Fast Training of Structured SVM Using Fixed-Threshold Sequential Minimal Optimization

Changki Lee and Myung-Gil Jang

In this paper, we describe a fixed-threshold sequential minimal optimization (FSMO) for structured SVM problems. FSMO is conceptually simple, easy to implement, and faster than the standard support vector machine (SVM) training algorithms for structured SVM problems. Because FSMO uses the fact that the formulation of structured SVM has no bias (that is, the threshold *b* is fixed at zero), FSMO breaks down the quadratic programming (QP) problems of structured SVM into a series of smallest QP problems, each involving only one variable. By involving only one variable, FSMO is advantageous in that each QP sub-problem does not need subset selection. For the various test sets, FSMO is as accurate as an existing structured SVM implementation (SVM-Struct) but is much faster on large data sets. The training time of FSMO empirically scales between O(*n*) and O($n^{1.2}$), while SVM-Struct scales between O($n^{1.5}$) and O($n^{1.8}$).

Keywords: Support vector machines, structured SVM, fixed-threshold sequential minimal optimization.

I. Introduction

In recent years, there has been a lot of interest in support vector machines (SVMs) [1]-[6]. SVMs have empirically been shown to give good generalization performance on a wide variety of problems such as text categorization [6], spam filtering [7], taxonomic text classification [4], learning to rank [8], image retrieval [9], and spoken language understanding [10].

The formulation of SVM is based on a two-class problem; hence, SVM is basically a binary classifier. Taskar and others presented a discriminative approach to parsing inspired by the large-margin criterion underlying SVMs in which the loss function is factorized analogous to the decoding process [11]. Tsochantaridis and others proposed large-margin models based on SVMs for the structured classification problem (structured SVM) in general and apply it for multiclass classification, the syntactic parsing problem, named entity recognition, taxonomic text classification, and sequence alignment [4]. Yue and others use structured SVM to globally optimize mean average precision (MAP) [8].

Chunking is the first decomposition method used in standard SVM training [1]. It starts with a random subset (chunk) of data which we define as *B* and trains an initial SVM. Support vectors in the chunk are retained while non-support vectors are replaced by patterns in *N* violating the Karush-Kuhn-Tucker (KKT) conditions. Then, the SVM is re-trained and the whole procedure is repeated. Chunking suffers from the problem that all of the support vectors that have been identified still need to be trained together at the end of the training process.

Osuna and others proposed another decomposition algorithm that fixes the size of working set *B* [2]. At each iteration, variables corresponding to patterns in *N* are frozen, while those

Manuscript received May 13, 2008; revised Feb. 16, 2009; accepted Feb. 18, 2009.

Changki Lee (phone: +82 42 860 6879, email: leeck@etri.re.kr) and Myung-Gil Jang (email: mgjang@etri.re.kr) are with Software & Content Research Laboratory, ETRI, Daejeon, Rep. of Korea.

in B are optimized in a quadratic programming (QP) sub-problem. After that, a new point in N violating the KKT conditions will replace some point in B. The SVM-light software [6] follows the same scheme, though with a slightly different subset selection heuristic.

Platt introduced the sequential minimal optimization (SMO) algorithm which breaks the original large QP into a series of smallest possible QPs, each involving only two variables [3]. The first variable is chosen among points that violate the KKT conditions, while the second variable is chosen so as to have a large increase in the dual objective. By involving only two variables, SMO is advantageous in that each QP sub-problem can be solved analytically in an efficient way, without the use of a numerical QP solver. In addition, SMO requires no extra matrix storage at all. Platt also introduced fixed-threshold SVM, but did not apply it to structured SVM.

The idea of core vector machines (CVM) was proposed, in which the two-category classification problem was formulated as an approximate minimum enclosing ball (MEB) problem in computational geometry [12]. The resulting algorithm is very fast and is especially useful for very large datasets. However, the algorithm is an approximation for SVM training that has an approximation ratio of $(1+e)^2$.

Tsochantaridis and others proposed large-margin models based on SVMs for the structured classification problem (structured SVM) in general and applied it to the syntactic parsing problem, named entity recognition, taxonomic text classification, and sequence alignment [4]. However, they used a standard SVM solver (SVM-light) to solve the dual form of structured SVM, despite the fact that structured SVM has no bias (that is, the threshold b is fixed at zero).

Recently, Joachims proposed a joint constraint algorithm for linear SVMs which trains in linear time [13]. It is based on an alternative formulation of the SVM optimization problem that exhibits a different form of sparsity compared to the conventional formulation. However, we do not explore it here because this method has a high constant.

In this paper, we describe a fast training algorithm of structured SVM called *fixed-threshold sequential minimal optimization* (FSMO). FSMO is conceptually simple, easy to implement, and faster than the standard SVM training algorithms for structured SVM problems. FSMO uses the fact that the formulation of structured SVM has no bias, that is, that the threshold b is fixed at zero. Therefore, FSMO breaks the QPs of structured SVM into a series of smallest QPs, each involving only one variable. By involving only one variable, FSMO is advantageous in that each QP sub-problem does not need subset selection.

The rest of this paper is organized as follows. Section II describes structured SVM. Section III describes our proposed FSMO algorithm for structural SVM. Section IV gives application and experimental results. The final section gives some concluding remarks.

II. Structured SVM

The formulation of standard SVM is based on a two-class problem; hence, SVM is basically a binary classifier. Tsochantaridis and others proposed large-margin models based on SVMs for the structured classification problem (structured SVM) [4]. In this section, we briefly describe structured SVM.

