Efficient Algorithms for Linear Summed Error Structural SVMs

P. Balamurugan
Computer Science and Automation
Indian Institute of Science
Bangalore, India.
Email: balamurugan@csa.iisc.ernet.in

Shirish Shevade
Computer Science and Automation
Indian Institute of Science
Bangalore, India.
Email: shirish@csa.iisc.ernet.in

T. Ravindra Babu
E-Com Research Lab
Education and Research, Infosys Ltd.,
Bangalore, India.
Email: Ravindrababu_T@infosys.com

***Abstract*—Structural Support Vector Machines (SSVMs) have become a popular tool in machine learning for predicting structured objects like parse trees, Part-of-Speech (POS) label sequences and image segments. Various efficient algorithmic techniques have been proposed for training SSVMs for large datasets. The typical SSVM formulation contains a regularizer term and a composite loss term. The loss term is usually composed of the Linear Maximum Error (LME) associated with the training examples. Other alternatives for the loss term are yet to be explored for SSVMs. We formulate a new SSVM with Linear Summed Error (LSE) loss term and propose efficient algorithms to train the new SSVM formulation using primal cutting-plane method and sequential dual coordinate descent method. Numerical experiments on benchmark datasets demonstrate that the sequential dual coordinate descent method is faster than the cutting-plane method and reaches the steady-state generalization performance faster. It is thus a useful alternative for training SSVMs when linear summed error is used.**

***Index Terms*—Structural SVMs, Dual method, Cutting-plane method.**

I. Introduction

The task of classifying structured inputs like images, sentences, etc., into structured outputs like image segments, parse trees and POS-tag sequences is called Structured Classification. Structured classification has recently gained wide popularity in the machine learning community. Structured classification comprises of the learning stage known as structured output learning and the inference stage known as structured prediction. Though prediction of certain specific output structures like finding cyclic permutation in route-prediction problems might itself be a very hard computational task, there are real-world examples like sequence labeling, parse tree classification and image segmentation, where prediction is tractable.

We introduce structured classification through a simple example, namely, the POS-tagging for an English sentence. The sentence forms the structured input and the POS-tagging forms the structured output. The space of structured inputs $\mathscr{X}$ for this example is the space of all English sentences. The space $\mathscr{Y}$ of structured outputs is the space of all possible POS-tagging. Hence every $\mathbf{x} \in \mathscr{X}$ is a sentence and is composed of multiple parts $\mathbf{x} = (x^1, x^2, \cdots, x^T)$. The corresponding structured output $\mathbf{y} \in \mathscr{Y}$ is the POS-tagging for the sentence $\mathbf{x}$. The structured output $\mathbf{y}$ could be represented as a collection of parts $(y^1, y^2, \cdots, y^T)$ corresponding to the parts of $\mathbf{x}$. Each $\{y^t\}_{t=1}^T$ is from some finite alphabet of size σ. The size $|\mathscr{Y}|$ of the output space $\mathscr{Y}$ is exponential in T for many real-world examples. The output $\mathbf{y} = (y^1, y^2, \cdots, y^T)$ is called *structured* because the prediction of each y^i cannot be done independent of other y^j, $j \neq i$; rather each component y^i is interdependent on one or more of the other components of $\mathbf{y}$ and the prediction has to be consistent with this interaction among the various components of $\mathbf{y}$.

Structured output learning involves learning a discriminant function $h : \mathscr{X} \times \mathscr{Y} \rightarrow \mathbb{R}$ which can take a linear form $h = \mathbf{w}^T f(\mathbf{x}, \mathbf{y})$, where $\mathbf{w}$ is a suitable parameter vector and $f(\mathbf{x}, \mathbf{y})$ is a task-dependent feature representation of the structured input $\mathbf{x}$ in accordance with the output $\mathbf{y}$. We could thus represent the parameterized discriminant function as $h(\mathbf{w}; \mathbf{x}, \mathbf{y})$. Hence the prediction task becomes

$$g(\mathbf{x}) = \arg\max_{\mathbf{y}} h(\mathbf{w}; \mathbf{x}, \mathbf{y}) = \arg\max_{\mathbf{y}} \mathbf{w}^T f(\mathbf{x}, \mathbf{y}) \qquad \text{(I.1)}$$

For the POS-tagging example, a suitable feature vector representation could be

$$f(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{T} \begin{pmatrix} x^t \otimes y^t \\ I(y^t = 1) I(y^{t-1} = 1) \\ I(y^t = 1) I(y^{t-1} = 2) \\ \vdots \\ I(y^t = \sigma) I(y^{t-1} = \sigma) \end{pmatrix} \qquad \text{(I.2)}$$

where $\otimes$ represents the positioning of feature vector x^t at the y^t-th position and $I(p) = 1$ if p is true and 0 otherwise. If we consider the sentence and its corresponding POS-tagging as a chain of nodes and edges, $x^t \otimes y^t$ in the feature-vector construction in (I.2) represents the node-features for node t and the other features correspond to the edge $(t-1, t)$. The construction of $f(\mathbf{x}, \mathbf{y})$ through (I.2) also assumes a dummy node y^0. Thus, for the structured classification of an English sentence into its corresponding POS-tagging, first a suitable parameter vector $\mathbf{w}$ is found by structured output learning and this parameter is then used in the prediction using (I.1).

The $\arg\max$ computation in (I.1) usually forms an intrinsic part of the learning algorithms[11][1]. This computation could be NP-hard for some real-world applications[6]. Even for tractable applications like sequence labeling and parse tree

classification, this computation could be expensive. Hence it plays an important role in the overall performance of any learning algorithm.

Several advances have been made in developing learning algorithms for structured classification. Hidden Markov Models (HMMs)[5], Max-Entropy Markov Models (MEMMs)[15] and Conditional Random Fields (CRFs)[12] are some of the early probabilistic discriminative models developed and studied for structured classification. Margin-based discriminative models like Max-Margin Markov Networks (M3Ns)[19] and Structural Support Vector Machines (SSVMs)[21][11] have also been developed. Various results[11][1] have indicated that margin based discriminative methods for structured classification outperform probabilistic discriminative methods in terms of training time and generalization performance.

In this work, we focus on Structural SVMs for structured classification. Various algorithms like cutting-plane method[11], sequential dual method (SDM)[1], bundle methods[20] and exponentiated gradient method[2] have been developed to solve structural SVMs. All these algorithms work on the standard (or an equivalent) SSVM formulation which optimizes the sum of a regularized parameter vector and the collective loss summed over the training examples. The loss typically used in this formulation is composed of Linear Max Error (LME) associated with each training example. A single LME term is associated with each training example and is shared across the constraints corresponding to this example. There are other alternatives to the loss term which include but are not limited to Quadratic Max Error (QME), Linear Summed Error (LSE) and Quadratic Summed Error (QSE)[7]. However, SSVM formulations involving these alternative loss terms have not yet been explored owing to the computational effort involved in solving such problems.

