Adaptive Stochastic Dual Coordinate Ascent for Conditional Random Fields

Rémi Le Priol
MILA and DIRO
Université de Montréal, Canada

Alexandre Piché
MILA and DIRO
Université de Montréal, Canada

Simon Lacoste-Julien
MILA and DIRO
Université de Montréal, Canada

Abstract

This work investigates the training of conditional random fields (CRFs) via the stochastic dual coordinate ascent (SDCA) algorithm of Shalev-Shwartz and Zhang (2016). SDCA enjoys a linear convergence rate and a strong empirical performance for binary classification problems. However, it has never been used to train CRFs. Yet it benefits from an "exact" line search with a single marginalization oracle call, unlike previous approaches. In this paper, we adapt SDCA to train CRFs, and we enhance it with an adaptive non-uniform sampling strategy based on block duality gaps. We perform experiments on four standard sequence prediction tasks. SDCA demonstrates performances on par with the state of the art, and improves over it on three of the four datasets, which have in common the use of sparse features.

1 INTRODUCTION

The conditional random field (CRF) model (Lafferty et al., 2001) is a common tool in natural language processing and computer vision for structured prediction. The optimization of this model is notoriously challenging. Schmidt et al. (2015) describes a practical implementation of the stochastic average gradient (SAG) algorithm (Roux et al., 2012) for CRFs and proposes a non-uniform sampling scheme that boosts performance. This algorithm (SAG-NUS) is currently the state of the art for CRFs optimization and we refer to Schmidt et al. (2015) for a detailed review of competing methods.

Deterministic (batch) methods such as L-BFGS (Sha and Pereira, 2003; Wallach, 2002) have linear convergence rate but the cost per iteration is large. On the other hand, the online exponentiated gradient method (OEG) (Collins et al., 2008) and SAG are both members of a family of algorithms with cheap stochastic updates and linear convergence rates, and they have both been applied to the training of CRFs. They are called variance reduced algorithms, because their common point is to use memory to reduce the variance of the stochastic update direction as they get closer from the optimum. Johnson and Zhang (2013) coined the name stochastic variance reduced gradient (SVRG) and Defazio et al. (2014) unified the family.

The stochastic dual coordinate ascent (SDCA) algorithm proposed by Shalev-Shwartz and Zhang (2013b, 2016) is a member of this family that has not yet been applied to CRFs. It is closely related to OEG in that it also does block-coordinate ascent on the dual objective. Yet an interesting advantage of SDCA over OEG (and SAG) is that the form of its update makes it possible to perform an "exact" line search with only *one* call to the *marginalization oracle*, i.e. the computation of the marginal probabilities for the CRF. This is in contrast to both SAG and OEG where each step size change requires a new call to the marginalization oracle. We thus propose in this paper to investigate the performance of SDCA for training CRFs.

Contributions. We adapt the multiclass variant of SDCA to the CRF setting by considering the marginal probabilities over the cliques of the graphical model. We provide a novel interpretation of SDCA as a relaxed fixed point update and highlights the block separability of the duality gap. We propose to enhance SDCA with an adaptive non-uniform sampling strategy based on the block gaps, and analyze its theoretical convergence improvement over uniform sampling. We compare the state-of-the-art methods on four prediction tasks with a sequence structure. SDCA with uniform sampling performs comparably with OEG and SAG. When SDCA is enhanced with the adaptive sampling strategy, it outperforms its competitors in terms of number of parameters updates on three of the tasks. These three tasks are all about natural language with handcrafted sparse features. We hypothesize that the efficiency of the dual methods can be related to the sparsity of these features.

Related work. Our proposed gap sampling strategy is similar to the one from Osokin et al. (2016) in the context of SDCA applied to the structured SVM objective, which reduces to the block-coordinate Frank-Wolfe (BCFW) algorithm (Lacoste-Julien et al., 2013). Dünner et al. (2017) recently analyzed a general adaptive sampling scheme for approximate block coordinate ascent that generalizes SDCA. Their proposed sampling scheme (which basically chooses the biggest gap) was motivated in the different context of mixed GPU and CPU computations, which does not apply to our setting. Our proposed practical strategy takes in consideration the staleness of the gaps and is more robust in our experimental setting. Csiba et al. (2015) proposes an adaptive sampling scheme for SDCA for binary classification which unfortunately cannot be generalized to the CRF setting due to an intractable computation. Closely related to our work is Perekrestenko et al. (2017) who analyzed several adaptive sampling strategies for a generalization of the primal-dual SDCA setup, including our proposed gap sampling scheme. However their analysis was focused on the single coordinate descent method (e.g. binary SDCA) and on sublinear convergence results obtained when strong convexity is not assumed. We cover instead the block-coordinate approach relevant to CRFs, and one of our notable results is to show that the linear convergence rate for gap sampling **dominates** the one for uniform sampling, in contrast to what happens in the sublinear regime studied by Perekrestenko et al. (2017).

Outline. We review the optimization problem for CRFs as well as provide novel insights on the primal-dual optimization structure in Section 2. We present SDCA for CRFs in Section 3 and discuss important implementation aspects in Section 4. We present and analyze various adaptive sampling schemes for SDCA in Section 5. We provide experiments in Section 6 and discuss the implications in Section 7.

2 CONDITIONAL RANDOM FIELDS

In this section, we review the CRF model and its associated primal and dual optimization problems. We then derive some interesting properties which motivate several optimization algorithms.

2.1 DEFINITION

A CRF models the conditional probability of a structured output $y \in \mathcal{Y}$ (e.g. a sequence) given an input $x \in \mathcal{X}$ with a Markov random field that uses an exponential family parameterization with sufficient statistics $F(x, y) \in \mathbb{R}^d$ and parameters $\boldsymbol{w} \in \mathbb{R}^d$: $p(y|x; \boldsymbol{w}) \propto \exp(\boldsymbol{w}^\top F(x, y))$. The feature vector F decomposes as a sum over the cliques $C \in \mathcal{C}$ of the graphical model for y: $F(x, y) = \sum_C F_C(x, y_C)$, where y_C denotes the subset of coordinates of y selected by the indices from the set C. See Figure 1 for an illustration.

