Int. J. Appl. Math. Comput. Sci., 2014, Vol. 24, No. 4, 917–930
DOI: 10.2478/amcs-2014-0067

A PRIMAL SUB–GRADIENT METHOD FOR STRUCTURED CLASSIFICATION WITH THE AVERAGED SUM LOSS

DEJAN MANČEV, BRANIMIR TODOROVIĆ

Faculty of Sciences and Mathematics
University of Niš, Višegradska 33, Niš, Serbia
e-mail: dejan.mancev@pmf.edu.rs, branimirtodorovic@yahoo.com

We present a primal sub-gradient method for structured SVM optimization defined with the averaged sum of hinge losses inside each example. Compared with the mini-batch version of the Pegasos algorithm for the structured case, which deals with a single structure from each of multiple examples, our algorithm considers multiple structures from a single example in one update. This approach should increase the amount of information learned from the example. We show that the proposed version with the averaged sum loss has at least the same guarantees in terms of the prediction loss as the stochastic version. Experiments are conducted on two sequence labeling problems, shallow parsing and part-of-speech tagging, and also include a comparison with other popular sequential structured learning algorithms.

Keywords: structured classification, support vector machines, sub-gradient methods, sequence labeling.

1. Introduction

A structured classification problem considers learning a mapping from the input to the output of structured objects, where the output structures incorporate different relationships among their classes. These algorithms, such as conditional random fields (Lafferty *et al*., 2001), the structured perceptron (Collins, 2002) or structured support vector machines (SSVMs) (Tsochantaridis *et al*., 2005), are proved to outperform the standard binary and multiclass classifiers, but they are usually more complex to train and require inference during the training procedure. They are applicable to different domains such as natural language processing (Daume, 2006), computer vision (Nowozin and Lampert, 2011), speech recognition (Sas and Żołnierek, 2013) and bioinformatics (Li *et al*., 2007). Besides easy training for the perceptron algorithm, training the SSVM assumes constrained optimization with possibly exponentially many constraints.

There are several ways to efficiently deal with such optimization. For the special case of a linearly decomposable loss, this problem can be presented with an equivalent polynomial-size formulation (Taskar *et al*., 2004) by introducing marginal variables on which we can apply sequential minimal optimization (SMO) (Platt, 1999). On the other hand, without the previous assumption, we can seek a small set of constraints that is sufficient to approximate a solution by increasing the working set of constraints through iterations. Joachims *et al*. (2009) use the cutting plane method on the equivalent formulation with one slack variable shared across all data and build the working set of constraints with a separation oracle. Even though the algorithm finds a solution where constraints are violated by no more than ϵ after $\mathcal{O}(1/\epsilon)$ iterations, each iteration assumes finding a separation oracle, which can be time consuming for a larger number of examples.

For large-scale problems there exist more suitable versions of online algorithms which simply sequentially perform parameter updates concerning only the most violated structure at a time, such as a perceptron (Collins, 2002) with a fixed step size, the passive-aggressive (PA) algorithm (Crammer *et al*., 2006) with an optimal step size analytically found in dual by considering only one constraint corresponding to the 'best' structure, the primal sub-gradient descent method (Ratliff *et al*., 2006) with a predefined step size followed by a projection which transfers the parameter back into the feasible region.

Shalev-Shwartz *et al*. (2011) proposed the Pegasos algorithm which takes a sub-gradient step with a predetermined step size and which can work in the mini-batch variant by choosing a set of examples and performing a sub-gradient step on it. Its structured

version was successfully applied to various problems: dependency parsing (Martins *et al*., 2011), semantic role labeling (Lim *et al*., 2013), part-of-speech tagging (Ni *et al*., 2010), optical character recognition (Jaggi *et al*., 2012), and named entity recognition (Lee *et al*., 2011). The empirical performance indicated fast convergence with the results comparable with those of other structured algorithms, while Ratliff *et al*. (2006) show that the cumulative prediction loss for the structured sub-gradient method grows only sublinearly in time.

Besides a single best inference which uses all previous algorithms during the training procedure, Crammer *et al*. (2005) introduce the k-best MIRA, which deals with the k-best structures at a time. The algorithm minimizes the norm of the parameter change while satisfying constraints corresponding to k-best outputs. McDonald *et al*. (2005) successfully applied it to dependency parsing, concluding that even small values of k are sufficient to achieve close to best performance. Another common feature of all algorithms is that they can be seen as minimization of a differently chosen regularized loss function. There are various loss functions which are used in the structured case, such as the structured hinge loss or its squared version (Tsochantaridis *et al*., 2005), the log loss (Lafferty *et al*., 2001), the softmax-margin as a log loss with a cost function (Gimpel and Smith, 2010), or the structured ramp loss (Do *et al*., 2008).

In this paper we shall consider the averaged sum of hinge losses over the structures inside one example and an approximate primal objective function on which the sub-gradient method is applied. Such changes in the loss function result in the fact that the algorithm can consider multiple structures inside one example (similar to the k-best MIRA variant). For this version we provide the cumulative bound of prediction losses and perform experiments with other popular sequential structured learning algorithms.

The paper is organized as follows. In Section 2, we define basic notations and the problem of max-margin structured classifiers. After reviewing the existing version of Pegasos for the structured case, in Section 3 we introduce the Pegasos algorithm with the averaged sum loss. Next, we provide a theoretical analysis for the introduced algorithm, followed by implementation concerns for sparse updates and the calculation of averaged parameters. In Section 6, we present experiments on sequence labeling problems, and conclude the paper in the last section.

2. Problem definition

Let $\mathscr{D} = ((\boldsymbol{x}^n, \boldsymbol{y}^n))_{n=1}^N$ be a training set, where each input $\boldsymbol{x}^n$ has the corresponding output structure $\boldsymbol{y}^n$. The set of all possible structures over $\boldsymbol{x}^n$ is denoted by $\mathscr{Y}(\boldsymbol{x}^n)$ and $\mathscr{Y}_{-n} = \mathscr{Y}(\boldsymbol{x}^n) \setminus \boldsymbol{y}^n$. In the case of sequence labeling, for example, $\boldsymbol{x}^n \in \mathcal{X}^{T_n}$ represents an input sequence of length T_n and $\mathscr{Y}(\boldsymbol{x}^n) = \mathcal{Y}^{T_n}$, where $\mathcal{Y}$ is a set of possible labels for an element of the input alphabet $\mathcal{X}$.

The problem of minimizing the regularized empirical risk over the set $\mathscr{D}$ is

$$\min_{\mathbf{w}} f(\mathbf{w}) = \min_{\mathbf{w}} \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{N} \sum_{n=1}^{N} \ell_n(\mathbf{w}), \quad (1)$$

where $\ell_n(\mathbf{w})$ represents a loss function on the n-th example with parameters $\mathbf{w}$. As inside each example there are many output structures, the loss function can be defined for each one separately. Let $\ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y}))$ represent a loss for the structure $\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)$ with parameters $\mathbf{w}$. We will define the *hinge loss for a structure* $\boldsymbol{y}$ as

$$\ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})) = \max\left(0, L(\boldsymbol{y}^n, \boldsymbol{y}) - \mathbf{w}^\mathsf{T} \Delta\mathbf{F}_n(\boldsymbol{y})\right), \quad (2)$$

with $\Delta\mathbf{F}_n(\boldsymbol{y}) = \mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}^n) - \mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y})$, where $\mathbf{F}(\boldsymbol{x}, \boldsymbol{y})$ represents a *global feature vector* measuring the compatibility of $\boldsymbol{x}$ and $\boldsymbol{y}$, while the function $L(\boldsymbol{y}^n, \boldsymbol{y})$ represents the cost of assigning the output $\boldsymbol{y}$ to observation $\boldsymbol{x}^n$ instead of $\boldsymbol{y}^n$.

Since inside each example there are many output structures, usually we deal only with those which provide the maximum loss on the current example. In that case the loss function, called the *max-margin* (MM) *loss*[1], is defined as

$$\ell_n^{\mathrm{MM}}(\mathbf{w}) = \max_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})) = \ell(\mathbf{w}; (\boldsymbol{x}^n, \widetilde{\boldsymbol{y}}_n)), \quad (3)$$

where $\widetilde{\boldsymbol{y}}_n$ is the 'best' structure for $\boldsymbol{x}^n$ with respect to the loss function, i.e.,

$$\widetilde{\boldsymbol{y}}_n = \arg\max_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})). \quad (4)$$

For the problem (1) and the previous loss function (3), the corresponding constrained optimization is

$$\min_{\mathbf{w}, \boldsymbol{\xi}} \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{N} \sum_{n=1}^{N} \xi_n \quad (5)$$

subject to

$$\mathbf{w}^\mathsf{T} \Delta\mathbf{F}_n(\boldsymbol{y}) \geq L(\boldsymbol{y}^n, \boldsymbol{y}) - \xi_n, \quad \forall n, \forall \boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n). \quad (6)$$

According to the constraints, the original structure $\boldsymbol{y}^n$ should produce a greater score $\mathbf{w}^\mathsf{T}\mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}^n)$ than any

[1] In the literature this loss is called the structured hinge loss (Taskar *et al*., 2004) or the max-margin loss for the structured case (Collins *et al*., 2008). Even though the former name is more common, we will prefer the latter one in this paper to avoid confusion with the hinge loss for a structure that is already defined in (2).

other structure, at least for the size of the margin for that structure $L(\boldsymbol{y}^n, \cdot)$, while the introduced N slack variables ξ_n should handle the non-separable case.

