.16±0.01	0.31	-	12.9	0.31±0.02	1.30±0.25	0.86±0.02
optdigits 1-7	0.02±0.01	0.17±0.03	0.33	-	21.3	0.18±0.04	1.78±0.21	0.62±0.03
optdigits 2-7	0.01±0.07	0.15±0.01	0.31	-	19.4	0.12±0.01	1.27±0.24	1.37±0.07
optdigits 8-9	0.01±0.01	0.18±0.01	0.29	-	20.9	0.72±0.02	2.33±0.35	0.79±0.02
optdigits 3-9	0.4±0.01	1.39±0.03	3.70	-	239.5	2.37±0.44	3.43±0.54	4.27±0.43
ionosphere	0.08±0.008	0.12±0.01	0.28	-	19.8	0.07±0.01	5.79±0.72	0.45±0.05
svmguide1-a (s)	0.04±0.007	0.7±0.03	7.54	-	588.6	0.29±0.02	2.35±0.41	2.49±0.18
ringnorm (s)	0.07±0.03	0.54±0.02	6.58	-	506.0	1.51±0.47	10.85±5.78	7.75±3.23
image	0.02±0.01	0.63±0.02	6.39	-	1054.0	0.48±0.16	12.31±3.69	2.54±0.23
letter	0.02±0.008	1.39±0.03	25.21	-	1747.0	3.66±0.67	251.2±17.2	14.51±1.24
satellite	0.05±0.01	3.97±0.18	93.82	-	5590.0	2.65±0.12	361.4±20.5	11.83±0.87
svmguide1-a	0.13±0.02	4.83±0.52	221.0	-	-	3.3±0.04	18.63±9.57	36.25±2.18
ringnorm	0.57±0.2	40.53±0.53	38.2±43.76	-	-	56.9±7.5	1120±9.8	148.4±25.8

Implementations of the iterSVM and iterSVR are modified from the LIBSVM package[7] (version 2.85) [7], while that of iterLS-SVM is from the LS-SVMlab package[8] (version 1.5) [32]. The implementation of the MMC is provided by the authors of [38]. Both the MMC and the GMMC use the state-of-the-art SDP solver of SDPT3[9] (version 4.0) [33]. Moreover, kernel evaluations are implemented in C++. All experiments are performed on a 2.13-GHz Intel Core 2 Duo PC with 3-GB memory running Windows XP.

Recall that the initial class labels for the iterative SVM/SVR/LS-SVM are obtained from the k-means clustering procedure (with $k = 2$). Hence, they are susceptible to the local minimum problem. On the other hand, NC, which is based on eigendecomposition, and MMC/GMMC, which are based on convex SDPs, are not plagued by local minima. In our implementation, this problem is alleviated by requiring the initial prototypes of k-means clustering to be sufficiently far away. These local optimization methods are then repeated ten times with different initial seeds for the k-means clustering on each data set, and then the averaged results are reported.

1) Results: Clustering errors on the various data sets are shown in Table IV. Data sets that cannot be run on our PC because of insufficient memory are marked with "–." Indeed, the standard implementation of NC also cannot be run on the ringnorm data and, here, its memory requirement is reduced by using the Nyström approximation[10] [15]. The last row summarizes the number of times each algorithm performs best (i.e., has the smallest clustering error). As can be seen, both iterSVR and iterLS-SVM are competitive with (and on some tasks even better than) GMMC and outperform the other methods.

As mentioned in Section I, another key problem with MMC and GMMC is that the solving of the corresponding SDPs is very slow. Table V compares the central processing unit (CPU) time consumption of the different algorithms. As can be seen, iterSVR is more efficient than GMMC (about 500 times faster on satellite, the largest data set that can be run by GMMC on our machine) and MMC (about 10 000 times faster on the data sets that MMC can handle). Thus, while iterSVR is competitive with GMMC in terms of clustering error, iterSVR is much more advantageous than GMMC in terms of training time. In general, iterLS-SVM is slower than iterSVR, as the LS-SVM solution is not sparse and each LS-SVM iteration has to solve a linear system involving all the patterns. On the other hand, it is well known that the SVR solution is sparse and thus each SVR iteration of iterSVR only involves optimizing the support vectors. Moreover, as can be seen from Table VI, the iterative methods usually converge very fast (in fewer than ten iterations). Note that although iterSVM converges faster than iterSVR, its clustering performance is much inferior (Table IV).