1. Support Vector Machines

Vapnik invented support vector machines. In its simplest, linear form, an SVM is a hyperplane that separates a set of positive examples from a set of negative examples with maximum margin. Margin maximization can be expressed as given in [1] as

$$\begin{aligned} &\min_{w,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_i^n \xi_i, \\ &\text{s.t. } \forall i,\ \ \xi_i \ge 0, \qquad \forall i,\ \ y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \ge 1-\xi_i. \end{aligned} \tag{1}$$

Using a Lagrangian multiplier, this optimization problem can be converted into a dual form which is a QP problem, where the objective function L_1 is solely dependent on a set of Lagrangian multipliers α :

$$\max_\alpha L_1(\alpha) = \sum_i^n \alpha_i - \frac{1}{2}\sum_i^n\sum_j^n \alpha_i\alpha_j\mathbf{x}_i\mathbf{x}_j, \tag{2}$$

subject to the inequality constraints,

$$\forall i,\ 0 \le \alpha_i \le \frac{C}{n}, \tag{3}$$

and one linear equality constraint,

$$\sum_i^n y_i\alpha_i = 0. \tag{4}$$

There is a one-to-one relationship between each Lagrangian multiplier and each training example. Once the Lagrangian multipliers are determined, the normal vector $\mathbf{w}$ and the threshold b can be derived from the Lagrangian multipliers:

$$\mathbf{w} = \sum_i^n y_i\alpha_i\mathbf{x}_i,\ \ b = -\mathbf{w}\cdot\mathbf{x}_k + y_k \quad \text{for some } \alpha_k > 0. \tag{5}$$

2. Structured SVM

Structured classification is the problem of predicting **y** from **x** when **y** has a meaningful internal structure. Elements $\mathbf{y} \in \mathbf{Y}$ may be, for instance, sequences, strings, labeled trees, lattices, or graphs. The approach we pursue is to learn a discriminant function *F*:**X**×**Y** → *R* over *<input, output>* pairs from which we can derive a prediction by maximizing *F* over the response variable for a specific given input *x*. Hence, the general form of our hypotheses *f* is

$$f(\mathbf{x};\mathbf{w}) = \arg\max_{\mathbf{y}\in\mathbf{Y}} F(\mathbf{x},\mathbf{y};\mathbf{w}),$$

where **w** denotes a parameter vector.

As the principle of the maximum-margin presented in [1], in the structured classification problem, Tsochantaridis and others proposed several maximum-margin optimization problems [4]. For convenience, we define

$$\delta\Psi_i(\mathbf{x}_i,\mathbf{y}) \equiv \Psi(\mathbf{x}_i,\mathbf{y}_i) - \Psi(\mathbf{x}_i,\mathbf{y}),$$

where ($\mathbf{x}_i$, $\mathbf{y}_i$) is the training data.

The hard-margin optimization problem can be written as

$$\begin{aligned} &SVM_0 : \min_w \frac{1}{2}\|\mathbf{w}\|^2, \\ &\qquad \text{s.t. } \forall i, \forall \mathbf{y} \in \mathbf{Y}\backslash\mathbf{y}_i, \quad \mathbf{w}\cdot\delta\Psi_i(\mathbf{x}_i,\mathbf{y}) > 1. \end{aligned} \tag{6}$$

The soft-margin criterion was proposed to allow errors in the training set by introducing slack variables [4].

$$\begin{aligned} &SVM_1 : \min_{w,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_i^n \xi_i, \\ &\qquad \text{s.t. } \forall i, \xi_i \ge 0, \\ &\qquad\qquad \forall i, \forall \mathbf{y} \in \mathbf{Y}\backslash\mathbf{y}_i, \quad \mathbf{w}\cdot\delta\Psi_i(\mathbf{x}_i,\mathbf{y}) \ge 1-\xi_i. \end{aligned} \tag{7}$$

Alternatively, using a quadratic term $\frac{C}{2n}\sum_i \xi_i^2$ to penalize margin violations, we obtained SVM$_2$ [4]. Here, $C > 0$ is a constant that controls the tradeoff between training error minimization and margin maximization.

To deal with problems in which |**Y**| is very large, such as semantic parsing, Tsochantaridis and others proposed two approaches that generalize the formulation SVM$_0$ and SVM$_1$ to the cases of arbitrary loss function [4]. The first approach is to re-scale the slack variables according to the loss incurred in each of the linear constraints:

$$\begin{aligned} &SVM^{\Delta s} : \min_{w,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_i^n \xi_i, \\ &\qquad \text{s.t. } \forall i, \xi_i \ge 0, \\ &\qquad\qquad \forall i, \forall \mathbf{y} \in \mathbf{Y}\backslash\mathbf{y}_i, \quad \mathbf{w}\cdot\delta\Psi_i(\mathbf{x}_i,\mathbf{y}) \ge 1-\frac{\xi_i}{\Delta(\mathbf{y}_i,\mathbf{y})}. \end{aligned} \tag{8}$$

The second approach to include loss function is to re-scale the margin as a special case of Hamming loss. The margin constraints in this setting take the following form:

$$\begin{aligned} &SVM^{\Delta m} : \min_{w,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_i^n \xi_i, \\ &\qquad \text{s.t. } \forall i, \xi_i \ge 0, \\ &\qquad\qquad \forall i, \forall \mathbf{y} \in \mathbf{Y}\backslash\mathbf{y}_i, \quad \mathbf{w}\cdot\delta\Psi_i(\mathbf{x}_i,\mathbf{y}) \ge \Delta(\mathbf{y}_i,\mathbf{y}) - \xi_i. \end{aligned} \tag{9}$$