In this paper we will consider the *average sum* (AS) *loss* $\ell_n^{\text{AS}}(\mathbf{w})$ defined as

$$\ell_n^{\text{AS}}(\mathbf{w}) = \frac{1}{|\mathscr{Y}_{-n}|} \sum_{\boldsymbol{y} \in \mathscr{Y}_{-n}} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})), \tag{7}$$

which represents the expected hinge loss for structures inside the n-th example. If the AS loss is used in problem (1), it leads to the corresponding constrained optimization problem:

$$\min_{\mathbf{w}, \boldsymbol{\xi}} \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{N} \sum_{n=1}^{N} \frac{1}{|\mathscr{Y}_{-n}|} \sum_{\boldsymbol{y} \in \mathscr{Y}_{-n}} \xi_{n,\boldsymbol{y}} \tag{8}$$

subject to

$$\begin{aligned} \mathbf{w}^{\mathsf{T}} \Delta\mathbf{F}_n(\boldsymbol{y}) &\geq L(\boldsymbol{y}^n, \boldsymbol{y}) - \xi_{n,\boldsymbol{y}}, \\ \xi_{n,\boldsymbol{y}} &\geq 0, \quad \forall n, \forall \boldsymbol{y} \in \mathscr{Y}_{-n}, \end{aligned} \tag{9}$$

where now one non-negative slack variable is assigned to each output structure. Using one slack variable per output structure inside one example can be seen as a structural generalization of the Weston and Watkins (1998) multi-class SVM, where slack variables are assigned to possible classes inside an example.

3. Structured Pegasos algorithms

3.1. Pegasos with the max-margin loss. Pegasos is a sub-gradient method introduced by Shalev-Shwartz *et al.* (2007). At each iteration t the algorithm chooses a set $A_t \subseteq \{1, \ldots, N\}$ of cardinality k. Then the objective function (1) is approximated with

$$f^{\text{MM}}(\mathbf{w}, A_t) = \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{k} \sum_{n \in A_t} \ell(\mathbf{w}; (\boldsymbol{x}^n, \widetilde{\boldsymbol{y}}_n)) \tag{10}$$

and optimized using the sub-gradient descent $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \nabla_t^{\text{MM}}$, with the value of the approximate objective sub-gradient

$$\nabla_t^{\text{MM}} = \lambda \mathbf{w}_t - \frac{1}{k} \sum_{n \in A_t^+} \Delta\mathbf{F}_n(\widetilde{\boldsymbol{y}}_n), \tag{11}$$

$A_t^+ = \{n \in A_t : \ell(\mathbf{w}_t; (\boldsymbol{x}^n, \widetilde{\boldsymbol{y}}_n)) > 0\}$, where the step size is set to $\eta_t = 1/(\lambda t)$.

After each sub-gradient step, the parameters can be optionally projected on the ball of radius $1/\sqrt{\lambda}$ with the update

$$\mathbf{w}_{t+1} \leftarrow \min\left\{1, \frac{1/\sqrt{\lambda}}{\|\mathbf{w}_{t+1}\|}\right\} \mathbf{w}_{t+1}. \tag{12}$$

The pseudocode is presented in Algorithm 1. In the case of $k = 1$ the update corresponds to the stochastic version, for $k = N$ this is the standard (batch) version and for $1 < k < N$ it is called the mini-batch version.

Algorithm 1: Structured Pegasos with the MM loss (Shalev-Shwartz *et al.*, 2011).

Input : Training data: $\mathscr{D} = ((\boldsymbol{x}^n, \boldsymbol{y}^n))_{n=1}^N$, parameter $\lambda \in \mathbb{R}^+$, $k \in \mathbb{N}$
Number of iterations: T
Output: Model parameters: $\mathbf{w}$

$\mathbf{w} := \mathbf{0}$;
for $t := 1$ **to** T **do**
  Choose $A_t \subseteq \{1, \ldots N\}$ so that $|A_t| = k$;
  foreach $n \in A_t$ **do**
    Find $\widetilde{\boldsymbol{y}}_n = \arg\max_{\boldsymbol{y}} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y}))$;
    `/* single best decoding */`
  $A_t^+ := \{n \in A_t : \ell(\mathbf{w}; (\boldsymbol{x}^n, \widetilde{\boldsymbol{y}}_n)) > 0\}$;
  $\eta_t := 1/(\lambda t)$;
  $\mathbf{w} := (1 - \eta_t \lambda)\mathbf{w} + \frac{\eta_t}{k} \sum_{n \in A_t^+} \Delta\mathbf{F}_n(\widetilde{\boldsymbol{y}}_n)$;
  [Optional: $\mathbf{w} := \min\left\{1, \frac{1/\sqrt{\lambda}}{\|\mathbf{w}\|}\right\} \mathbf{w}$];
  `/* projection */`

3.2. Pegasos with the averaged sum loss. Let us consider using the AS loss and approximate the objective function (1) with

$$\frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{|A_t|} \sum_{n \in A_t} \frac{1}{|B_n|} \sum_{\boldsymbol{y} \in B_n} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})), \tag{13}$$

where $B_n \subseteq \mathscr{Y}_{-n}$ and $A_t \subseteq \{1, \ldots, N\}$ contains the set of examples on which the approximation is made. Further on, we will consider the previous approximation restricted only to the n-th example, i.e., where we choose $A_t = \{n\}$ and define

$$f^{\text{AS}}(\mathbf{w}, B_n) = \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{|B_n|} \sum_{\boldsymbol{y} \in B_n} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})). \tag{14}$$

This restriction allows us to obtain an on-line algorithm through examples with mini-batch optimization inside each example according to set B_n, while the selection of B_n allows us to choose which structures we will consider in the optimization process. We consider a sub-gradient of the approximate objective (14) given by

$$\nabla^{\text{AS}} = \lambda \mathbf{w} - \frac{1}{|B_n|} \sum_{\boldsymbol{y} \in B_n^+} \Delta\mathbf{F}_n(\boldsymbol{y}), \tag{15}$$

where $B_n^+ = \{\boldsymbol{y} \in B_n : \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})) > 0\}$. Thus the parameter update for the structured Pegasos with the AS loss in the t-th iteration on the n-th example is

$$\mathbf{w}_{t+1} = (1 - \eta_t \lambda)\mathbf{w}_t + \frac{\eta_t}{|B_n|} \sum_{\boldsymbol{y} \in B_n^+} \Delta\mathbf{F}_n(\boldsymbol{y}). \tag{16}$$

Let us define the *prediction violation set of structures*, S_n, as

$$S_n = \{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n) : \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})) \geq \ell(\mathbf{w}; (\boldsymbol{x}^n, \widehat{\boldsymbol{y}}_n))\},$$

where the *prediction structure* $\widehat{\boldsymbol{y}}_n$ is given by

$$\widehat{\boldsymbol{y}}_n = \arg \max_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \mathbf{w}^{\mathsf{T}} \mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}). \tag{17}$$

In theoretical analysis, we will consider the version of Pegasos with the AS loss where the selection of the set B_n is not from all $\mathscr{Y}_{-n}$ structures but only from S_n, and to such a restriction we will refer as the *restricted Pegasos* algorithm. The way we choose the set B_n from S_n of size k is not important for further analysis. Note that by choosing $B_n = \{\widetilde{\boldsymbol{y}}_n\}$ the algorithm is reduced to stochastic Pegasos with the MM loss. Also note that it is possible to select A_t with a cardinality greater than one, and the algorithm will operate over multiple structures inside each of selected examples in one update.

Pegasos with the *k*-best loss. Let $Best_n^k$ denote the set of k structures with the highest score on the n-th example, i.e., the structures which maximize the value of $\ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y}))$. Further, we can define the *k-best loss* as

$$\ell_n^{k\text{best}}(\mathbf{w}) = \frac{1}{k} \sum_{\boldsymbol{y} \in Best_n^k} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})),$$

and the corresponding objective function restricted to the n-th example as

$$f^{Best_n^k}(\mathbf{w}) = f^{\text{AS}}(\mathbf{w}, Best_n^k). \tag{18}$$

According to (18), we can see the k-best objective as a special case of the AS objective approximation (14) which is made on the $Best_n^k$ set. Also we can see that the k-best loss lies between the MM loss and the AS loss, i.e., $\ell_n^{\text{AS}}(\mathbf{w}) \leq \ell_n^{k\text{best}}(\mathbf{w}) \leq \ell_n^{\text{MM}}(\mathbf{w})$.