TABLE VI
NUMBER OF ITERATIONS REQUIRED FOR CONVERGENCE OF THE ITERATIVE PROCEDURES

	iterSVM	iterLS-SVM	iterSVR
optdigits 3-8 (s)	2 ± 0	2 ± 0	3 ± 0
optdigits 1-7 (s)	5 ± 0	5 ± 0	5 ± 0
optdigits 2-7 (s)	7 ± 0	3 ± 0	5 ± 0
optdigits 8-9 (s)	3 ± 0	3 ± 0	7 ± 0
ionosphere (s)	3 ± 0	3 ± 0	5 ± 0
optdigits 3-8	3 ± 0	2 ± 0	3 ± 0
optdigits 1-7	2 ± 0	2 ± 0	3 ± 0
optdigits 2-7	3 ± 0	3 ± 0	5 ± 0
optdigits 8-9	6 ± 0	4 ± 0	5 ± 0
optdigits 3-9	3.9 ± 0.8	5.15 ± 0.7	6.4 ± 1.8
ionosphere	2 ± 0	3 ± 0	4 ± 0
svmguide1-a (s)	5 ± 0	5 ± 0	5 ± 0
ringnorm (s)	5.7±1.1	6.2±2.2	13.5±5.7
image	5 ± 0	9 ± 0	4 ± 0
letter	6 ± 0	4 ± 0	15 ± 0
satellite	11 ± 0	13 ± 0	5 ± 0
svmguide1-a	4 ± 0	4 ± 0	10 ± 0
ringnorm	4.8 ± 0.5	5.4±0.5	7.4±0.8

C. Experiments on Large Data Sets

To demonstrate the scaling properties of the different algorithms, we use the adult data set from the sequential minimization optimization (SMO) webpage.[11] This version of

[7]http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[8]http://www.esat.kuleuven.ac.be/sista/lssvmlab/

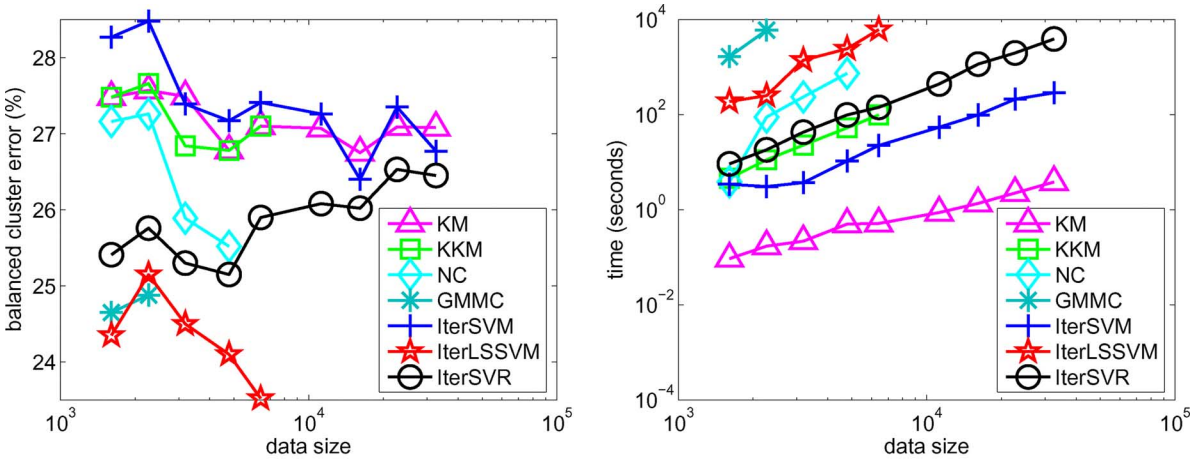
[9]http://www.math.nus.edu.sg/~mattohkc/sdpt3.html

[10]The Nyström approximation uses 500 random samples and the reported result is averaged over ten repetitions.

[11]http://research.microsoft.com/users/jplatt/smo.html

TABLE VII
NUMBER OF PATTERNS IN THE adult SUBSETS

adult-1a	adult-2a	adult-3a	adult-4a	adult-5a	adult-6a	adult-7a	adult-8a	adult-a
1,605	2,265	3,185	4,781	6,414	11,220	16,100	22,696	32,561



(a) (b)

Fig. 8. Comparison of the clustering algorithms on the adult subsets. (a) Balanced cluster error. (b) Time consumption.

