Maximum Margin Clustering Made Practical

Kai Zhang TWINSEN@CSE.UST.HK
Ivor W. Tsang IVOR@CSE.UST.HK
James T. Kwok JAMESK@CSE.UST.HK
Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong

Abstract

Maximum margin clustering (MMC) is a recent large margin unsupervised learning approach that has often outperformed conventional clustering methods. Computationally, it involves non-convex optimization and has to be relaxed to different semidefinite programs (SDP). However, SDP solvers are computationally very expensive and only small data sets can be handled by MMC so far. 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 non-convex problem. A key step to avoid premature convergence is on the use of SVR with the Laplacian loss, instead of SVM with the hinge loss, in the inner optimization subproblem. Experiments on a number of synthetic and real-world data sets demonstrate that the proposed approach is often more accurate, much faster and can handle much larger data sets.

1. Introduction

Clustering has long been an active research area in machine learning (Jain & Dubes, 1988). Given a set of observations, clustering aims at identifying dominant structures in the data and grouping similar instances together. This is extremely valuable for data analysis in practice, and has been widely used in diverse domains such as information retrieval, computer vision, and bioinformatics.

Over the decades, a battery of clustering approaches have been developed, such as the k-means clustering algorithm, mixture models, and spectral clustering. Recently, motivated by the success of large margin methods in supervised learning (Schölkopf & Smola, 2002), there is growing interest in extending large margin methods to unsupervised learning. Xu et al. (2005) proposed a novel approach called maximum margin clustering (MMC), which performs clustering by simultaneously finding maximum margin hyperplanes in the data. Experimental results showed that this can often outperform conventional clustering methods. Moreover, it can also be extended to a general framework for both unsupervised and semi-supervised learning (Xu & Schuurmans, 2005).

While large margin supervised learning methods are usually formulated as convex optimization problems (notably, quadratic programs for support vector machines), large margin unsupervised learning is much more difficult and leads to non-convex integer programs. Existing maximum margin clustering methods (Xu et al., 2005; Xu & Schuurmans, 2005; Valizadegan & Jin, 2007) all rely on reformulating and relaxing the non-convex optimization problem as semidefinite programs (SDP) (Boyd & Vandenberghe, 2004), which can then be solved by standard SDP solvers such as SDPT3 and SeDuMi. In particular, the generalized maximum margin clustering (GMMC) method (Valizadegan & Jin, 2007) reduces the number of parameters in the SDP formulation from n^2 in (Xu et al., 2005) 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, solving SDPs is still computationally very expensive. Thus, maximum margin clustering 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). On the other hand, there can be an enormous amount of data available in many real-world learning problems. For example, in image segmentation, a small 200×200 image already has 40,000 pixels (examples) to be clustered. In web and data mining, even medium-sized

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007.

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 maximum margin clustering by avoiding the use of SDP relaxations. Instead, we revisit a natural approach that was considered ineffective (Xu et al., 2005), namely, by performing alternating optimization (Bezdek & Hathaway, 2003) directly on the original non-convex problem. While a straightforward implementation easily gets stuck in poor locally optimal solutions, our key modification is to replace SVM by support vector regression with the Laplacian loss. As will be seen, this discourages premature convergence and allows greater flexibility in exploring the search space. Computationally, the proposed procedure involves only a sequence of quadratic programs for SVR training. The resultant implementation is fast and scales well compared to existing approaches.

The rest of the paper is organized as follows. Section 2 gives a brief review on maximum margin clustering. Section 3 then describes the proposed method. Experimental results are presented in Section 4, and the last section gives some concluding remarks.

2. Maximum Margin Clustering

Large margin methods, notably the support vector machines (SVM), has been highly successful in supervised learning. Given a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i$ is the input and $y_i \in \{\pm 1\}$ is the output, the SVM finds a large margin hyperplane $f(\mathbf{x}) = \mathbf{w}^\top \varphi(\mathbf{x}) + b$ (where φ is the mapping induced by a kernel k) by solving the following optimization problem:

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

Here, ξ_i's are slack variables for the errors, $C > 0$ is a regularization parameter and $\mathbf{e}$ is the vector of ones.