Algorithm 1. Cutting plane algorithm for solving structured SVM within tolerance ε [4].

```
1: Input: $(\mathbf{x}_1,\mathbf{y}_1),\cdots,(\mathbf{x}_n,\mathbf{y}_n)$, $C$, $\varepsilon$
2: $S_i \leftarrow \phi$ for all $i = 1,\cdots,n$
3: repeat
4:   for $i = 1,\cdots,n$ do
5:     set up cost function
         $SVM_1^{\Delta s} : H(\mathbf{y}) \equiv (1-\mathbf{w}\cdot\delta\Psi_i(\mathbf{x}_i,\mathbf{y}))\Delta(\mathbf{y}_i,\mathbf{y})$
         $SVM_2^{\Delta s} : H(\mathbf{y}) \equiv (1-\mathbf{w}\cdot\delta\Psi_i(\mathbf{x}_i,\mathbf{y}))\sqrt{\Delta(\mathbf{y}_i,\mathbf{y})}$
         $SVM_1^{\Delta m} : H(\mathbf{y}) \equiv (\Delta(\mathbf{y}_i,\mathbf{y}) - \mathbf{w}\cdot\delta\Psi_i(\mathbf{x}_i,\mathbf{y}))$
         $SVM_2^{\Delta m} : H(\mathbf{y}) \equiv (\sqrt{\Delta(\mathbf{y}_i,\mathbf{y})} - \mathbf{w}\cdot\delta\Psi_i(\mathbf{x}_i,\mathbf{y}))$
           where $\mathbf{w} = \sum_{i,\mathbf{y}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}}\delta\Psi_i(\mathbf{x}_i,\mathbf{y})$
6:     compute $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathbf{Y}} H(\mathbf{y})$
7:     compute $\xi_i = \max\{0, \max_{\mathbf{y}\in S_i} H(\mathbf{y})\}$
8:     if $H(\hat{\mathbf{y}}) > \xi_i + \varepsilon$ then
9:       $S_i \leftarrow S_i \cup \{\hat{\mathbf{y}}\}$
10:      $\alpha_S \leftarrow$ optimize dual over $S = \cup_i S_i$
11:    end if
12:  end for
13: until no $S_i$ changes during iteration
```

3. SVM Learning

The support vector learning algorithm aims to find a small set of active constraints that ensures a sufficiently accurate solution.

The pseudocode of the algorithm is given in algorithm 1 [4]. The algorithm applies to all the SVM formulations previously discussed. The only difference is in the way the cost function is set up in step 5. The algorithm maintains a working set S_i for each training example ($\mathbf{x}_i$, $\mathbf{y}_i$) to keep track of the selected constraints which define the current relaxation. Iterating through the training examples ($\mathbf{x}_i$, $\mathbf{y}_i$), the algorithm proceeds by finding the (potentially) "most violated" constraint, involving some output value $\hat{\mathbf{y}}$ (line 6). If the (appropriately scaled) margin violation of this constraint exceeds the current value of ξ_i by more than *e* (line 8), the dual variable corresponding to $\hat{\mathbf{y}}$ is added to the working set (line 9). Once a constraint has been added, the solution is recomputed with regard to *S* by SVM-light (line 10) [4]. The algorithm stops if no constraint is violated by more than *e*.

III. Fixed-Threshold SMO for Structured SVM

In this section, we describe the FSMO algorithm for solving structured SVM. Instead of SVM-light, FSMO is used to solve the dual problem of structured SVM in the cutting plane algorithm (line 10 in algorithm 1).

We can solve the optimization problem of structured SVM presented in (6) through (9) with Lagrangian multipliers. We only describe the case of margin re-scaling due to space limitation:

$$\begin{aligned}\min_{\mathbf{w},\xi} L_2(\mathbf{w},\xi)=&\frac{1}{2}\|\mathbf{w}\|^2+\frac{C}{n}\sum_{i}^{n}\xi_i\\&+\sum_{i,\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}}(\Delta(\mathbf{y}_i,\mathbf{y})-\mathbf{w}\cdot\delta\Psi_i(\mathbf{x}_i,\mathbf{y})-\xi_i)\\&-\sum_{i}^{n}\beta_i\xi_i\\&\text{s.t. }\ \alpha_{i\mathbf{y}}\geq 0,\quad \beta_i\geq 0,\end{aligned}\qquad(10)$$

$$\frac{\partial L_2}{\partial \mathbf{w}}=\mathbf{w}-\sum_{i,\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}}\delta\Psi_i(\mathbf{x}_i,\mathbf{y})=0,\qquad(11)$$

$$\frac{\partial L_2}{\partial \xi_i}=\frac{C}{n}-\sum_{\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}}-\beta_i=0.\qquad(12)$$

$$\left(\therefore\frac{C}{n}=\sum_{\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}}+\beta_i,\quad 0\leq\sum_{\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}}\leq\frac{C}{n}\right).$$

From (10) through (12), we obtain the following dual form which is a QP problem where the objective function L_3 is solely dependent on a set of Lagrangian multipliers α:

$$\begin{aligned}\max_{\alpha} L_3(a)=&\sum_{i,\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}}\Delta(\mathbf{y}_i,\mathbf{y})\\&-\frac{1}{2}\sum_{i,\mathbf{y}\neq\mathbf{y}_i}\sum_{j,\bar{\mathbf{y}}\neq\mathbf{y}_j}\alpha_{i\mathbf{y}}\alpha_{j\bar{\mathbf{y}}}\delta\Psi_i(\mathbf{x}_i,\mathbf{y})\cdot\delta\Psi_j(\mathbf{x}_j,\bar{\mathbf{y}}),\\&\text{s.t. }\forall i,\ \ 0\leq\sum_{\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}}\leq\frac{C}{n},\quad\forall i\mathbf{y},\ \alpha_{i\mathbf{y}}>0.\end{aligned}\qquad(13)$$