The k-best loss is convex (Boyd and Vandenberghe, 2004), and we can apply the Pegasos algorithm for optimization with the sub-gradient and parameter update defined with (15) and (16) by setting $B_n = Best_n^k$. If we choose B_n to be $Best_n^k$, not a subset from S_n, such a version can also be called k-best Pegasos, as it works in a similar framework as the k-best MIRA by Crammer *et al*. (2005) and it will directly optimize the k-best loss. Moreover, we can use k-best decoding to find structures from the prediction violation set. Since we need k output structures (if they exist) with the loss greater than that for the prediction structure, we do this by finding $Best_n^k$ and removing structures which do not belong to S_n. The pseudocode is presented in Algorithm 2.

Algorithm 2: (Restricted) Structured Pegasos with the AS loss.

Input : Training data: $\mathscr{D} = ((\boldsymbol{x}^n, \boldsymbol{y}^n))_{n=1}^N$,
parameter $\lambda \in \mathbb{R}^+$, $k \in \mathbb{N}$
Number of iterations: T
Output: Model parameters: $\mathbf{w}$

$\mathbf{w} := \mathbf{0}$;
for $t := 1$ **to** T **do**
  Choose n from $\{1, \dots N\}$;
  Select $B_n \subset \mathscr{Y}(\boldsymbol{x}^n)$ of size k
  e.g. $B_n := k\text{-}\arg\max_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y}))$;
  `/* k-best decoding */`
  $\widehat{\boldsymbol{y}}_n := \arg\max_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \mathbf{w}^{\mathsf{T}} \mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y})$;
  `/* prediction sequence */`
  $B_n := \{\boldsymbol{y} \in B_n : \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})) \geq \ell(\mathbf{w}; (\boldsymbol{x}^n, \widehat{\boldsymbol{y}}_n))\}$; `/* for restricted version */`
  $B_n^+ := \{\boldsymbol{y} \in B_n : L(\boldsymbol{y}^n, \boldsymbol{y}) > \mathbf{w}^{\mathsf{T}} \Delta \mathbf{F}_n(\boldsymbol{y})\}$;
  $\eta_t := 1/(\lambda t)$;
  $\mathbf{w} := (1 - \eta_t \lambda)\mathbf{w} + \frac{\eta_t}{|B_n|} \sum_{\boldsymbol{y} \in B_n^+} \Delta \mathbf{F}_n(\boldsymbol{y})$;
  [Optional: $\mathbf{w} := \min\left\{1, \frac{1/\sqrt{\lambda}}{\|\mathbf{w}\|}\right\} \mathbf{w}$];
  `/* projection */`

4. Theoretical analysis

In the structured case we care about the cumulative bound of prediction losses through the iterations between the prediction structure $\widehat{\boldsymbol{y}}_n$ and the true structure $\boldsymbol{y}^n$, i.e., the sum over $L(\boldsymbol{y}^n, \widehat{\boldsymbol{y}}_n)$. This bound for the stochastic sub-gradient method with the MM loss is given by Ratliff *et al*. (2006), and we will provide a bound for restricted Pegasos with the AS loss. First we need the following lemma by Shalev-Shwartz *et al*. (2011). Recall that a function f is λ-strongly convex if $f(\mathbf{w}) - \frac{\lambda}{2} \|\mathbf{w}\|^2$ is a convex function.

Lemma 1. (Shalev-Shwartz *et al.*, 2011) *Let $f_1, \dots, f_T$ be a sequence of λ-convex functions and D be a closed convex set. Define $\Pi_D(\mathbf{w}) = \arg\min_{\mathbf{w}' \in D} \|\mathbf{w} - \mathbf{w}'\|$. Let $\mathbf{w}_1, \dots, \mathbf{w}_{T+1}$ be a sequence of vectors such that $\mathbf{w}_1 \in D$ and, for $t \geq 1$, $\mathbf{w}_{t+1} = \Pi_D(\mathbf{w}_t - \eta_t \nabla_t)$, where ∇_t belongs to the sub-gradient set of f_t at $\mathbf{w}_t$ and $\eta_t = 1/\lambda t$. Assume that, for all t, $\|\nabla_t\| \leq G$. Then, for all $\mathbf{u} \in D$ it follows that*

$$\frac{1}{T} \sum_{t=1}^{T} f_t(\mathbf{w}_t) \leq \frac{1}{T} \sum_{t=1}^{T} f_t(\mathbf{u}) + \frac{G^2(1 + \ln T)}{2\lambda T}.$$

Theorem 1. *Let $(\boldsymbol{x}^1, \boldsymbol{y}^1), \dots, (\boldsymbol{x}^N, \boldsymbol{y}^N)$ be a sequence of examples where $\|\Delta \mathbf{F}_n(\boldsymbol{y})\| \leq R$ and $L(\boldsymbol{y}^n, \boldsymbol{y}) \leq 1$ for all $\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)$, $n = 1, \dots, N$ and $\mathbf{w}^* = \arg\min_{\mathbf{w}} f(\mathbf{w})$, where $f(\mathbf{w})$ is defined with loss function $\ell_n^{AS}(\mathbf{w})$. Then, for the update (16) with the optional pro-*

jection step (12) it follows that

$$\frac{1}{N}\sum_{n=1}^{N} f^{AS}(\mathbf{w}_n, B_n) \leq \frac{1}{N}\sum_{n=1}^{N} f^{AS}(\mathbf{w}^*, B_n) + \frac{c(1+\ln N)}{2\lambda N}$$

where $c = (\sqrt{\lambda} + R)^2$ *if we perform the projection step and* $c = 4R^2$ *otherwise.*

Proof. We first show that the conditions of Lemma 1 are satisfied. The function $f^{\mathrm{AS}}(\mathbf{w}_n, B_n)$ is a λ-convex function by definition. Further, if we use the projection step, then it follows that $\|\mathbf{w}_n\| \leq 1/\sqrt{\lambda}$ and $\|\nabla_n\| \leq \lambda + R$. If we do not use it, with a similar technique as employed by Shalev-Shwartz *et al*. (2011), we get $\|\mathbf{w}_n\| \leq R/\sqrt{\lambda}$ and $\|\nabla_n\| \leq 2R$.

Next, we want to show that $\mathbf{w}^* \in D$, which is obvious if we do not use the projection. If we use it, then for the primal problem (8)–(9) we have the corresponding dual problem

$$\max_{\boldsymbol{\alpha}} \sum_{n=1}^{N} \sum_{\boldsymbol{y}\in\mathscr{Y}_{-n}} \alpha_{n,\boldsymbol{y}} L_{n,\boldsymbol{y}} - \frac{1}{2}\Big\| \sum_{n=1}^{N} \sum_{\boldsymbol{y}\in\mathscr{Y}_{-n}} \alpha_{n,\boldsymbol{y}} \Delta\mathbf{F}_n(\boldsymbol{y}) \Big\|^2 \quad (19)$$

subject to

$$0 \leq \alpha_{n,\boldsymbol{y}} \leq \frac{C}{N|\mathscr{Y}_{-n}|}, \quad \forall n, \forall \boldsymbol{y} \in \mathscr{Y}_{-n}, \quad (20)$$

where $C = 1/\lambda$, $L_{n,\boldsymbol{y}}$ is an abbreviation for $L(\boldsymbol{y}^n, \boldsymbol{y})$, and the connection between primal and dual parameters is $\mathbf{w} = \sum_{n=1}^{N}\sum_{\boldsymbol{y}\in\mathscr{Y}_{-n}} \alpha_{n,\boldsymbol{y}} \Delta\mathbf{F}_n(\boldsymbol{y})$. If $(\mathbf{w}^*, \xi^*)$ is an optimal point for the primal problem and $\boldsymbol{\alpha}^*$ is an optimum for the dual, then from the strong duality at the optimum there is an equality between the primal and the dual objective value

$$\frac{1}{2}\|\mathbf{w}^*\|^2 \leq \frac{1}{2}\|\mathbf{w}^*\|^2 + \frac{C}{N}\sum_{n=1}^{N}\frac{1}{|\mathscr{Y}_{-n}|}\sum_{\boldsymbol{y}\in\mathscr{Y}_{-n}} \xi^*_{n,\boldsymbol{y}} = \sum_{n=1}^{N}\sum_{\boldsymbol{y}\in\mathscr{Y}_{-n}} \alpha^*_{n,\boldsymbol{y}} L_{n,\boldsymbol{y}} - \frac{1}{2}\|\mathbf{w}^*\|^2$$

where the first inequality is due to $\xi^*_{n,\boldsymbol{y}} \geq 0$.