Motivated by its success in supervised learning, maximum margin clustering (Xu et al., 2005) aims at extending large margin methods to unsupervised learning. However, as the class labels $\mathbf{y} = [y_1, \ldots, y_n]^\top$ are unknown, a trivially "optimal" solution is to assign all patterns to the same class, and the resultant margin will be infinite. To prevent such a meaningless solution, Xu et al. (2005) 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 constant controlling the class imbalance. Then, the margin can be maximized by optimizing both the unknown $\mathbf{y}$ and the unknown SVM parameter $(\mathbf{w}, b)$ together, as:

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

Recall that the dual of the SVM is

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

where $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_n]^\top$, $\mathbf{0}$ is the zero vector, $\mathbf{K}$ is the kernel matrix, and $\circ$ denotes the elementwise product between matrices. Hence, (2) can also be written as

$$\begin{array}{ll} \min_{\mathbf{y}} \max_{\boldsymbol{\lambda}} & 2\boldsymbol{\lambda}^\top \mathbf{e} - \boldsymbol{\lambda}^\top (\mathbf{K} \circ \mathbf{y}\mathbf{y}^\top)\boldsymbol{\lambda} \\ \text{s.t.} & \boldsymbol{\lambda}'\mathbf{y} = 0, \;\; C\mathbf{e} \geq \boldsymbol{\lambda} \geq \mathbf{0}, \\ & y_i = \{\pm 1\}, \; -\ell \leq \mathbf{e}^\top \mathbf{y} \leq \ell. \end{array} \tag{4}$$

As (4) is non-convex and thus difficult to solve, Xu et al. (2005) relaxed it as a convex integer problem, and then further turned this into an equivalent semidefinite program (SDP) as in (Lanckriet et al., 2004):

$$\begin{array}{ll} \min_{\mathbf{M},\delta,\boldsymbol{\mu},\boldsymbol{\nu}} & \delta \\ \text{s.t.} & \begin{bmatrix} \mathbf{M} \circ \mathbf{K} & \mathbf{e} + \boldsymbol{\mu} - \boldsymbol{\nu} \\ (\mathbf{e} + \boldsymbol{\mu} - \boldsymbol{\nu})^\top & \delta - 2C\boldsymbol{\nu}'\mathbf{e} \end{bmatrix} \succeq 0, \\ & \mathrm{diag}(\mathbf{M}) = \mathbf{e}, \mathbf{M} \succeq 0, -\ell\mathbf{e} \leq \mathbf{M}\mathbf{e} \leq \ell\mathbf{e}. \end{array} \tag{5}$$

Here, $\mathbf{M} \in \mathbb{R}^{n \times n}$, $\boldsymbol{\mu}, \boldsymbol{\nu} \in \mathbb{R}^n$ and the symbol $\succeq 0$ denotes positive semidefinite. The class labels can then be recovered from $\mathbf{M}$ by eigen-decomposition. While initially developed for this two-class case, MMC has also been extended to the multi-class formulation (Xu & Schuurmans, 2005), and again leads to a SDP.

Note that as $\mathbf{M} \in \mathbb{R}^{n \times n}$, the number of parameters in (5) scales quadratically with n, the number of examples. Another deficiency of (Xu et al., 2005) is that the bias b is assumed to be zero. Very recently, these problems are alleviated by the generalized maximum margin clustering (GMMC) (Valizadegan & Jin, 2007). The bias term can now be included in the classification boundary. Moreover, the number of parameters in the SDP is reduced from n^2 to n, which leads to significant computational savings.

3. Proposed Procedure

In this section, we first revisit the use of alternating optimization to solve the un-relaxed non-convex MMC problem (Section 3.1). Computationally, this allows the problem to be solved as quadratic programs which

are much more efficient. However, empirically, it suffers from premature convergence and easily gets stuck in poor local optima. Our key proposal, which is to replace SVM by SVR with the Laplacian loss, is then introduced in Section 3.2. An efficient procedure to enforce the class balance constraint in the proposed method is discussed in Section 3.3, and finally some complexity analysis is in Section 3.4.

3.1. Approach 1: Iterative SVM

A natural way to solve (2) is by using a simple iterative approach based on alternating optimization (Bezdek & Hathaway, 2003), similar to the one proposed in (Xu et al., 2006). First, fix $\mathbf{y}$ and maximize (3) w.r.t. $\boldsymbol{\lambda}$, which is just standard SVM training. Then, fix $\boldsymbol{\lambda}$ and minimize (2) w.r.t. $\mathbf{y}$. Note that with a fixed $\boldsymbol{\lambda}$, both $\mathbf{w}$ and b can be determined from the KKT conditions, so the second step reduces to

$$\begin{aligned} \min_{\mathbf{y},\xi_i} \quad & \boldsymbol{\xi}^\top \mathbf{e} \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \varphi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \;\; \xi_i \geq 0. \end{aligned}$$

Obviously, this yields $y_i = \text{sign}(\mathbf{w}^\top \varphi(\mathbf{x}_i) + b)$ and $\xi_i = 1 - y_i(\mathbf{w}^\top \varphi(\mathbf{x}_i) + b)$. The two steps are then repeated until convergence (Algorithm 1).

Algorithm 1 Iterative SVM procedure.

