

Multitask Multiclass Support Vector Machines

You Ji, Shiliang Sun

Department of Computer Science and Technology, East China Normal University,
500 Dongchuan Road, Shanghai 200241, P.R. China
Email: jiyou09@gmail.com, slsun@cs.ecnu.edu.cn

Abstract—In this paper, we present a new classification method named multitask multiclass support vector machines based on the regularization principle. Our starting point is the recent success of multitask learning which has shown that learning multiple related tasks simultaneously can get better results than learning these tasks independently. We cast multitask multiclass problems as a constrained optimization problem with a quadratic objective function. Unlike most approaches which typically decompose a multitask multiclass problem into multiple multitask binary classification problems, our approach can learning multitask multiclass problems directly and effectively. This paper also derives the dual optimization which indicates the relations between tasks. The linear multitask multiclass learning method can be generalized to non-linear cases by the kernel trick. Experimental results indicate that the new approach can get encouraging results for multitask multiclass problems.

Index Terms—Multiclass classification; Multitask learning; Support vector machine; Kernel; Regularization

I. INTRODUCTION

The principal goal of multitask learning is to improve the generalization performance by leveraging the domain-specific information contained in the related tasks [1]. One way to reach the goal is learning multiple related tasks simultaneously while using a shared representation. In fact, the training signals for extra tasks serve as an inductive bias which is important if we wish to learn complex and real-world tasks together [1]. Past empirical work has shown that, when there are relations between tasks to learn, it is beneficial to learn them simultaneously instead of learning each task independently [1]–[4]. There are also a lot of attempts to theoretically study multitask learning [5]–[7]. Multitask learning methods can be natural extensions of existing kernel based on learning methods. In [6], it demonstrates that methods based on multitask kernel can lead to significant performance gains.

A successful instance of multitask learning is the adaptation to support vector machines (SVMs) by multitask kernel methods. The way to solve the multitask multiclass problem using regularized multitask learning was based on reducing the multitask multiclass problem into multitask binary classification problems. Then some multiclass classification strategies such as one-against-one and one-against-all were used to solve multiclass problems. However, while it provides a simple and powerful framework it can not capture correlations between different classes [8]. Multiclass kernel-based vector machines (MKVMs) not only capture these correlations, but also solve multiclass classification in one dual optimization problem instead of reducing multiclass problems into multiple binary classification problems [8].

In this paper we develop and discuss in detail a direct approach for learning multitask multiclass SVMs (M^2 SVMs) with on the basis of MKVMs. Up to recent, multitask support vector learning algorithms have been designed for multitask binary problems [5]. This paper mainly makes attempts to generalize multitask binary SVMs to M^2 SVMs. We cast M^2 SVMs as a constrained optimization problem with a quadratic objective function. Correlations between the different classes and relationships between these related tasks can be well captured by M^2 SVMs. Experimental results of two real-world datasets demonstrate the effectiveness of the proposed method.

The rest of this paper is organized as follows. Section II briefly reviews works on multiclass problems and multitask learning. Section III thoroughly describes the multitask multiclass support vector machines and the process of derivation in detail. Section IV talks about the dual optimization problem of M^2 SVMs. Section V extends our method to non-linear multitask learning by the kernel trick. Experiments and the analysis of experimental results are introduced in Section VI. Conclusions and future work are presented in Section VII.

II. RELATED WORK

M^2 SVMs build on previous research on regularized multitask learning and multiclass learning. In this section, the related work about regularized multitask learning and multiclass learning methods are briefly introduced as preliminary knowledge, from which we reach our M^2 SVMs approach.

A. Regularized Multitask Learning

Regularized multitask learning is presented by Evgeniou et al. [5] that extends single task SVMs to multitask learning. Suppose that there are T different but related tasks. All the data for the T tasks come from the same space $X \times Y$ where $X \subset R^d, Y \subset \{-1, 1\}$. For simplicity, we assume that each example just belongs to one task. Input $(x_i, y_i), (i \in \{1, 2, 3, \dots, m\})$ stands for the i th example whose task index is $\varphi(i)$. m stands for the size of training set. Regularized multitask learning learns T classifiers w_1, \dots, w_T , where each classifier w_t is specifically dedicated to task t where $t \in \{1, \dots, T\}$ [5]. With w_0 which captures the common information among all tasks, all w_t can be rewritten as $w_t = w_0 + v_t$. Then v_t and w_0 are get by solving the optimization problem

$$\min_{w_0, \dots, w_T} \sum_{t=0}^T \lambda_t \|w_t\|_2^2 + \sum_i^m [1 - y_i (w_0 + v_{\varphi(i)}) x_i]_+, \quad (1)$$

where $[a]_+$ means the hinge loss. Constants $\lambda_t \geq 0$ trade-off the regularization of related tasks. There are two extreme cases. If $\lambda_0 \rightarrow +\infty$ then $w_0 \rightarrow 0$ which indicates that all tasks are decoupled. Then there is no related information between tasks. If $\lambda_0 \rightarrow 0$ then $v_t \rightarrow \vec{0}$ which means all tasks share the same decision function with w_0 .

By the strategy of feature map, the above multitask learning can be generalized to include non-linear multitask learning. Kernel methods can be used to model relations between tasks and lead to linear and non-linear multitask learning algorithms [6], [9], [10]. A wide variety of kernels which are useful for applications are proposed in [6], [9].

Although regularized multitask learning formulation and experimental results are superior to their single task counterparts. It requires all tasks to share the same set of class labels. When it refers to multiclass problems, a common method is to build a set of binary classifiers where each classifier distinguishes between one of the labels to the rest (or to one of the other labels). These approaches can not capture correlations between the different classes since it breaks a multiclass problem into multiple independent binary problems [8].