A. Contributions

We propose an alternative formulation for Structural SVMs with a composite loss term called the Linear Summed Error(LSE). We call this new formulation the LSE-SSVM. The resultant optimization problem is still convex with exponentially many constraints and contains exponentially many optimization variables compared to those in LME-SSVMs. To handle the exponentially many optimization variables in the primal problem, we propose an equivalent formulation of the primal problem with only a finite number of optimization variables. A primal cutting-plane method to solve the new equivalent primal formulation of LSE-SSVMs is proposed. We also devise a sequential dual coordinate descent method to solve the dual LSE-SSVM formulation. Numerical experiments were carried out to compare the proposed methods on multi-class and sequence labeling problems. These experiments demonstrate that the sequential dual coordinate descent method outperforms the cutting-plane method in terms of training time. Also, it achieves steady-state generalization performance faster than the cutting-plane method.

The rest of the paper is organized as follows. The next section describes Structural SVMs and related formulations and discusses various algorithms to solve these formulations. In Section III, we introduce Linear Summed Error Structural SVMs and illustrate a primal cutting-plane method to solve the formulation. We discuss the dual formulation of LSE-SSVMs and give a sequential dual coordinate descent algorithm to solve dual LSE-SSVMs in Section IV. In Section V, details of empirical results of the proposed algorithms through various experiments conducted on a variety of datasets are presented. Section VI concludes the paper.

II. Structural SVMs

Given the training data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ of structured inputs and outputs, structural SVMs[19][21] solve a convex Quadratic Program (QP), which is (**OP1**):

$$\begin{aligned} \min_{\mathbf{w},\xi_i} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \\ \text{s.t.} \quad & \xi_i = \max_{\mathbf{y}} \left\{\max\left(0, l_i(\mathbf{y}) - \mathbf{w}^T\Delta f_i(\mathbf{y})\right)\right\} \quad \forall\, i \end{aligned} \tag{II.1}$$

where $C > 0$ is a regularization parameter and $\Delta f_i(\mathbf{y}) = f(\mathbf{x}_i, \mathbf{y}_i) - f(\mathbf{x}_i, \mathbf{y})$. The term $l_i(\mathbf{y})$ is the decomposable Hamming-loss function

$$l_i(\mathbf{y}) = \sum_{t=1}^{T} I(y_i^t \neq y^t) \tag{II.2}$$

The dual problem of **OP1** is (**OP1-Dual**):

$$\begin{aligned} \min \quad & \frac{1}{2}\|\sum_{i,\mathbf{y}} \boldsymbol{\alpha}_i(\mathbf{y})\Delta f_i(\mathbf{y})\|^2 - \sum_{i,\mathbf{y}} \boldsymbol{\alpha}_i(\mathbf{y}) l_i(\mathbf{y}) \\ \text{s.t.} \quad & \sum_{\mathbf{y}} \boldsymbol{\alpha}_i(\mathbf{y}) = C\ \forall\, i, \ \ \boldsymbol{\alpha}_i(\mathbf{y}) \geq 0\ \forall\, i, \mathbf{y} \end{aligned} \tag{II.3}$$

A cutting-plane algorithm[11] has been proposed for an equivalent formulation of **OP1** called the one-slack formulation (**OP1-one-slack**):

$$\begin{aligned} \min_{\mathbf{w},\xi\geq 0} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \\ \text{s.t.} \quad & \frac{1}{n}\mathbf{w}^T\sum_{i=1}^{n}\Delta f_i(\bar{\mathbf{y}}_i) \geq \frac{1}{n}\sum_{i=1}^{n} l_i(\bar{\mathbf{y}}_i) - \xi, \\ & \forall\, (\bar{\mathbf{y}}_1, \ldots, \bar{\mathbf{y}}_n) \in \mathscr{Y}^n \end{aligned} \tag{II.4}$$

There are other formulations which modify the regularizer term of **OP1**. One such formulation is the L1-norm regularized structural SVM[24]. The problem has the following form (**OP1-L1**):

$$\begin{aligned} \min_{\mathbf{w},\xi_i} \quad & \|\mathbf{w}\|_1 + C\sum_i \xi_i \\ \text{s.t.} \quad & \xi_i = \max_{\mathbf{y}} \left\{\max\left(0, l_i(\mathbf{y}) - \mathbf{w}^T\Delta f_i(\mathbf{y})\right)\right\} \quad \forall\, i \end{aligned} \tag{II.5}$$

The L1-norm regularizer in the above formulation results in sparse primal solution. Various methods to solve this formulation have been proposed in Zhu et al.[24]. One solution method is the cutting-plane method similar to that described

in [21]. There are other methods like projected sub-gradient and EM-style algorithm in [24] which are used to solve equivalent or modified formulations of **OP1-L1**. Another algorithm proposed in Wang et al.[22] uses a $0/1$ loss instead of the Hamming loss term of **OP1-L1** and uses the extra-gradient method to solve the resultant formulation.

Other formulations which modify the loss term associated with **OP1** have also been studied. The classical Conditional Random Fields[12] optimize the log-linear loss. Non-decomposable losses like Precision and Recall and ROC area are discussed in [17] and algorithms have been proposed to find the most violating constraint using such loss functions. A hybrid loss for Structural SVMs and CRFs with consistency properties has been proposed in [18].

However, we note that a more natural alternative to the linear max error loss term in **OP1** is the linear summed error loss term. Instead of optimizing the LME, we could optimize the summed error for every example calculated for every structured output $\mathbf{y} \in \mathscr{Y}$. Optimizing linear summed error loss is well-known for multi-class SVMs[23]. To the best of our knowledge, the use of linear summed error loss term for Structural SVMs has not yet been studied. In this work, we formulate a Structural SVM with the linear summed error loss term and propose efficient algorithms to solve the resultant formulation.