1: Initialize the labels $\mathbf{y}$ by simple clustering method.
2: Fix $\mathbf{y}$ and perform standard SVM training.
3: Compute $\mathbf{w}$ and b from the KKT conditions.
4: Assign the labels as $y_i = \text{sign}(\mathbf{w}^\top \varphi(\mathbf{x}_i) + b)$.
5: Repeat steps 2-4 until convergence.

3.1.1. Poor Empirical Performance

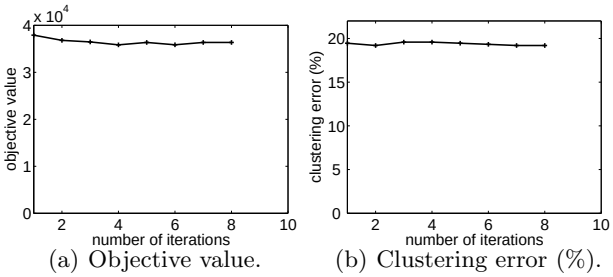
In practice, the performance of the iterative SVM is not satisfactory, since its improvement over the initial labeling is usually insignificant. An example is shown in Figure 1. Here, the task is to separate digits 3 and 9 (from the UCI optdigits data set). Each digit has about 390 samples. We initialize with the normalized cut algorithm (Shi & Malik, 2000), which yields an error rate of 19.46%. Figure 1 shows that both the objective value and the clustering quality change little during the iterations.

3.1.2. Premature Convergence

To understand this poor performance, we consider the hinge loss (Figure 2(a)) used by the SVM:

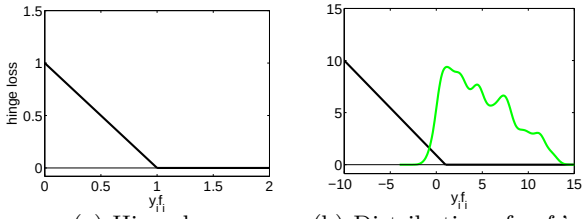
$$L_h = \begin{cases} 0 & \text{if } \; y_i f_i \geq 1, \\ 1 - y_i f_i & \text{otherwise,} \end{cases}$$

where $f_i = f(\mathbf{x}_i)$. Because of this loss, 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 is evidenced from the empirical distribution of $y_i f_i$'s on the task shown earlier (Figure 2(b)). If we want to flip the label of a sample, the loss changes from the blue line to the red line in Figure 3. This will be very large as most of them are far away from the elbow (e.g., point "b"). Therefore, the SVM is unwilling to flip the class labels. In other words, the procedure over-commits to the initial label estimates, and thus easily gets stuck in local optima.



(a) Objective value. (b) Clustering error (%).

Figure 1. Poor performance of the iterative SVM procedure in Algorithm 1.



(a) Hinge loss. (b) Distribution of $y_i f_i$'s.

Figure 2. Loss function and empirical distribution of $y_i f_i$'s on the digits task for SVM.

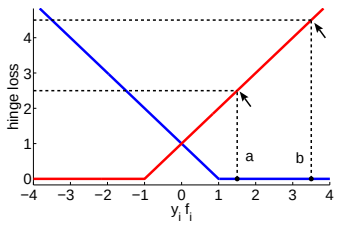


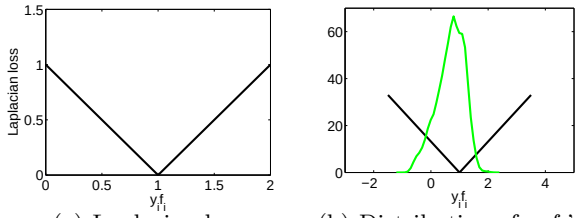
Figure 3. Flipping the labels.

3.2. Approach 2: Iterative SVR

3.2.1. Discouraging Premature Convergence

To solve this problem, we have to change the loss function to discourage $y_i f_i$'s from lying on the far right of the elbow. This can be done by using the ϵ-insensitive loss used in support vector regression (SVR). In particular, when $\epsilon = 0$, it reduces to the Laplacian loss (Figure 4(a)) $L_s = |f_i - y_i|$. In this case, 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. As L_s is symmetric around the point where $y_i f_i = 1$, the $y_i f_i$'s will tend to lie around this point in order to reduce the loss. Figure 4(b) plots the resultant empirical distribution of $y_i f_i$'s on the same digits task. Because the $y_i f_i$ values are now closer to the origin (e.g., point "a" in Figure 3), it is easier to flip the labels if needed, and so SVR can more easily get out of a poor solution.



(a) Laplacian loss. (b) Distribution of $y_i f_i$'s.

Figure 4. Loss function and empirical distribution of $y_i f_i$'s on the digits task for SVR with the Laplacian loss.

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

3.2.2. Relaxing the Constraint

Another motivation for using SVR with the Laplacian loss is that it can be regarded as SVM training with relaxed constraints. This enables SVR to have greater flexibility than SVM in exploring the search space.