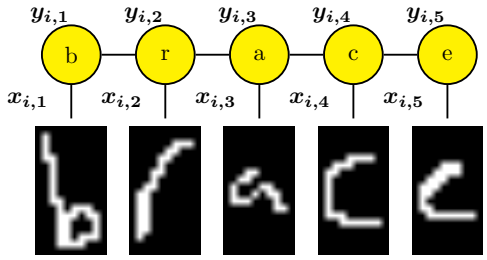


Figure 1: Example of graphical model for the optical character recognition (OCR) task. We want to exploit the structure of the word to predict that $y_{i,5}$ is an "e" and not a "c". This can be done by working on the pairs $y_{i,\{t,t+1\}} = (y_{i,t}, y_{i,t+1})$, the cliques of that model.

2.2 PRIMAL PROBLEM

We have a data set $(x_i, y_i)_{i\in[1,n]}$ of n i.i.d. input and structured output pairs. The parameter is learned by minimizing the ℓ_2-regularized negative log-likelihood:

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} -\log\left(p(y_i|x_i;\boldsymbol{w})\right) . \quad (1)$$

We now rewrite it using the notation for the SDCA setup for multi-class classification from Shalev-Shwartz and Zhang (2016). Denote $M_i = |\mathcal{Y}_i|$ the number of labelings for sequence i. Denote A_i the $d \times M_i$ matrix whose columns are the *corrected features* $\{\psi_i(y) := F(x_i, y_i) - F(x_i, y)\}_{y\in\mathcal{Y}_i}$. Denote also $\phi_i(s) := \log\left(\sum_{y\in\mathcal{Y}_i} \exp(s_y)\right)$ the log-partition function for the scores $s \in \mathbb{R}^{M_i}$. The negative log-likelihood can be written $-\log(p(y_i|x_i;\boldsymbol{w})) = \phi_i(-A_i^\top \boldsymbol{w})$. The primal objective function to minimize over $\boldsymbol{w} \in \mathbb{R}^d$ thus becomes:

$$\mathcal{P}(\boldsymbol{w}) := \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} \phi_i(-A_i^\top \boldsymbol{w}) . \quad (2)$$

2.3 DUAL FORMULATION

The above minimization problem (2) has an equivalent *Fenchel convex dual* problem (Lebanon and Lafferty, 2002). Denote Δ_M the probability simplex over M elements. Denote $\alpha_i \in \Delta_{M_i}$ the set of dual variables for a given x_i. The dual problem handles directly the probability of the labels for the training set. The dual objective to maximize over the choice of $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in$

$\Delta_{|\mathcal{Y}_1|} \times \ldots \times \Delta_{|\mathcal{Y}_n|}$ is:

$$\mathcal{D}(\boldsymbol{\alpha}) := -\frac{\lambda}{2}\|\frac{1}{n\lambda}\sum_i A_i\alpha_i\|^2 + \frac{1}{n}\sum_{i=1}^n H(\alpha_i)\,, \quad (3)$$

where $H(\alpha_i) := -\sum_{y\in\mathcal{Y}_i}\alpha_i(y)\log(\alpha_i(y))$ is the entropy of the probability distribution α_i. The negative entropy appears as the convex conjugate of the softmax: $-H = \phi^*$.

2.4 OPTIMALITY CONDITION

We define the *conjugate weight* function $\hat{w}$ as follows:

$$\hat{w}(\boldsymbol{\alpha}) := \frac{1}{n\lambda}\sum_i A_i\alpha_i = \frac{1}{\lambda n}\sum_{i=1}^n \mathbb{E}_{y\sim\alpha_i}[\psi_i(y)]$$
$$= \frac{1}{\lambda}\left(\frac{1}{n}\sum_{i=1}^n F(x_i,y_i) - \frac{1}{n}\sum_{i=1}^n \mathbb{E}_{y\sim\alpha_i}[F(x_i,y)]\right).$$

It is the difference between the average of the ground truth features, and the average of the expected features for the dual variable, up to a factor $\frac{1}{\lambda}$. We can show that $\hat{w}(\boldsymbol{\alpha}^\star) = \boldsymbol{w}^\star$ where $\boldsymbol{w}^\star$ and $\boldsymbol{\alpha}^\star$ are respectively the optimal primal parameters and the optimal dual parameters.

We can also define the *conjugate* probabilities $\hat{\alpha}_i$ as follows:

$$\forall i, \quad \hat{\alpha}_i(\boldsymbol{w}) := \nabla_s\phi_i(-A_i^\top\boldsymbol{w}) = p(.|x_i;\boldsymbol{w}). \quad (4)$$

We get another optimality condition $\hat{\alpha}(\boldsymbol{w}^\star) = \boldsymbol{\alpha}^\star$. These two optimality conditions can be deduced directly from the structure of the duality gaps.

2.5 DUALITY GAPS

Note that $\mathcal{P}(\boldsymbol{w}) \geq \mathcal{D}(\boldsymbol{\alpha})$ is always true, with equality at the optimum. The *duality gap* is defined by:

$$g(\boldsymbol{w},\boldsymbol{\alpha}) = \mathcal{P}(\boldsymbol{w}) - \mathcal{D}(\boldsymbol{\alpha})\,. \quad (5)$$

Note that we can rewrite the primal gradient as following:

$$\nabla\mathcal{P}(\boldsymbol{w}) = \lambda(\boldsymbol{w} - \hat{w}\circ\hat{\alpha}(\boldsymbol{w}))\,. \quad (6)$$

One can verify that:

$$g(\boldsymbol{w},\hat{\alpha}(\boldsymbol{w})) = \frac{\lambda}{2}\|\boldsymbol{w} - \hat{w}(\hat{\alpha}(\boldsymbol{w}))\|^2 \quad (7)$$
$$= \frac{1}{2\lambda}\|\nabla\mathcal{P}(\boldsymbol{w})\|^2\,. \quad (8)$$

This structure of the gap for the primal weights and its conjugate dual probabilities have an equivalent in the dual. Denote the Fenchel duality gap of ϕ_i for the scores $s_i = -A_i^T\boldsymbol{w}$ and probabilities $\boldsymbol{\alpha}_i$:

$$F_i(s_i,\alpha_i) := \phi_i(s_i) + \phi_i^*(\alpha_i) + s_i^T\alpha_i \geq 0. \quad (9)$$