III. Linear Summed Error - Structural SVMs

Hill and Doucet [7] have proposed an overarching framework for all multi-category classification problems. They have also proposed various alternatives for empirical risk loss terms which could be used in the optimization problems for multi-category SVMs. Since structured classification can be considered a generalization of the multi-category classification problem with exponentially large number of classes, we could use the loss terms discussed in [7] for structured prediction optimization problems. This is possible because the framework in [7] easily extends to multi-category classification with exponential number of distinct classes. We formulate LSE-SSVMs as a convex QP (**OP2**):

$$\begin{aligned} \min_{\mathbf{w},\xi_i(\mathbf{y})\geq 0} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i,\mathbf{y}} \xi_i(\mathbf{y}) \\ \text{s.t.} \quad & \mathbf{w}^T\Delta f_i(\mathbf{y}) \geq l_i(\mathbf{y}) - \xi_i(\mathbf{y}) \ \ \forall\, i, \mathbf{y} \end{aligned} \tag{III.1}$$

We note that the problem size of **OP2** in terms of the primal variables is much larger when compared to **OP1**. This is primarily because of associating a primal slack variable $\xi_i(\mathbf{y})$ with each structured output $\mathbf{y} \in \mathscr{Y}$. The main complexity in solving **OP2** formulation is optimizing the sum of $\xi_i(\mathbf{y})$ for all possible structured outputs over the entire output space $\mathscr{Y}$ which might be of an exponential size. To handle this difficulty, we propose a one-slack equivalent formulation of the multiple-slack formulation in problem **OP2**, similar to that proposed in [11].

The first step is to convert the problem **OP2** into an n-slack formulation with an equivalent set of constraints. We could formulate **OP2** equivalently as (**OP2-n-slack**):

$$\begin{aligned} \min_{\mathbf{w},\xi_i\geq 0} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \\ \text{s.t.} \quad & \sum_{\mathbf{y}\in S_i} \mathbf{w}^T\Delta f_i(\mathbf{y}) \geq \sum_{\mathbf{y}\in S_i} l_i(\mathbf{y}) - \xi_i \ \ \forall\, i, \forall\, S_i \in 2^{\mathscr{Y}} \end{aligned} \tag{III.2}$$

where $2^{\mathscr{Y}}$ represents the power-set of $\mathscr{Y}$. S_i in **OP2-n-slack** represents a collection $\{\mathbf{y}\}$ of structured outputs associated with an example i and hence is a subset of $\mathscr{Y}$. The sum of slack variables $\sum_{\mathbf{y}} \xi_i(\mathbf{y})$ corresponding to an example i in the problem **OP2** is replaced with a single slack ξ_i in the problem **OP2-n-slack**. The constraints in problem **OP2** associated with a single example i are also grouped accordingly. We note that each slack $\xi_i(\mathbf{y}) \geq l_i(\mathbf{y}) - \mathbf{w}^T\Delta f_i(\mathbf{y})$. Since ξ_i corresponds to the sum of all slack variables $\xi_i(\mathbf{y})$, an intuitive constraint formulation for **OP2-n-slack** is that ξ_i must be greater than every possible collection $\{\xi_i(\mathbf{y})\}$ of outputs $\{\mathbf{y}\}$ corresponding to an example i. Hence if we consider any collection S_i of outputs $\mathbf{y}$ for an example i, then the grouped constraint $\xi_i \geq \sum_{\mathbf{y}\in S_i}[l_i(\mathbf{y}) - \mathbf{w}^T\Delta f_i(\mathbf{y})]$ must hold. We provide a proof of the equivalence of formulations **OP2** and **OP2-n-slack** next.

Theorem 1: The problems OP2 and OP2-n-slack are equivalent.

Proof: Adapting the proof in [9], we prove that OP2 and OP2-n-slack have an equivalent set of constraints and have the same objective value. For a given $\mathbf{w}$, the $\xi_i(\mathbf{y})$ in OP2 can be optimized individually and the optimum value for $\xi_i(\mathbf{y})$ is $\max\{0, l_i(\mathbf{y}) - \mathbf{w}^T\Delta f_i(\mathbf{y})\}$. If we assume that the size $|\mathscr{Y}|$ of output space is m, and the outputs can be enumerated as $\mathbf{y}^1, \mathbf{y}^2, \cdots, \mathbf{y}^m$ then we could write the constraints of problem OP2-n-slack as

$$\forall s \in \{0,1\}^m,\ \sum_{j=1}^{m} s_j \mathbf{w}^T\Delta f_i(\mathbf{y}^j) \geq \sum_{j=1}^{m} s_j l_i(\mathbf{y}^j) - \xi_i \ \ \forall i$$

where s is an m-sized vector. Hence for any given $\mathbf{w}$, the optimal ξ_i for problem OP2-n-slack could be given by

$$\xi_i = \max_{s\in\{0,1\}^m} \left\{ \sum_{j=1}^{m} s_j l_i(\mathbf{y}^j) - \sum_{j=1}^{m} s_j \mathbf{w}^T\Delta f_i(\mathbf{y}^j) \right\}$$

This linear function in s_j can be optimized for each s_j individually. Hence the optimal ξ_i is

$$\begin{aligned} \min \xi_i &= \sum_{j=1}^{m} \max_{s_j\in\{0,1\}} \left\{ s_j l_i(\mathbf{y}^j) - s_j \mathbf{w}^T\Delta f_i(\mathbf{y}^j) \right\} \\ &= \sum_{j=1}^{m} \max\left\{0, l_i(\mathbf{y}^j) - \mathbf{w}^T\Delta f_i(\mathbf{y}^j)\right\} \\ &= \min \sum_{j=1}^{m} \xi_i(\mathbf{y}^j) \\ &= \min \sum_{\mathbf{y}\in\mathscr{Y}} \xi_i(\mathbf{y}) \end{aligned}$$

Hence for any given $\mathbf{w}$, given the optimal ξ_i and $\xi_i(\mathbf{y})$, the objective functions of OP2 and OP2-n-slack are same and hence their optimal values. ■

In the next step, we convert the problem **OP2-n-slack** into an 1-slack formulation with an equivalent set of constraints. We formulate **OP2-n-slack** equivalently as (**OP2-1-slack**):

$$\begin{aligned} \min_{\mathbf{w},\xi\geq 0} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \\ \text{s.t.} \quad & \sum_{i=1}^{n}\sum_{\mathbf{y}\in S_i} \mathbf{w}^T\Delta f_i(\mathbf{y}) \geq \sum_{i=1}^{n}\sum_{\mathbf{y}\in S_i} l_i(\mathbf{y}) - \xi \\ & \forall (S_1, S_2, \cdots, S_n) \in (2^{\mathscr{Y}})^n \end{aligned} \tag{III.3}$$

In this 1-slack formulation, we replace the sum of n-slack variables $\sum_{i=1}^{n}\xi_i$ of **OP2-n-slack** with a single slack ξ. Hence the constraint grouping in 1-slack formulation must be consistent with every possible combination of the n-slack variables of **OP2-n-slack**. Thus the grouping in **OP2-1-slack** handles all possible n-tuples of collections of outputs $(S_1, S_2, \cdots, S_n)$. We prove the equivalence of problems **OP2-n-slack** and **OP2-1-slack** in the next theorem.

Theorem 2: The problems OP2-n-slack and OP2-1-slack are equivalent.