Recall that the primal of SVR, with the use of the Laplacian loss, is (Schölkopf & Smola, 2002):

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

where ξ_i, ξ_i^* are the slack variables. Its dual is:

$$\begin{aligned} \max_{\boldsymbol{\alpha}\boldsymbol{\alpha}^*} \quad & (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top\mathbf{y} - \frac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top\mathbf{K}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ \text{s.t.} \quad & (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top\mathbf{e} = 0, \;\; C\mathbf{e} \ge \boldsymbol{\alpha}, \boldsymbol{\alpha}^* \ge \mathbf{0}, \end{aligned} \tag{6}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]^\top$ and $\boldsymbol{\alpha}^* = [\alpha_1^*, \ldots, \alpha_n^*]^\top$.

From the KKT conditions, α_i and α_i^* (which are non-negative) cannot be both zero. Thus, we can treat $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}^*$ as the positive and negative parts of a variable $\mathbf{u}$, i.e., $\mathbf{u} = \boldsymbol{\alpha} - \boldsymbol{\alpha}^*$ and $|\mathbf{u}| = \boldsymbol{\alpha} + \boldsymbol{\alpha}^*$. Then (6) can be rewritten as

$$\begin{aligned} \max_{\mathbf{u}} \quad & \mathbf{u}^\top\mathbf{y} - \frac{1}{2}\mathbf{u}^\top\mathbf{K}\mathbf{u} \\ \text{s.t.} \quad & \mathbf{u}^\top\mathbf{e} = 0, \;\; C\mathbf{e} \ge \mathbf{u} \ge -C\mathbf{e}. \end{aligned} \tag{7}$$

As the targets are binary ($y_i \in \{\pm 1\}$) in this regression problem, we can factorize u_i as $u_i = u_i y_i^2 = \lambda_i y_i$ where $\lambda_i = u_i y_i$ and $C \ge \lambda_i \ge -C$. Then, (7) becomes

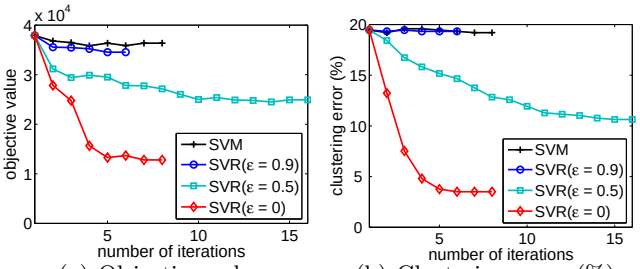
$$\begin{aligned} \max_{\boldsymbol{\lambda}} \quad & 2\boldsymbol{\lambda}^\top\mathbf{e} - \boldsymbol{\lambda}^\top(\mathbf{K} \circ \mathbf{y}\mathbf{y}^\top)\boldsymbol{\lambda} \\ \text{s.t.} \quad & \boldsymbol{\lambda}^\top\mathbf{y} = 0, \;\; C\mathbf{e} \ge \boldsymbol{\lambda} \ge -C\mathbf{e}. \end{aligned}$$

This is very similar to SVM's dual in (3), except that the constraint $C\mathbf{e} \ge \boldsymbol{\lambda} \ge \mathbf{0}$ in (3) is relaxed to $C\mathbf{e} \ge \boldsymbol{\lambda} \ge -C\mathbf{e}$.

3.2.3. Demonstration

Here, we first demonstrate the efficacy of the proposed modification. We follow the same algorithm shown in Algorithm 1, except that SVM training in Step 2 is now changed to SVR training. Besides using $\epsilon = 0$ in the ϵ-insensitive loss, we also experiment with different values of ϵ. Experiment is performed on the same digit task as in Section 3.1.1.

Figure 5 plots the evolutions of the objective value[1] in (2) and the clustering error. As can be seen, SVR using the Laplacian loss leads to better objective values and clustering performance; while SVR using a large ϵ has poor performance similar to the SVM.



(a) Objective value. (b) Clustering error (%).

Figure 5. Comparing iterative SVM and SVR procedures.

