Block-Coordinate Frank-Wolfe Optimization for Structural SVMs

Simon Lacoste-Julien* INRIA - SIERRA project-team, École Normale Supérieure, Paris, France
Martin Jaggi* CMAP, École Polytechnique, Palaiseau, France
Mark Schmidt INRIA - SIERRA project-team, École Normale Supérieure, Paris, France
Patrick Pletscher Machine Learning Laboratory, ETH Zurich, Switzerland

* Both authors contributed equally.

Abstract

We propose a randomized block-coordinate variant of the classic Frank-Wolfe algorithm for convex optimization with block-separable constraints. Despite its lower iteration cost, we show that it achieves a similar convergence rate in duality gap as the full Frank-Wolfe algorithm. We also show that, when applied to the dual structural support vector machine (SVM) objective, this yields an online algorithm that has the same low iteration complexity as primal stochastic subgradient methods. However, unlike stochastic subgradient methods, the block-coordinate Frank-Wolfe algorithm allows us to compute the *optimal* step-size and yields a computable duality gap guarantee. Our experiments indicate that this simple algorithm outperforms competing structural SVM solvers.

1. Introduction

Binary SVMs are amongst the most popular classification methods, and this has motivated substantial interest in optimization solvers that are tailored to their specific problem structure. However, despite their wider applicability, there has been much less work on solving the optimization problem associated with *structural* SVMs, which are the generalization of SVMs to structured outputs like graphs and other combinatorial objects (Taskar et al., 2003; Tsochantaridis et al., 2005). This seems to be due to the difficulty of dealing with the exponential number of constraints in the primal problem, or the exponential number of variables in the dual problem. Indeed, because they achieve an $\tilde{O}(1/\varepsilon)$ convergence rate while only requiring a single call to the so-called *maximization oracle* on each iteration, basic stochastic subgradient methods are still widely used for training structural SVMs (Ratliff et al., 2007; Shalev-Shwartz et al., 2010a). However, these methods are often frustrating to use for practitioners, because their performance is very sensitive to the sequence of step sizes, and because it is difficult to decide when to terminate the iterations.

To solve the dual structural SVM problem, in this paper we consider the Frank-Wolfe (1956) algorithm, which has seen a recent surge of interest in machine learning and signal processing (Mangasarian, 1995; Clarkson, 2010; Jaggi, 2011; 2013; Bach et al., 2012), including in the context of binary SVMs (Gärtner & Jaggi, 2009; Ouyang & Gray, 2010). A key advantage of this algorithm is that the iterates are *sparse*, and we show that this allows us to efficiently apply it to the dual structural SVM objective even though there are an exponential number of variables. A second key advantage of this algorithm is that the iterations only require optimizing linear functions over the constrained domain, and we show that *this is equivalent to the maximization oracle* used by subgradient and cutting-plane methods (Joachims et al., 2009; Teo et al., 2010). Thus, the Frank-Wolfe algorithm has the same wide applicability as subgradient methods, and can be applied to problems such as low-treewidth graphical models (Taskar et al., 2003), graph matchings (Caetano et al., 2009), and associative Markov networks (Taskar, 2004). In contrast, other approaches must use more expensive (and potentially intractable) oracles such as computing marginals over labels (Collins et al., 2008; Zhang et al., 2011) or doing a Bregman projection onto the space of structures (Taskar et al., 2006). Interestingly, for structural SVMs we also show that existing batch subgradient and cutting-plane methods are *special cases* of Frank-Wolfe algorithms, and this leads to stronger and simpler $O(1/\varepsilon)$ convergence rate guarantees for these existing algorithms.

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

As in other batch structural SVM solvers like cutting-plane methods (Joachims et al., 2009; Teo et al., 2010) and the excessive gap technique (Zhang et al., 2011) (see Table 1 at the end for an overview), each Frank-Wolfe iteration unfortunately requires calling the appropriate oracle once for *all* training examples, unlike the single oracle call needed by stochastic subgradient methods. This can be prohibitive for data sets with a large number of training examples. To reduce this cost, we propose a novel randomized block-coordinate version of the Frank-Wolfe algorithm for problems with block-separable constraints. We show that this algorithm still achieves the $O(1/\varepsilon)$ convergence rate of the full Frank-Wolfe algorithm, and in the context of structural SVMs, it only requires a single call to the maximization oracle. Although the stochastic subgradient and the novel block-coordinate Frank-Wolfe algorithms have a similar iteration cost and theoretical convergence rate for solving the structural SVM problem, the new algorithm has several important advantages for practitioners:

- The *optimal* step-size can be efficiently computed in closed-form, hence no step-size needs to be selected.
- The algorithm yields a *duality gap* guarantee, and (at the cost of computing the primal objective) we can compute the duality gap as a proper stopping criterion.
- The convergence rate holds even when using *approximate* maximization oracles.