B. Multiclass Learning

Support vector machines are originally designed for the binary classification. How to extend binary SVMs to the multiclass classification is still a research topic [11]. There are two ways to solve multiclass classification problems. One way is to construct a multiclass classifier by combining several binary classifiers. There are three methods based on the binary classification: one-against-all, one-against-one, and DAGSVM [12]. The other way is to consider all classes together [8], [13]. It cast multiclass categorization problems as a constrained optimization problem with a quadratic objective function which yields a direct method for training multiclass predictors [8]. Usually, with this strategy, a much larger dual optimization problem is required to solve. Algorithmic implementations which are able to incorporate kernels with a compact set of constraints and decompose the dual problem into multiple optimization problems of reduced size are proposed in [8], [11]. Because this strategy is related to our work, we briefly review it below. A comparison of methods for multiclass support vector machines is given in [11].

MKVMs are generalizations of separating hyperplanes and margins to the scenario of multiclass problems. Input (x_i, y_i) belongs to $X \times Y$, where $X \in R^d$, $Y \in \{1, 2, 3, \dots, K\}$ and $i \in \{1, \dots, m\}$. This framework uses classifiers of the form

$$H_M(x) = \arg \max_k^K \{M_k x\} \quad (2)$$

where M is a matrix of size $K \times d$. M_k is the k th row of M . The quadratic optimization problem is defined as

$$\begin{aligned} \min_M \quad & \frac{1}{2} \beta \|M\|_2^2 + \sum_{i=1}^m \varepsilon_i \\ \text{s.t.} \quad & \forall k, i, \\ & M_{y_i} x_i + \delta_{y_i, k} - M_k x_i \geq 1 - \varepsilon_i, \end{aligned} \quad (3)$$

where K means that there are K different labels, $\delta_{p,q}$ is equal to 1 if $p = q$ and 0 otherwise, and $\beta > 0$ is a regularization constant. For $k = y_i$, the inequality constraints become $\varepsilon_i \geq 0$. M_k stands for the k th row of M . $M_k x_i$ means the confidence and the similarity core for the k th class. Therefore, according to the equation above, the predicted label we want to get is the index of the row which attains the highest similarity score of samples [8]. Its goal is to find a matrix M that attains a small empirical error on the training samples and also generalizes well. The advantage of this strategy is solving multiclass classification problems so that it can capture the information between different classes directly. Results in [11] also show that for large-scale problems methods considering all classes at once need fewer support vectors than methods constructed by binary classifiers.

III. M²SVMs

All the data for T tasks comes from the same distribution P on $X \times Y$ where $X \subset R^d$, $Y \subset \{1, 2, 3, \dots, K\}$. For simplicity, we assume that each sample just belongs to one task. For each task we have $m_t, t \in \{1, 2, 3, \dots, T\}$ instances sampled from P_t . We assume that P_t is different from each other but P_t has relationships with other tasks [5], [14]. When $T = 1$, it is the same with the single task learning problem. Our framework uses classifiers as

$$\begin{aligned} H_{M_t}(x) &= \arg \max_k^K \{M_{t,k} x_t\} \\ M_t &= M_0 + V_t \end{aligned} \quad (4)$$

where M_t is a matrix of size $K \times d$. $M_{t,k}$ stands for the k th row of M_t , and M_t comes from a particular probability distribution P_t . This implies that all M_t are close to the mean matrix M_0 . M_0 shares classification information between tasks. When the tasks are very similar to each other, matrixes V_t are very small. To this end we solve the following optimization problem which is similar to (3):

$$\begin{aligned} \min \quad & \beta \sum_{k=1}^K \left[\sum_{t=1}^T \rho_t \|V_{t,k}\|^2 + \|M_{0,k}\|^2 \right] + \sum_{t=1}^T \sum_{i=1}^{m_t} \varepsilon_{t,i} \\ \text{s.t.} \quad & \forall t, k, i, \\ & (V_{t,y_i} + M_{0,y_i}) x_{t,i} + \delta_{k,y_i} - (V_{t,k} + M_{0,k}) x_{t,i} \geq 1 - \varepsilon_{t,i} \end{aligned} \quad (5)$$

where $\delta_{p,q}$ is equal to 1 if $p = q$ and 0 otherwise, and ρ_t is the weighted parameter between V_t and M_0 . $\beta > 0$ and $\rho_t > 0$ are regularization constants. For $k = y_i$, inequality constraints become $\varepsilon_{t,i} \geq 0$. Solving the question directly is not a good choice. We find that it is better to solve the Eq. (5)'s dual optimization problem. Eq. (6) to Eq. (21) describes the details how we achieve the dual optimization problem. The full dual optimization problem is given in Eq. (22). We add a dual set of variables, one for each constraint and get the Lagrangian

of the optimization problem as

$$\begin{aligned}
L(V_{t,k}, M_{0,k}, \varepsilon_{t,i}) &= \beta \sum_{k=1}^K \left[\sum_{t=1}^T \rho_t \|V_{t,k}\|^2 + \|M_{0,k}\|^2 \right] \\
&- \sum_{k=1}^K \sum_{t=1}^T \sum_{i=1}^{m_t} a_{k,t,i} \left[\begin{aligned} &(V_{t,y_i} + M_{0,y_i}) x_{t,i} + \delta_{k,y_i} \\ &- (V_{t,k} + M_{0,k}) x_{t,i} - 1 + \varepsilon_{t,i} \end{aligned} \right] \\
&+ \sum_{t=1}^T \sum_{i=1}^{m_t} \varepsilon_{t,i} \\
\text{s.t. : } &\forall t, k, i \\
&a_{k,t,i} \geq 0.
\end{aligned} \tag{6}$$