The positivity comes from the definition of convex conjugates. The gap is zero when s_i and α_i are conjugate variables for ϕ_i, e.g. $\alpha_i = \nabla\phi_i(s_i)$. For any smooth loss ϕ_i, the duality gap between $\hat{w}(\boldsymbol{\alpha})$ and $\boldsymbol{\alpha}$ decomposes as a sum of Fenchel gaps (Shalev-Shwartz and Zhang, 2013a):

$$g(\hat{w}(\boldsymbol{\alpha}),\boldsymbol{\alpha}) = \frac{1}{n}\sum_i F(-A_i^T\hat{w}(\boldsymbol{\alpha}),\alpha_i). \quad (10)$$

The log-sum-exp and the entropy are a special pair of conjugates. Their Fenchel duality gap is also equal to the Bregman divergence generated by $\phi_i^* = -H$, the Kullback-Leibler divergence: $F_i(s_i,\alpha_i) = D_{KL}(\alpha_i||\nabla\phi_i(s_i))$. Writing this for the same pair of conjugate variables yields:

$$g(\hat{w}(\boldsymbol{\alpha}),\boldsymbol{\alpha}) = \frac{1}{n}\sum_i D_{KL}(\alpha_i||\hat{\alpha}_i(\hat{w}(\boldsymbol{\alpha}))). \quad (11)$$

The duality gaps (7) and (11) are typically used to monitor the optimization. In Appendix D, we explain how one can transfer a convergence guarantee on the primal or dual suboptimality to a convergence guarantee on the duality gap.[1] Moreover, the block-separability of gaps from (11) can motivate an adaptive sampling scheme, as we describe in Section 5.

2.6 INTERPRETATION

The primal formulation chooses a $\boldsymbol{w}$ of small norm so as to maximise the conditional probability of observing the labels. Conversely, the dual formulation chooses conditional probabilities of the labels so as to minimize the ℓ_2 distance between the expected features and empirical expectation of the ground truth features. The optimal distribution would be the empirical distribution, if not for the entropic regularization that favors more uniform probabilities. This is the regularized version of the classical duality between maximum-likelihood and maximum-entropy for exponential families.

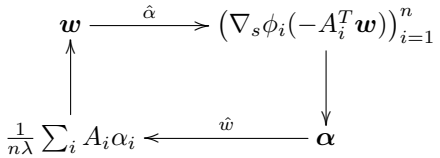
The optimality conditions show that the solution of the primal Problem (2) is also a *fixed point* for the function $\hat{w}\circ\hat{\alpha}$. Because of the gradient form (6), the gradient descent update can also be written as a *relaxed* fixed point update:

$$\boldsymbol{w}^+ = \boldsymbol{w} - \gamma\nabla\mathcal{P}(\boldsymbol{w}) \quad (12)$$
$$= (1-\gamma\lambda)\boldsymbol{w} + \gamma\lambda\ \hat{w}\circ\hat{\alpha}(\boldsymbol{w})\,. \quad (13)$$

The algorithm SDCA described in the next section also admits a relaxed fixed point update on the block α_i

[1] This implies that convergence results on the dual problem directly translates to convergence results on the primal and vice-versa; a fact apparently missed in the linear rate comparison of Schmidt et al. (2015).

(see (14)). More generally, optimization algorithms for Problem (2) can often be interpreted as a back and forth between the conjugate variables w and $\hat{w}(\hat{\alpha}(\boldsymbol{w}))$ (primal methods) or α and $\hat{\alpha}(\hat{w}(\boldsymbol{\alpha}))$ (dual methods). For instance, one could interpret OEG as a relaxed fixed point iteration over the score variables $s_i = -A_i^T \boldsymbol{w}$.



Most of the results presented in this section and in Section 5 can be transposed to other kinds of loss and regularization, under some regularity assumptions. Our focus in this paper is the application of SDCA to CRF models and thus we focused the discussion on the log-likelihood setting and the ℓ_2 norm, which are widely used.

3 PROXIMAL STOCHASTIC DUAL COORDINATE ASCENT

We first describe the SDCA in its general setting, and then describe the necessary modifications for training a CRF.

3.1 GENERAL SETTING

The stochastic dual coordinate ascent algorithm (SDCA) updates one dual coordinate at a time so as to maximize the dual objective. SDCA was originally proposed for binary classification (Shalev-Shwartz and Zhang, 2013b) where each dual variable α_i lives in $\Delta_2 = [0, 1]$. In this case, it is possible to do exact coordinate maximization of the dual objective over a single α_i with standard one dimensional optimization.

In the multi-class setting however, there is no simple way to maximize the dual objective over the block $\alpha_i \in \Delta_K$. The algorithm with the surprising name of Proximal-SDCA[2], option II (Shalev-Shwartz and Zhang, 2016) proposes a solution to this problem. It updates α_i in a clever direction derived from the primal-dual relationship, which amounts to a relaxed fixed point update. See Algorithm 1.

We now describe the idea. At all time, we maintain the pair of dual and primal variables $(\boldsymbol{\alpha}, \boldsymbol{w} = \hat{w}(\boldsymbol{\alpha}))$. At each step, we sample a training point i. We compute $\beta_i = \nabla_s \phi_i(-A_i^T \boldsymbol{w}) = \hat{\alpha}_i \circ \hat{w}(\boldsymbol{\alpha})$, the next fixed point iterate. We then define the dual ascent direction by $\delta_i := \beta_i - \alpha_i$. Finally we update the block α_i with the right step size so as to increase the dual objective $\mathcal{D}(\boldsymbol{\alpha})$ using a relaxed fixed point update:

$$\alpha_i^+ \leftarrow \alpha_i + \gamma \delta_i = (1 - \gamma)\alpha_i + \gamma \hat{\alpha}_i \circ \hat{w}(\boldsymbol{\alpha}) \,. \quad (14)$$