The extremum of the object function L_3 is at

$$\frac{\partial L_3(\alpha)}{\partial\alpha_{i\mathbf{y}}}=\Delta(\mathbf{y}_i,\mathbf{y})-\mathbf{w}\cdot\delta\Psi_i(\mathbf{x}_i,\mathbf{y})=0.\qquad(14)$$

Algorithm 2. FSMO algorithm for solving dual form of structured SVM.

```
Input: $(\mathbf{x}_1,\mathbf{y}_1),\cdots,(\mathbf{x}_n,\mathbf{y}_n)$, $S$, $\alpha_S$, $C$
$\mathbf{w}=\sum_{i,\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}}\delta\Psi_i(\mathbf{x}_i,\mathbf{y})$
repeat
    for $(\mathbf{x}_i,\hat{\mathbf{y}})$ in $S$ do
        if $(\mathbf{x}_i,\hat{\mathbf{y}})$ violates the KKT condition do
            update $\alpha_{i\hat{\mathbf{y}}}$:
                $SVM_1^{\Delta m}:\alpha_{i\hat{\mathbf{y}}}\leftarrow\alpha_{i\hat{\mathbf{y}}}^{old}+\frac{\Delta(\mathbf{y}_i,\hat{\mathbf{y}})-\mathbf{w}\cdot\delta\Psi_i(\mathbf{x}_i,\hat{\mathbf{y}})}{\|\delta\Psi_i(\mathbf{x}_i,\hat{\mathbf{y}})\|^2}$,
                    $s.t.\ 0\leq\sum_{\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}}\leq\frac{C}{n}$
                $SVM_1^{\Delta s}:\alpha_{i\hat{\mathbf{y}}}\leftarrow\alpha_{i\hat{\mathbf{y}}}^{old}+\frac{1-\mathbf{w}\cdot\delta\Psi_i(\mathbf{x}_i,\hat{\mathbf{y}})}{\|\delta\Psi_i(\mathbf{x}_i,\hat{\mathbf{y}})\|^2}$,
                    $s.t.\ 0\leq\sum_{\mathbf{y}\neq\mathbf{y}_i}\frac{\alpha_{i\mathbf{y}}}{\Delta(\mathbf{y}_i,\hat{\mathbf{y}})}\leq\frac{C}{n}$
            $\mathbf{w}\leftarrow\mathbf{w}^{old}+(\alpha_{i\hat{\mathbf{y}}}-\alpha_{i\hat{\mathbf{y}}}^{old})\delta\Psi_i(\mathbf{x}_i,\hat{\mathbf{y}})$
        end if
    end for
until no $\alpha_{i\hat{\mathbf{y}}}$ changes during iteration
```

Let $\alpha_{i\mathbf{y}}^{new}=\alpha_{i\mathbf{y}}+s$ and $\mathbf{w}^{new}=\mathbf{w}+s\delta\Psi_i(\mathbf{x}_i,\mathbf{y})$. Then we can get the following update equation from (14).

$$\begin{aligned}\mathbf{w}^{new}\cdot\delta\Psi_i(\mathbf{x}_i,\mathbf{y})&=\mathbf{w}\cdot\delta\Psi_i(\mathbf{x}_i,\mathbf{y})+s\|\delta\Psi_i(\mathbf{x}_i,\mathbf{y})\|^2\\&=\Delta(\mathbf{y}_i,\mathbf{y}),\\s&=\frac{\Delta(\mathbf{y}_i,\mathbf{y})-\mathbf{w}\cdot\delta\Psi_i(\mathbf{x}_i,\mathbf{y})}{\|\delta\Psi_i(\mathbf{x}_i,\mathbf{y})\|^2}.\\\therefore\alpha_{i\mathbf{y}}^{new}&=\alpha_{i\mathbf{y}}+\frac{\Delta(\mathbf{y}_i,\mathbf{y})-\mathbf{w}\cdot\delta\Psi_i(\mathbf{x}_i,\mathbf{y})}{\|\delta\Psi_i(\mathbf{x}_i,\mathbf{y})\|^2}.\end{aligned}\qquad(15)$$

For the optimization with slack re-scaling, the loss function affects the linear part of the objective function and inequality constrains (13) as follows:

$$\begin{aligned}\max_{\alpha} L_3(a)=&\sum_{i,\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}}\\&-\frac{1}{2}\sum_{i,\mathbf{y}\neq\mathbf{y}_i}\sum_{j,\bar{\mathbf{y}}\neq\mathbf{y}_j}\alpha_{i\mathbf{y}}\alpha_{j\bar{\mathbf{y}}}\delta\Psi_i(\mathbf{x}_i,\mathbf{y})\cdot\delta\Psi_j(\mathbf{x}_j,\bar{\mathbf{y}}),\\&\text{s.t. }\forall i,\ \ 0\leq\sum_{\mathbf{y}\neq\mathbf{y}_i}\frac{\alpha_{i\mathbf{y}}}{\Delta(\mathbf{y}_i,\mathbf{y})}\leq\frac{C}{n},\ \forall i\mathbf{y},\ \alpha_{i\mathbf{y}}>0.\end{aligned}\qquad(16)$$

Then we can calculate the update equation for slack re-scaling from (16) as

$$\alpha_{i\mathbf{y}}^{new} = \alpha_{i\mathbf{y}} + \frac{1-\mathbf{w}\cdot\delta\Psi_i(\mathbf{x}_i,\mathbf{y})}{\left\|\delta\Psi_i(\mathbf{x}_i,\mathbf{y})\right\|^2}. \tag{17}$$

Quadratic slack penalties (SVM$_2$) can be applied, but we skip them due to the space limitation.