3.3. Enforcing the Class Balance Constraint

As discussed in Section 2, one has to enforce the class balance constraint (1) to avoid trivially "optimal" MMC solutions. Hence, we require each of the $\mathbf{y}$'s obtained throughout the proposed iterative process to satisfy (1). To guarantee this, we cannot compute

[1] For the SVR case, this objective value is computed by first training a SVM using the label $\mathbf{y}$ obtained by SVR.

the bias b of the SVR model from the KKT conditions as usual. Instead, we obtain b (and $\mathbf{y}$) by minimizing

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

With $\mathbf{w}$ at hand, this can be reduced to

$$\begin{aligned} \min_{\mathbf{y},b} \quad & \sum_{i=1}^{n} \left|\mathbf{w}^\top\varphi(\mathbf{x}_i) + b - y_i\right| \qquad (8) \\ \text{s.t.} \quad & y_i \in \{\pm 1\}, \ \ -\ell \le \mathbf{e}^\top\mathbf{y} \le \ell. \end{aligned}$$

This optimization problem can be solved easily, based on the observation that b influences the assignment of y_i's by effectively defining a boundary on the sorted list of $\mathbf{w}^\top\varphi(\mathbf{x}_i)$'s. Patterns whose $f(\mathbf{x}_i)$ values are smaller than this boundary should be labeled negative, while the rest are labeled positive.[2] Thus, we can simply shift b in the range such that the class balance constraint is satisfied, compute at each position the corresponding objective value in (8), and set b to be the position with the minimum objective. Now, sorting takes $O(n\log(n))$ time, and the search takes $O(\ell n)$ time. So this procedure takes $O(n\log(n) + \ell n)$ time. The complete procedure is shown in Algorithm 2.

Algorithm 2 Iterative SVR.

1: Initialize the labels $\mathbf{y}$ by simple clustering method.
2: Fix $\mathbf{y}$, and perform SVR with Laplacian loss (6).
3: Compute $\mathbf{w}$ from the KKT condition.
4: Compute the bias b as described in Section 3.3.
5: Assign the labels as $y_i = \text{sign}(\mathbf{w}^\top\varphi(\mathbf{x}_i) + b)$.
6: Repeat steps 2–5 until convergence.

3.4. Time Complexities

Consider a SDP problem of the form: $\min_{\mathbf{x}} \mathbf{f}^T\mathbf{x}$ subject to $\sum_{j=1}^{l} x_j\mathbf{A}_j^i \succeq 0,\ i = 1,\ldots,l$, where $\mathbf{x} \in \mathbb{R}^l, \mathbf{f} \in \mathbb{R}^l$ and $\mathbf{A}_j^i \in \mathbb{R}^{n_i\times n_i}$. On using primal-dual methods to solve such SDPs, it is known that the time complexity per iteration is $O(l^2\sum_{i=1}^{m} n_i^2)$ (Lobo et al., 1998), and the number of iterations is usually $O(\sqrt{\sum_i n_i})$ (Nesterov & Nemirovskii, 1994).

[2]Here, we outline the proof. Suppose that there are two points $\mathbf{x}_j, \mathbf{x}_k$ with $f(\mathbf{x}_j) \ge 0 \ge f(\mathbf{x}_k)$, but the prediction is $y_j = -1$ and $y_k = 1$. Then objective in (8) is non-optimal because $|f(\mathbf{x}_j) + 1| + |f(\mathbf{x}_k) - 1| + \sum_{i\ne j,k}|f(\mathbf{x}_i) - y_i| > |f(\mathbf{x}_j) - 1| + |f(\mathbf{x}_k) + 1| + \sum_{i\ne j,k}|f(\mathbf{x}_i) - y_i|$.

With n examples to be clustered, $l = n^2$ and $n_i = n$ for MMC. So the total time complexity is $O(n^6\cdot n) = O(n^7)$. Similarly, for GMMC, the time complexity is $O(n^4\cdot n^{0.5}) = O(n^{4.5})$. Thus, both methods 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) (Platt, 1999). The number of iterations in iterSVR is usually small (around 10) in practice. Hence, iterSVR is computationally very efficient.

4. Experiments

In this section, experiments are performed on a number of data sets from the UCI repository (ionosphere, digits, letter and satellite), the LIBSVM data[3] (svmguide1-a and w1b) and another benchmark repository[4] (ringnorm and image). For the digits data, we follow (Valizadegan & Jin, 2007) and focus on those pairs (3 vs 8, 1 vs 7, 2 vs 7, and 8 vs 9) that are difficult to differentiate. For the letter and satellite data sets, there are multiple classes and we use their first two classes only (A vs B, and C1 vs C2, respectively). Finally, the w1b, svmguide1-a and ringnorm data sets are relatively large. So a subset, with 500 samples randomly selected from both the positive and negative classes, is used for each data.

We use the LIBSVM package for implementing the iterSVR. The regularization parameter C is fixed at 500 and the Gaussian kernel $\exp\left(-\|\mathbf{z}\|^2/\sigma^2\right)$ is used. To choose the kernel width σ, we first obtain a crude estimate[5] (D) on the maximum distance between samples, and then set σ to be 2 to 5 times of D. The initial $\mathbf{y}$ labels are obtained from k-means clustering (with $k = 2$). Experiments are run on a 2.13GHz Intel Core™2 Duo PC running Windows XP.