In order to find the minimum for the primal variables V_t, M_0, ε , we require,

$$\frac{\partial L}{\partial \varepsilon_{t,i}} = 1 - \sum_{k=1}^K a_{k,t,i} = 0 \Rightarrow \sum_{k=1}^K a_{k,t,i} = 1. \tag{7}$$

Similarly, for $M_{0,k}$, we require,

$$\begin{aligned}
\frac{\partial L}{\partial M_{0,k}} &= 2\beta M_{0,k} + \sum_{t=1}^T \sum_{i=1}^{m_t} a_{k,t,i} x_{t,i} \\
&- \sum_{k=y_i}^{y_i} \sum_{t=1}^T \sum_{i=1}^{m_t} \left[\underbrace{\left(\sum_{e=1}^K a_{e,t,i} \right)}_{=1} x_{t,i} \right] \\
&= 2\beta M_{0,k} + \sum_{t=1}^T \sum_{i=1}^{m_t} a_{k,t,i} x_{t,i} - \sum_{k=y_i}^{y_i} \sum_{t=1}^T \sum_{i=1}^{m_t} x_{t,i} \\
&= 2\beta M_{0,k} + \sum_{t=1}^T \sum_{i=1}^{m_t} a_{k,t,i} x_{t,i} - \sum_{t=1}^T \sum_{i=1}^{m_t} \delta_{k,y_i} x_{t,i} = 0
\end{aligned} \tag{8}$$

which results in the following form

$$\begin{aligned}
M_{0,k} &= \frac{1}{2\beta} \left(\sum_{t=1}^T \sum_{i=1}^{m_t} \delta_{k,y_i} x_{t,i} - \sum_{t=1}^T \sum_{i=1}^{m_t} a_{k,t,i} x_{t,i} \right) \\
&= \frac{1}{2\beta} \sum_{t=1}^T \sum_{i=1}^{m_t} (\delta_{k,y_i} - a_{k,t,i}) x_{t,i}.
\end{aligned} \tag{9}$$

Next, for $V_{t,k}$, we need,

$$\begin{aligned}
\frac{\partial L}{\partial V_{t,k}} &= 2\beta \rho_t V_{t,k} + \sum_{i=1}^{m_t} a_{k,t,i} x_{t,i} \\
&- \sum_{k=y_i}^{y_i} \sum_{i=1}^{m_t} \left[\underbrace{\left(\sum_{e=1}^K a_{e,t,i} \right)}_{=1} x_{t,i} \right] \\
&= 2\beta \rho_t V_{t,k} + \sum_{i=1}^{m_t} a_{k,t,i} x_{t,i} - \sum_{k=y_i}^{y_i} \sum_{i=1}^{m_t} x_{t,i} \\
&= 2\beta \rho_t V_{t,k} + \sum_{i=1}^{m_t} a_{k,t,i} x_{t,i} - \sum_{i=1}^{m_t} \delta_{k,y_i} x_{t,i} \\
&= 0
\end{aligned} \tag{10}$$

which results in the form

$$\begin{aligned}
V_{t,k} &= \frac{1}{2\beta \rho_t} \left(\sum_{i=1}^{m_t} \delta_{k,y_i} x_{t,i} - \sum_{i=1}^{m_t} a_{k,t,i} x_{t,i} \right) \\
&= \frac{1}{2\beta \rho_t} \sum_{i=1}^{m_t} (\delta_{k,y_i} - a_{k,t,i}) x_{t,i}.
\end{aligned} \tag{11}$$

To solve the optimization problem, we have to rewrite $L(V_{t,k}, M_{0,k}, \varepsilon_{t,i})$ in Eq. (6) as

$$\begin{aligned}
L(V_{t,k}, M_{0,k}, \varepsilon_{t,i}) &= \beta \sum_{k=1}^K \sum_{t=1}^T \left[\rho_t \|V_{t,k}\|^2 + \frac{1}{\beta} \sum_{i=1}^{m_t} (a_{k,t,i} V_{t,k} x_{t,i} - a_{k,t,i} V_{t,y_i} x_{t,i}) \right] \\
&+ \beta \sum_{k=1}^K \left[\|M_0\|^2 + \frac{1}{\beta} \sum_{t=1}^T \sum_{i=1}^{m_t} (a_{k,t,i} M_{0,k} x_{t,i} - a_{k,t,i} M_{0,y_i} x_{t,i}) \right] \\
&+ \sum_{t=1}^T \sum_{i=1}^{m_t} \varepsilon_{t,i} - \sum_{k=1}^K \sum_{t=1}^T \sum_{i=1}^{m_t} a_{k,t,i} [\delta_{k,y_i} - 1 + \varepsilon_{t,i}] \\
&= \beta \sum_{k=1}^K \left[\underbrace{\sum_{t=1}^T \left(\rho_t \|V_{t,k}\|^2 + \frac{1}{\beta} \sum_{i=1}^{m_t} a_{k,t,i} V_{t,k} x_{t,i} \right)}_{\triangleq S_1} \right] \\
&\quad + \underbrace{\|M_0\|^2 + \frac{1}{\beta} \sum_{t=1}^T \sum_{i=1}^{m_t} a_{k,t,i} M_{0,k} x_{t,i}}_{\triangleq S_2} \\
&- \sum_{k=1}^K \sum_{t=1}^T \sum_{i=1}^{m_t} \left[\underbrace{a_{k,t,i} V_{t,y_i} x_{t,i} + a_{k,t,i} M_{0,y_i} x_{t,i}}_{\triangleq S_3} \right] \\
&+ \sum_{k=1}^K \sum_{t=1}^T \sum_{i=1}^{m_t} a_{k,t,i} \delta_{k,y_i} + C
\end{aligned} \tag{12}$$

where C is a constant variable which is equal to the whole training data size of all tasks as

$$C = \sum_{k=1}^K \sum_{t=1}^T \sum_{i=1}^{m_t} a_{k,t,i} = \sum_{t=1}^T \sum_{i=1}^{m_t} \underbrace{\sum_{k=1}^K a_{k,t,i}}_{=1} = \sum_{t=1}^T m_t. \tag{13}$$