The update equation forces the output of the structured SVM to be $\Delta(\mathbf{y}_i,\mathbf{y})$ and 1 in the cases of margin re-scaling and slack re-scaling, respectively. After the new α is computed, it is clipped to the [0,*C*/*n*] interval in margin re-scaling and to the [0,1] interval in slack re-scaling.

Because structured SVM has no bias in (6) through (9), (13) and (16) do not have the linear equality constraint (4) of standard SVM. Therefore, FSMO can optimize only one Lagrange multiplier at a time.

The pseudocode of FSMO is given in algorithm 2. The algorithm applies to the slack re-scaling and margin re-scaling formulations previously discussed. The only difference is the update equation in step 6. Iterating through the training examples $(\mathbf{x}_i,\hat{\mathbf{y}})$ in working set *S* (line 4), the algorithm proceeds by finding the constraint which violates the KKT conditions [5]. Equations (18) and (19) are the KKT conditions for the QP problems (13) and (16). The QP problems are solved when, for all *i*,

$$\begin{aligned} SVM_1^{\Delta m}: \alpha_{i\mathbf{y}} &= 0 \Leftrightarrow u_{i\mathbf{y}} \geq \Delta(\mathbf{y}_i,\mathbf{y}),\\ 0 < \sum_{\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}} &< \frac{C}{n} \Leftrightarrow u_{i\mathbf{y}} = \Delta(\mathbf{y}_i,\mathbf{y}),\\ \sum_{\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}} &= \frac{C}{n} \Leftrightarrow u_{i\mathbf{y}} \leq \Delta(\mathbf{y}_i,\mathbf{y}). \end{aligned} \tag{18}$$

$$\begin{aligned} SVM_1^{\Delta s}: \alpha_{i\mathbf{y}} &= 0 \Leftrightarrow u_{i\mathbf{y}} \geq 1,\\ 0 < \sum_{\mathbf{y}\neq\mathbf{y}_i}\frac{\alpha_{i\mathbf{y}}}{\Delta(\mathbf{y}_i,\mathbf{y})} &< \frac{C}{n} \Leftrightarrow u_{i\mathbf{y}} = 1,\\ \sum_{\mathbf{y}\neq\mathbf{y}_i}\frac{\alpha_{i\mathbf{y}}}{\Delta(\mathbf{y}_i,\mathbf{y})} &= \frac{C}{n} \Leftrightarrow u_{i\mathbf{y}} \leq 1, \end{aligned} \tag{19}$$

where $u_{i\mathbf{y}} = \mathbf{w}\cdot\delta\Psi_i(\mathbf{x}_i,\mathbf{y})$.

Note that due to the KKT conditions, it is not necessary to re-train on well classified examples that are outside the margins [5]. If the constraint does not satisfy the KKT condition (line 5), the $\alpha_{i\hat{\mathbf{y}}}$ variable corresponding to $(\mathbf{x}_i,\hat{\mathbf{y}})$ and **w** are updated by using update (15) and (17) (lines 6, 7). The algorithm stops if no $\alpha_{i\hat{\mathbf{y}}}$ changes during iteration.

The FSMO algorithm is used to solve the dual problem of structured SVM in the cutting plane algorithm (line 10 in algorithm 1).

IV. Application and Experiments

We implemented structured SVM using FSMO (in C++) to solve QP problems. For comparison, we run the SVM-Struct that uses SVM-light for solving QP problems [4], maximum entropy (ME), and conditional random fields (CRF) [14]. We also run the LIBSVM which uses the SMO method [15]. For FSMO, SVM-Struct, and LIBSVM, a linear kernel is used. Table 1 summarizes the characteristics of the data sets used.

1. Multiclass Classification

We implemented the conventional winner-takes-all (WTA) multiclass classification [16] as follows. Let $\mathbf{Y} = \{\mathbf{y}_1,\cdots,\mathbf{y}_k\}$ and $\mathbf{w} = (\mathbf{w}_1,\cdots,\mathbf{w}_k)$ be a stack of vectors, $\mathbf{w}_k$ being a weight vector associated with the *k*-th class $\mathbf{y}_k$. Then we define $F(\mathbf{x},\mathbf{y}_k;\mathbf{w}) = \mathbf{w}_k\cdot\Phi(\mathbf{x})$, where $\Phi(\mathbf{x})\in R^D$ denotes an arbitrary input representation. We define the multiclass SVM as

$$\begin{aligned} &SVM^{muticlass}: \min_{w,\xi}\frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_i^n \xi_i,\ s.t.\ \forall i, \xi_i \geq 0,\\ &\forall i, \mathbf{w}_{\mathbf{y}_i}\cdot\Phi(\mathbf{x}_i) \geq \mathbf{w}_{\mathbf{y}}\cdot\Phi(\mathbf{x}_i) + \Delta(\mathbf{y}_i,\mathbf{y}) - \xi_i, \end{aligned}$$

where $\mathbf{y} = \arg\max_{\mathbf{y}\in\mathbf{Y}/\mathbf{y}_i}\{\mathbf{w}_{\mathbf{y}}\cdot\Phi(\mathbf{x}_i)\}$.