TABLE VIII
EMPIRICAL TIME COMPLEXITY OF THE VARIOUS ALGORITHMS ON THE adult DATA SET

KM	KKM	NC	GMMC	iterSVM	iterLS-SVM	iterSVR
$O(n^{1.16})$	$O(n^{2.19})$	$O(n^{2.83})$	$O(n^{3.72})$	$O(n^{1.68})$	$O(n^{2.62})$	$O(n^{2.01})$

the adult data has predefined subsets, ranging in size from the smallest of 1605 to the largest of 32 561 (Table VII). As in Section IV-B, we compare a variety of algorithms including: 1) k-means clustering, 2) kernel k-means clustering, 3) normalized cut, 4) GMMC, 5) iterSVM, 6) iterLS-SVM, and 7) iterSVR. On the other hand, MMC cannot be run even on the smallest subset because of memory problem.

As the data is very skewed,[12] the clustering error is inappropriate for performance evaluation. Instead, the balanced clustering error, which is defined as the cluster error averaged over the two classes, is reported here. Experimental results are shown in Fig. 8. In terms of the (balanced) clustering error, both iterLS-SVM and GMMC are very good; iterSVR is slightly inferior but still outperforms all the other methods. In terms of speed, GMMC is the slowest. Note that many algorithms (such as kernel k-means clustering, normalized cut, GMMC, and iterLS-SVM) cannot be run with the larger subsets because of insufficient memory. The empirical time complexities of the various methods are shown in Table VIII. Overall, iterSVR yields good clustering performance and is fast.

Finally, we demonstrate the ability of the iterSVR procedure on large data sets by considering image segmentation as a clustering problem. Experiments are performed on four images[13] with the RGB values (in the range $[0, 255]$) as features. The sizes of the images are reported in Table IX. As can be seen, there are a large number of pixels to be clustered and methods including MMC, GMMC, and iterLS-SVM are unable to handle these large data sets. For comparison, we run k-means clustering and iterSVM. As in Section IV-B, the Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2/\sigma^2)$ is used. A fixed $\sigma = 500$ and $C = 500$ is used for iterSVR on all the images, while the other algorithms have the extra freedom in choosing their best parameters. As can be seen from Fig. 9, the segmentation results of iterSVR are visually more satisfactory than those of the other algorithms.

In the same manner as [29], the iterSVR algorithm can be used for multiclass problems by performing binary clustering recursively. For example, as shown in Fig. 10, the woman image is first segmented to background and foreground (from left to right), then the skin and clothes, and so on.

V. CONCLUSION

In this paper, we propose an efficient approach for solving MMC via alternating optimization. The key step is on the use of SVR with the Laplacian loss, or LS-SVM with the square loss, in the inner optimization subproblem. In contrast to the traditional hinge loss used in the SVM, these symmetric loss functions discourage premature convergence by penalizing overconfident predictions. While existing MMC algorithms are formulated as very expensive SDPs, our approach is formulated as a sequence of efficient kernel regression (involving either QPs or linear system). Empirically, the clustering accuracies of the proposed iterative procedures are competitive or even better than existing MMC and traditional clustering algorithms. Moreover, it is much faster (by tens of thousands of times compared with MMC and hundreds of times compared with GMMC) and can handle much larger data sets (hundreds of times larger than the largest data set reported in the MMC literature).

In the future, we will investigate how to extend our clustering method to the semisupervised learning setting. Another problem

[12]The numbers of positive and negative samples in the full set (adult − a) are 7841 and 24 720, respectively.

[13]The horse and roo images are from the benchmark Berkeley image segmentation database whose url is http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/; zebra and squirrel are commonly used in vision literature.