We substitute Eq. (11) into Eq. S_1 and get,

$$\begin{aligned}
S_1 &= \rho_t \|V_{t,k}\|^2 + \frac{1}{\beta} \sum_{i=1}^{m_t} a_{k,t,i} V_{t,k} x_{t,i} \\
&= \frac{1}{4\beta^2 \rho_t} \sum_{i=1}^{m_t} \sum_{j=1}^{m_t} (\delta_{k,y_i} - a_{k,t,i}) (\delta_{k,y_j} - a_{k,t,j}) \langle x_{t,i}, x_{t,j} \rangle \\
&+ \frac{1}{2\beta^2 \rho_t} \sum_{i=1}^{m_t} \sum_{j=1}^{m_t} a_{t,k,i} (\delta_{k,y_j} - a_{k,t,j}) \langle x_{t,i}, x_{t,j} \rangle \\
&= \frac{1}{4\beta^2 \rho_t} \sum_{i=1}^{m_t} \sum_{j=1}^{m_t} \left[\begin{aligned} &(\delta_{k,y_i} + a_{k,t,i}) \\ &\times (\delta_{k,y_j} - a_{k,t,j}) \langle x_{t,i}, x_{t,j} \rangle \end{aligned} \right].
\end{aligned} \tag{14}$$

Then we substitute Eq. (9) into S_2 and get,

$$\begin{aligned}
S_2 &= \|M_0\|^2 + \frac{1}{\beta} \sum_{t=1}^T \sum_{i=1}^{m_t} a_{k,t,i} M_{0,k} x_{t,i} \\
&= \frac{1}{4\beta^2} \sum_{t=1}^T \sum_{s=1}^T \sum_{i=1}^{m_t} \sum_{j=1}^{m_s} (\delta_{k,y_i} - a_{k,t,i}) (\delta_{k,y_j} - a_{k,s,j}) \langle x_{t,i}, x_{s,j} \rangle \\
&+ \frac{1}{2\beta^2} \sum_{t=1}^T \sum_{s=1}^T \sum_{i=1}^{m_t} \sum_{j=1}^{m_s} a_{k,t,i} (\delta_{k,y_j} - a_{k,s,j}) \langle x_{t,i}, x_{s,j} \rangle \\
&= \frac{1}{4\beta^2} \sum_{t=1}^T \sum_{s=1}^T \sum_{i=1}^{m_t} \sum_{j=1}^{m_s} \left[\begin{aligned} &(\delta_{k,y_i} + a_{k,t,i}) \\ &\times (\delta_{k,y_j} - a_{k,s,j}) \langle x_{t,i}, x_{s,j} \rangle \end{aligned} \right].
\end{aligned} \tag{15}$$

After substituting Eq. (9) and Eq. (11) into S_3 and get,

$$\begin{aligned}
S_3 &= a_{k,t,i} V_{t,y_i} x_{t,i} + a_{k,t,i} M_{0,y_i} x_{t,i} \\
&= \frac{1}{2\beta\rho_t} \sum_{j=1}^{m_t} a_{k,t,i} (\delta_{y_i,y_j} - a_{y_i,t,i}) \langle x_{t,i}, x_{t,j} \rangle \\
&+ \frac{1}{2\beta} \sum_{s=1}^T \sum_{j=1}^{m_s} a_{k,t,i} (\delta_{y_i,y_j} - a_{y_i,s,j}) \langle x_{t,i}, x_{s,j} \rangle \\
&= \frac{1}{2\beta} \sum_{s=1}^T \sum_{j=1}^{m_s} \left[\left(1 + \frac{\delta_{t,s}}{\rho_t}\right) a_{k,t,i} \right. \\
&\quad \left. \times (\delta_{y_i,y_j} - a_{y_i,s,j}) \langle x_{t,i}, x_{s,j} \rangle \right]. \tag{16}
\end{aligned}$$

If we set $S_4 = \sum_{t=1}^T S_1 + S_2$, then we get,

$$\begin{aligned}
S_4 &= \sum_{t=1}^T S_1 + S_2 \\
&= \frac{1}{4\beta^2\rho_t} \sum_{t=1}^T \sum_{i=1}^{m_t} \sum_{j=1}^{m_t} \left[(\delta_{k,y_i} + a_{k,t,i}) \right. \\
&\quad \left. \times (\delta_{k,y_j} - a_{k,t,j}) \langle x_{t,i}, x_{t,j} \rangle \right] \\
&+ \frac{1}{4\beta^2} \sum_{t=1}^T \sum_{s=1}^T \sum_{i=1}^{m_t} \sum_{j=1}^{m_s} \left[(\delta_{k,y_i} + a_{k,t,i}) \right. \\
&\quad \left. \times (\delta_{k,y_j} - a_{k,s,j}) \langle x_{t,i}, x_{s,j} \rangle \right] \\
&= \frac{1}{4\beta^2} \sum_{t=1}^T \sum_{s=1}^T \sum_{i=1}^{m_t} \sum_{j=1}^{m_s} \left[\left(1 + \frac{\delta_{t,s}}{\rho_t}\right) \right. \\
&\quad \left. \times (\delta_{k,y_i} + a_{k,t,i}) \right. \\
&\quad \left. \times (\delta_{k,y_j} - a_{k,s,j}) \langle x_{t,i}, x_{s,j} \rangle \right]. \tag{17}
\end{aligned}$$