Figure 1 shows log-log plots of how training time increases with the size of the training set on the MNIST data set. We use *C*=2000 and *e*=0.01 for FSMO and SVM-Struct, *C*=2000/*n* (*n* is a training set size) for LIBSVM, and *g*=1 (*g* is a Gaussian priority) for ME. FSMO is substantially faster than SVM-Struct and LIBSVM in most cases. LIBSVM is faster than FSMO in the small training set size of 100, but it is slower than other methods in middle and large training set sizes. FSMO is 34 times faster than SVM-Struct and 42 times faster than LIBSVM, when the training set size is 60,000. The FSMO training time scales as O(*n*), while SVM-Struct and LIBSVM scale as O($n^{1.6}$) and O($n^{1.9}$), respectively.

Figure 2 shows the training times on the news20 data set (we use *C*=100 and *e*=0.01 for FSMO and SVM-Struct, *C*=100/*n*

Table 1. Data sets used in the experiments.

Data set	Task	Size	# class	# attribute
MNIST	Multi-classification	60,000	10	780
News20	Multi-classification	10,000	20	62,061
Rcv1-binary	Multi-classification	80,000	2	47,236
English chunking	Sequence labeling	100,000	22	387,875
Korean spacing	Sequence labeling	1,000,000	2	228,260

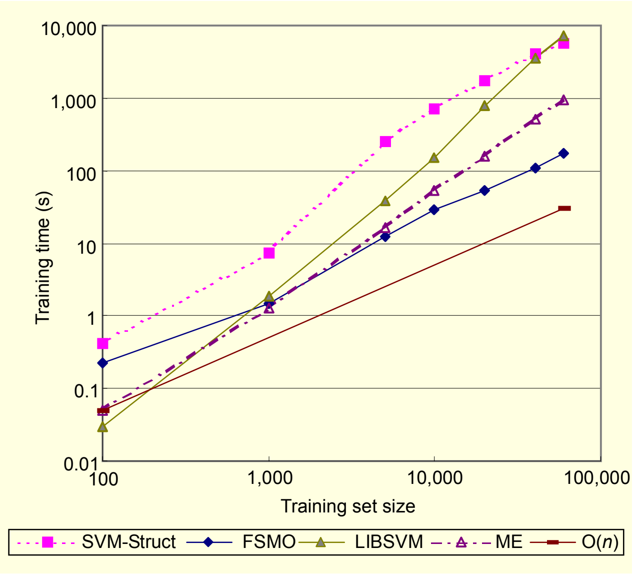

Fig. 1. Training times of the MNIST data set.

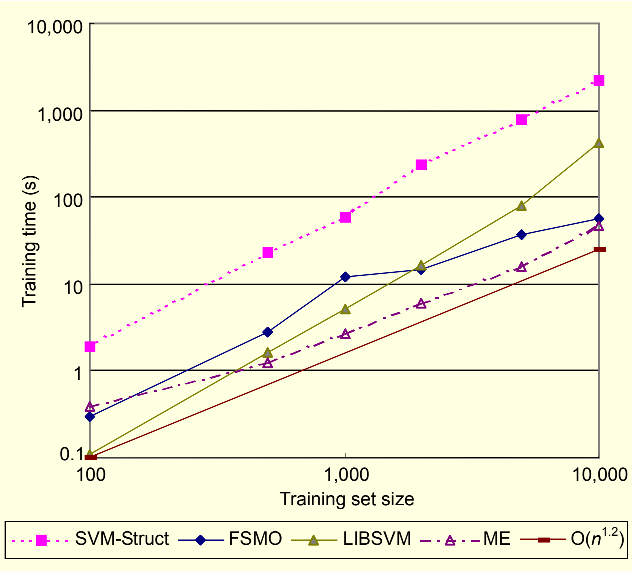

Fig. 2. Training times of the news20 data set.

for LIBSVM, and g=0.02 for ME). FSMO is about 40 times faster than SVM-Struct and 7 times faster than LIBSVM, when the training set size is 10,000. The FSMO training time scales as $O(n^{1.2})$, while SVM-Struct and LIBSVM scale as $O(n^{1.6})$ and $O(n^{1.7})$, respectively.

Figure 3 shows the training time on the Rcv1-binary data set (we use C=100 and e=0.01 for FSMO and SVM-Struct, C=100/n for LIBSVM, and g=1 for ME). FSMO is about 13 times faster than SVM-Struct and 80 times faster than LIBSVM, when the training set size is 80,000. The FSMO training time scales as $O(n)$, while SVM-Struct and LIBSVM scale as $O(n^{1.5})$ and $O(n^{1.6})$, respectively.

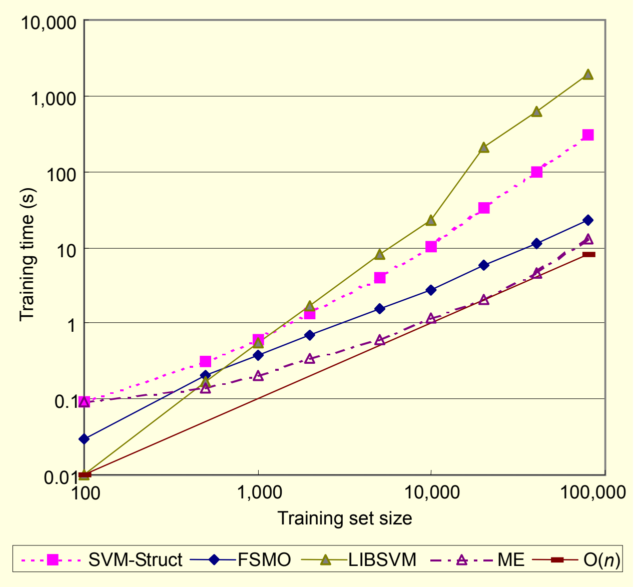

Fig. 3. Training times of the Rcv1-binary data set.