TABLE IX
SIZES OF THE IMAGES USED IN FIGS. 9 AND 10

	horse	zebra	squirrel	roo	woman	flower
size	240×160	214×160	288×209	288×209	261×116	214×160
number of pixels	38,400	34,240	60,192	60,192	30,276	34,240

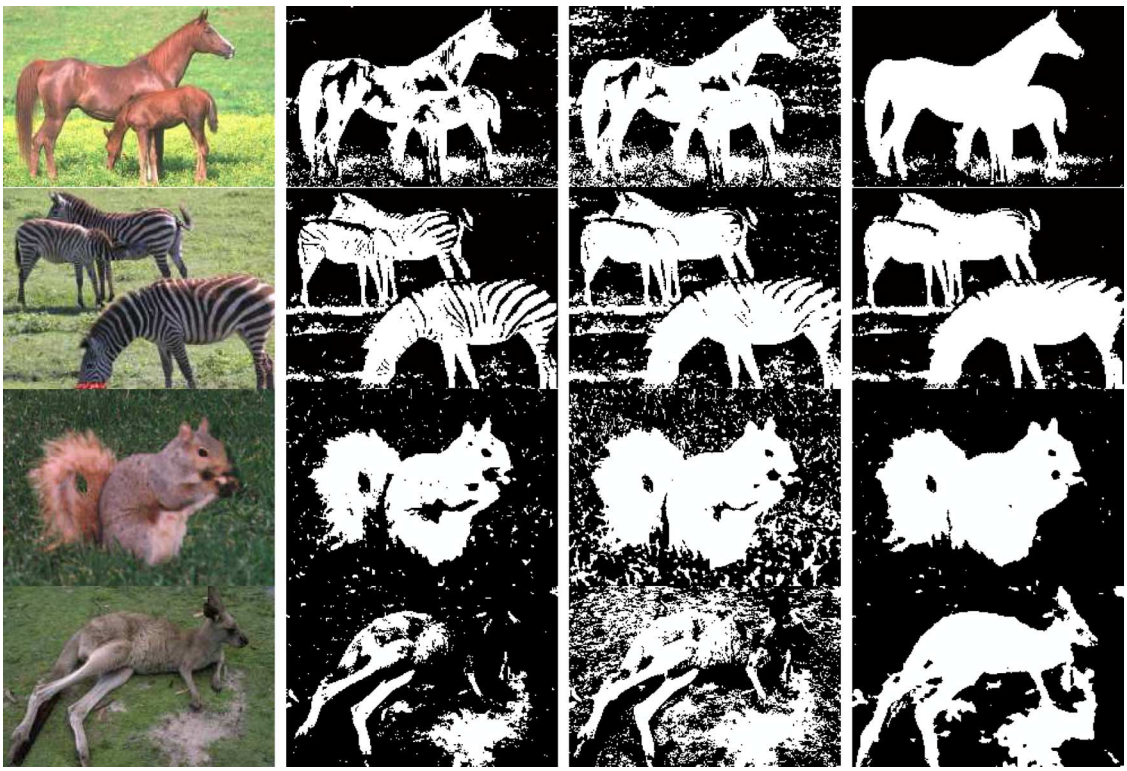


Fig. 9. Segmentation results obtained by k-means clustering (second column), iterSVM (third column), and iterSVR (fourth column).

worth investigation is how to scale up the iterative LS-SVM procedure that has performed quite well in our experiments. Currently, we can only use the LS-SVMlab package on data sets with a maximum of about 7000 samples. We plan to incorporate techniques such as low-rank approximation [13] and large scale decomposition methods [8] so that it can be used on much larger real-world data sets. Moreover, we will study the extension to the multiclass scenario, where the balance constraints need to be enforced among the multiple decision functions.

APPENDIX
CONCAVE-CONVEX PROCEDURE FOR MAXIMUM MARGIN CLUSTERING

The *concave-convex procedure* [42] is an optimization tool for problems whose objective function can be expressed as a difference of convex functions. In the optimization literature, it is often known as the *difference of convex functions* algorithm (DCA) [26]. While the concave-convex procedure only considered linear constraints [42], Smola *et al.* generalized this to the CCCP for problems with a concave-convex objective function with concave-convex constraints [30].