Then we substitute S_3, S_4 into Eq. (12) and get,

$$\begin{aligned}
Q(a) &= \frac{1}{4\beta} \sum_{k=1}^K \sum_{t=1}^T \sum_{s=1}^T \sum_{i=1}^{m_t} \sum_{j=1}^{m_s} \left[\left(1 + \frac{\delta_{t,s}}{\rho_t}\right) \right. \\
&\quad \left. \times (\delta_{k,y_i} + a_{k,t,i}) \right. \\
&\quad \left. \times (\delta_{k,y_j} - a_{k,s,j}) \langle x_{t,i}, x_{s,j} \rangle \right] \\
&- \frac{1}{2\beta} \sum_{k=1}^K \sum_{t=1}^T \sum_{i=1}^{m_t} \sum_{s=1}^T \sum_{j=1}^{m_s} \left[\left(1 + \frac{\delta_{t,s}}{\rho_t}\right) a_{k,t,i} \right. \\
&\quad \left. \times (\delta_{y_i,y_j} - a_{y_i,s,j}) \langle x_{t,i}, x_{s,j} \rangle \right] \\
&+ \sum_{k=1}^K \sum_{t=1}^T \sum_{i=1}^{m_t} a_{k,t,i} \delta_{k,y_i} \\
&= \frac{1}{4\beta} \sum_{k=1}^K \sum_{t=1}^T \sum_{s=1}^T \sum_{i=1}^{m_t} \sum_{j=1}^{m_s} \left[\left(\delta_{k,y_i} + a_{k,t,i} \right) \right. \\
&\quad \left. \times (\delta_{k,y_j} - a_{k,s,j}) \right. \\
&\quad \left. - 2a_{k,t,i} (\delta_{y_i,y_j} - a_{y_i,s,j}) \right. \\
&\quad \left. \times \left(1 + \frac{\delta_{t,s}}{\rho_t}\right) \langle x_{t,i}, x_{s,j} \rangle \right] \\
&+ \sum_{k=1}^K \sum_{t=1}^T \sum_{i=1}^{m_t} a_{k,t,i} \delta_{k,y_i}. \tag{18}
\end{aligned}$$

In order to simplify the expression of $Q(a)$ in Eq. (18), we set

$$\begin{aligned}
S_5 &= \sum_{k=1}^K \left[(\delta_{k,y_i} + a_{k,t,i}) (\delta_{k,y_j} - a_{k,s,j}) \right. \\
&\quad \left. - 2a_{k,t,i} (\delta_{y_i,y_j} - a_{y_i,s,j}) \right] \\
&= \sum_{k=1}^K (\delta_{k,y_i} + a_{k,t,i}) (\delta_{k,y_j} - a_{k,s,j}) \\
&\quad - 2 \underbrace{\sum_{k=1}^K a_{k,t,i}}_{=1} (\delta_{y_i,y_j} - a_{y_i,s,j}) \\
&= \sum_{k=1}^K (\delta_{k,y_i} + a_{k,t,i}) (\delta_{k,y_j} - a_{k,s,j}) \\
&\quad - 2 (\delta_{y_i,y_j} - a_{y_i,s,j}) \\
&= \sum_{k=1, k \neq y_i}^K \left(\underbrace{\delta_{k,y_i}}_{=0} + a_{k,t,i} \right) (\delta_{k,y_j} - a_{k,s,j}) \\
&\quad + \left(\underbrace{\delta_{y_i,y_i}}_{=1} + a_{y_i,t,i} - 2 \right) (\delta_{y_i,y_j} - a_{y_i,s,j}) \\
&= \sum_{k=1, k \neq y_i}^K a_{k,t,i} (\delta_{k,y_j} - a_{k,s,j}) \\
&\quad + (a_{y_i,t,i} - 1) (\delta_{y_i,y_j} - a_{y_i,s,j}) \\
&= \sum_{k=1, k \neq y_i}^K \left(a_{k,t,i} - \underbrace{\delta_{k,y_i}}_{=0} \right) (\delta_{k,y_j} - a_{k,s,j}) \\
&\quad + \left(a_{y_i,t,i} - \underbrace{\delta_{y_i,y_i}}_{=1} \right) (\delta_{y_i,y_j} - a_{y_i,s,j}) \\
&= \sum_{k=1}^K (a_{k,t,i} - \delta_{k,y_i}) (\delta_{k,y_j} - a_{k,s,j}). \tag{19}
\end{aligned}$$

After we substitute S_5 into Eq. (18), we get,

$$\begin{aligned}
Q(a) &= \frac{1}{4\beta} \sum_{k=1}^K \sum_{t=1}^T \sum_{s=1}^T \sum_{i=1}^{m_t} \sum_{j=1}^{m_s} \left[(a_{k,t,i} - \delta_{k,y_i}) (\delta_{k,y_j} - a_{k,s,j}) \right. \\
&\quad \left. \times \left(1 + \frac{\delta_{t,s}}{\rho_t}\right) \langle x_{t,i}, x_{s,j} \rangle \right] \\
&+ \sum_{k=1}^K \sum_{t=1}^T \sum_{i=1}^{m_t} a_{k,t,i} \delta_{k,y_i}. \tag{20}
\end{aligned}$$