Now, we obtain

$$\|\mathbf{w}^*\|^2 \leq \sum_{n=1}^{N}\sum_{\boldsymbol{y}\in\mathscr{Y}_{-n}} \alpha^*_{n,\boldsymbol{y}} L_{n,\boldsymbol{y}} \leq \sum_{n=1}^{N}\sum_{\boldsymbol{y}\in\mathscr{Y}_{-n}} \alpha^*_{n,\boldsymbol{y}} \leq \frac{1}{\lambda}$$

and $\|\mathbf{w}^*\| \leq \sqrt{1/\lambda}$. We can now apply Lemma 1 and get the desired bound. ■

Theorem 2. *Let the conditions from the previous theorem be satisfied and let* B_n *be chosen as* $B_n \subseteq S_n$. *Then it follows that*

$$\sum_{n=1}^{N} L(\boldsymbol{y}^n, \widehat{\boldsymbol{y}}_n) \leq \frac{\lambda}{2} N \|\mathbf{w}^*\|^2 + \frac{c(1+\ln N)}{2\lambda} + \sum_{n=1}^{N}\frac{1}{|B_n|}\sum_{\boldsymbol{y}\in B_n} \ell(\mathbf{w}^*; (\boldsymbol{x}^n, \boldsymbol{y})). \quad (21)$$

Proof. According to the definition of $f^{\mathrm{AS}}(\mathbf{w}_n, B_n)$, from the previous theorem we have

$$\begin{aligned}\frac{\lambda}{2N}\sum_{n=1}^{N}\|\mathbf{w}_n\|^2 + \frac{1}{N}\sum_{n=1}^{N}\frac{1}{|B_n|}\sum_{\boldsymbol{y}\in B_n}\ell(\mathbf{w}_n; (\boldsymbol{x}^n, \boldsymbol{y})) \\ \leq \frac{\lambda}{2}\|\mathbf{w}^*\|^2 + \frac{1}{N}\sum_{n=1}^{N}\frac{1}{|B_n|}\sum_{\boldsymbol{y}\in B_n}\ell(\mathbf{w}^*; (\boldsymbol{x}^n, \boldsymbol{y})) \\ + \frac{c(1+\ln N)}{2\lambda N}. \quad (22)\end{aligned}$$

Using the definition of $\widehat{\boldsymbol{y}}_n$, it follows that

$$\mathbf{w}^{\mathsf{T}}\mathbf{F}(\boldsymbol{x}^n, \widehat{\boldsymbol{y}}_n) \geq \mathbf{w}^{\mathsf{T}}\mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}), \quad \forall \boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n), \quad (23)$$

which leads to $\mathbf{w}^{\mathsf{T}}\Delta\mathbf{F}_n(\widehat{\boldsymbol{y}}_n) \leq 0$. Therefore,

$$\begin{aligned}L(\boldsymbol{y}^n, \widehat{\boldsymbol{y}}_n) &\leq \ell(\mathbf{w}; (\boldsymbol{x}^n, \widehat{\boldsymbol{y}}_n)) \\ &\leq \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})), \quad \forall \boldsymbol{y} \in B_n,\end{aligned}$$

where the last inequality follows since B_n is a subset from S_n. Now, we have

$$\begin{aligned}&\sum_{n=1}^{N} L(\boldsymbol{y}^n, \widehat{\boldsymbol{y}}_n) \\ &\leq \sum_{n=1}^{N}\frac{1}{|B_n|}\sum_{\boldsymbol{y}\in B_n}\ell(\mathbf{w}_n; (\boldsymbol{x}^n, \boldsymbol{y})) \\ &\leq \frac{\lambda}{2}\sum_{n=1}^{N}\|\mathbf{w}_n\|^2 \\ &\quad + \sum_{n=1}^{N}\frac{1}{|B_n|}\sum_{\boldsymbol{y}\in B_n}\ell(\mathbf{w}_n; (\boldsymbol{x}^n, \boldsymbol{y})),\end{aligned}$$

which, in combination with (22), provides the desired bound. ■

From the previous theorem and using the inequality

$$\frac{1}{|B_n|}\sum_{\boldsymbol{y}\in B_n}\ell(\mathbf{w}^*; (\boldsymbol{x}^n, \boldsymbol{y})) \leq \ell_n^{\mathrm{MM}}(\mathbf{w}^*), \quad \forall B_n \subseteq \mathscr{Y}_{-n}, \quad (24)$$

we get the following corollary.

Corollary 1. *Let the conditions from the previous theorem be satisfied. Then it follows that*

$$\sum_{n=1}^{N} L(\boldsymbol{y}^n, \widehat{\boldsymbol{y}}_n) \leq \frac{\lambda}{2} N \left\|\mathbf{w}^*\right\|^2 + \sum_{n=1}^{N} \ell_n^{MM}(\mathbf{w}^*) + \frac{c(1+\ln N)}{2\lambda}, \quad (25)$$

as well as

$$\sum_{n=1}^{N} L(\boldsymbol{y}^n, \widehat{\boldsymbol{y}}_n) \leq \sqrt{\frac{c(1+\ln N)}{N}} \left\|\mathbf{w}^*\right\| + \sum_{n=1}^{N} \ell_n^{MM}(\mathbf{w}^*),$$

by choosing

$$\lambda = \sqrt{\frac{c(1+\ln N)}{N \left\|\mathbf{w}^*\right\|^2}}.$$

If we set $B_n = \{\widetilde{\boldsymbol{y}}_n\}$ for each n, then the equality holds in (24) and the previous bound reduces to that of the stochastic version provided by Ratliff *et al*. (2006). For the other selection of B_n, according to the inequality (24), the right-hand side of (21) is at most the right-hand side of (25), so Corollary 1 states that Pegasos with the AS loss has at most the same bound of cumulative prediction losses as the stochastic Pegasos algorithm.

The Pegasos algorithm of Shalev-Shwartz *et al*. (2011) picks examples uniformly at random. Even though uniform sampling is not used in the previous theorems, it can improve the convergence rate of the method. Also, picking examples uniformly at random can be very helpful to eliminate problems in a dataset when the examples are grouped by some criteria in parts of the corpus and come in a particular order.

5. Implementation issues

Regarding implementation, there are two main operations that are performed over the parameter vector: *scaling*, when we first scale $\mathbf{w}_t$ by factor $(1 - \eta_t\lambda)$ and optionally once again in the projection step, and the operation *add*, where we add scaled feature vectors to current parameters multiple times. Shalev-Shwartz *et al*. (2011) present a sparse implementation where *scaling* can be done in $\mathcal{O}(1)$ and *add* a new feature vector in $\mathcal{O}(d)$, where d is the number of non-zero elements in the feature vector. This is done by representing the parameter vector as $\mathbf{w} = a\mathbf{v}$. They also consider averaged parameters,

$$\overline{\mathbf{w}}_T = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t, \quad (26)$$

and state that, in practice, the final hypothesis $\mathbf{w}_T$ often provides better results.

We do not have a theoretical analysis for averaged parameters, since we do not bound the overall objective $f(\overline{\mathbf{w}}_T)$ in the structured case with the AS loss. However, we provide an experimental analysis for averaged and non-averaged parameters in the next section. In order to calculate averaged parameters, we should not simply apply the formula (26) because we will not get the sparse updates. Xu (2011) presents an efficient procedure to find averaged parameters using a linear transformation where the addition of a new feature vector is also done in $\mathcal{O}(d)$. In practical implementations, both averaged and non-averaged parameters require rescaling from time to time since the variables can go out of range. In a non-averaged implementation it can be easily done by rescaling a to one, while rescaling for averaged parameters can be found in the implementation of Bottou (2008). Note that rescaling is not a sparse operation.

The algorithm also requires selecting the set B_n from the prediction violation set S_n. Checking if the structure belongs to the set S_n is an easy task, although building such a set can be a problem as we need to collect all structures with the score greater than that for the prediction structure. Fortunately, the algorithm does not require the calculation of the whole set as we need only an arbitrary portion of its elements to approximate the objective function. Since we need structures with the highest score, we can use a k-best inference to create the $Best_n^k$ set with the top k structures in a descending order, and then we can easily remove structures which do not belong to S_n to get the required set B_n from $S_n \cap Best_n^k$.

For sequence labeling, the Viterbi (1967) algorithm can be straightforwardly extended to the k-best variant by keeping its k-best partial scores at every position. Storing the k-best partial scores at each position of a sequence of length T can be done using the matrix, leading to the time complexity $\mathcal{O}(|Y|^2Tk)$ and the memory complexity $\mathcal{O}(|Y|Tk)$, where Y represents the set of possible labels for each observation. Another approach, which we apply in our experiments, is to use the A* search to generate k-best paths on trellis (Soong and Huang, 1991; Nagata, 1994). This algorithm can also be adapted to generate exact k-best paths with the involved loss function. The total time complexity is $\mathcal{O}(|Y|^2T + |Y|Tk\log k)$ and the total memory complexity is $\mathcal{O}(|Y|T + kT)$, which can be a better choice than the k-best Viterbi algorithm.

6. Experimental results

We present experimental results on shallow parsing (Tjong Kim Sang and Buchholz, 2000) on the CONLL-2000 corpus[2] and part-of-speech (POS) tagging on the Brown corpus[3]. We choose these problems for experiments as they are important tasks which come usually as first steps in pipeline structures of natural language processing problems. POS tagging belongs to

[2] http://www.cnts.ua.ac.be/conll2000/chunking.

[3] http://khnt.aksis.uib.no/icame/manuals/brown/.

Table 1. Templates used for generating features at position t in a sequence, where w_t denotes the current word, a_t attributes for the current word and y_t the corresponding label.