[2]We simply call it SDCA in the rest of this paper

Algorithm 1 Prox-SDCA (option II) called SDCA here

Initialize $\alpha_i^{(0)} \in \Delta_{M_i}, \forall i$
Let $\boldsymbol{w}^{(0)} = \hat{w}(\boldsymbol{\alpha}^{(0)}) = \frac{1}{\lambda n} \sum_i A_i \alpha_i$
for $t = 0, 1 \ldots$ **do**
  Sample i uniformly at random in $\{1, \ldots, n\}$
  Let $\beta_i := \hat{\alpha}_i(\boldsymbol{w}) = \nabla_s \phi(-A_i^T \boldsymbol{w})$
  Let $\delta_i = \beta_i - \alpha_i^{(t)}$ {dual ascent direction}
  Let $\boldsymbol{v}_i = \frac{1}{\lambda n} A_i \delta_i$ {primal direction}
  Solve Equation (15) to get γ^* {Line Search}
  Update $\alpha_i^{(t+1)} := \alpha_i^{(t)} + \gamma^* \delta_i$
  Update $\boldsymbol{w}^{(t+1)} := \hat{w}(\boldsymbol{\alpha}^{(t+1)}) = \boldsymbol{w}^{(t)} + \gamma^* \boldsymbol{v}_i$

The dual ascent direction is guaranteed to increase $\mathcal{D}(\boldsymbol{\alpha})$, unless $\delta_i = 0$ (this actually means that the block is already optimal, see (11)). The primal weights $\boldsymbol{w} = \hat{w}(\boldsymbol{\alpha})$ are related to $\boldsymbol{\alpha}$ by a linear transformation. Define the primal direction $\boldsymbol{v}_i = \frac{1}{\lambda n} A_i \delta_i \in \mathbb{R}^d$. One can update the weights directly: $\boldsymbol{w}^+ \leftarrow \boldsymbol{w} + \gamma \boldsymbol{v}_i$.

The step size $\gamma \in [0, 1]$ is either fixed, or found via line search. In practice the fixed step size for which convergence is guaranteed is really small. The line search is relatively cheap as we are looking at only one block:

$$\gamma^* := \arg\max_{\gamma \in [0,1]} -\phi_i^*(\alpha_i + \gamma \delta_i) - \frac{\lambda n}{2} \|\boldsymbol{w} + \gamma \boldsymbol{v}_i\|^2. \quad (15)$$

Note that one can decompose the quadratic term and precompute $\langle \boldsymbol{w}, \boldsymbol{v}_i \rangle$ and $\|\boldsymbol{v}_i\|^2$ to accelerate the optimisation. The bottleneck remains the computation of ϕ_i^* (and its derivatives).

3.2 ADAPTATION TO CRF

In the CRF setting, the dual variable α_i is exponentially large in the input size x_i. For a sequence x_i of length T where each node can take up to K values, the number of possible labels is $|\mathcal{Y}_i| = K^T$. It might not even fit in memory. Instead, the standard approach used in OEG and SAG is to consider the marginal probabilities $(\mu_C)_{C \in \mathcal{C}}$ on the cliques of the graphical model. Similarly, we replace $\boldsymbol{\alpha}$ by $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_n)$, where $\mu_i \in \prod_C \Delta_C$ is the concatenation of all the clique marginal vectors for the sample i. For the same sequence x_i, this reduces the memory cost to $K^2(T-1)$ for the pair marginals. We denote $m_i = \sum_C |\mathcal{Y}_{i,C}|$ this new memory fingerprint. For a sequence long enough, we have $m_i \ll M_i$. The associated weight vector can still be expressed as function of $\boldsymbol{\mu}$ thanks to the separability of the features:

$$\hat{w}(\boldsymbol{\mu}) = \frac{1}{\lambda n} \sum_i \sum_C \mathbb{E}_{\mu_{i,C}}[\psi_{i,C}] = \frac{1}{\lambda n} \sum_i B_i \mu_i, \quad (16)$$

where $B_i = (\psi_{i,C}(y_C))_{C, y_C} \in \mathbb{R}^{d \times m_i}$ is the horizontal concatenation of the cliques feature vectors.

Algorithm 2 SDCA for CRF

Initialize $\mu_i^{(0)} \in \prod_C \Delta_C$ consistently $\forall i$ {use (21)}
Set $\boldsymbol{w}^{(0)} := \hat{w}(\boldsymbol{\mu}^{(0)}) = \frac{1}{\lambda n} \sum_i B_i \mu_i^{(0)}$ {See (16)}
(Optional) Let $g_i = 100, \forall i$
for $t = 0, 1 \ldots$ **do**
  Sample i uniformly at random in $\{1, \ldots, n\}$
  (Alternatively) Sample i proportionally to g_i
  Let $\nu_{i,C}(y_C) := p(y_C | x_i; \boldsymbol{w}^{(t)}), \forall C \in \mathcal{C}$ {**oracle**}
  (Optional) Let $g_i = \tilde{D}(\mu_i || \nu_i)$ {duality gap (19)}
  Let $\delta_i = \nu_i - \mu_i^{(t)}$ {ascent direction}
  Let $\boldsymbol{v}_i = \frac{1}{\lambda n} \hat{w}(\delta_i)$ {primal direction}
  Solve Equation (20) to get γ^* {Line Search}
  Update $\mu_i^{(t+1)} := \mu_i^{(t)} + \gamma^* \delta_i$
  Update $\boldsymbol{w}^{(t+1)} := \hat{w}(\boldsymbol{\mu}^{(t+1)}) = \boldsymbol{w}^{(t)} + \gamma^* \boldsymbol{v}_i$

Now, assume that the graph has a *junction tree* structure $T = (\mathcal{C}, \mathcal{S})$ (Koller and Friedman, 2009, Def. 10.3), where $\mathcal{C}$ is the set of maximal cliques and $\mathcal{S}$ the set of separators. We can then run message passing on the junction tree to infer the new marginals given weights $\boldsymbol{w}$: $\hat{\mu}_i(\boldsymbol{w}) = p(y_C = . | x_i; \boldsymbol{w})$. We can also now recover the joint probability $\alpha_i(y)$ as a function of its marginals $\mu_{i,C}$ (Koller and Friedman, 2009, Def. 10.6):

$$\alpha_i(y) = \frac{\prod_{C \in \mathcal{C}} \mu_{i,C}(y_C)}{\prod_{S \in \mathcal{S}} \mu_{i,S}(y_S)}. \tag{17}$$