IV. DUAL OPTIMIZATION PROBLEM

We may rewrite Eq. (20) as

$$\begin{aligned}
Q(a) &= \frac{1}{4\beta} \sum_{i=1}^m \sum_{j=1}^m \left[(a_{k,i} - \delta_{k,y_i}) (\delta_{k,y_j} - a_{k,j}) \right. \\
&\quad \left. \times \left(1 + \frac{\delta_{\varphi(i),\varphi(j)}}{\rho_{\varphi(i)}}\right) \langle x_i, x_j \rangle \right] \\
&+ \sum_{i=1}^m \sum_{j=1}^m a_{k,i} \delta_{k,y_i} \tag{21}
\end{aligned}$$

where $m = \sum_{t=1}^T m_t$, $i, j \in \{1, 2, 3, \dots, m\}$, m means the training data size and $\varphi(i)$ means that the i th sample belongs to $\varphi(i)$ th task. We obtain the following objective function of

the dual program,

$$\begin{aligned} \min Q(a) &= \frac{1}{4\beta} \sum_{i=1}^m \sum_{j=1}^m \left[\begin{aligned} &(a_{k,i} - \delta_{k,y_i}) (\delta_{k,y_j} - a_{k,j}) \\ &\times \left(1 + \frac{\delta_{\varphi(i),\varphi(j)}}{\rho_{\varphi(i)}} \right) \langle x_i, x_j \rangle \end{aligned} \right] \\ &+ \sum_{i=1}^m \sum_{j=1}^m a_{k,i} \delta_{k,y_i} \\ \text{s.t. : } \forall i, j, k, \quad &a_{k,i} \geq 0, \quad \sum_{k=1}^K a_{k,i} = 1. \end{aligned} \quad (22)$$

V. NON-LINEAR MULTITASK LEARNING

The dual optimization problem in Eq. (22) has one shortcoming: a linear decision function may be unsuitable to classify different examples especially when training sets are not linearly separable. But an important characteristic of SVMs is that they can be used to estimate highly non-linear functions through the use of kernels [5]. This method is named the kernel trick. The kernel trick can be used to avoid computing feature maps directly. In fact, every algorithm based on inner products can be kernelized [15]. By the same strategy in [5], we will extend our strategy to non-linear multitask learning as

$$\begin{aligned} \min Q(a) &= \frac{1}{4\beta} \sum_{i=1}^m \sum_{j=1}^m \left[\begin{aligned} &(a_{k,i} - \delta_{k,y_i}) (\delta_{k,y_j} - a_{k,j}) \\ &\times K_{\varphi(i),\varphi(j)} \langle x_i, x_j \rangle \end{aligned} \right] \\ &+ \sum_{i=1}^m \sum_{j=1}^m a_{k,i} \delta_{k,y_i} \\ \text{s.t. : } \forall i, j, k, \quad &a_{k,i} \geq 0, \quad \sum_{k=1}^K a_{k,i} = 1 \end{aligned} \quad (23)$$

where K is given as

$$K_{\varphi(i),\varphi(j)}(x_i, x_j) = \left(1 + \frac{\delta_{\varphi(i),\varphi(j)}}{\rho_{\varphi(i)}} \right) \langle x_i, x_j \rangle \quad (24)$$

We may replace the dot product with a non-linear kernel as is done for standard SVMs [5]. It is noteworthy that this non-linear multitask kernel is different from the one in [5]. All tasks share the same parameter μ in [5]. Because tasks' weights should be different in multitask learning, we use a separate parameter ρ_t for each task. We rewrite the classifier as

$$\begin{aligned} H_t(x) &= \arg \max_{k=1}^K \{M_{t,k}x\} \\ &= \arg \max_{k=1}^K \left\{ \frac{1}{2\beta} \sum_{i=1}^m (\delta_{k,y_i} - a_{k,i}) K_{\varphi(i),t}(x_i, x) \right\}. \end{aligned} \quad (25)$$

It is an easy way to implement our strategy by replacing kernels used in MKVMs. MKVMs have two implementations. One is described in [8] and the other one is given in [11]. In experiments, we choose the implementation in [11] as our solver.

VI. EXPERIMENTS

We evaluate M²SVMs on two datasets taken from the UCI Machine Learning Repository [16]. The first one is Isolet spoken alphabet recognition while the second one is Spoken Arabic Digit (SAD). We first provide a brief overview of the two datasets and then present results in our experimental settings.

A. Details about Datasets

The Isolet dataset was collected from 150 subjects uttering all English alphabet twice. Each speaker contributed 52 training examples. Because three examples are historically missing, there are 7797 examples in total. These speakers are grouped into sets of 30 speakers each. These groups are referred to as isolet1 to isolet5. The original task is to classify which letter has been uttered based on features such as spectral coefficients, contour features, sonorant features, pre-sonorant features, and post-sonorant features. The representation of Isolet lends itself to the multitask multiclass learning [4]. So we treat each of the subsets as its own classification task ($T = 5$) with $K = 26$ labels. These five tasks are highly related with each other because all the data is collected from same utterances [4]. Because groups of speakers vary greatly in the way of speaking the English alphabets, five tasks are different from each other. These five tasks are labeled from 1 to 5. In order to remove low variance noise and to speed up computation time we preprocess the Isolet data with principal component analysis. We capture 98% of the data variance while reducing the dimensionality from 617 to 290.

The SAD dataset was collected from 88 speakers with 44 males and 44 females Arabic native speakers between the ages 18 and 40 to represent ten spoken Arabic digits from 0 to 9. Each speaker spoken one Arabic digit 10 times. So there are 8800 (10 digits \times 10 repetitions \times 88 speakers) samples in the dataset. The original task is to classify which Arabic digit has been uttered based on 13 mel-frequency cepstral coefficients. The representation of SAD lends itself to the multitask multiclass learning. We split the original task into two tasks. One task (Task 1) is to classify Arabic digits spoken by males while the other one (Task 2) is to classify ones spoken by females. These two tasks are highly related with each other because these data are sampled from the same region. Because the voice, intonation and volume are usually different between male and female, we may think that these two tasks are also different between each other. In the original dataset, the feature of one spoken digit is a matrix of size $row \times 13$ ($4 \leq row \leq 93$). For simplicity, we compute the average value of each column of this matrix. Then we get a vector of size 1×13 to represent one digit's feature. Details about these two datasets are listed in Table I.