Table 2. Performance (F1) on the data sets.

Algorithm	MNIST	News20	Rcv1-binary
FSMO	92.59	82.82	96.42
SVM-Struct	92.68	82.90	96.41
LIBSVM	91.61	83.12	96.50
ME	92.30	82.24	96.38

In multiclass classification, FSMO has much better scaling than SVM-Struct and LIBSVM. One potential worry is that the speedup of FSMO over SVM-Struct could come at the expense of prediction accuracy; however this is not the case. Table 2 shows the performance (F1) of FSMO, SVM-Struct, LIBSVM, and ME on the data sets. FSMO and SVM-Struct show very similar performance. The differences between them are not statistically significant.

2. Label Sequence Learning

Label sequence learning deals with the problem of predicting a sequence of labels, $\mathbf{y} = (\mathbf{y}^1, \cdots, \mathbf{y}^m), \mathbf{y}^k \in \Sigma$, from a given sequence of inputs, $\mathbf{x} = (\mathbf{x}^1, \cdots, \mathbf{x}^m)$ and $\mathbf{w}$ = ($\mathbf{w}_1$,⋯, $\mathbf{w}_k$), where $\mathbf{w}_k$ is a weight vector associated with $\mathbf{y}_k$. It subsumes problems like segmenting or annotating observation sequences and has widespread applications in optical character recognition, natural language processing, information extraction, and computational biology. In the setup followed in [17], the joint feature map $\Psi(\mathbf{x}, \mathbf{y})$ is the histogram of state transition plus a set of features describing the emissions. An adapted version of the Viterbi algorithm is used to solve the argmax in line 6 of algorithm 1. We define structured SVM for sequence tagging as

$$SVM^{hmm}: \min_{w,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i}^{n}\xi_i,$$

$$\text{s.t. } \forall i,\ \xi_i \geq 0,$$

$$\forall i, \sum_j \mathbf{w}_{\mathbf{y}_i^j} \cdot \Phi(\mathbf{x}_i^j) + \mathbf{w}_{trans} \cdot \varphi_{trans}(\mathbf{y}_i^{j-1}, \mathbf{y}_i^j)$$

$$\geq \sum_j \mathbf{w}_{\mathbf{y}_i^j} \cdot \Phi(\mathbf{x}_i^j) + \mathbf{w}_{trans} \cdot \varphi_{trans}(\mathbf{y}^{j-1}, \mathbf{y}^j)$$

$$+ \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i,$$

where

$$\mathbf{y} = \arg\max_{\mathbf{y}\in\mathbf{Y}/\mathbf{y}_i} \{\sum_j \mathbf{w}_{\mathbf{y}^j} \cdot \Phi(\mathbf{x}_i^j) + \mathbf{w}_{trans} \cdot \varphi_{trans}(\mathbf{y}^{j-1}, \mathbf{y}^j)\}.$$

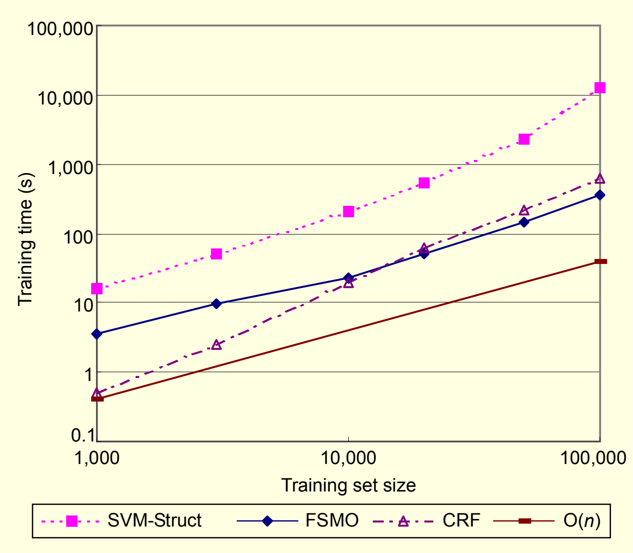

Fig. 4. Training time of the English chunking data set.

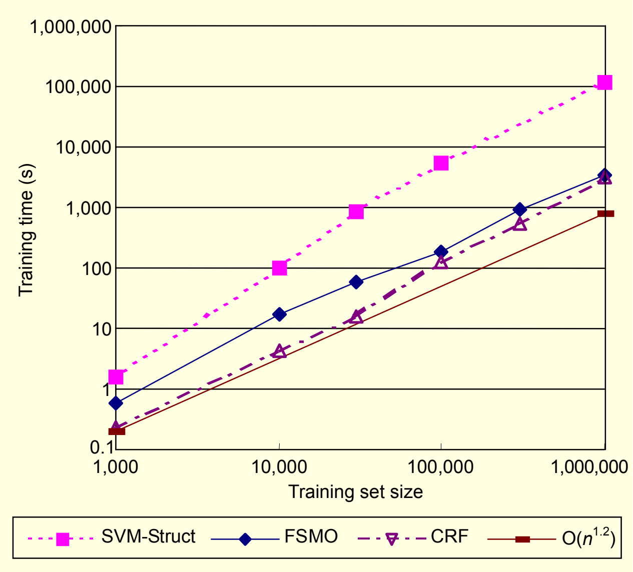

Fig. 5. Training time of the Korean spacing data set.