4.1. Importance of Class Balance Constraint

Before a comprehensive comparison of the methods, we first demonstrate the importance of the class balance constraint as discussed in Section 3.3. We use the same data set as in Section 3.1, but this time with a suboptimal kernel width for the normalized cut. This produces a very poor clustering result (with 24.5% error) for initializing iterSVR. As can be seen from Figure 6, the absence of the class balance constraint leads

[3]http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/

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

[5]We set $D = (\sum_{k=1}^{d}[\max\{X^k\} - \min\{X^k\}]^2)^{1/2}$, where $\{X^k\}$ is the kth attribute values of the data set.

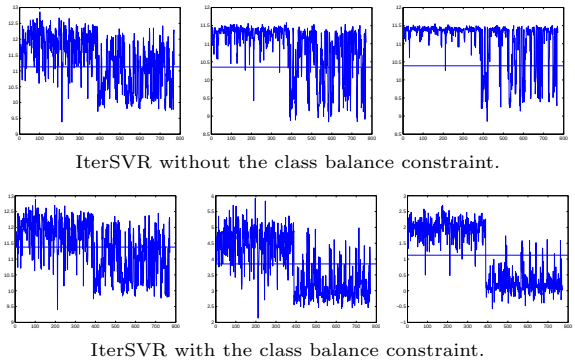


Figure 6. Iterative SVR with/without the class balance constraint. The patterns are ordered along the x-axis, and the y-axis is the prediction. Here, a perfect predictor should produce a (minus) sign curve. From left to right is in the order of iterations.

to a confusion of the two classes. After adding the constraint, iterSVR can separate the two classes well. In the experiments, the balance parameter is chosen as $\ell = 0.03n$ for balanced data, and $\ell = 0.3n$ for unbalanced data.

4.2. Clustering Accuracy

The following methods are compared: (1) k-means clustering (KM); (2) normalized cut (NC) (Shi & Malik, 2000), using the Gaussian affinity function and with all points included in the neighborhood; (3) MMC (Xu et al., 2005); (4) GMMC (Valizadegan & Jin, 2007); (5) iterSVM; (6) iterLS-SVM; (7) iterSVR. 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; while iterLS-SVM uses the square loss. The iterLS-SVM is modified from the least-square SVM (Suykens & Vandewalle, 1999)[6] and serves to demonstrate the importance of the loss function.

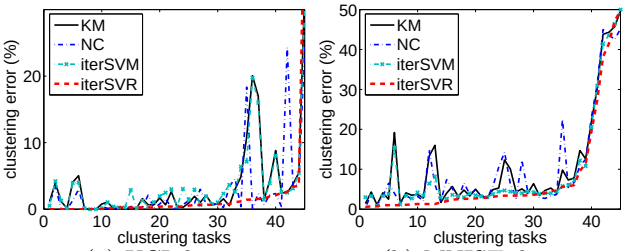
Recall that the initial class labels for the iterative SVM/LS-SVM/SVR are obtained from the k-means clustering procedure. Hence, these local optimization methods, along with the k-means clustering procedure, are susceptible to the problem of local minimum; while NC/MMC/GMMC do not. In the implementation, this problem is alleviated by requiring the initial prototypes of k-means clustering to be sufficiently far away. To demonstrate the effect, these local optimization methods are run 10 times on each data set, with different initial seeds for the k-means clustering, and then the averaged results reported.

Parameters of NC, iterSVM/LS-SVM are chosen from a set of candidates that gives the best performance. For NC/MMC/GMMC, results on the digits and ionosphere data are simply copied from (Valizadegan & Jin, 2007). As for the other data sets, because they are large and both MMC/GMMC are very slow, the results of these two methods will not be reported.

Results are shown in Table 1. Due to the extra care taken in initializing the k-means clustering, in most cases, all the local optimization procedures consistently yield the same clustering result over the 10 runs. Moreover, iterSVR is the most accurate in most cases.

Following (Valizadegan & Jin, 2007), we have only focused on several digit pairs above. Here, we give a more thorough comparison by considering all 45 pairs of digits 0–9. This experiment is also repeated with the MNIST digits[7] (of image size 28×28 and pixel values in $[0, 2]$). The best tuned kernel width parameter is used for the NC. Again, MMC and GMMC are not run because they are too slow.

Figure 7 plots the clustering errors on the 45 clustering tasks. As can be seen, iterSVR works well even on the difficult tasks. Table 2 reports the average clustering error of the algorithms. Again, iterSVR is the best.



(a) UCI digits. (b) MNIST digits.

Figure 7. Comparison using all digit pairs from the UCI and MNIST data sets. The 45 clustering tasks are ordered along the x-axis so that the clustering errors obtained by iterSVR are increasing.

Table 2. Average clustering error (%) over all pairs of UCI and MNIST digits.