TABLE I
DETAILS OF DATASETS.

Name	Attributes	Instances	Classes	Tasks
Isolet	290	7797	26	5
SAD	13	8800	10	2

B. Experimental Setting

Scaling before applying SVMs is very important [17]. The main advantage of scaling is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges. Another advantage is to avoid computing difficulties. We scale each attribute to the range $[-1, +1]$ as in [17].

In our experiments, we use 3-fold cross validation to get the average error rate. The multitask kernel used in our experiments is based on the widely used radial basis function (RBF). The kernel can be written as

$$\begin{aligned} K_{\varphi(i)\varphi(j)}(x_i, x_j) &= \left(1 + \frac{\delta_{\varphi(i), \varphi(j)}}{\rho_t}\right) k(x_i, x_j) \\ k(x_i, x_j) &= \exp\left(-\gamma\|x_i - x_j\|^2\right), \gamma > 0 \end{aligned} \quad (26)$$

were γ is kernel parameter. There are other two parameters ρ_t, β for our model.

It is usually unknown beforehand which parameters are best for problems [17]. We used a grid search strategy to select best parameters which are also recommend in [17]. Various pairs of pairs of (ρ_t, β, γ) values are tried. Then we prefer to use the one with the best cross validation results on the training set. Usually, we use a coarse grid first. After we find the better region on the grid, a finer grid search on that region can be conducted. In our experiments, firstly we take a coarse grid search for parameters ρ_t, β, γ in the region $[2^{-10}, 2^{14}]$ with exponent growth 0.5. Then we take the finer grid search on the neighborhood of the best results of the coarse grid search such as $[2^{-1}, 2^3]$. The exponent growth is 0.01. Although the process of grid search usually takes a lot of time during experiments, it is an effective and straightforward method which avoid doing more consuming exhaustive parameter search.

C. Results and Analysis

We compared the performance of our multitask multiclass SVMs algorithm with different baselines in Table II and Table III. *ALL* means just taking all the T tasks' data together and taking these data with no tasks' differences. Multitask learning is denoted as *MT* in Table II and Table III. For example, 3.66% in Table II is the result of multitask learning which is based on one-against-all multiclass classification strategy. 2.70% in Table I is the result of M²SVMs.

From Table II, because of the increasing of training samples, we find that accuracy is highly improved when we just take instances of all tasks together and ignore relationships between tasks. In Table III, collecting instances of all tasks together just gets error rates which seem to be the average results of two tasks'. But if relationships between tasks are captured by multiclass learning, multitask learning get better results than methods which just ignore these useful information between tasks in Table II and Table III. This means the importance of relationship between tasks.

If we compare results by columns, we can get that MKVMs can get comparable results with the other two popular methods. Because MKVMs can capture relationship between different classes, it can get better results than other two methods.

In Table III, the result of M²SVMs is 2.97% while MKVMs' is 2.73% of Task 1. It seems M²SVMs get a worse result than learning tasks independently. But if we consider Task 1 and Task 2 together, there are $1467 \times 2.73\% + 1467 \times 5.28\% = 117$ instances misclassified. With M²SVMs, there are only $2934 \times 2.97\% = 87$ instances misclassified. (There are 1467 examples

for testing in Task 1 and Task 2, because we use 3-fold cross validation and the data size of each task is 4400. Therefore, there are 2934 examples for testing in *MT* and *ALL*.) If we just take instances of Task 1 and Task 2 together, MKVMs get $2934 \times 4.10\% = 120$ instances misclassified. This is even worse than learning tasks independently. This comparison also indicates the importance of exploiting correlations between tasks.

Because M²SVMs not only can capture correlations between different classes, but also can use common information between tasks to prediction, it get the best results both in Table II and Table III.

TABLE II
ERROR RATES ON ISOLET DATASET.

Isolet	one-against-one	one-against-all	MKVMs
Task 1	7.12%	7.18%	5.51%
Task 2	6.47%	6.73%	5.00%
Task 3	9.70%	9.68%	9.70%
Task 4	10.40%	10.98%	11.23%
Task 5	10.33%	9.81%	10.29%
ALL	4.84%	4.23%	4.10%
MT	3.23%	3.66%	2.70%

TABLE III
ERROR RATES ON SAD DATASET.

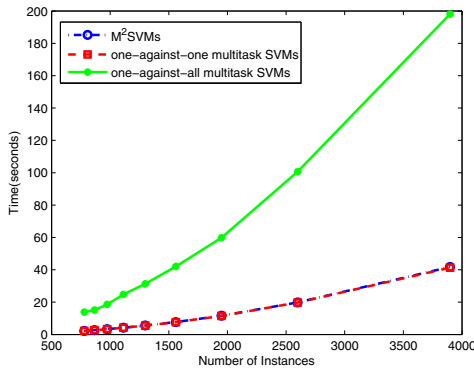
SAD	one-against-one	one-against-all	MKVMs
Task 1	3.46%	3.07%	2.73%
Task 2	5.32%	5.10%	5.28%
ALL	4.97%	4.32%	4.10%
MT	3.10%	3.25%	2.97%

Because the complexity of the dual optimization problem, one may think that M²SVMs may take a lot of computing time. We also compare these methods on the computing time. The computational experiments were done on a Intel Core 2 with 1024 RAM using gcc compiler. Note that the results shown in Fig. 1(a) and Fig. 1(b) are the average time of 3-fold cross validation with instances randomly selected repeated by 10 times. From Fig. 1(a) and Fig. 1(b), we may easily find that M²SVMs take the similar time with multitask SVMs based on one-against-one strategy. Multitask SVMs based on one-against-all strategy take more time than the other two methods.