Table 3. Performance (F1) on the data sets.

Algorithm	English chunking (%)	Korean spacing (%)
FSMO	92.76	97.03
SVM-Struct	92.76	97.04
CRF	92.79	96.91

Figure 4 shows the training time on the English chunking data set (we use *C*=1000 and *e*=0.01 for FSMO and SVM-Struct, and *g*=1 for CRF). FSMO is substantially faster than SVM-Struct on all training set sizes. FSMO is 72 times faster than SVM-Struct and similar to CRF when the training set size is 100,000. The FSMO training time scales as O(n), while SVM-Struct scales as O($n^{1.5}$).

Figure 5 shows the training time on the Korean spacing data set (we use *C*=1000 and *e*=0.1 for FSMO and SVM-Struct, and *g*=1 for CRF). FSMO is about 40 times faster than SVM-Struct and similar to CRF when the training set size is 1,000,000. The FSMO training time scales as O($n^{1.2}$), while SVM-Struct scales as O($n^{1.7}$).

In sequence tagging, FSMO also has much better scaling than SVM-Struct. Table 3 shows the performance (F1) of FSMO, SVM-Struct, and CRF on the data sets. The performance of FSMO, SVM-Struct, and CRF are very similar. The differences between FSMO and SVM-Struct are not statistically significant.

V. Conclusion

This paper presented FSMO for structured SVM problems. FSMO is simple and faster than the standard SVM training algorithms for structured SVM problems. For various test sets, FSMO is as accurate as an existing structured SVM implementation (SVM-Struct) but is much faster on large data sets. FSMO is 10 to 70 times faster than SVM-Struct and 7 to 80 times faster than LIVSVM which uses the SMO method. The training time of FSMO empirically scales between O(n) and O($n^{1.2}$), while SVM-Struct scales between O($n^{1.5}$) and O($n^{1.8}$).

References

[1] V. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.

[2] E. Osuna, R. Freund, and F. Girosi, “Training Support Vector Machines: An Application to Face Detection,” *Proc. CVPR*, 1997, pp. 130-136.

[3] J. Platt, “Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines,” Microsoft Research Technical Report MSR-TR-98-14, 1998.

[4] I. Tsochantaridis et al., "Support Vector Machine Learning for Interdependent and Structured Output Spaces," *Proc. ICML*, 2004, p. 104.

[5] B. Scholkopf and A. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, 2001.

[6] T. Joachims, "A Statistical Learning Model of Text Classification with Support Vector Machines," *Proc. SIGIR*, 2001, pp. 128-136.

[7] D. Sculley and G.M. Wachman, "Relaxed Online SVMs for Spam Filtering," *Proc. SIGIR*, 2007, pp. 415-422.

[8] Y. Yue et al., "A Support Vector Method for Optimizing Average Precision," *Proc. SIGIR*, 2007, pp. 271-278.

[9] D. Kim, J. Song, and B. Choi, "Support Vector Machine Learning for Region-Based Image Retrieval with Relevance Feedback," *ETRI J.*, vol. 29, no. 5, 2007, pp. 700-702.

[10] C. Lee et al., "A Multi-Strategic Concept-Spotting Approach for Robust Understanding of Spoken Korean," *ETRI J.*, vol. 29, no. 2, 2007, pp. 179-188.

[11] B. Taskar, C. Guestrin, and D. Koller, "Max Margin Markov Networks," *NIPS*, vol. 16, 2004.

[12] I.W. Tsang, J.T. Kwok, and P.M. Cheung, "Core Vector Machines: Fast SVM Training on Very Large Data Sets," *Journal of Machine Learning Research*, vol. 6, 2005, pp. 363-392.

[13] T. Joachims, "Training Linear SVMs in Linear Time," *Proc. KDD*, 2006.

[14] C. Lee et al., "Fine-Grained Named Entity Recognition Using Conditional Random Fields for Question Answering," *Proc. AIRS*, 2006, pp. 581-587.

[15] C. Chang and C. Lin, "Training *v*-Support Vector Classifiers," *Neural Computation*, vol. 13, no. 9, 2001, pp. 2119-2147.

[16] K. Crammer and Y. Singer, "On the Learnability and Design of Output Codes for Multiclass Problems," *Journal of Machine Learning*, vol. 47, 2004, pp. 201-233.

[17] Y. Altun, I. Tsochantaridis, and T. Hofmann, "Hidden Markov Support Vector Machines," *Proc. ICML*, 2003.

Changki Lee received the BS degree in computer science from KAIST, Korea, in 1999. He received the MS and PhD degrees in computer engineering from POSTECH, Korea, in 2001 and 2004, respectively. Since 2004, he has been with Electronics and Telecommunications Research Institute (ETRI), Korea, as a senior member of research staff. He has served as a reviewer for some international journals, such as *Information System*, *Information Processing and Management*, and *ETRI Journal*. His research interests are natural language processing, information retrieval, data mining, and machine learning.

Myung-Gil Jang received the BS and MS degrees in computer science and statistics from Pusan National University, Korea, in 1988 and 1990. He received the PhD degree in information science from Chungnam National University in 2002. He was with System Engineering Research Institute (SERI), Korea, from 1990 to 1997 as a researcher. Since 1998, he has been with Electronics and Telecommunications Research Institute (ETRI), Korea, as a Senior/Principle Member of Research Staff. His research interests are natural language processing, information retrieval, question answering, knowledge and dialogue processing, media retrieval/management, and semantic web.