Further, our experimental results show that the optimal step-size leads to a significant advantage during the first few passes through the data, and a systematic (but smaller) advantage in later passes.

2. Structural Support Vector Machines

We first briefly review the standard convex optimization setup for structural SVMs (Taskar et al., 2003; Tsochantaridis et al., 2005). In structured prediction, the goal is to predict a structured object $\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})$ (such as a sequence of tags) for a given input $\boldsymbol{x} \in \mathcal{X}$. In the standard approach, a structured feature map $\boldsymbol{\phi} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$ encodes the relevant information for input/output pairs, and a linear classifier with parameter $\boldsymbol{w}$ is defined by $h_{\boldsymbol{w}}(\boldsymbol{x}) = \operatorname{argmax}_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle$. Given a labeled training set $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$, $\boldsymbol{w}$ is estimated by solving

$$\begin{aligned} \min_{\boldsymbol{w}, \boldsymbol{\xi}} \quad & \frac{\lambda}{2} \|\boldsymbol{w}\|^2 + \frac{1}{n} \sum_{i=1}^n \xi_i \\ \text{s.t.} \quad & \langle \boldsymbol{w}, \boldsymbol{\psi}_i(\boldsymbol{y}) \rangle \geq L(\boldsymbol{y}_i, \boldsymbol{y}) - \xi_i \quad \forall i, \forall \boldsymbol{y} \in \overbrace{\mathcal{Y}(\boldsymbol{x}_i)}^{=: \mathcal{Y}_i}, \end{aligned} \tag{1}$$

where $\boldsymbol{\psi}_i(\boldsymbol{y}) := \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y}_i) - \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y})$, and $L_i(\boldsymbol{y}) := L(\boldsymbol{y}_i, \boldsymbol{y})$ denotes the task-dependent structured error of predicting output $\boldsymbol{y}$ instead of the observed output $\boldsymbol{y}_i$ (typically a Hamming distance between the two labels). The slack variable ξ_i measures the surrogate loss for the i-th datapoint and λ is the regularization parameter. The convex problem (1) is what Joachims et al. (2009, Optimization Problem 2) call the n-slack structural SVM with margin-rescaling. A variant with *slack-rescaling* was proposed by Tsochantaridis et al. (2005), which is equivalent to our setting if we replace all vectors $\boldsymbol{\psi}_i(\boldsymbol{y})$ by $L_i(\boldsymbol{y})\boldsymbol{\psi}_i(\boldsymbol{y})$.

Loss-Augmented Decoding. Unfortunately, the above problem can have an exponential number of constraints due to the combinatorial nature of $\mathcal{Y}$. We can replace the $\sum_i |\mathcal{Y}_i|$ linear constraints with n *piecewise-linear* ones by defining the structured hinge-loss:

$$\text{'max oracle'} \qquad \tilde{H}_i(\boldsymbol{w}) := \max_{\boldsymbol{y} \in \mathcal{Y}_i} \underbrace{L_i(\boldsymbol{y}) - \langle \boldsymbol{w}, \boldsymbol{\psi}_i(\boldsymbol{y}) \rangle}_{=: H_i(\boldsymbol{y}; \boldsymbol{w})}. \tag{2}$$

The constraints in (1) can thus be replaced with the non-linear ones $\xi_i \geq \tilde{H}_i(\boldsymbol{w})$. The computation of the structured hinge-loss for each i amounts to finding the most 'violating' output $\boldsymbol{y}$ for a given input $\boldsymbol{x}_i$, a task which can be carried out efficiently in many structured prediction settings (see the introduction). This problem is called the *loss-augmented decoding* subproblem. In this paper, we only assume access to an efficient solver for this subproblem, and we call such a solver a *maximization oracle*. The equivalent non-smooth unconstrained formulation of (1) is:

$$\min_{\boldsymbol{w}} \quad \frac{\lambda}{2} \|\boldsymbol{w}\|^2 + \frac{1}{n} \sum_{i=1}^n \tilde{H}_i(\boldsymbol{w}). \tag{3}$$

Having a maximization oracle allows us to apply subgradient methods to this problem (Ratliff et al., 2007), as a subgradient of $\tilde{H}_i(\boldsymbol{w})$ with respect to $\boldsymbol{w}$ is $-\boldsymbol{\psi}_i(\boldsymbol{y}_i^*)$, where $\boldsymbol{y}_i^*$ is any maximizer of the loss-augmented decoding subproblem (2).