Equation (17) in turn allows us to compute the entropy and the divergences of the joints, using only the marginals. Let μ_i and ν_i be the marginals of respectively α_i and β_i, then the entropy and the Kullback-Leibler divergence are given by:

$$\tilde{H}(\mu_i) := H(\alpha_i) = \sum_C H(\mu_{i,C}) - \sum_S H(\mu_{i,S}) \tag{18}$$

and

$$\begin{aligned} &\tilde{D}(\mu_i || \nu_i) := D_{KL}(\alpha_i || \beta_i) \\ &= \sum_C D_{KL}(\mu_{i,C} || \nu_{i,C}) - \sum_S D_{KL}(\mu_{i,S} || \nu_{i,S}). \end{aligned} \tag{19}$$

With this expression of the entropy (18), we can compute the dual objective, and thus perform the line search:

$$\gamma^* = \underset{\gamma \in [0,1]}{\arg\max}\, \tilde{H}(\mu_i^{(t)} + \gamma \delta_i) - \frac{\lambda n}{2} \|\boldsymbol{w}^{(t)} + \gamma \boldsymbol{v}_i\|^2. \tag{20}$$

With the Kullback-Leibler divergence (19), we can compute efficiently the individual duality gaps from (11). Algorithm 2 describes this variation of SDCA, with as an option a non-uniform sampling strategy defined in Section 5.3.

4 IMPLEMENTATION

We provide in Appendix A a discussion of various important implementation aspects summarized here.

1. The initialization of dual methods for CRFs can significantly influence their performance. As explained in Appendix A, we use:
$$\boldsymbol{\alpha}^{(0)} := \varepsilon \boldsymbol{u} + (1 - \varepsilon) \boldsymbol{\delta}\,, \tag{21}$$
where $\boldsymbol{u}$ is the uniform distribution on each block, $\boldsymbol{\delta}$ is a unit mass on each ground truth label and ε is a small number.
2. Storing the dual variable may be expensive and one should allocate a decent amount of memory.
3. The line search requires computing the entropy of the marginals. This is costly and we used Newton-Raphson algorithm to minimize the number of iterations. This in turn requires storing the logarithm of the dual variable.

5 ADAPTIVE SAMPLING FOR SDCA

Recently, there has been a lot of attention on non-uniform sampling for stochastic methods. The general goal is to sample more often points which are harder to classify and can bring more progress on the objective. These methods are said to be *adaptive* when the sampling probability changes during the optimization. SDCA itself has had several adaptive schemes proposed. In the following, we attempt to explain and relate these methods, and suggest new schemes that work well on our problem.

5.1 ASCENT LEMMA

We start by restating the ascent lemma from Equation (25) in Shalev-Shwartz and Zhang (2013a). This lemma inspires and supports all the strategies.

Ascent after sampling i: At iteration t, if we sample i and take a step of size $\gamma_i \in [0, 1]$, we can lower bound the resulting dual improvement:

$$\begin{aligned} &n(\mathcal{D}(\boldsymbol{\alpha}^+) - \mathcal{D}(\boldsymbol{\alpha})) \\ &\geq \gamma_i \underbrace{\left[\phi(-A_i^T \boldsymbol{w}) + \phi^*(\alpha_i) + \boldsymbol{w}^T A_i \alpha_i\right]}_{\text{Fenchel gap} =: g_i} \\ &\quad + \gamma_i \left(\frac{(1 - \gamma_i)}{2} - \frac{\gamma_i R_i}{2 \lambda n} \right) \|\beta_i - \alpha_i\|_1^2 \end{aligned} \tag{22}$$

where $R_i := \|A_i\|_{1 \to 2}^2 = \max_{y \in \mathcal{Y}_i} \|\psi_i(y)\|_2^2$ is the squared radius of the corrected features for sample i.

Note that compared to the original text, we used the fact that the regularizer is the ℓ_2 norm and the loss is 1-smooth

with respect to the ℓ_∞ norm. We define $R := \max_i R_i$, $\bar{R} := \frac{1}{n}\sum_i R_i$ and $\bar{g} := \frac{1}{n}\sum_i g_i$ the true duality gap (see (9)-(10)). We also introduce $L_i := \lambda + \frac{R_i}{n}$ an upper bound on the smoothness of loss i plus regularizer for the ℓ_2 norm. We recall from Section 2.5 that $g_i = D_{KL}(\alpha_i||\beta_i)$ (11). We give the name *residual* to $d_i := \|\beta_i - \alpha_i\|_1^2$.

This lemma is derived with standard assumptions and inequalities on the smoothness of the loss and the strong convexity of the regularizer. The first term of the lower bound is the ascent guarantee while the other term gives condition on the step-size to ensure progress. We refer the reader to the original paper for more details.

To get the expected progress (conditioned on the past) after sampling with probability $\boldsymbol{p}$, we simply need to take the sum of the inequality above after multiplying both sides by p_i. Our goal is to maximize this lower bound by choosing the right probability $\boldsymbol{p}$ and step sizes $\boldsymbol{\gamma}$. To be able to conclude the proof with the original method, we also want some constants time the duality gap $\bar{g}$ to appear in the lower bound – the gap is lower bounded by the dual suboptimality and thus this constant will give the linear rate of convergence. The lemma can then transpose this result from the dual sub-optimality to the duality gap as described in Appendix D. From there on there are two general approaches: importance sampling and duality gap sampling.

5.2 IMPORTANCE AND RESIDUAL SAMPLING

With the importance sampling approach, the goal is to set the step-size and the probability so that they cancel each other out: $\gamma_i = \frac{\gamma}{p_i}$. One then get an unbiased estimate of the true duality gap from (11) as the first term of the upper bound. What is left is maximizing the second term with respect to $\boldsymbol{p}$. This is the approach proposed by Zhao and Zhang (2015) (Importance Sampling, left term below) and generalized by Csiba et al. (2015) (Residual sampling, a.k.a. AdaSDCA for binary classification, right term):

$$p_i \propto L_i \quad \text{or} \quad p_i \propto d_i\sqrt{L_i}. \tag{23}$$

These sampling schemes somehow allow to maximize the second term of (22). Intuitively, they replace a dependency on R in the convergence rate by a dependency on $\bar{R}$. They can give good results on binary and multi-class logistic regression. There are a few issues though.

- One needs an accurate estimate of the L_i.
- Importance sampling is not adaptive.
- In the CRF setting, the residual is $d_i = \|\beta_i - \alpha_i\|_1^2$. It is the squared ℓ^1 norm of a vector of exponential size. We are not aware of any trick to compute it efficiently.