DATA	KM	NC	ITERSVM	ITERSVR
UCI	3.62	2.43	3.49	**1.82**
MNIST	10.79	10.08	9.49	**7.59**

4.3. Speed

The main problem with both MMC and GMMC is that they are based on SDPs and are very slow. In

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

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

Table 1. Clustering errors (%) on the various data sets.

DATA	SIZE	KM	NC	MMC	GMMC	ITERSVM	ITERLS-SVM	ITERSVR
DIGITS 3–8	357	5.32 ± 0	35	10	5.6	4.2 ± 0	3.92 ± 0	$\mathbf{3.36 \pm 0}$
DIGITS 1–7	361	$\mathbf{0.55} \pm 0$	45	31.25	2.2	$\mathbf{0.55} \pm 0$	0.55 ± 0	$\mathbf{0.55} \pm 0$
DIGITS 2–7	356	3.09 ± 0	34	1.25	0.5	3.09 ± 0	2.25 ± 0	$\mathbf{0.0} \pm 0$
DIGITS 8–9	354	9.32 ± 0	48	3.75	16.0	9.6 ± 0	8.76 ± 0	$\mathbf{3.67} \pm 0$
IONOSPHERE	351	32±17.9	25	**21.25**	23.5	31.6± 24.9	24.4 ± 2.83	32.3±16.6
W1B	1000	44.1 ± 0	41.8	-	-	39.4 ± 0	-	$\mathbf{36} \pm 0$
SVMGUIDE1-A	1000	23.5 ± 0	12.5	-	-	23.6 ± 0	$\mathbf{6.8} \pm 0$	$\mathbf{6.8} \pm 0$
LETTER	1555	17.94 ± 0	23.2	-	-	14.6 ± 0	7.33 ± 0	$\mathbf{7.2} \pm 0$
SATELLITE	2236	4.07 ± 0	4.21	-	-	4.05 ± 0	3.7 ± 0	$\mathbf{3.18} \pm 0$
IMAGE	1010	43.5 ± 0	41.2	-	-	38.08 ± 0	41.2 ± 0	$\mathbf{28.6} \pm 0$
RINGNORM	1000	24±5.83	22.3	-	-	27.2 ± 6.43	29.8 ± 11.3	$\mathbf{9.3 \pm 5.87}$

this section, we demonstrate the speed advantage of iterSVR. Because of the need to run MMC, we only use 4 small data subsets, which are obtained by sampling 50 samples for each class. The other settings are the same as in (Xu et al., 2005). From Table 3, iterSVR is thousands of times faster than MMC. As reported in (Valizadegan & Jin, 2007), GMMC is about < 100 times faster than MMC on data sets of this size. Hence, iterSVR is still faster than GMMC by about one to two orders of magnitude.

Table 3. Wall clock time (in seconds) and clustering error (%) for the various methods. Number in brackets is the speedup of iterSVR relative to MMC.

	DATA	NC	MMC	ITERSVR
TIME	DIGITS 3–9	0.05	1218	0.2 (6090)
	DIGITS 8–9	0.09	957	0.3 (3190)
	DIGITS 2–7	0.08	1079	0.14 (7707)
	IONOSPHERE	0.11	764	0.12 (6367)
ERROR (%)	DIGITS 3–9	21	7	3
	DIGITS 8–9	9	3	3
	DIGITS 2–7	6	15	8
	IONOSPHERE	29	25	25

4.4. Image Segmentation

In this section, we use the proposed clustering algorithm for color image segmentation. Experiments are performed on five images[8] (Figure 8), with the RGB values ($[0, 255]$) as features. A fixed $\sigma = 500$ is used for all images. For comparison, we run the k-means clustering algorithm and an approach[9] (Figueiredo & Jain, 2002) based on the Expectation-Maximization (EM) algorithm. MMC, GMMC and NC (which requires storing an $n \times n$ affinity matrix) cannot handle such large data sets and so are not compared.

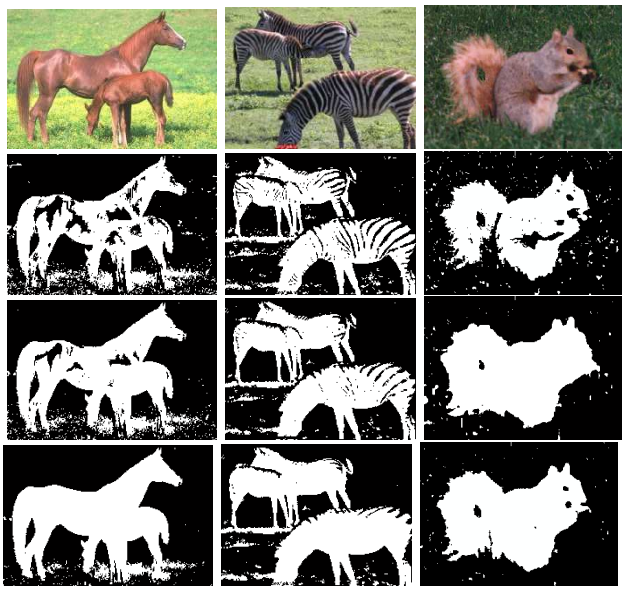


Figure 8. Segmentation results obtained by k-means clustering (2nd row), EM algorithm (3rd row), and iterSVR (bottom row).