The Dual. The Lagrange dual of the above n-slack-formulation (1) has $m := \sum_i |\mathcal{Y}_i|$ variables or potential 'support vectors'. Writing $\alpha_i(\boldsymbol{y})$ for the dual variable associated with the training example i and potential output $\boldsymbol{y} \in \mathcal{Y}_i$, the dual problem is given by

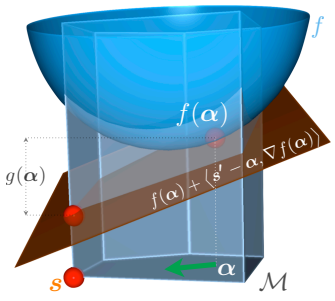
$$\begin{aligned} \min_{\substack{\boldsymbol{\alpha} \in \mathbb{R}^m \\ \boldsymbol{\alpha} \geq 0}} \quad & f(\boldsymbol{\alpha}) := \frac{\lambda}{2} \left\| A\boldsymbol{\alpha} \right\|^2 - \boldsymbol{b}^T \boldsymbol{\alpha} \\ \text{s.t.} \quad & \textstyle\sum_{\boldsymbol{y} \in \mathcal{Y}_i} \alpha_i(\boldsymbol{y}) = 1 \quad \forall i \in [n] \,, \end{aligned} \tag{4}$$

where the matrix $A \in \mathbb{R}^{d \times m}$ consists of the m columns $A := \left\{ \frac{1}{\lambda n} \boldsymbol{\psi}_i(\boldsymbol{y}) \in \mathbb{R}^d \;\middle|\; i \in [n], \boldsymbol{y} \in \mathcal{Y}_i \right\}$, and the vector $\boldsymbol{b} \in \mathbb{R}^m$ is given by $\boldsymbol{b} := \left(\frac{1}{n} L_i(\boldsymbol{y}) \right)_{i \in [n], \boldsymbol{y} \in \mathcal{Y}_i}$. Given a

dual variable vector $\boldsymbol{\alpha}$, we can use the Karush-Kuhn-Tucker optimality conditions to obtain the corresponding primal variables $\boldsymbol{w} = A\boldsymbol{\alpha} = \sum_{i,\,\boldsymbol{y}\in\mathcal{Y}_i} \alpha_i(\boldsymbol{y}) \frac{\boldsymbol{\psi}_i(\boldsymbol{y})}{\lambda n}$, see Appendix E. The gradient of f then takes the simple form $\nabla f(\boldsymbol{\alpha}) = \lambda A^T A\boldsymbol{\alpha} - \boldsymbol{b} = \lambda A^T \boldsymbol{w} - \boldsymbol{b}$; its $(i,\boldsymbol{y})$-th component is $-\frac{1}{n} H_i(\boldsymbol{y};\boldsymbol{w})$, cf. (2). Finally, note that the domain $\mathcal{M} \subset \mathbb{R}^m$ of (4) is the product of n probability simplices, $\mathcal{M} := \Delta_{|\mathcal{Y}_1|} \times \ldots \times \Delta_{|\mathcal{Y}_n|}$.