Proof: Adapting the proof in [11], we prove that OP2-n-slack and OP2-1-slack have an equivalent set of constraints and have the same objective value. For a given $\mathbf{w}$, the ξ_i in OP2-n-slack can be optimized individually and the optimum value for ξ_i is

$$\max_{S_i\in 2^{\mathscr{Y}}} \left\{ \sum_{\mathbf{y}\in S_i} l_i(\mathbf{y}) - \sum_{\mathbf{y}\in S_i} \mathbf{w}^T\Delta f_i(\mathbf{y}) \right\}$$

For the problem OP2-1-slack, the optimal value of ξ is given for a $\mathbf{w}$ by

$$\min \xi = \max_{(S_1,\cdots,S_n)\in(2^{\mathscr{Y}})^n} \left\{ \sum_{i,\mathbf{y}\in S_i} l_i(\mathbf{y}) - \sum_{i,\mathbf{y}\in S_i} \mathbf{w}^T\Delta f_i(\mathbf{y}) \right\}$$

This function can be decomposed linearly in S_i for any given $\mathbf{w}$, and each S_i can be optimized individually.

$$\begin{aligned} \min \xi &= \sum_{i=1}^{n} \max_{S_i\in 2^{\mathscr{Y}}} \left\{ \sum_{\mathbf{y}\in S_i} l_i(\mathbf{y}) - \sum_{\mathbf{y}\in S_i} \mathbf{w}^T\Delta f_i(\mathbf{y}) \right\} \\ &= \min \sum_{i=1}^{n} \xi_i \end{aligned}$$

Hence for any given $\mathbf{w}$, given the optimal ξ_i and ξ, the objective functions of OP2-n-slack and OP2-1-slack are same and hence their optimal values. ■

Hence, instead of solving the problem **OP2**, one could solve the single slack formulation **OP2-1-slack**. We give a cutting-plane algorithm similar to that proposed in [11] for solving the problem **OP2-1-slack**.

A. Cutting-Plane method for LSE-SSVMs

We solve the 1-slack formulation **OP2-1-slack** using a cutting-plane method. We observe that though the problems **OP2-1-slack** and **OP2** are theoretically equivalent, **OP2-1-slack** still contains an infinite number of constraints grouped across examples and across outputs. The cutting-plane procedure explained in [11] cannot be directly applied to problem **OP2-1-slack** because of a different constraint formulation. In [11], the maximum violating sequence $\max_{\mathbf{y}}\{l_i(\mathbf{y}) - \mathbf{w}^T\Delta f_i(\mathbf{y})\}$ is found for every example i and then a violating constraint is obtained by summing the $\Delta f_i(\mathbf{y})$ vectors and the corresponding $l_i(\mathbf{y})$ terms for all examples. Such a procedure might not be sufficient for **OP2-1-slack** because the constraints in **OP2-1-slack** are much more general than those in **OP1-one-slack** as they consider all possible n-tuples of all possible collections of outputs $\mathbf{y}$. Hence we adopt the procedure as explained in Algorithm 1 to find a violating constraint for the cutting-plane algorithm for **OP2-1-slack**.

We maintain a working set $\mathscr{W}$ of constraints. We also maintain a cache of violators V_i for each example i. In every iteration of the algorithm, we find $\arg\max_{\mathbf{y}}\{l_i(\mathbf{y}) - \mathbf{w}^T\Delta f_i(\mathbf{y})\}$ for every example i and add it to V_i if it is not already present. We build the sets S_i by finding the violating outputs for each example i as described in Step 14 of Algorithm 1. After passing through all examples, we add the n-tuple $(S_1, S_2, \cdots, S_n)$ as the violated constraint to the set $\mathscr{W}$. We terminate the algorithm when this violated constraint is not violated by more than a desired precision ϵ.

We employ certain speed-up heuristics in Algorithm 1. Step 9 in Algorithm 1 is expensive for various real-world applications. For sequence labeling applications, Viterbi decoding is performed in Step 9. For parse-tree classification, the Cook-Kasami-Younger (CKY) algorithm is used. All these algorithms are computationally demanding and hence need to be used only when it is absolutely essential. Hence in our implementation, we use a two-loop approach in Algorithm 1 similar to that proposed in [1]. In type-I loop, we perform Step 9 whereas in type-II loop, we do not perform Step 9 and use only V_i to find the most violating constraint. In our implementation of Algorithm 1, the optimization in Step 7 was done by solving the restricted dual problem using a Hildreth-despo QP solver used in [10]. Other methods like a fixed-threshold SMO as used in [13] could also be tried.

IV. Dual LSE-Structural SVMs

We now discuss a method to solve the dual problem of **OP2**, referred to as **(OP3)**:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} D(\boldsymbol{\alpha}) \quad &= \quad \frac{1}{2}\Big\|\sum_{i,\mathbf{y}} \boldsymbol{\alpha}_i(\mathbf{y})\Delta f_i(\mathbf{y})\Big\|^2 - \sum_{i,\mathbf{y}} \boldsymbol{\alpha}_i(\mathbf{y}) l_i(\mathbf{y}) \\ \text{s.t.} \quad & \quad 0 \leq \boldsymbol{\alpha}_i(\mathbf{y}) \leq C \;\forall\, i, \mathbf{y} \end{aligned} \tag{IV.1}$$

This dual problem has a nice structure and hence we could solve this dual problem instead of solving the primal problem **OP2** or **OP2-1-slack**. It is also important to note that the equality constraint present in **OP1-Dual** is absent in **OP3**. This

Algorithm 1 *Primal cutting-plane method to solve OP2-1-slack*

Input $S = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, C, ϵ
$\mathbf{w} = 0, V_i = \{\mathbf{y}_i\},\ i = 1, 2, \cdots, n$
$S_i = \phi,\ i = 1, 2, \cdots, n$
$\mathscr{W} = \phi$
$iter = 0, stopflag = 0$
repeat
  Solve **OP2-1-slack** with constraints in $\mathscr{W}$
  for $i = 1, \ldots, n$ **do**
    $\hat{\mathbf{y}}_i = \arg\max_{\mathbf{y}} \{l_i(\mathbf{y}) - \mathbf{w}^T \Delta f_i(\mathbf{y})\}$.
    if $\hat{\mathbf{y}}_i \notin V_i$ **then**
      $V_i = V_i \cup \{\hat{\mathbf{y}}_i\}$
    end if
    for $\mathbf{y} \in V_i$ **do**
      if $\mathbf{w}^T \Delta f_i(\mathbf{y}) < l_i(\mathbf{y})$ **then**
        $S_i = S_i \cup \{\mathbf{y}\}$
      end if
    end for
  end for
  if $\sum_{i, \mathbf{y} \in S_i} \left\{ l_i(\mathbf{y}) - \mathbf{w}^T \Delta f_i(\mathbf{y}) \right\} \leq \xi + \epsilon$ **then**
    stopflag=1
  else
    $\mathscr{W} = \mathscr{W} \cup \{(S_1, S_2, \cdots, S_n)\}$
  end if
  $iter := iter + 1$
until $stopflag == 1$

makes the formulation **OP3** more suitable for online learning applications compared to **OP1-Dual**. We now give a sequential dual coordinate-descent method for solving **OP3**.