(y_t, y_{t-1})	
(y_t, w_t)	(y_t, a_t)
(y_t, y_{t-1}, w_t)	(y_t, y_{t-1}, a_t)
$(y_t, y_{t-1}, w_t, w_{t-1})$	$(y_t, y_{t-1}, w_t, a_{t-1})$
$(y_t, y_{t-1}, a_t, w_{t-1})$	$(y_t, y_{t-1}, a_t, a_{t-1})$

sequence labeling problems where we need to assign a single label to each member of the observed sequence. The label represents a grammatical category, i.e., part of speech, for the corresponding word and its context in a sentence. Shallow parsing identifies non-overlapping text segments which correspond to certain syntactic units. It is usually a step preceding full parsing and following POS tagging. The results are presented in terms of the F-measure, as the harmonic mean of precision and recall computed over tokens belonging to a chunk, while for POS tagging they are presented as accuracy, i.e., the proportion of correctly classified labels over all tokens in a sentence.

Choosing an example for the update in the Pegasos algorithm with the average sum loss is done sequentially, while for the MM loss the training set is partitioned into parts of size k on which the updates are performed sequentially. One pass through all training examples will be referred to as an *epoch*. In the forthcoming discussion we use the following abbreviations to specify the case we tested: the prefix *avg* before the algorithm name means that the test results are produced with the averaged parameters (26), the Pegasos algorithm will be abbreviated with Peg, its restricted version with resPeg, the suffixes MML and ASL after the algorithm name will respectively refer to the max-margin loss and the average sum loss, and -WP at the end will denote that the Pegasos algorithm is used without the projection step.

6.1. Features. In sequence labeling problems we can write a global feature vector as the sum over feature functions from all positions t in a sequence, i.e.,

$$\mathbf{F}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{t=1}^{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t).$$

Each feature vector at specific position $\boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t) \in \mathbb{R}^M$ is usually presented with binary-coded active features of the current transition and the current context around the t-th observation. Table 1 shows the templates used for generating features at each position in a sentence.

In addition to its label, each observation (word) will also have a corresponding *attribute*. The attribute will represent specific characteristics of a word. For part-of-speech tagging we used standard characteristics for describing words: if a token is written in all caps, the initial cap or lowercase, if a token contains digits, if it represents a special character, a punctuation mark or an abbreviation; and we did not use external linguistic resources. For shallow parsing, we used the corresponding POS tag which is already given in the corpora, in combination with previously described characteristics of a word and its belonging to specific external linguistic dictionaries.

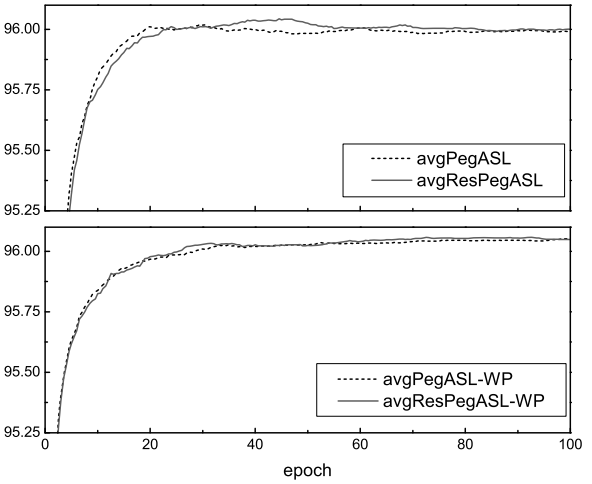

Fig. 1. Results in terms of the F-measure through epochs for the restricted and non-restricted version of the Pegasos algorithm with the averaged sum loss on shallow parsing. The curves are drawn with $k = 10$.

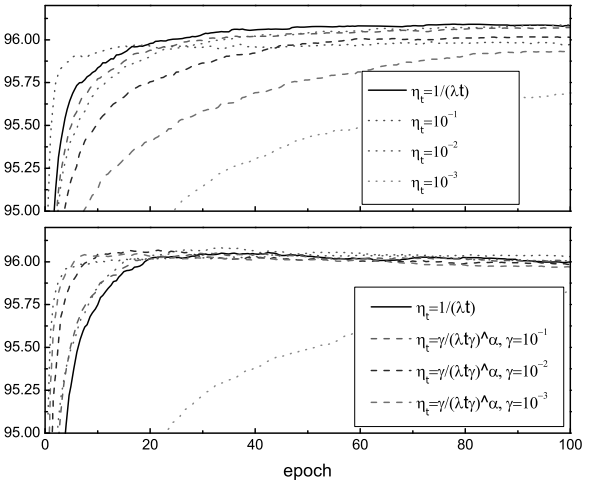

Fig. 2. Results in terms of the F-measure through epochs with different step sizes for the stochastic Pegasos algorithm without a projection step (top panel) and with a projection step (bottom panel). The specified step sizes are common for both panels with $\alpha = 0.75$, but are shown in two legends for better clarity.

6.2. Restricted vs. non-restricted version. We first compare the results for Pegasos with the AS loss and the corresponding restricted version. Recall that the restriction is made by selecting the set B_n as the subset of S_n, and that it was needed for the theoretical analysis. Also note that using the restricted version implies the calculation of the prediction sequence (17), see Algorithm 2, which increases the training time since the additional Viterbi decoding must be performed. In Fig. 1 we see minor differences in results whenever Pegasos with the AS loss is used with or without restriction. The specific parameter k is presented in caption, and there were similar small differences with other parameters we tested. Since the results are so similar, with the only difference in training time, in further analysis we only include restricted Pegasos in time comparison with other algorithms and in the last table with the results of all methods.

6.3. Dependence of the regularization parameter λ. Figure 3 presents the influence of the regularization parameter λ on the results for shallow parsing and POS tagging. Small values of the regularization parameters need more iterations, i.e., long runtimes, which is mentioned by Shalev-Shwartz *et al*. (2011) and can be clearly seen from Fig. 3 for both shallow parsing and POS tagging. However, large regularization parameters produce almost no difference in results after 20 and 100 epochs, but the outcomes are not satisfying. Interestingly, the best results on both the datasets are achieved when the projection is not used, even if theoretical analysis provides similar bounds for both the versions.

In order to select the regularization parameter in further experiments, we perform cross-validation. We use 5-fold cross-validation to find the optimal parameter for each method separately, and then we employ this optimal parameter in a test scenario in all figures. When we present curves as the dependence of results through epochs, we use the optimal parameter provided after 100 training epochs in cross-validation. Further, in experiments we will include a case when we present only final results, and then the optimal number of training epochs will also be selected and included in cross-validation, which will be described later (see Section 6.8).

6.4. Different step sizes. The Pegasos algorithm in the t-th iteration changes parameters with the step size $\eta_t = 1/(\lambda t)$. This step can be generalized as $\eta_t = 1/(\lambda t)^\alpha$ with $\alpha = 1$. However, other values of $\alpha \in (1/2, 1]$ can be used, as suggested by Bach and Moulines (2011). The step can be further generalized as $\eta_t = \gamma/(\lambda\gamma t)^\alpha$ with a constant γ which is used in the ASGD implementation of Bottou (2008) with $\alpha = 0.75$. Another approach is to employ a constant small step size in each iteration (Ratliff *et al*., 2006).

In Fig. 2 we present an experiment where we tested different step sizes. As noticed by Nemirovski *et al*. (2009), with $\alpha = 1$ the choice of the regularization parameter is critical. However, as we have described before, we perform cross-validation to choose the optimal regularization parameter, and the results using the Pegasos step are very similar to those with the constant step size equal to 10^{-2} or using $\eta_t = \gamma/(\lambda\gamma t)^\alpha$ with $\gamma = 10^{-1}$ when the projection step is not performed (top panel). A larger constant step size $\eta_t = 10^{-1}$ can provide faster convergence in the first few epochs, but in the end it does not achieve very good results. In contrast to the results without projection where other step sizes do not seem to be beneficial, we can see from the bottom panel the improvement of other step sizes in the first iterations when the projection step is included. After all epochs in that case, all step sizes provide very similar results, except a very small constant size $\eta_t = 10^{-3}$, with the best results achieved with $\eta_t = 10^{-1}$.

6.5. Projection and averaged parameters. Using the averaged parameters helps to avoid oscillations in the test results. Figure 4 shows the oscillations when the averaged parameters are not used. The Pegasos algorithm with averaged parameters slowly converges, but after 100 epochs it usually provides results similar to the non-averaged case. It is also possible to use a mixed approach that combines both the cases by starting averaging after some portion of training iterations, as suggested by Rakhlin *et al*. (2012), which should result in faster convergence. An open question is when or whether or not to start averaging (Shamir, 2012). Rakhlin *et al*. (2012) use averaging after $T/2$ iterations, Xu (2011) employs a comparison of the moving average of the empirical loss of non-averaged parameters and exponential moving average parameters to determine when to start averaging, while the implementation of Bottou (2008) is by the default set to averaging after the first epoch. Additionally, Shamir and Zhang (2012) propose a simple averaging scheme which can be performed with other stopping criteria on-the-fly, with an unknown number of training iterations in advance.