3. The Frank-Wolfe Algorithm

We consider the convex optimization problem $\min_{\boldsymbol{\alpha}\in\mathcal{M}} f(\boldsymbol{\alpha})$, where the convex feasible set $\mathcal{M}$ is *compact* and the convex objective f is *continuously differentiable*. The Frank-Wolfe algorithm (1956) (shown in Algorithm 1) is an iterative optimization algorithm for such problems that only requires optimizing *linear* functions over $\mathcal{M}$, and thus has wider applicability than projected gradient algorithms, which require optimizing a quadratic function over $\mathcal{M}$. At every iteration, a feasible search corner $\boldsymbol{s}$ is first found by minimizing over $\mathcal{M}$ the *linearization* of f at the current iterate $\boldsymbol{\alpha}$ (see picture in inset). The next iterate is then obtained as a convex combination of $\boldsymbol{s}$ and the previous iterate, with step-size γ. These simple updates yield two interesting properties. First, every iterate $\boldsymbol{\alpha}^{(k)}$ can be written as a convex combination of the starting point $\boldsymbol{\alpha}^{(0)}$ and the search corners $\boldsymbol{s}$ found previously. The parameter $\boldsymbol{\alpha}^{(k)}$ thus has a sparse representation, which makes the algorithm suitable even for cases where the dimensionality of $\boldsymbol{\alpha}$ is exponential. Second, since f is convex, the minimum of the linearization of f over $\mathcal{M}$ immediately gives a lower bound on the value of the yet unknown optimal solution $f(\boldsymbol{\alpha}^*)$. Every step of the algorithm thus computes for free the following 'linearization duality gap' defined for any feasible point $\boldsymbol{\alpha} \in \mathcal{M}$ (which is in fact a special case of the Fenchel duality gap as explained in Appendix D):



$$g(\boldsymbol{\alpha}) := \max_{\boldsymbol{s}'\in\mathcal{M}} \langle \boldsymbol{\alpha} - \boldsymbol{s}', \nabla f(\boldsymbol{\alpha}) \rangle = \langle \boldsymbol{\alpha} - \boldsymbol{s}, \nabla f(\boldsymbol{\alpha}) \rangle. \quad (5)$$

As $g(\boldsymbol{\alpha}) \geq f(\boldsymbol{\alpha}) - f(\boldsymbol{\alpha}^*)$ by the above argument, $\boldsymbol{s}$ thus readily gives at each iteration the current duality gap as a *certificate* for the current approximation quality (Jaggi, 2011; 2013), allowing us to monitor the convergence, and more importantly to choose the theoretically sound stopping criterion $g(\boldsymbol{\alpha}^{(k)}) \leq \varepsilon$.

In terms of convergence, it is known that after $O(1/\varepsilon)$ iterations, Algorithm 1 obtains an ε-approximate solution (Frank & Wolfe, 1956; Dunn & Harshbarger, 1978) as well as a guaranteed ε-small duality gap (Clarkson, 2010; Jaggi, 2013), along with a certificate to (5). For the convergence results to hold, the internal linear subproblem does not need to be solved exactly, but only to some error. We review and generalize the convergence proof in Appendix C. The constant hidden in the $O(1/\varepsilon)$ notation is the *curvature constant* C_f, an affine invariant quantity measuring the maximum deviation of f from its linear approximation over $\mathcal{M}$ (it yields a weaker form of Lipschitz assumption on the gradient, see e.g. Appendix A for a formal definition).

Algorithm 1 Frank-Wolfe on a Compact Domain

Let $\boldsymbol{\alpha}^{(0)} \in \mathcal{M}$
for $k = 0 \ldots K$ **do**
  Compute $\boldsymbol{s} := \operatorname{argmin}_{\boldsymbol{s}'\in\mathcal{M}} \left\langle \boldsymbol{s}', \nabla f(\boldsymbol{\alpha}^{(k)}) \right\rangle$
  Let $\gamma := \frac{2}{k+2}$, or optimize γ by line-search
  Update $\boldsymbol{\alpha}^{(k+1)} := (1-\gamma)\boldsymbol{\alpha}^{(k)} + \gamma \boldsymbol{s}$

4. Frank-Wolfe for Structural SVMs

Note that classical algorithms like the projected gradient method cannot be tractably applied to the dual of the structural SVM problem (4), due to the large number of dual variables. In this section, we explain how the Frank-Wolfe method (Algorithm 1) can be efficiently applied to this dual problem, and discuss its relationship to other algorithms. The main insight here is to notice that the linear subproblem employed by Frank-Wolfe is actually directly equivalent to the loss-augmented decoding subproblem (2) for each datapoint, which can be solved efficiently (see Appendix B.1 for details). Recall that the optimization domain for the dual variables $\boldsymbol{\alpha}$ is the product