If we assume that the size $|\mathscr{Y}|$ of output space is m, and the outputs can be enumerated as $\mathbf{y}^1, \mathbf{y}^2, \cdots, \mathbf{y}^m$, the dual problem **OP3** could be simply written as **(OP3a)**:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2} \left\| \sum_{i=1}^{n} \sum_{j=1}^{m} \boldsymbol{\alpha}_i(\mathbf{y}^j) \Delta f_i(\mathbf{y}^j) \right\|^2 - \sum_{i=1}^{n} \sum_{j=1}^{m} \boldsymbol{\alpha}_i(\mathbf{y}^j) l_i(\mathbf{y}^j) \\ \text{s.t.} \quad & 0 \leq \boldsymbol{\alpha}_i(\mathbf{y}^j) \leq C \ \ \forall\, i, j \end{aligned} \tag{IV.2}$$

The optimal primal and dual variables of **OP2** and **OP3** are associated using the relation

$$\mathbf{w}^* = \sum_{i, \mathbf{y}} \boldsymbol{\alpha}_i^*(\mathbf{y}) \Delta f_i(\mathbf{y}) \tag{IV.3}$$

Hsieh et.al [8] have proposed a dual coordinate descent method for large scale classification SVMs. If we have all the outputs $\mathbf{y}^1, \mathbf{y}^2, \cdots, \mathbf{y}^m$ at our disposal, we could apply the dual coordinate descent method in [8] to our formulation in a straight-forward fashion. We discuss the details assuming that all outputs $\mathbf{y}^1, \mathbf{y}^2, \cdots, \mathbf{y}^m$ in $\mathscr{Y}$ are available to us. We then discuss a way to handle the exponential size m of the output space $\mathscr{Y}$.

A. Dual Coordinate Descent method for LSE-Structural SVMs

We follow a dual coordinate descent method similar to that described in [8]. The algorithm runs in stages $k = 1, 2, \cdots, \infty$ and we could consider the vector of dual variables $\boldsymbol{\alpha}^k = [\alpha_i^k(\mathbf{y}^j)], \forall i, j$ at each stage k of the algorithm. Such a vector is of length $p = n \times m$. At each stage k of the algorithm, we optimize all the variables of the vector $\boldsymbol{\alpha}^k$ in a sequential fashion. During the sequential optimization, a component of the vector $\boldsymbol{\alpha}^k$ is selected and optimized keeping all the other components fixed. If we assume that the l^{th} component of $\boldsymbol{\alpha}^k$ has been selected, and if we consider the selected component as $\alpha_i^k(\mathbf{y}^j)$, then we optimize the amount δ of change which could be associated with $\alpha_i^k(\mathbf{y}^j)$ without violating the constraints. To optimize δ, we solve a simple QP in δ which is **(OP4)**:

$$\begin{aligned} D(\boldsymbol{\alpha}^k + \delta \mathbf{e}_l) \quad &= \quad \frac{1}{2} \Delta f_i(\mathbf{y}^j)^T \Delta f_i(\mathbf{y}^j) \delta^2 + \nabla_l D(\boldsymbol{\alpha}^k) \delta + b \\ \text{s.t.} \quad & \quad 0 \leq (\alpha_i^k(\mathbf{y}^j) + \delta) \leq C \end{aligned} \tag{IV.4}$$

where $\mathbf{e}_l$ is a vector of length p with 1 at position l and zeros in the remaining positions. b is a constant term and $\nabla_l D(\boldsymbol{\alpha}^k)$ is the l^{th} component of the gradient ∇D. We see that for the change δ to be zero, the l^{th} component $\nabla_l^P D(\boldsymbol{\alpha}^k)$ of projected gradient $\nabla^P D(\boldsymbol{\alpha}^k)$ must be zero. The l^{th} component of the projected gradient is given by

$$\nabla_l^P D(\boldsymbol{\alpha}^k) = \begin{cases} \nabla_l D(\boldsymbol{\alpha}^k) & \text{if } 0 < \alpha_i^k(\mathbf{y}^j) < C \\ min(0, \nabla_l D(\boldsymbol{\alpha}^k)) & \text{if } \alpha_i^k(\mathbf{y}^j) = 0 \\ max(0, \nabla_l D(\boldsymbol{\alpha}^k)) & \text{if } \alpha_i^k(\mathbf{y}^j) = C \end{cases}$$

$\nabla_l D(\boldsymbol{\alpha}^k)$, the gradient with respect to a particular variable $\alpha_i^k(\mathbf{y}^j)$ is given by

$$\nabla_l D(\boldsymbol{\alpha}^k) = \mathbf{w}^T \Delta f_i(\mathbf{y}^j) - l_i(\mathbf{y}^j) \tag{IV.5}$$

If $\nabla_l^P D(\boldsymbol{\alpha}^k) \neq 0$, then we update $\alpha_i^k(\mathbf{y}^j)$ to $\alpha_i^k(\mathbf{y}^j)_{new}$ using

$$\min \left(\max \left(\alpha_i^k(\mathbf{y}^j) - \frac{\nabla_l D(\boldsymbol{\alpha}^k)}{\Delta f_i(\mathbf{y}^j)^T \Delta f_i(\mathbf{y}^j)}, 0 \right), C \right) \tag{IV.6}$$

After finding $\alpha_i^k(\mathbf{y}^j)_{new}$, we update $\mathbf{w}$ using the formula

$$\mathbf{w}_{new} = \mathbf{w}_{old} + (\alpha_i^k(\mathbf{y}^j)_{new} - \alpha_i^k(\mathbf{y}^j)_{old}) \Delta f_i(\mathbf{y}^j) \tag{IV.7}$$

However all these calculations are possible only for a finite sized vector $\boldsymbol{\alpha}^k$. The difficulty in solving **OP3** arises because of the exponential size m of $\mathscr{Y}$ which gives rise to an equally exponential sized dual variable set. Hence a direct application of the dual coordinate scheme discussed above is not possible for **OP3**. We handle the problem of exponential sized dual variable set by restricting our attention to a subset containing only r dual variables, $r \ll m$.