6.6. Max-margin loss vs. averaged sum loss. Figure 5 presents the results for the Pegasos algorithm with the MM and AS loss. In the left panel we see that both algorithms converge to the same results irrespective of whether or not we use the projection. As expected, the convergence in terms of the number of epochs through the training set is faster for Pegasos with the AS loss, since it considers many more structures for the update. On the other hand, when comparing convergence in terms

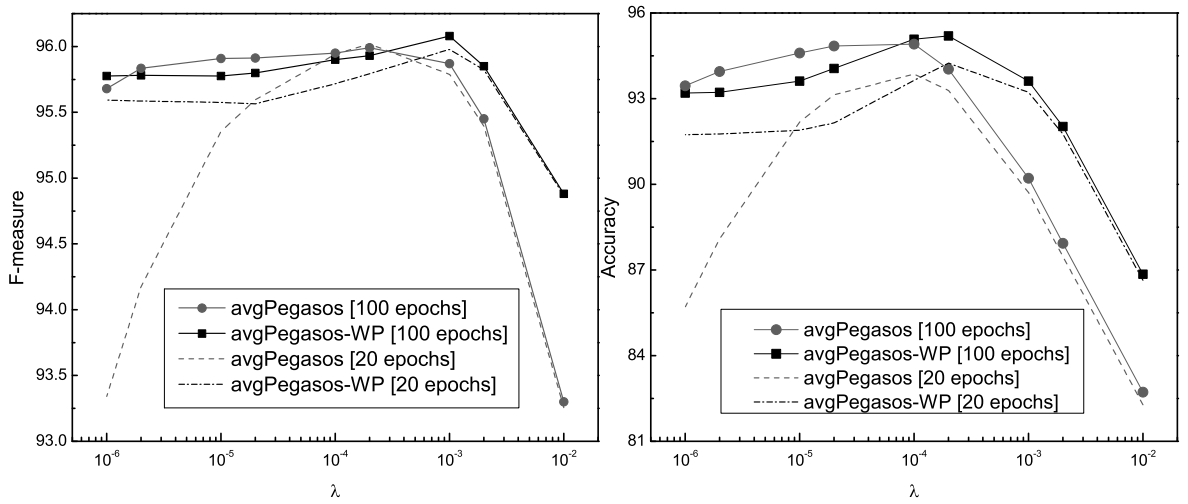

Fig. 3. Dependence on the regularization parameter λ for the stochastic Pegasos algorithm. The results are presented with the averaged parameters after 20 and 100 training epochs. Shallow parsing (left) and POS tagging (right).

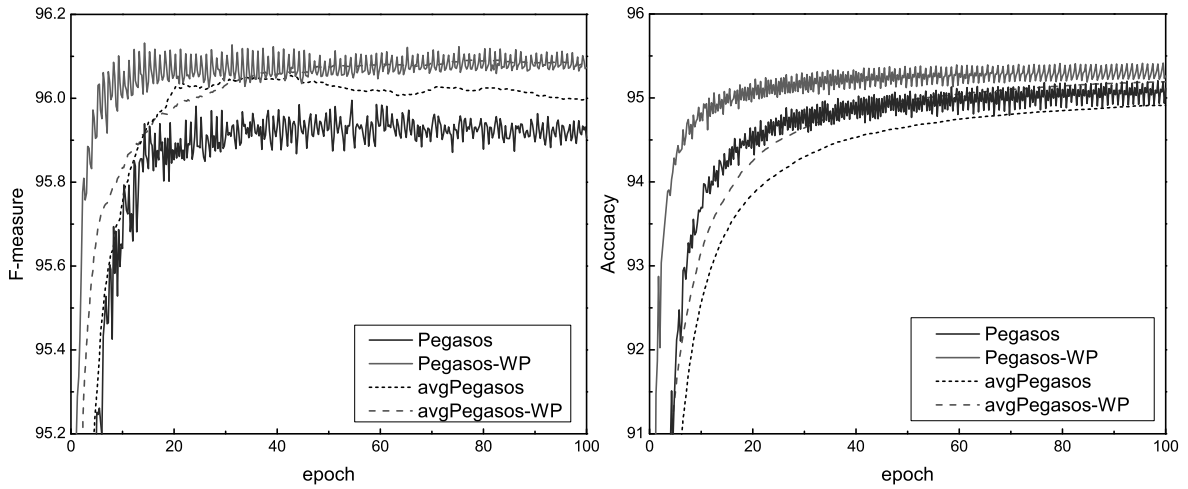

Fig. 4. Results for shallow parsing (left) and POS tagging (right) through the iterations for the stochastic Pegasos algorithm dependent on the use of projection and averaged parameters.

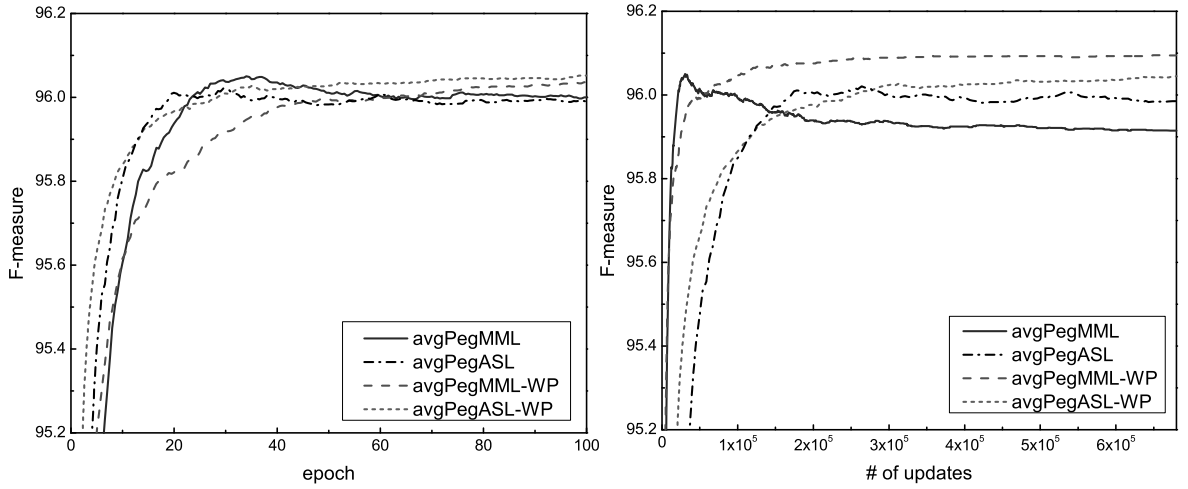

Fig. 5. Comparison between Pegasos algorithms with the MM and AS loss. The dependence of the F-measure through epochs (left) and the dependence on the number of parameter updates (right). All curves are presented with the averaged parameters and with $k = 10$ for the shallow parsing problem.

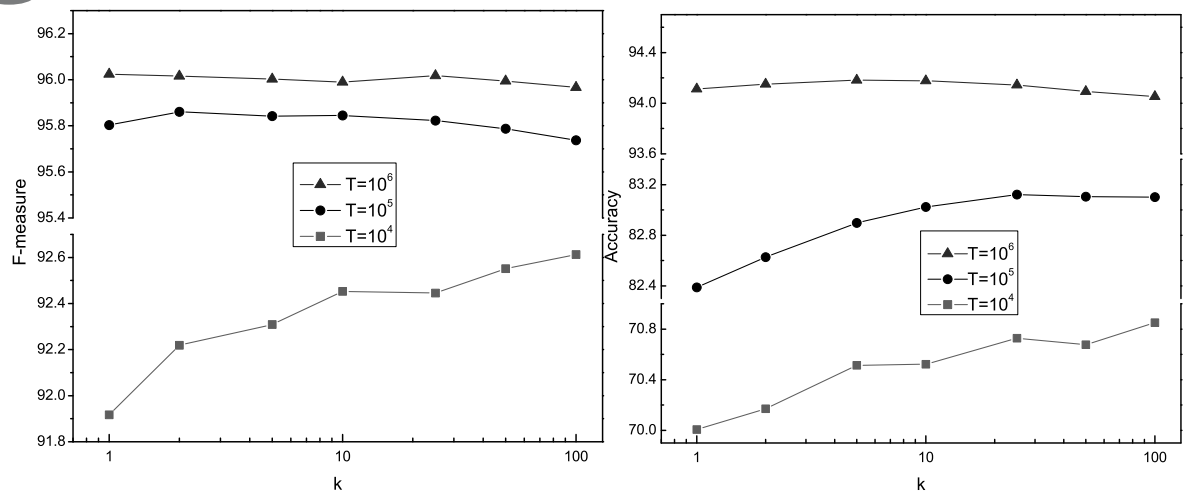

Fig. 6. Results for the Pegasos algorithm for different values of parameter k after a fixed number of iterations specified in the legend. Results on shallow parsing (left) and POS tagging (right).

of the number of updates, the convergence is faster when the MM loss is used, since it considers structures from k different training examples in one update. Even if they converge to the same values, the choice of the algorithm will depend on whether we want to obtain the maximum results with a fewer number of updates or with a fewer number of examples presented to the algorithm. However, as we will see in further analysis, if we disregard the number of epochs/updates performed by the algorithm and consider only the training time, then the stochastic version will be more suitable since the decoding time plays an important role in the overall runtime.

6.7. Dependence of parameter *k*. The influence of parameter k is shown in Fig. 6 with similar behavior on the given corpora. The optimal regularization parameter is found during the cross-validation separately for each point in the figure, which is defined with k provided on the x axis and the number of training iterations provided in legend. With a smaller number of iterations, the results get better as we increase k for both shallow parsing and POS tagging. This is expected since we extract more information from each example and learn based on k-best structures.