5.3 GAP SAMPLING

To make sure that the second term is positive, the original proof of uniform SDCA sets $\gamma_i = \gamma = (1 + \frac{R}{\lambda n})^{-1}$ to obtain:

$$n\mathbb{E}_p[D(\boldsymbol{\alpha}^+) - D(\boldsymbol{\alpha})] \geq \gamma \sum_i p_i g_i. \tag{24}$$

Assuming a full knowledge of the duality gaps g_i, the optimal decision is to sample the point with maximum duality gap. This was done by Dünner et al. (2017) in the context of multi-class classification on a pair CPU-GPU. While the GPU computes the update, the CPU updates as many duality gaps as possible. This lead to impressive acceleration over massive datasets.

However, this is not our current setting. We know and update only one gap at a time (for efficiency). Because of staleness of the gaps, our experiments with this method did not even converge for the most part (see Section 6.3). We need a more robust method.

We take inspiration from what was done by Osokin et al. (2016) to improve the Block-Coordinate Frank-Wolfe (BCFW) algorithm (Lacoste-Julien et al., 2013). We propose to bias sampling towards examples whose duality gaps are large: $p_i \propto g_i$. If we know all the duality gaps, the expected improvement reads:

$$n\mathbb{E}_p[D(\boldsymbol{\alpha}^+) - D(\boldsymbol{\alpha})] \geq \chi(\boldsymbol{g})^2\,\gamma\,\bar{g}, \tag{25}$$

where $\chi(\boldsymbol{g}) = \sqrt{\frac{\frac{1}{n}\sum_i g_i^2}{\bar{g}^2}} \in [1, \sqrt{n}]$ is the non-uniformity of the duality gaps, as defined in Osokin et al. (2016, Section 3.1). The value $\chi(\boldsymbol{g})^2\gamma$ is the value that will appear in the linear convergence rate of this method. It means that the convergence rate for gap sampling **dominates** the one for uniform sampling. This is different from what was observed for BCFW where they could not prove dominance in general.

In practice we use stale estimates of the gaps and there are no convergence guarantees. We discuss more this issue in section 6.3.

We also explored a combination of gap sampling and importance sampling. We could get similar convergence rate where a trade-off appeared between the mean smoothness and the non-uniformity. We detail these considerations as a technical report in Appendix F for the interested reader.

6 EXPERIMENTS

We conducted these experiments to answer three questions: (1) How does the line search influence SDCA? (2) How do the non-uniform sampling schemes compare with each other? and (3) How does SDCA compare with SAG and OEG on sequence prediction?

Table 1: Dataset summary. d is the dimension of $\boldsymbol{w}$. n is the number of data points (sequences). N is the number of nodes (e.g. sum of sequences length). K is the number of possible labels for each node. A is the number of attributes (see Appendix B). a is the maximum number of attributes extracted from one node. Mem. is the memory required by the pairwise marginals stored as float 64. The pairwise marginals dominate the memory cost.

Dataset	OCR	CONLL	NER	POS
d	4,082	1.6×10^6	2.8×10^6	8.6×10^6
n	6,202	8,936	15,806	38,219
N	52,827	2.1×10^5	2×10^5	9.1×10^5
K	26	22	9	45
A	128	74,658	3.1×10^5	1.9×10^5
a	128	19	20	13
Mem.(GiB)	0.2	0.7	0.1	13

6.1 EXPERIMENTAL SETTING

We applied the experimental setup outlined by Schmidt et al. (2015). We implemented SDCA to train a classifier on four CRF training tasks: (1) the optical character recognition (OCR) dataset (Taskar et al., 2004), (2) the CoNLL-2000 shallow parse chunking dataset (CONLL), (3) the CoNLL-2002 Dutch named-entity recognition dataset (NER), and (4) a part-of-speech (POS) tagging task using the Penn Treebank Wall Street Journal data. Additional details regarding these datasets are provided in Table 1. Note that the tasks (2), (3), (4) are about language understanding. They use sparse features (the ratio a/A from the table is small). The sparsest data set is NER. Note that POS is considerably larger than other datasets. All experiments are performed with a regularization factor $\lambda = 1/n$. We used our own implementation[3] of SDCA coded in plain Python and Numpy (Walt et al., 2011). In most plots we report the logarithm base 10 of the primal sub-optimality. We got the optimum by running L-BFGS a large number of iterations.

6.2 EFFECT OF THE LINE SEARCH

We implemented the safe bounded Newton-Raphson method from Press et al. (1992, Section 9.4) on the derivative of the line search function. A natural question to ask is : how precise should the line search be? The stopping criterion for this algorithm is the size of the last step taken so there is no proper precision parameter. We refer to this stopping criterion for the line search as the sub-precision of SDCA.

[3]The code to reproduce our experiments is available at: https://remilepriol.github.io/research/sdca4crf.html.

We discovered experimentally that the convergence of SDCA is mostly independent of the sub-precision. On all datasets, if we ask 0.01 sub-precision or less, SDCA converges with the same rate. An explanation is that the accuracy of the optimization arises from iterates $\boldsymbol{\alpha}$ and $\hat{\alpha}(\hat{w}(\boldsymbol{\alpha}))$ getting closer to each other in the simplex with each iteration.

Reaching 0.01 or 0.001 takes on average 2 iterations. Each iteration of Newton's method require the computation of the first and second derivative of the line search objective (20). In the following we report results with sub-precision 0.001 to be on the safe side. These 2 iterations were taking about 30% of the algorithms running time for each dataset.[4]

We also performed experiments with only one step of the Newton update. The convergence was not affected on OCR, CONLL and POS, but convergence failed on NER (see Figure 8 of Appendix E). This phenomenon could be related to sparsity.

6.3 COMPARISON OF SAMPLING SCHEMES

We compare the performance of four sampling strategies with 20% of uniform sampling against the full Uniform approach, on the OCR dataset (see results in Figure 2):

- *Importance:* sample proportionally to the smoothness constants $L_i = \lambda + \frac{R_i}{n}$. We report how we evaluated the radii R_i in Appendix C.
- *Gap:* sample proportionally to our current estimate of the duality gaps.[5]
- *Gap* $\times$ *importance:* sample proportionally to the product of the gap and smoothness constants.
- *Max:* sample deterministically the variable with the largest recorded gap (Dünner et al., 2017).