In SDM[1] to solve SSVMs, a cache V_i is maintained for every training example i and the optimization is done for the dual variables corresponding to $\mathbf{y} \in V_i$. However because of the summation constraint in **OP1-Dual**, at least two variables have to be selected for optimization. The problem **OP3** does not have such a summation constraint and hence each variable could be optimized independently. Similar to SDM[1], we

TABLE I

SUMMARY OF MULTI-CLASS DATA SETS. n DENOTES THE SIZE OF THE TRAINING DATA, d IS THE INPUT DIMENSION, k DENOTES THE NUMBER OF CLASSES AND N IS THE FEATURE VECTOR DIMENSION

Data set	n	d	k	N
Covertype	581012	54	7	378
Sector	6412	55197	105	5795685

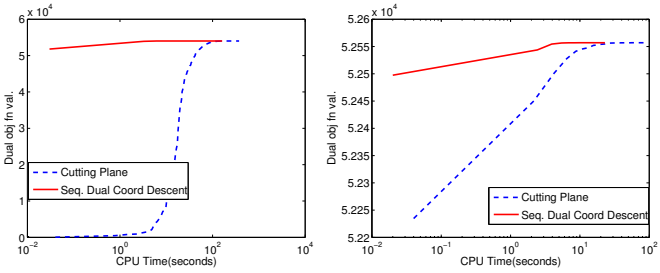

Fig. 1. **Comparison of *CP* and *SDCD* methods on multi-class data sets.**(C=0.01) *Left*: **Covertype**, *Right*: **Sector**.

maintain a set V_i for each training example i and perform optimization on the $\alpha_i(\mathbf{y})$ variables for $\mathbf{y} \in V_i$. Hence we solve the reduced optimization problem (**OP5**):

$$\min_{\boldsymbol{\alpha}} D'(\boldsymbol{\alpha}) \quad = \quad \frac{1}{2}\| \sum_{i,\mathbf{y}\in V_i} \boldsymbol{\alpha}_i(\mathbf{y})\Delta f_i(\mathbf{y})\|^2 - \sum_{i,\mathbf{y}\in V_i} \boldsymbol{\alpha}_i(\mathbf{y}) l_i(\mathbf{y})$$
$$\text{s.t.} \qquad 0 \leq \boldsymbol{\alpha}_i(\mathbf{y}) \leq C \;\forall\, i, \mathbf{y} \in V_i$$

To construct the set V_i for each example i, we find a violator $\mathbf{y} \in \mathscr{Y}$ using $\arg\max_{\mathbf{y}}\{l_i(\mathbf{y}) - \mathbf{w}^T\Delta f_i(\mathbf{y})\}$. Hence our algorithm operates by visiting an example i, constructing or updating the set V_i and then performing dual coordinate descent method on dual variables corresponding to that example i. We illustrate the procedure in Algorithm 2.

The sequential dual coordinate descent method described in Algorithm 2 contains added speed-up heuristics. The basic algorithm scheme is to visit each example i, find a new violator $\hat{\mathbf{y}}_i$ using Step 10. If the violator is not present in V_i, we add it to V_i and perform coordinate descent for variables corresponding to $\mathbf{y} \in V_i$. The entire procedure terminates when there is no significant change in the dual variables for all examples. Various other terminating conditions could be used; but we found that this termination criterion was a good heuristic to stop the algorithm. As discussed earlier, Step 10 is still a computationally intensive task and hence should be used less frequently. Hence we follow a two loop approach with type-I loop and type-II loop. In type-I loop, we always find a violator and in type-II loop, we do not perform Step 10 and perform coordinate descent method for dual variables associated with the set V_i. We alternate between the two loops such that we spend more time in type-II loops.

V. EXPERIMENTS

In this section, we give details about the various experiments conducted to show the efficacy of both primal and dual optimization methods for LSE-SSVMS. We performed experiments on various multi-class and sequence-labeling datasets using Cutting-Plane (CP) and the Sequential Dual Coordinate Descent (SDCD) methods. The values of the parameter C in **OP2-1-slack** and **OP3** were chosen based on the validation set performance. Both the algorithms were implemented in C with double precision. All experiments were run on a dual-CPU quad-core 2.4GHz Intel Xeon Processor with a $16GB$ shared main memory under Linux.

Algorithm 2 *Sequential Dual Coordinate Descent Algorithm to solve* **OP5**

1: Input $S = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, C
2: $\mathbf{w} = 0, V_i = \{\mathbf{y}_i\}$, $\alpha_i(\mathbf{y}_i) = C \;\; \forall i = 1, 2, \cdots, n$
3: $\kappa_1 = 10, \kappa_2 = 5$
4: $iter = 0, stopflag = 0$
5: GetMaxY = 1,deltathresh=10^{-4}
6: **repeat**
7: ChangedInOuterLoop=0
8: **for** $i = 1, \ldots, n$ **do**
9: **if** GetMaxY == 1 **then**
10: $\hat{\mathbf{y}}_i = \arg\max_{\mathbf{y}}\{l_i(\mathbf{y}) - \mathbf{w}^T\Delta f_i(\mathbf{y})\}$.
11: **if** $\hat{\mathbf{y}}_i \notin V_i$ **then**
12: $V_i = V_i \cup \{\hat{\mathbf{y}}_i\}$, $\boldsymbol{\alpha}_i(\hat{\mathbf{y}}_i) = 0$.
13: **end if**
14: **end if**
15: alphachange=0
16: **for** $\mathbf{y} \in V_i$ **do**
17: Solve (OP4) to find δ
18: Update $\boldsymbol{\alpha}_i(\mathbf{y}) = \boldsymbol{\alpha}_i(\mathbf{y}) + \delta$
19: Update $\mathbf{w}$ using (IV.7)
20: alphachange += $\delta \times \delta$
21: **end for**
22: **if** GetMaxY == 1 AND alphachange>deltathresh **then**
23: ChangedInOuterLoop = 1
24: **end if**
25: **end for**
26: **if** GetMaxY == 1 **then**
27: **if** ChangedInOuterLoop == 0 **then**
28: stopflag=1
29: **else**
30: **if** $iter \geq \kappa_1$ **then**
31: GetMaxY=0
32: **end if**
33: **end if**
34: **else**
35: **if** $(iter - \kappa_1)\%\kappa_2 == 0$ **then**
36: GetMaxY=1
37: **end if**
38: **end if**
39: $iter := iter + 1$
40: **until** $stopflag == 1$

A. Multi-class Datasets

First, we compare the performance of the cutting-plane and SDCD method on multi-class datasets. We use Covertype and Sector datasets from [4]. The details of the datasets are given

TABLE II

SUMMARY OF SEQUENCE-LABELING DATA SETS. n AND n_{test} DENOTE THE SIZES OF THE TRAINING AND TEST DATA RESPECTIVELY, d IS THE INPUT DIMENSION, k DENOTES THE NUMBER OF ALPHABETS AND N IS THE FEATURE VECTOR DIMENSION