Also when the number of iterations is large, i.e., when the algorithm converges to its best result, all values of parameter k provide similar results with slight degradation for a very large k, e.g., 25, 50, 100. The reason is probably that including a lot of structures into the training procedure over numerous iterations creates many active features and may lead to overfitting. Thus, in this scenario, increasing k is not useful and the single best version is enough to get very good results. However, achieving better results by increasing k with a smaller number of iterations shows us where the k-best version

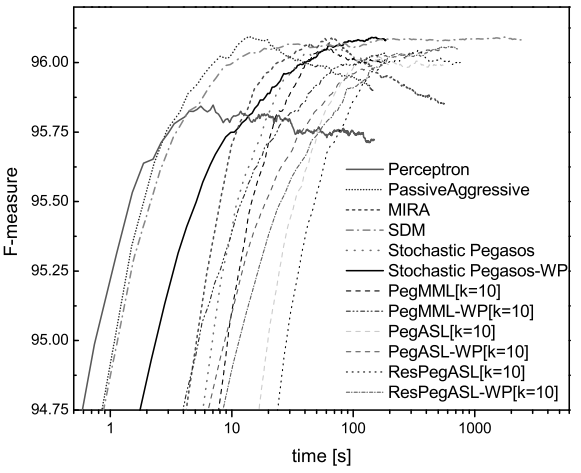

Fig. 7. Training time comparison for different algorithms. For all algorithms, the results are presented with averaged parameters. The horizontal axis represents the training time, and the curves show the F-measure on test corpora after various time spent on training. All curves represent the training time for 100 epoch on the shallow parsing problem and $k = 10$ is set for the MIRA. (Best seen in color.)

can be useful. If we want to train sequentially in one pass, or if we have a model which should be online corrected when the new example is presented, dealing with k higher than one should increase results as it includes more information from the new example.

6.8. Comparison with other algorithms. Finally we compare the Pegasos algorithm with other popular sequential structured algorithms briefly described in the following text.

The **Perceptron algorithm** for structured

classification was introduced by Collins (2002). It always updates the parameter vector on the 'best' structure $\widetilde{\boldsymbol{y}}_n$ with a fixed step size. It is the easiest to implement and fastest algorithm, and it does not have additional hyperparameters to be tuned. Even though it is a simple method, it can provide very good results. Also, other methods which include optimization of the step size and incorporate a loss function, might further improve the results. Since Perceptron changes parameters without a predetermined step size in an online manner, using averaged parameters is very important, and they significantly contribute to avoiding very large oscillations in results.

The **passive aggressive (PA) algorithm** was introduced by Crammer *et al*. (2006). This is an online algorithm that considers optimization with only one constraint at a time generated on the 'best' structure $\widetilde{\boldsymbol{y}}_n$. It also deals with the soft-margin by introducing a slack variable into the optimization problem. Since only one constraint is used, the solution can be found in analytical form.

The **sequential dual method (SDM)** for structural SVMs, described in Balamurugan *et al*. (2011), traverses through training examples and, at the n-th example, its working set of constraints is extended by the one generated for the best structure $\widetilde{\boldsymbol{y}}_n$. Then sequential minimal optimization (SMO) (Platt, 1999) is applied to optimize dual variables associated with structures inside the working set. The algorithm can also be sped up by introducing heuristics to control the growth of the working set as described by the authors. In our experiments we do not use any additional heuristics.

The ***k*-best margin infused relaxed algorithm (MIRA)** (Crammer *et al*., 2005) is an online algorithm which restricts optimization not only to one (as the PA algorithm does) but to k-best (most violated) constraints. At each example, after decoding k outputs with the highest score, the algorithm minimizes the norm of parameter change while satisfying constraints on generated outputs. The minimization problem cannot be solved analytically and an appropriate solver is needed. We use SMO for this purpose. Note that for $k = 1$ the solution can be found analytically, and thus the MIRA in this case is reduced to the PA algorithm.

We have implemented all algorithms in C++ for comparison. The experiments are performed on a computer with an Intel Core i7-3612QM CPU 2.10 GHz and 8 GB RAM. Figure 7 shows the dependence of the F-measure through the time spent on training for previously mentioned algorithms together with different versions of the Pegasos algorithm. All implemented algorithms share the same structures and operators when working with feature and parameter vectors. Thus the time comparison shown in Fig. 7 can be considered reliable.

Figure 7 shows the amount of time that is needed with $k = 10$ for the AS and MM loss in comparison with stochastic Pegasos. Also the restricted version is included, which is one of the slowest since it needs additional decoding. For other online algorithms, we can see that after achieving the best results there is a degradation in the F-measure on test corpora. They all need a few epochs to converge, so they need not be trained for 100 epochs since it can raise the problem of overfitting. This is especially evident with the MIRA, because at each iteration it must satisfy all k new generated constraints in an online manner, which can notably make changes from the previous parameters. On the other hand, Pegasos algorithms need more iterations to achieve the highest results with averaged parameters, as we have seen before. Speaking of the highest results, we can see that all algorithms, except for the Perceptron, obtain very similar highest F-measures with slight differences among them. However, they mostly differ in terms of training time, the F-measure dependence on the number of training epochs and on the regularization parameter.

We can get the same conclusions from Table 2. It provides final results for each method on both datasets with the corresponding parameters selected via cross-validation. In order to select the stopping criteria during the cross-validation up to 100 epochs for shallow parsing (200 epochs for POS tagging), we selected the best results between $\{10, 20, 30, 50, 100\}$ epochs for shallow parsing (including 200 for POS tagging). Therefore, the combination of the best pair (regularization, number of epochs) is presented and then used in a test scenario. If a method has a hyperparameter k, then it is also cross-validated from the set $\{2, 5, 10\}$. As mentioned before, we can see that different versions of Pegasos with projection consistently provide lower results than the corresponding versions without projection, even though the theoretical analysis provides similar bounds for both versions. Also, the optimal values of parameter k in Table 2 are 2 or 5 for all algorithms, which shows that the problems do not need a lot of additional structures to achieve the best results with enough training epochs.

From the previous experiments, we can see that the results are consistent on both the problems. The dependence of the regularization parameter, parameter k, whether we use the averaging option, the restricted version or the projection, shows similar behavior on the both corpora.

We have presented the results for sequence labeling problems. However, the method is applicable to other domains wherever we can define a structured classification problem and define a prediction violation set by k-best inference exactly or by some approximation.

Table 2. Results for different algorithms and their corresponding parameters (regularization parameters, parameter k, number of training epochs) obtained from 5-fold cross-validation for each dataset. For all algorithms, the results are presented with averaged parameters, so the *avg* prefix is omitted in the algorithm name. The parameter C_d denotes the dual regularization parameter, where in the PA algorithm it means that the dual parameters are upper bound with C_d, while in the MIRA and SDM it means that the sum of dual parameters is equal to C_d inside one example.

	Shallow parsing				Pos tagging			
Method	Reg.	k	# epoch	F-measure	Reg.	k	# epoch	Accuracy
Perceptron	–	–	20	95.798	–	–	30	94.829
PassiveAggressive	$C_d = 1$	–	10	96.093	$C_d = 10^{-2}$	–	100	95.189
MIRA	$C_d = 10^{-2}$	2	30	96.095	$C_d = 10^{-2}$	5	50	95.214
SDM	$C_d = 10^{-1}$	–	100	96.084	$C_d = 10^{-1}$	–	200	95.292
Stochastic Peg	$\lambda = 2 \cdot 10^{-3}$	–	30	96.041	$\lambda = 10^{-4}$	–	200	95.065
Stochastic Peg-WP	$\lambda = 10^{-3}$	–	100	96.082	$\lambda = 2 \cdot 10^{-3}$	–	200	95.320
PegMML	$\lambda = 2 \cdot 10^{-3}$	5	30	96.044	$\lambda = 10^{-4}$	2	200	95.068
PegMML-WP	$\lambda = 10^{-3}$	2	100	96.094	$\lambda = 2 \cdot 10^{-3}$	2	200	95.302
PegASL	$\lambda = 2 \cdot 10^{-3}$	2	30	96.047	$\lambda = 10^{-4}$	5	200	95.070
PegASL-WP	$\lambda = 10^{-3}$	2	50	96.076	$\lambda = 2 \cdot 10^{-3}$	5	200	95.317
ResPegASL	$\lambda = 2 \cdot 10^{-3}$	5	30	96.034	$\lambda = 10^{-4}$	2	200	95.066
ResPegASL-WP	$\lambda = 10^{-3}$	2	50	96.078	$\lambda = 2 \cdot 10^{-3}$	5	200	95.298

7. Conclusion

We have presented an iterative primal sub-gradient method to optimize a structured SVM with the averaged sum of hinge losses, which can deal with k different structures at a time inside one example. In the theoretical analysis, we have shown that the bound of cumulative prediction losses is at most like the bound for the stochastic version, while the empirical evaluation suggests that, with a smaller number of iterations, increasing k contributes to improving the results.