Consider the following optimization problem:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f_0(\mathbf{x}) - g_0(\mathbf{x}) \\ \text{s.t.} \quad & f_i(\mathbf{x}) - g_i(\mathbf{x}) \le c_i, \qquad i = 1, \ldots, n \end{aligned}$$

where f_i, g_i $(i = 0, \ldots, n)$ are real-valued, convex, and differentiable functions on $\mathbb{R}^d$, and $c_i \in \mathbb{R}$. Given an initial $\mathbf{x}_0$, CCCP computes $\mathbf{x}_{t+1}$ from $\mathbf{x}_t$ by replacing $g_i(\mathbf{x})$ with its first-order Taylor expansion at $\mathbf{x}_t$, and then setting $\mathbf{x}_{t+1}$ to the solution of the relaxed optimization problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & f_0(\mathbf{x}) - \left[g_0(\mathbf{x}_t) + \nabla g_0(\mathbf{x}_t)^\top (\mathbf{x} - \mathbf{x}_t)\right] \\ \text{s.t.} \quad & f_i(\mathbf{x}) - \left[g_i(\mathbf{x}_t) + \nabla g_i(\mathbf{x}_t)^\top (\mathbf{x} - \mathbf{x}_t)\right] \le c_i, \\ & i = 1, \ldots, n. \end{aligned} \tag{18}$$

Here $\nabla g(\hat{\mathbf{x}})$ is the gradient of the function g at $\hat{\mathbf{x}}$. Each CCCP iteration then becomes a convex optimization problem. For non-smooth functions g_i, it can be easily shown that the gradient in (18) should then be replaced by the subgradient (a subgradient of f at $\mathbf{x}$ is any vector γ that satisfies the inequality $f(\mathbf{y}) \ge f(\mathbf{x}) + \gamma^t op(\mathbf{y} - \mathbf{x})$ for all $\mathbf{y}$ [4]). It can be shown that the objective (18) obtained from each iteration decreases monotonically and converges to a local minimum [30]. In practice, CCCP can often converge to a global solution [26].

On dropping the balance constraint, the MMC problem in (2) can be written as

$$\begin{aligned} \min_{\mathbf{w}, b, \boldsymbol{\xi}} \quad & \|\mathbf{w}\|^2 + 2C\boldsymbol{\xi}^\top \mathbf{e} \\ \text{s.t.} \quad & \left|\mathbf{w}^\top \varphi(\mathbf{x}_i) + b\right| \ge 1 - \xi_i \\ & \xi_i \ge 0, \qquad i = 1, \ldots, n. \end{aligned} \tag{19}$$

Fig. 10. Recursive segmentation results by iterSVR.

Let $\mathbf{w}_t, b_t$ be the solution at the tth iteration. Then

$$\begin{aligned}\partial_{\mathbf{w}} \left|\mathbf{w}^\top \varphi(\mathbf{x}_i) + b\right|_{\substack{\mathbf{w}=\mathbf{w}_t \\ b=b_t}} &= \operatorname{sign}\left(\mathbf{w}_t^\top \varphi(\mathbf{x}_i) + b_t\right) \varphi(\mathbf{x}_i) \\ \partial_b \left|\mathbf{w}^\top \varphi(\mathbf{x}_i) + b\right|_{\substack{\mathbf{w}=\mathbf{w}_t \\ b=b_t}} &= \operatorname{sign}\left(\mathbf{w}_t^\top \varphi(\mathbf{x}_i) + b_t\right).\end{aligned}$$

Following CCCP, the $\left|\mathbf{w}^\top \varphi(\mathbf{x}_i) + b\right|$ term in (19) is replaced by its first-order Taylor expansion

$$\begin{aligned}&\left|\mathbf{w}_t^\top \varphi(\mathbf{x}_i) + b_t\right| + \operatorname{sign}\left(\mathbf{w}_t^\top \varphi(\mathbf{x}_i) + b_t\right) \\ &\quad \cdot \left((\mathbf{w} - \mathbf{w}_t)^\top \varphi(\mathbf{x}_i) + (b - b_t)\right) \\ &= \operatorname{sign}\left(\mathbf{w}_t^\top \varphi(\mathbf{x}_i) + b_t\right)\left(\mathbf{w}^\top \varphi(\mathbf{x}_i) + b\right).\end{aligned}$$

The optimization problem in (19) then becomes

$$\begin{aligned}\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \|\mathbf{w}\|^2 + 2C\boldsymbol{\xi}^\top \mathbf{e} \\ \text{s.t.} \quad & \hat{y}_t\left(\mathbf{w}^\top \varphi(\mathbf{x}_i) + b\right) \geq 1 - \xi_i \\ & \xi_i \geq 0, \qquad i = 1, \ldots, n\end{aligned}$$

where $\hat{y}_t = \operatorname{sign}(\mathbf{w}_t^\top \varphi(\mathbf{x}_i) + b_t)$. This is a standard SVM problem with the labels $\hat{y}_t$'s obtained by the SVM in the previous iteration, and one can observe the close similarity between this and Algorithm 1. However, such CCCP solution ignores the balance constraint, which could lead to trivial solutions as mentioned in Section II. It is an open issue on how to apply CCCP while keeping the balance constraint.