Data set	n	n_{test}	d	k	N
OCR	6877	55310	128	26	4004
POS	7200	1681	404990	42	17011344
WSJPOS	35531	1681	446180	42	18741324
Brown	48242	9098	290843	185	53840180

in Table I. The following feature vector representation for the data was used:

$$f(\mathbf{x}, \mathbf{y}) = \left(\; x^1 \otimes y^1 \; \right) \tag{V.1}$$

Multi-class feature representation could be considered as the feature representation for POS-tagging of a single word sentence where $\mathbf{x} = (x^1)$ and $\mathbf{y} = (y^1)$. Hence the construction in (V.1) uses no edge-related information. SDCD method uses the standard Hamming-loss illustrated in (II.2), whereas cutting-plane method uses the scaled Hamming-loss as in [11] with a scaling factor of 10. A simple 0-1 loss for the cutting-plane method might give a poor generalization performance due to the grouped constraints in **OP2-1-slack**. Hence the need for scaling. We performed the experiments with C=0.01. We included all classes in V_i for both the methods and skipped the violator finding steps (Step 9 in Algorithm 1 and Step 10 in Algorithm 2). For the cutting-plane method, $\epsilon = 0.1$ was used. The results are presented in Fig. 1. The results from our experiments clearly indicate that SDCD method outperforms the cutting-plane method by reaching the optimum objective value an order of magnitude faster.

B. Sequence Labeling Datasets

We also performed experiments with sequence labeling datasets in addition to those on multi-class datasets. We used OCR[19], POS[16], WSJPOS[14] and Brown[3] datasets for our experiments. We used the ten-part partition of OCR dataset as was used in [19], where each partition has a train and a test dataset. The dataset details are given in Table II. We used the feature vector representation discussed for POS-tagging example in Section I for all our experiments. For OCR datasets, we used $\epsilon = 0.1$ for cutting-plane method. However for POS, WSJPOS and Brown datasets, $\epsilon = 1.0$ proved to be a sufficient stopping threshold for the cutting-plane method.

1) Training Time Comparison: To compare the training times of both the methods, we compute the relative dual objective function value difference $\frac{|f^* - f|}{|f^*|}$, where f^* is the optimal objective function value obtained by the respective optimization method. We present the plots in Fig. 2 which give the decrease in the relative dual objective value as the time increases. Similar performance is observed for OCR dataset as indicated by results in Table III. Our plots clearly indicate that SDCD method is an order of magnitude faster than the cutting-plane method for all datasets and achieves the desired test set accuracy faster.

2) Test set accuracy Comparison: We compare the test set accuracy obtained by both the methods on all datasets. The behaviour of test set accuracy as time progresses is given in Fig. 2. The plots indicate that SDCD method achieves a better test set accuracy much earlier than the cutting-plane method. This is primarily because of the online nature of the SDCD method which helps it to learn a useful model during its initial iterations of training. Usually, the SDCD method gives a better test set accuracy after being trained for the first few iterations itself. This property is clearly lacking in cutting-plane method which gives a comparable test set generalization performance much later in its training stage.

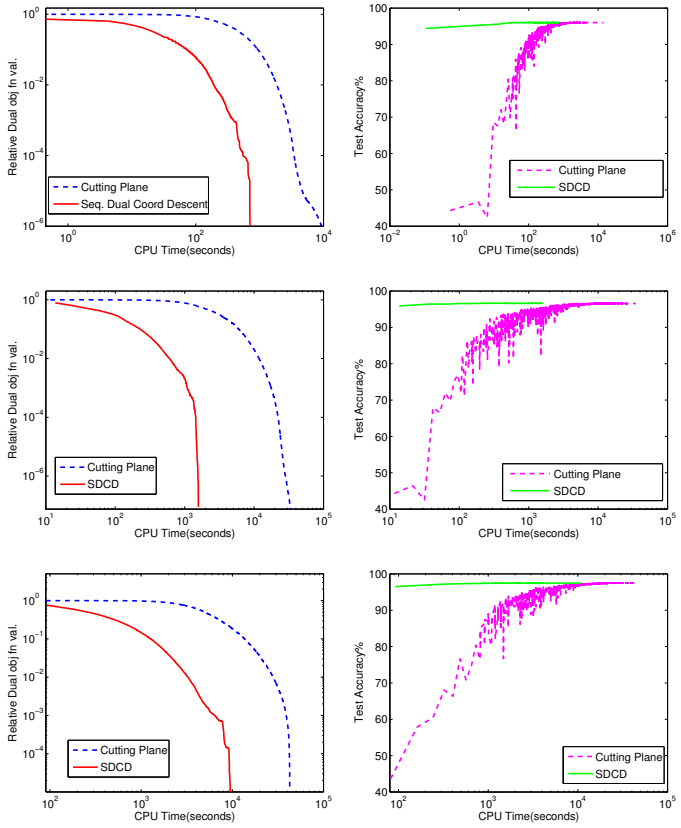

Fig. 2. **Comparison of *CP* and *SDCD* methods on sequence labeling data sets.**(C=0.01) *Row 1*: **POS**, *Row 2*: **WSJPOS**, *Row 3*: **Brown**.

3) Effect of the regularization parameter C: We performed experiments with C=0.01, 0.1 and 1.0 for partitioned OCR datasets. We present the runtime comparison results for both the methods in Table III and the test set accuracy results in Table IV. The results indicate that as we increase the regularization parameter C, the time taken for training by both the methods increases. However the increase in the training times observed for SDCD is lesser compared to that observed for the cutting-plane method. The results also indicate a decrease in the test set accuracy as C increases.