As discussed in Section 5.3, Max sampling is not robust enough to the staleness of the gap estimates and fails to converge here. We also observe that Importance performs worse than Uniform, and that Gap $\times$ Importance performs worse than Gap. This indicates that the smoothness upper bounds we estimated are not informative of the difficulty of optimizing a point for SDCA. Overall, Gap sampling gives the best performance and this is what we use in the following experiments.

The ratio of uniform sampling is here to mitigate the fact that we sample proportionally to stale gaps. This is

[4] We also tried initializing the line search with 0.5 or with the previous step size. There was no significant difference.

[5] For the gap approaches, we initialize the gap estimates with large values (100) so as to perform a pass over the whole dataset before starting to sample proportionally to the stale estimates.

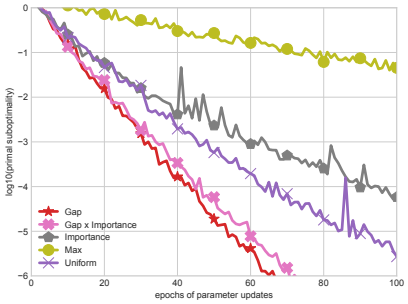


Figure 2: Performance of competing sampling schemes on the OCR dataset with 80% of non-uniformity. Sampling proportionally to the gap gives the best performance.

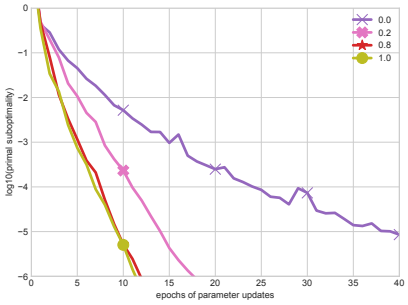


Figure 3: SDCA with Gap sampling applied on NER with various fractions of non-uniform sampling, as indicated by the number in the legend. Increasing the fraction only improves the performance, up to a certain point.

the strategy adopted by SAG-NUS (Schmidt et al., 2015) which samples uniformly half of the time. Another strategy used by Osokin et al. (2016) is to update all the duality gaps at once every 10 epochs or so. Our experiments indicate that these strategies are not needed for SDCA-GAP. Increasing the ratio of non-uniformity up to 1 only improves the performance on all datasets, though after 0.8 the improvements are marginal, as illustrated by Figure 3 for the NER dataset.

In fact, the estimate of the total gap maintained by SDCA is somewhat accurate, as illustrated for different datasets in Figure 9 of Appendix E. Empirically, it always remains within a factor 2 of the true duality gap. This accuracy is a good news because one can use this estimate of the duality gap as a stopping criterion for the whole algorithm. Once it reaches a certain precision threshold, one just has to perform one last batch update to check the real value. This is similar in spirit to SAG, which uses the norm of its estimate of the true gradient as a stopping criterion. Both are duality gaps estimators (see Equation (7)).

6.4 COMPARISON AGAINST SAG AND OEG

We downloaded the code for OEG and SAG-NUS as implemented by Schmidt et al. (2015) from the SAG4CRF project page.[6] We used our own implementation of SDCA with a line search sub-precision of 0.001. We provide the comparison in Figure 4 according to two different measures of complexity which are implementation independent.

Oracle calls. Schmidt et al. (2015) compared the algorithms on the basis of the number of oracle calls. We report these on OCR and NER in Figures 4a and 4d. Results on the other datasets are in Figure 6 in Appendix E. This metric was suitable for the methods they compared. Both OEG and SAG-NUS use a line search where they call an oracle on each step. SDCA does not need the oracle to perform its line search. However the oracle is message passing on a junction tree. It has a cost proportional to the size of the marginals. Each iteration of the line search require computing the entropy of these marginals, or their derivatives. These costs are roughly the same. Comparing the number of oracle calls for each method is thus unfairly advantaging SDCA by hiding the cost of its line search. It becomes a relevant comparison when a marginalization oracle becomes much more expensive than approximating the entropy (see the discussion in Section 7). When this cost is hidden, SDCA-GAP is on par with SAG-NUS* on OCR and it is much faster on the sparse datasets.

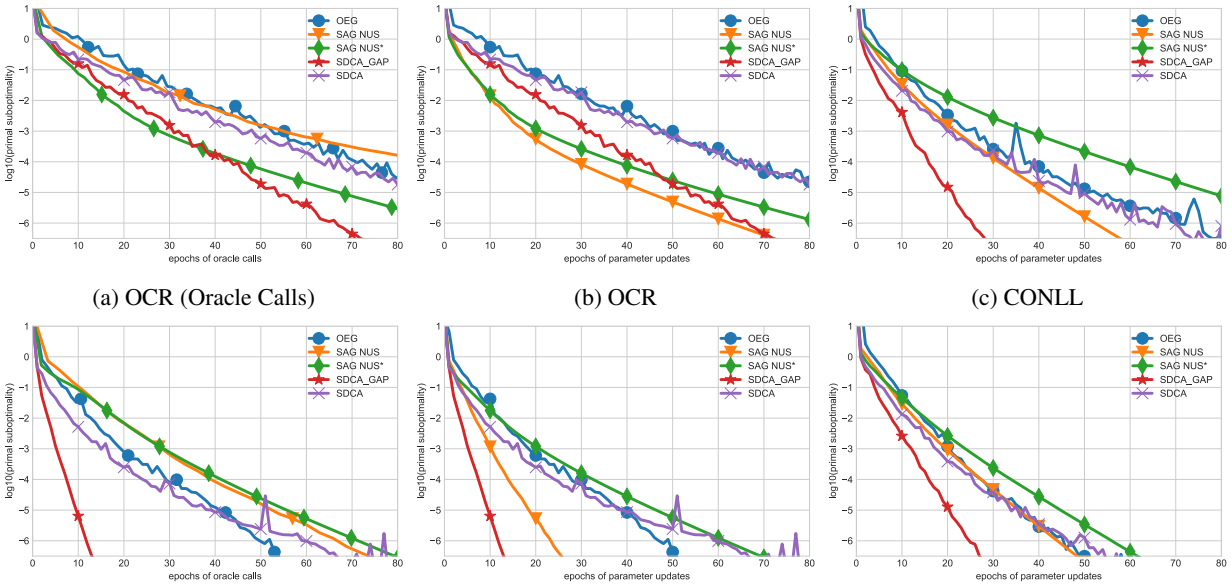
Parameter updates. To give a different perspective, we report the log of the sub-optimality against the number of parameter updates in Figures 4b, 4c, 4e and 4f. This removes the additional cost of the line search for all methods.[7]

We observe that uniform SDCA and OEG need roughly the same number of parameters update on all four datasets. When we add the adaptive gap sampling, SDCA outperforms OEG by a margin. On OCR, SDCA and SDCA-GAP do not perform as well as SAG-NUS. On the three other datasets, SDCA-GAP needs less iterations. In fact, the more sparse the dataset, the less iterations are needed.