In contrast to the existent mini-batch version with the max-margin loss, which can be suitable for parallel implementation, the advantage of the version with the average sum loss is extracting more information from each example. It should be useful, for instance, when the existing model should be corrected online as a new example is presented. However, dealing with multiple structures is not quite useful when we train the model with many iterations and already include enough structures, where the stochastic version provides quite satisfying results with a simpler and faster training. Moreover, the proposed algorithm can be used in combination with mini-batch iterations taking advantages from both approaches, i.e., extracting more information from each training example, and can also be suitable for parallel processing of multiple examples.

Acknowledgment

This research was supported by the Ministry of Education, Science and Technological Development, Republic of Serbia, Grant No. 174013.

The authors would like to thank the anonymous reviewers for their valuable comments for improving the quality of the paper and suggestions for better presentation of results.

References

Bach, F. and Moulines, E. (2011). Non-asymptotic analysis of stochastic approximation algorithms for machine learning, *in* J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira and K.Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems (NIPS)*, Curran Associates, Inc., Red Hook, NY, pp. 451–459.

Balamurugan, P., Shevade, S., Sundararajan, S. and Keerthi, S.S. (2011). A sequential dual method for structural SVMs, *SDM 2011—Proceedings of the 11th SIAM International Conference on Data Mining, Mesa, AZ, USA*.

Bottou, L. (2008). SGD implementation, `http://leon.bottou.org/projects/sgd`.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*, Cambridge University Press, New York, NY.

Collins, M. (2002). Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms, *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing,* Vol. 10, Association for Computational Linguistics, Stroudsburg, PA, pp. 1–8.

Collins, M., Globerson, A., Koo, T., Carreras, X. and Bartlett, P.L. (2008). Exponentiated gradient algorithms for conditional random fields and max-margin Markov networks, *Journal of Machine Learning Research* **9**: 1775–1822.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S. and Singer, Y. (2006). Online passive-aggressive algorithms, *Journal of Machine Learning Research* **7**: 551–585.

Crammer, K., McDonald, R. and Pereira, F. (2005). Scalable large-margin online learning for structured classification,

NIPS Workshop on Learning with Structured Outputs, Vancouver/Whistler, Canada.

Daume, III, H.C. (2006). *Practical Structured Learning Techniques for Natural Language Processing*, Ph.D. thesis, University of Southern California, Los Angeles, CA.

Do, C.B., Le, Q.V., Teo, C.H., Chapelle, O. and Smola, A.J. (2008). Tighter bounds for structured estimation, *in* D. Koller (Ed.), *Advances in Neural Information Processing Systems*, Curran Associates, Inc., Red Hook, NY, pp. 281–288.

Gimpel, K. and Smith, N.A. (2010). Softmax-margin CRFs: Training log-linear models with cost functions, *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Los Angeles, CA, USA*, pp. 733–736.

Jaggi, M., Lacoste-Julien, S., Schmidt, M. and Pletscher, P. (2012). Block-coordinate Frank–Wolfe for structural SVMS, *NIPS Workshop on Optimization for Machine Learning, Lake Tahoe, NV, USA*.

Joachims, T., Finley, T. and Yu, C.-N.J. (2009). Cutting-plane training of structural SVMs, *Machine Learning* **77**(1): 27–59.

Lafferty, J.D., McCallum, A. and Pereira, F.C.N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data, *Proceedings of the 18th International Conference on Machine Learning, ICML'01, San Francisco, CA, USA*, pp. 282–289.

Lee, C., Ryu, P.-M. and Kim, H. (2011). Named entity recognition using a modified Pegasos algorithm, *Proceedings of the 20th ACM International Conference on Information and Knowledge Management, Glasgow, UK*, pp. 2337–2340.

Li, M., Lin, L., Wang, X. and Liu, T. (2007). Protein-protein interaction site prediction based on conditional random fields, *Bioinformatics* **23**(5): 597–604.

Lim, S., Lee, C. and Ra, D. (2013). Dependency-based semantic role labeling using sequence labeling with a structural SVM, *Pattern Recognition Letters* **34**(6): 696–702.

Martins, A.F.T., Smith, N.A., Xing, E.P., Aguiar, P.M.Q. and Figueiredo, M.A.T. (2011). Online learning of structured predictors with multiple kernels, *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA*, Vol. 15, pp. 507–515.

McDonald, R., Crammer, K. and Pereira, F. (2005). Online large-margin training of dependency parsers, *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, ACL'05, Ann Arbor, MI, USA*, pp. 91–98.

Nagata, M. (1994). A stochastic Japanese morphological analyzer using a forward-DP backward-A* N-best search algorithm, *Proceedings of the 15th Conference on Computational Linguistics, COLING '94, Kyoto, Japan*, Vol. 1, pp. 201–207.

Nemirovski, A., Juditsky, A., Lan, G. and Shapiro, A. (2009). Robust stochastic approximation approach to stochastic programming, *SIAM Journal on Optimization* **19**(4): 1574–1609.

Ni, Y., Saunders, C., Szedmak, S. and Niranjan, M. (2010). The application of structured learning in natural language processing, *Machine Translation* **24**(2): 71–85.

Nowozin, S. and Lampert, C.H. (2011). Structured learning and prediction in computer vision, *Foundations and Trends in Computer Graphics and Vision* **6**(3–4): 185–365.

Platt, J.C. (1999). Fast training of support vector machines using sequential minimal optimization, *in* B. Schölkopf, C.J.C. Burges and A.J. Smola (Eds.), *Advances in Kernel Methods*, MIT Press, Cambridge, MA, pp. 185–208.

Rakhlin, A., Shamir, O. and Sridharan, K. (2012). Making gradient descent optimal for strongly convex stochastic optimization, *in* J. Langford and J. Pineau (Eds.), *Proceedings of the 29th International Conference on Machine Learning (ICML-12), Edinburgh, UK*, pp. 449–456.

Ratliff, N.D., Bagnell, J.A. and Zinkevich, M.A. (2006). Subgradient methods for maximum margin structured learning, *ICML Workshop on Learning in Structured Output Spaces, Pittsburgh, PA, USA*.

Sas, J. and Żołnierek, A. (2013). Pipelined language model construction for Polish speech recognition, *International Journal of Applied Mathematics and Computer Science* **23**(3): 649–668, DOI: 10.2478/amcs-2013-0049.

Shalev-Shwartz, S., Singer, Y. and Srebro, N. (2007). Pegasos: Primal estimated sub-gradient solver for SVM, *Proceedings of the 24th International Conference on Machine Learning, ICML '07, Corvalis, OR, USA*, pp. 807–814.

Shalev-Shwartz, S., Singer, Y., Srebro, N. and Cotter, A. (2011). Pegasos: Primal estimated sub-gradient solver for SVM, *Mathematical Programming* **127**(1): 3–30.

Shamir, O. (2012). Open problem: Is averaging needed for strongly convex stochastic gradient descent?, *Journal of Machine Learning Research* **23**: 47–1.

Shamir, O. and Zhang, T. (2012). Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes, *arXiv preprint*, arXiv:1212.1824.

Soong, F.K. and Huang, E.-F. (1991). A tree-trellis based fast search for finding the N-best sentence hypotheses in continuous speech recognition, *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Toronto, Canada*, Vol. 1, pp. 705–708.

Taskar, B., Guestrin, C. and Koller, D. (2004). Max-margin Markov networks, *in* S. Thrun, L. Saul and B. Schölkopf (Eds.), *Advances in Neural Information Processing Systems 16*, MIT Press, Cambridge, MA, pp. 25–32.

Tjong Kim Sang, E.F. and Buchholz, S. (2000). Introduction to the CoNLL-2000 shared task: Chunking, *Proceedings of the 2nd Workshop on Learning Language in Logic/4th Conference on Computational Natural Language Learning, Lisbon, Portugal*, Vol. 7, pp. 127–132.

Tsochantaridis, I., Joachims, T., Hofmann, T. and Altun, Y. (2005). Large margin methods for structured and interdependent output variables, *Journal of Machine Learning Research* **6**: 1453–1484.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, *IEEE Transactions on Information Theory* **13**(2): 260–269.

Weston, J. and Watkins, C. (1998). Multi-class support vector machines, *Technical report*, Department of Computer Science, Royal Holloway, University of London, London.

Xu, W. (2011). Towards optimal one pass large scale learning with averaged stochastic gradient descent, *arXiv preprint*, arXiv:1107.2490.

Dejan Mančev is currently a teaching assistant and a Ph.D. student at the Faculty of Science and Mathematics, University of Niš, Serbia. He received an M.Sc degree in mathematics in 2008 from the same university. His research interests include training methods for structured classifiers with application in natural language processing, and usage of neural networks in stock market forecast.

Branimir Todorović is an associate professor at the Computer Science Department, Faculty of Mathematics and Sciences, University of Niš. He received his D.Sc. degree from the Faculty of Electrical Engineering, University of Belgrade. His research interest include sequential Bayesian training of feed forward and recurrent neural networks, blind source separation and deconvolution, on-line training of structural classifiers, active and semi-supervised learning algorithms.

Received: 5 November 2013
Revised: 14 April 2014
Re-revised: 26 May 2014