4) Comparison of SDCD for LSE-SSVMs with SDM for LME-SSVMs: We note that SDCD which solves **OP3** is very similar to the sequential dual method (SDM)[1] which solves **OP1-Dual**. Hence we compare the training time, test set accuracy and the number of active constraints present in the working set for SDCD and SDM algorithms. The details are

TABLE III
TRAINING TIME(SECONDS) COMPARISON OF SEQ. DUAL COORD DESCENT(SDCD) AND THE CUTTING-PLANE (CP) METHOD ON OCR DATA PARTITION

		$OCR0$	$OCR1$	$OCR2$	$OCR3$	$OCR4$	$OCR5$	$OCR6$	$OCR7$	$OCR8$	$OCR9$
C=0.01	CP	115.36	99.37	119.97	132.31	142.08	113.14	156	175.91	133.95	113.33
	$SDCD$	24.89	23.41	20.84	25.44	29.35	25.62	34.66	23.07	30.87	24.95
C=0.1	CP	595.13	628.05	474.26	432.04	1288.68	467.75	418.11	378.49	511.23	1193.12
	$SDCD$	37.37	38.73	36.73	49.69	38.71	33.47	54.11	68.39	49.2	37.63
C=1.0	CP	4181.79	9013.09	4928.94	9628	8971.55	7657.24	7134.54	4904.36	4118.72	3822.71
	$SDCD$	106.39	179.53	123.81	116.5	178.28	148.41	184.26	193.44	146.02	116.76

TABLE IV
TEST SET ACCURACY(%) COMPARISON OF SEQ. DUAL COORD DESCENT(SDCD) AND THE CUTTING-PLANE (CP) METHOD ON OCR DATA PARTITION

		$OCR0$	$OCR1$	$OCR2$	$OCR3$	$OCR4$	$OCR5$	$OCR6$	$OCR7$	$OCR8$	$OCR9$
C=0.01	CP	75.46	76.05	75.97	76.8	76.43	75.87	76.44	75.66	76.44	75.78
	$SDCD$	74.43	75.4	75.3	76.17	75.47	75.24	75.99	74.8	75.6	75.02
C=0.1	CP	73.46	74.14	74.1	75.19	75.28	73.52	74.2	73.78	75.5	73.84
	$SDCD$	73.86	74.54	74.66	75.56	75.44	73.88	74.43	74.02	75.62	74.57
C=1.0	CP	71.15	71.56	72.25	73	72.74	71.24	71.63	71.18	73.38	71.74
	$SDCD$	71.3	72.05	72.44	73.15	72.93	71.39	71.68	71.68	73.31	71.85

provided in Table V. We note from Table V that the training time required to solve LSE-SSVM is more than that required for LME-SSVMs. This is mainly due to the increase in the number of slack variables from n in **OP1** to $n|\mathscr{Y}|$ in **OP2**. Correspondingly, there is also an increase in the size of the working set. This is evident from Table V.

TABLE V
COMPARISON OF SDCD WITH SDM

Data set	Train time(sec)		Test Accuracy%		$\sum_i \lvert V_i \rvert$	
	SDCD (C=0.01)	SDM (C=0.1)	SDCD (C=0.01)	SDM (C=0.1)	SDCD (C=0.01)	SDM (C=0.1)
POS	715	53	96.02	96.03	133782	38891
WSJPOS	1588	242	96.61	96.67	305533	114891
Brown	10832	1836	97.46	97.56	395130	146127

VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a new formulation for Structural SVMs with a composite loss term made up of linear summed error associated with all possible structured outputs. We also give a primal cutting-plane and a sequential dual coordinate descent (SDCD) method to solve the formulation. Through various experiments, we demonstrate that the sequential dual coordinate descent method has a superior performance than the primal cutting-plane method in terms of training time, while achieving a comparable test set accuracy. The algorithms are general and can be used in the context of other structured outputs like parse trees. Though the SDCD method is slower than SDM for structured classification, we note that the SDCD method can directly be used in online learning of Structural SVMs. We are investigating the details of extending the SDCD method to online learning of Structural SVMs.

ACKNOWLEDGMENT

The work of the first author was partially supported by the grant from Infosys Ltd., India.

REFERENCES

[1] P. Balamurugan, S. K. Shevade, S. Sundararajan, S. S. Keerthi. A Sequential Dual Method for Structural SVMs, SDM, 2011.

[2] P. Bartlett, M. Collins, B. Taskar, D. McAllester. Exponentiated Gradient Algorithms For Large-margin Structured Classification. NIPS, 2004.

[3] Brown Corpus. Available at `http://nltk.googlecode.com/svn/trunk/nltk_data/index.xml`

[4] C.-C. Chang, C.-J. Lin. LIBSVM Data: Classification (Multi-class). `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html`

[5] D. Freitag, A. K. McCallum. Information Extraction using HMMs and shrinkage. AAAI, 1999.

[6] T. Gärtner, S. Vembu. On Structured Output Training: Hard Cases and an Efficient Alternative. ECML, 2009.

[7] S. I. Hill, A. Doucet. A Framework for kernel-based multi-category classification. Tech. rep. CUED/F-INFENG/TR.508, Engineering Dept., University of Cambridge. 2005.

[8] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, S. Sundararajan. A Dual Coordinate Descent Method for Large-scale Linear SVM, ICML,2008.

[9] T. Joachims. Training linear SVMs in linear time. SIGKDD, 2006.

[10] T. Joachims. SVM^{struct}: Software available at `http://www.cs.cornell.edu/People/tj/svm_light/svm_struct.html`

[11] T. Joachims, T. Finley, C-N. J. Yu. Cutting-Plane training of Structural SVMs. Machine Learning, 77(1):27-59, 2009.

[12] J. Lafferty, A. McCallum, F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. ICML, 2001.

[13] C. Lee, M-G. Jang. A Modified Fixed-threshold SMO for 1-Slack Structural SVMs. ETRI Journal, Volume 32, Number 1, February, 2010.

[14] M. Marcus, B. Santorini, M. A. Marcinkiewicz, Building a Large Annotated Corpus of English, Computational Linguistics, 1993.

[15] A. McCallum, D. Freitag, F. Pereira. Maximum Entropy Markov Models for Information Extraction and Segmentation. ICML, 2000.

[16] N. Nguyen, Y. Guo, Comparisons of Sequence-labeling Algorithms and Extensions, ICML, 2007.

[17] M. Ranjbar, G. Mori, Y. Wang. Optimizing Complex Loss Functions in Structured Prediction. ECCV, 2010.

[18] Q. Shi, M. Reid, T. Caetano, A. v. d. Hengel. A Hybrid Loss for Multiclass and Structured Prediction. Technique Report, School of Computer Science, The University of Adelaide. 2010.

[19] B. Taskar, C. Guestrin and D. Koller, Maximum-margin Markov networks, NIPS, 2003.

[20] G. H. Teo, S. V. N. Vishwanathan, A. Smola, Q. V. Le. Bundle Methods for Regularized Risk Minimization. JMLR, 11:311-365, 2010.

[21] I. Tsochantaridis, T. Joachims, T. Hoffmann, Y. Altun. Large Margin Methods for Structured and Inter-dependent Output Variables, JMLR, 6:1453:1484, 2005.

[22] Z. Wang, J. Shawe-Taylor. Large-Margin Structured Prediction via Linear Programming. AISTATS, 2009.

[23] J. Weston, C. Watkins. Multi-class support vector machines. In M. Verleysen, editor, Proceedings of EASNN99, Brussels, 1999. D. Facto Press.

[24] J. Zhu, E. P. Xing, B. Zhang. Primal Sparse Max-Margin Markov Networks. SIGKDD, 2009.