REFERENCES

[1] A. Asuncion and D. J. Newman, UCI Machine Learning Repository Irvine, CA, 2007 [Online]. Available: http://archive.ics.uci.edu/ml/

[2] A. Azran and Z. Ghahramani, "A new approach to data driven clustering," in *Proc. 23rd Int. Conf. Mach. Learn.*, Pittsburgh, PA, Jun. 2006, pp. 57–64.

[3] J. C. Bezdek and R. J. Hathaway, "Convergence of alternating optimization," *Neural Parallel Sci. Comput.*, vol. 11, no. 4, pp. 351–368, Dec. 2003.

[4] J. F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal, *Numerical Optimization*. Berlin, Germany: Springer-Verlag, 2003.

[5] L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, *Large-Scale Kernel Machines*. Cambridge, MA: MIT Press, 2007.

[6] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[7] C.-C. Chang and C.-J. Lin, LIBSVM: A Library for Support Vector Machines, 2007 [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[8] W. Chu, C. J. Ong, and S. S. Keerthi, "An improved conjugate gradient scheme to the solution of least squares SVM," *IEEE Trans. Neural Netw.*, vol. 16, no. 2, pp. 498–501, Mar. 2005.

[9] R. Collobert, F. Sinz, J. Weston, and L. Bottou, "Large scale transductive SVMs," *J. Mach. Learn. Res.*, vol. 7, pp. 1687–1712, Aug. 2006.

[10] R. Collobert, J. Weston, and L. Bottou, "Trading convexity for scalability," in *Proc. 23rd Int. Conf. Mach. Learn.*, Pittsburgh, PA, Jun. 2006, pp. 201–208.

[11] C. Elkan, "Clustering documents with an exponential-family approximation of the Dirichlet compound multinomial distribution," in *Proc. 23rd Int. Conf. Mach. Learn.*, Pittsburgh, PA, Jun. 2006, pp. 289–296.

[12] E. Elkan, "Using the triangular inequality to accelerate k-means," in *Proc. 21st Int. Conf. Mach. Learn.*, 2003, pp. 147–153.

[13] S. Fine and K. Scheinberg, "Efficient SVM training using low-rank kernel representations," *J. Mach. Learn. Res.*, vol. 2, pp. 243–264, Dec. 2001.

[14] , D. H. Fisher, Ed., *Unsupervised Learning*. Boston, MA: Kluwer, 2002.

[15] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the Nyström method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 2, pp. 214–225, Feb. 2004.

[16] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Boston, MA: Kluwer, 1992.

[17] C. Helmberg, F. Rendl, B. Vanderbei, and H. Wolkowicz, "An interior-point method for semidefinite programming," *SIAM J. Optim.*, vol. 6, no. 2, pp. 342–361, 1996.

[18] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

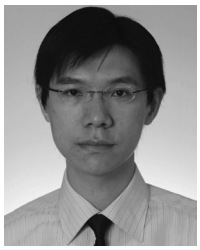
[19] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, Jul. 2001.

[20] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *J. Mach. Learn. Res.*, vol. 5, pp. 27–72, 2004.