This is likely explained by SDCA's ability to almost perfectly optimize each block separately due to its line search method. More specifically, as the datasets become sparser, the prediction between data points becomes less and less correlated (i.e. the label distribution for two points that share no attributes will not influence each other directly through their primal weights). In settings where no points

[6] https://www.cs.ubc.ca/~schmidtm/Software/SAG4CRF.html

[7] This is a penalty for SAG-NUS* which enforces a line-search skipping strategy.



(a) OCR (Oracle Calls) (b) OCR (c) CONLL

(d) NER (Oracle calls) (e) NER (f) POS

Figure 4: Primal sub-optimality as a function of the number of oracle calls (left) or parameters updates (center and right). SDCA refers to uniform sampling. SDCA-GAP refers to sampling Gap sampling 80% of the time. SAG-NUS performs a line search at every iteration. SAG-NUS* implements a line-search skipping strategy. It appears worse than SAG-NUS when we look at the number of updates, which hides the cost of the line search.

share any attributes (completely sparse), all methods optimize each point independently. SDCA may perform very well thanks to its precise line search.

In terms of test error, SDCA is on par with SAG, and a bit better than OEG. All methods reach maximum accuracy after a few epochs. We report the evolution of the test error in Figure 7 of Appendix E.

Comparing the number of parameters updates also has a disadvantage. It penalizes methods with line search skipping strategies likes OEG and SAG. The running time is highly implementation dependent and providing a fair comparison is non-trivial. We focused on implementation independent comparisons. SCDA, SAG and OEG have many common operations: the oracle, the computation of the scores and the primal direction. The fact that the line search took only 30% of SDCA's runtime indicates that the conclusion drawn from the number of updates may hold for other metrics.

7 DISCUSSION

In this work, we investigated using SDCA for training CRFs for the first time. The observed empirical convergence per parameter update was similar for standard SDCA and OEG. However, SDCA can be enhanced with an adaptive sampling scheme, consistently accelerating its convergence and also yielding faster convergence than SAG with non-uniform sampling on datasets with sparse features. It would be natural to also implement a gap sampling scheme for OEG, though several quantities needed for the computation are not readily available in standard OEG and would yield higher overhead in actual implementation. We leave finding a more efficient implementation of a gap sampling scheme for OEG as an interesting research direction.

A key feature of SDCA is to only require one marginalization oracle per line-search. This could become advantageous over SAG or OEG when the marginalization oracle becomes much more expensive than evaluating the entropy function from the marginals. Examples for this scenario include: when a parallel implementation is used for the entropy computation; or when the marginalization oracle uses an iterative approximate inference algorithms such as TRW BP whereas an approximation of the entropy is direct from the marginals (Krishnan et al., 2015). Investigating these scenarios with full timing comparison (which is implementation dependent) is a further interesting direction of future work.

We also note that acceleration schemes have been proposed for both SAG and SDCA (Lin et al., 2015; Shalev-Shwartz and Zhang, 2016), though they have not been tested yet for training CRFs.

Acknowledgments

We are thankful to Thomas Schweizer for his numerous software engineering advices. We thank Gauthier Gidel and Akram Erraqabi who started this project. We are indebted to Ahmed Touati for his involvement during the first phase of the project. Alexandre Piché was supported by the Open Philantropy Project. This work was partially supported by the NSERC Discovery Grant RGPIN-2017-06936.

References

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

D. Csiba, Z. Qu, and P. Richtárik. Stochastic dual coordinate ascent with adaptive probabilities. In *ICML*, 2015.

A. Defazio, F. Bach, and S. Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *NIPS*, 2014.

C. Dünner, T. Parnell, and M. Jaggi. Efficient use of limited-memory accelerators for linear learning on heterogeneous systems. In *NIPS*, 2017.

R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS*, 2013.

D. Koller and N. Friedman. *Probabilistic Graphical Models*. The MIT Press, 2009.

R. G. Krishnan, S. Lacoste-Julien, and D. Sontag. Barrier Frank-Wolfe for marginal inference. In *NIPS*, 2015.

S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher. Block-coordinate Frank-Wolfe optimization for structural SVMs. In *ICML*, 2013.

J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

G. Lebanon and J. D. Lafferty. Boosting and maximum likelihood for exponential models. In *NIPS*, 2002.

H. Lin, J. Mairal, and Z. Harchaoui. A universal catalyst for first-order optimization. In *NIPS*, 2015.

Y. Nesterov. *Introductory Lectures on Convex Optimization*. Applied Optimization. Springer US, 2004.

A. Osokin, J.-B. Alayrac, I. Lukasewitz, P. Dokania, and S. Lacoste-Julien. Minding the gaps for block Frank-Wolfe optimization of structured SVMs. In *ICML*, 2016.

D. Perekrestenko, V. Cevher, and M. Jaggi. Faster coordinate descent via adaptive importance sampling. In *AISTATS*, 2017.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C: the art of scientific computing*. Cambridge University Press, Cambridge, 2nd edition, 1992.

N. L. Roux, M. Schmidt, and F. R. Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *NIPS*, 2012.

M. Schmidt, R. Babanezhad, M. Ahmed, A. Defazio, A. Clifton, and A. Sarkar. Non-uniform stochastic average gradient method for training conditional random fields. In *AISTATS*, 2015.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *NAACL*, 2003.

S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *arXiv:1309.2375*, 2013a.

S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14, 2013b.

S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, 2016.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *NIPS*, 2004.

H. Wallach. Efficient training of conditional random fields. Master's thesis, University of Edinburgh, 2002.

S. v. d. Walt, S. C. Colbert, and G. Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 2011.

P. Zhao and T. Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *ICML*, 2015.