Due to lack of space, segmentation results for only three of five images are shown in Figure 8. As can be seen, segmentation results of iterSVR are visually more satisfactory than k-means clustering and EM algorithms. The time required for iterSVR is reported in Table 4.

The algorithm can be extended for multi-class problems by performing binary clustering recursively, in the same manner as (Shi & Malik, 2000). As an example, we perform segmentation on the 261×116 woman image. Figure 9 shows the segmentation results. From left to right, the woman is segmented out from the

[8] bird, palace and horse are from Berkeley image database (http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/), zebra and squirrel are commonly used in vision literature.

[9] Code is from http://www.lx.it.pt/~mtf/mixturecode.zip.

Table 4. Wall clock time (in minutes) for segmentation.

	BIRD	PALACE	HORSE	ZEBRA	SQUIRREL
TIME	6.9	19.5	43.6	37.8	58.5
SIZE	240×160	240×160	240×160	214×160	288×209

background, then the skin and clothes, and so on.

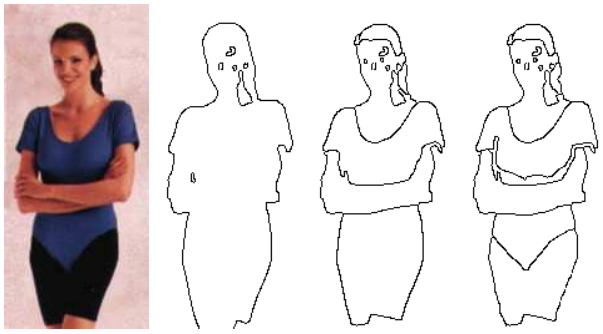


Figure 9. Recursive segmentation results by iterSVR.

5. Conclusion

In this paper, we propose an efficient approach for solving maximum margin clustering via alternating optimization. The key step is on the use of SVR with the Laplacian loss, instead of SVM with the hinge loss, in the inner optimization subproblem. While existing MMC algorithms are very computationally expensive and can only handle very small data sets, our algorithm is often more accurate, much faster (by thousands of times compared with MMC) 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 the learning parameters (C and σ) can be determined in a more disciplined manner.

Acknowledgments

This research has been partially supported by the Research Grants Council of the Hong Kong Special Administrative Region.

References

Bezdek, J., & Hathaway, R. (2003). Convergence of alternating optimization. *Neural, Parallel & Scientific Computations*, *11*, 351–368.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge, UK: Cambridge University Press.

Figueiredo, M., & Jain, A. (2002). Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*, 381–396.

Jain, A., & Dubes, R. (1988). *Algorithms for clustering data*. Englewood Cliffs, NJ: Prentice Hall.

Lanckriet, G., Cristianini, N., Bartlett, P., El Ghaoui, L., & Jordan, M. (2004). Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, *5*, 27–72.

Lobo, M., Vandenberghe, L., Boyd, S., & Lebret, H. (1998). Applications of second-order cone programming. *Linear Algebra Applications*, *284*, 193–228.

Nesterov, Y., & Nemirovskii, A. (1994). *Interior-point polynomial algorithms in convex programming*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges and A. Smola (Eds.), *Advances in kernel methods – Support vector learning*, 185–208. Cambridge, MA: MIT Press.

Schölkopf, B., & Smola, A. (2002). *Learning with kernels*. Cambridge, MA: MIT Press.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*, 888–905.

Suykens, J., & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters*, *9*, 293–300.

Valizadegan, H., & Jin, R. (2007). Generalized maximum margin clustering and unsupervised kernel learning. *Advances in Neural Information Processing Systems 19*. Cambridge, MA: MIT Press.

Xu, L., Neufeld, J., Larson, B., & Schuurmans, D. (2005). Maximum margin clustering. *Advances in Neural Information Processing Systems 17*. Cambridge, MA: MIT Press.

Xu, L., & Schuurmans, D. (2005). Unsupervised and semi-supervised multi-class support vector machines. *Proceedings of The Twentieth National Conference on Artificial Intelligence* (pp. 904–910). Pittsburgh, PA, USA.

Xu, L., Wilkinson, D., Southey, F., & Schuurmans, D. (2006). Discriminative unsupervised learning of structured predictors. *Proceedings of the 23rd International Conference on Machine Learning*.