[21] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret, "Applications of second-order cone programming," *Linear Algebra Appl.*, vol. 284, pp. 193–228, 1998.
[22] G. McLachlan and D. Peel, *Finite Mixture Models*. New York: Wiley, 2000.
[23] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*. Philadelphia, PA: SIAM, 1994.
[24] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. Cambridge, MA: MIT Press, 2002, vol. 14.
[25] D. Pelleg and A. Moore, "Accelerating exact k-means algorithms with geometric reasoning," *Knowl. Disc. Data Mining*, pp. 277–281, 1999.
[26] T. Pham Dinh and L. T. Hoai An, "A D.C. optimization algorithm for solving the trust-region subproblem," *SIAM J. Optim.*, vol. 8, no. 2, pp. 476–505, 1998.
[27] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods—Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 185–208.
[28] B. Schölkopf and A. J. Smola, *Learning With Kernels*. Cambridge, MA: MIT Press, 2002.
[29] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.
[30] A. J. Smola, S. V. N. Vishwanathan, and T. Hofmann, "Kernel methods for missing variables," in *Proc. 10th Int. Workshop Artif. Intell. Statist.*, Barbados, Jan. 2005, pp. 325–332.
[31] J. F. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optim. Methods Software*, vol. 11–12, Special Issue on Interior Point Methods, pp. 625–653, 1999.
[32] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, pp. 293–300, 1999.
[33] K. C. Toh, M. J. Todd, and R. H. Tutuncu, "SDPT3—A Matlab software package for semidefinite programming," *Optim. Methods Software*, vol. 11, pp. 545–581, 1999.
[34] K. Tsuda and T. Kudo, "Clustering graphs by weighted substructure mining," in *Proc. 23rd Int. Conf. Mach. Learn.*, Pittsburgh, PA, Jun. 2006, pp. 953–960.
[35] H. Valizadegan and R. Jin, "Generalized maximum margin clustering and unsupervised kernel learning," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2007, vol. 19.
[36] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.
[37] , W. Wu, H. Xiong, and S. Shekhar, Eds., *Clustering and Information Retrieval*. Norwell, MA: Kluwer, 2004.
[38] L. Xu, J. Neufeld, B. Larson, and D. Schuurmans, "Maximum margin clustering," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2005, vol. 17.
[39] L. Xu and D. Schuurmans, "Unsupervised and semi-supervised multi-class support vector machines," in *Proc. 20th Nat. Conf. Artif. Intell.*, Pittsburgh, PA, 2005, pp. 904–910.
[40] L. Xu, D. Wilkinson, F. Southey, and D. Schuurmans, "Discriminative unsupervised learning of structured predictors," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 1057–1064.
[41] D. Yarowsky, "Unsupervised word-sense disambiguation rivaling supervised methods," in *Proc. 33rd Annu. Meeting Assoc. Comput. Linguistics*, 1995, pp. 189–196.
[42] A. L. Yuille and A. Rangarajan, "The concave-convex procedure," *Neural Comput.*, vol. 15, no. 4, pp. 915–936, Apr. 2003.
[43] K. Zhang, I. W. Tsang, and J. T. Kwok, "Maximum margin clustering made practical," in *Proc. 24th Int. Conf. Mach. Learn.*, Corvallis, OR, Jun. 2007, pp. 1119–1126.
[44] X. Zhu, "Semi-supervised learning literature survey," Dept. Comput. Sci, Univ. Wisconsin-Madison, Madison, WI, Tech. Rep. 1530, 2005.

Kai Zhang received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology, Hong Kong, in 2008.

Then, he joined the Life Science Division, Lawrence Berkeley National Laboratory, Berkeley, CA. His research interests are machine learning and pattern recognition, in particular, large scale unsupervised learning and dimension reduction algorithms. He also works on applications in data mining, bioinformatics, and complex networks.



Ivor W. Tsang received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology (HKUST), Kowloon, Hong Kong, in 2007.

Currently, he is an Assistant Professor at the School of Computer Engineering, Nanyang Technological University, Nanyang, Singapore. His scientific interests include machine learning, kernel methods and large scale optimization, and their applications to data mining and pattern recognitions.

Dr. Tsang was awarded the prestigious IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding 2004 Paper Award in 2006. He was also awarded the Microsoft Fellowship in 2005, the Best Paper Award from the IEEE Hong Kong Chapter of Signal Processing Postgraduate Forum in 2006, and also the HKUST Honor Outstanding Student in 2001.

James T. Kwok received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology (HKUST), Kowloon, Hong Kong, in 1996.

He then joined the Department of Computer Science, Hong Kong Baptist University, Hong Kong, as an Assistant Professor. He returned to HKUST in 2000 and is now an Associate Professor at the Department of Computer Science and Engineering. His research interests include kernel methods, machine learning, pattern recognition, and artificial neural networks.

Dr. Kwok is an Associate Editors for the IEEE TRANSACTIONS ON NEURAL NETWORKS and *Neurocomputing*. He received the IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding 2004 Paper Award in 2006.