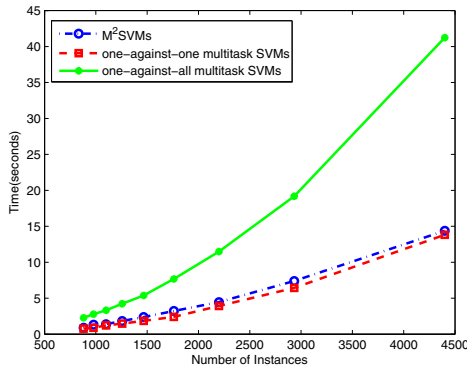
Because using less support vectors can improve the speed of prediction and avoid over-fitting, we also make comparison on the number of support vectors. After the training process, these three algorithms have different number of support vectors. From Fig. 2(a) and Fig. 2(b) we may find that M²SVMs and multitask SVMs based on one-against-one take less support vectors than multitask SVMs based on one-against-all.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we present a new method named M²SVMs based on the minimization of regularization principle. Our method is mainly based on MKVMs [8] and regularized



(a) Results of Isolet.



(b) Results of SAD.

Fig. 1. Comparison of computing time.

multitask learning [5]. We get a dual optimization problem which is optimized to give solutions. Then we extend it to non-linear multitask learning by the kernel trick. This method is very easy to implement by using existing tools.

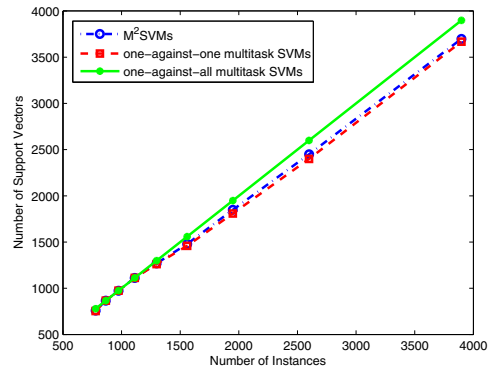
We compared our method with the other two popular multitask learning methods which are based on one-against-one and one-against-all strategies. We mainly compare these algorithms in three aspects: accuracy, computing time and the number of support vectors. The results show that our method get better results than the other two methods almost on all comparisons. How to extend multitask SVMs to label-incompatible multitask learning is still an open question. It is valuable to study in the future.

ACKNOWLEDGMENT

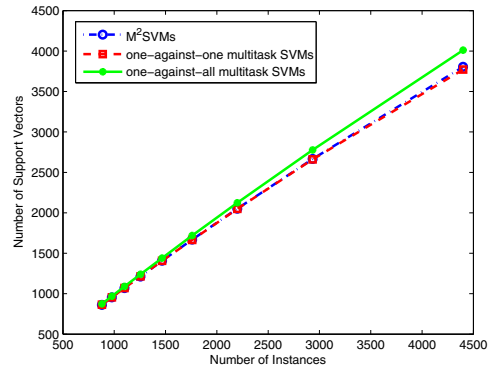
This work is supported in part by the National Natural Science Foundation of China under Project 61075005, and the Fundamental Research Funds for the Central Universities.

REFERENCES

- [1] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, pp. 41–75, 1997.
- [2] S. Sun, "Multitask learning for EEG-based biometrics," in *ICPR*, 2008, pp. 1–4.
- [3] X.T. Yuan and S. Yuan, "Visual classification with multi-task joint sparse representation," in *CVPR*, 2010, pp. 3493–3500.



(a) Results of Isolet.



(b) Results of SAD.

Fig. 2. Comparison of the number of support vectors.

- [4] S. Parameswaran and K.Q. Weinberger, "Large margin multi-task metric learning," in *NIPS*, 2010.
- [5] T. Evgeniou and M. Pontil, "Regularized multi-task learning," in *KDD*, 2004, pp. 109–117.
- [6] T. Evgeniou, C.A. Micchelli, and M. Pontil, "Learning multiple tasks with kernel methods," *Journal of Machine Learning Research*, vol. 6, pp. 615–637, 2005.
- [7] A. Argyriou, T. Evgeniou, and M. Pontil, "Convex multi-task feature learning," *Machine Learning*, vol. 73, pp. 243–272, 2008.
- [8] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of Machine Learning Research*, vol. 2, pp. 265–292, 2002.
- [9] C.A. Micchelli and M. Pontil, "Kernels for multi-task learning," in *NIPS*, 2005, vol. 17, pp. 921–928.
- [10] Q. Gu and J. Zhou, "Learning the shared subspace for multitask clustering and transductive transfer classification," in *ICDM*, 2009, pp. 159–168.
- [11] C.W. Hsu and C.J. Lin, "A comparison of methods for multiclass support vector machines," in *IEEE Transactions on Neural Networks*, 2002, vol. 13, pp. 415–425.
- [12] J.C. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin DAGs for multiclass classification," in *NIPS*, 2000, vol. 12, pp. 547–553.
- [13] V. Franc and V. Hlavác, "Multi-class support vector machine," in *ICPR*, 2002, vol. 2, pp. 236–239.
- [14] Ben-David and R. Schuller, "Exploiting task relatedness for multiple task learning," in *COLT*, 2003.
- [15] I. Steinwart, "Support vector machines are universally consistent," *Journal of Complexity*, vol. 18, no. 3, pp. 768–791, 2002.
- [16] A. Asuncion and D.J. Newman, "UCI machine learning repository," 2007, Datasets available at <http://archive.ics.uci.edu/ml/datasets.html>.
- [17] C.W. Hsu, C.C. Chang, C.J. Lin, et al., "A practical guide to support vector classification," 2003.