Algorithm 2 Batch Primal-Dual Frank-Wolfe Algorithm for the Structural SVM

Let $\boldsymbol{w}^{(0)} := \boldsymbol{0}$, $\ell^{(0)} := 0$
for $k = 0 \ldots K$ **do**
  for $i = 1 \ldots n$ **do**
    Solve $\boldsymbol{y}_i^* := \operatorname{argmax}_{\boldsymbol{y}\in\mathcal{Y}_i} H_i(\boldsymbol{y};\boldsymbol{w}^{(k)})$ cf. (2)
  Let $\boldsymbol{w}_{\boldsymbol{s}} := \sum_{i=1}^n \frac{1}{\lambda n} \boldsymbol{\psi}_i(\boldsymbol{y}_i^*)$ and $\ell_{\boldsymbol{s}} := \frac{1}{n} \sum_{i=1}^n L_i(\boldsymbol{y}_i^*)$
  Let $\gamma := \frac{\lambda(\boldsymbol{w}^{(k)} - \boldsymbol{w}_{\boldsymbol{s}})^T \boldsymbol{w}^{(k)} - \ell^{(k)} + \ell_{\boldsymbol{s}}}{\lambda \|\boldsymbol{w}^{(k)} - \boldsymbol{w}_{\boldsymbol{s}}\|^2}$ and clip to $[0,1]$
  Update $\boldsymbol{w}^{(k+1)} := (1-\gamma)\boldsymbol{w}^{(k)} + \gamma \boldsymbol{w}_{\boldsymbol{s}}$
    and $\ell^{(k+1)} := (1-\gamma)\ell^{(k)} + \gamma \ell_{\boldsymbol{s}}$

of n simplices, $\mathcal{M} = \Delta_{|\mathcal{Y}_1|} \times \ldots \times \Delta_{|\mathcal{Y}_n|}$. Since each simplex consists of a potentially exponential number $|\mathcal{Y}_i|$ of dual variables, we cannot maintain a dense vector $\boldsymbol{\alpha}$ during the algorithm. However, as mentioned in Section 3, each iterate $\boldsymbol{\alpha}^{(k)}$ of the Frank-Wolfe algorithm is a sparse convex combination of the previously visited corners $\boldsymbol{s}$ and the starting point $\boldsymbol{\alpha}^{(0)}$, and so we only need to maintain the list of previously seen solutions to the loss-augmented decoding subproblems to keep track of the non-zero coordinates of $\boldsymbol{\alpha}$, avoiding the problem of its exponential size. Alternately, if we do not use kernels, we can avoid the quadratic explosion of the number of operations needed in the dual by *not* explicitly maintaining $\boldsymbol{\alpha}^{(k)}$, but instead maintaining the corresponding *primal* variable $\boldsymbol{w}^{(k)}$.

A Primal-Dual Frank-Wolfe Algorithm for the Structural SVM Dual. Applying Algorithm 1 with line search to the dual of the structural SVM (4), but only maintaining the corresponding *primal* primal iterates $\boldsymbol{w}^{(k)} := A\boldsymbol{\alpha}^{(k)}$, we obtain Algorithm 2. Note that the Frank-Wolfe search corner $\boldsymbol{s} = (\mathbf{e}^{\boldsymbol{y}_1^*}, \ldots, \mathbf{e}^{\boldsymbol{y}_n^*})$, which is obtained by solving the loss-augmented subproblems, yields the update $\boldsymbol{w}_{\boldsymbol{s}} = A\boldsymbol{s}$. We use the natural starting point $\boldsymbol{\alpha}^{(0)} := (\mathbf{e}^{\boldsymbol{y}_1}, \ldots, \mathbf{e}^{\boldsymbol{y}_n})$ which yields $\boldsymbol{w}^{(0)} = \mathbf{0}$ as $\boldsymbol{\psi}_i(\boldsymbol{y}_i) = \mathbf{0}\ \forall i$.

The Duality Gap. The duality gap (5) for our structural SVM dual formulation (4) is given by

$$\begin{aligned} g(\boldsymbol{\alpha}) := & \max_{\boldsymbol{s}' \in \mathcal{M}} \langle \boldsymbol{\alpha} - \boldsymbol{s}', \nabla f(\boldsymbol{\alpha}) \rangle \\ = & (\boldsymbol{\alpha} - \boldsymbol{s})^T (\lambda A^T A \boldsymbol{\alpha} - \boldsymbol{b}) \\ = & \lambda (\boldsymbol{w} - A\boldsymbol{s})^T \boldsymbol{w} - \boldsymbol{b}^T \boldsymbol{\alpha} + \boldsymbol{b}^T \boldsymbol{s} \ , \end{aligned}$$

where $\boldsymbol{s}$ is an *exact* minimizer of the linearized problem given at the point $\boldsymbol{\alpha}$. This (Fenchel) duality gap turns out to be the same as the Lagrangian duality gap here (see Appendix B.2), and gives a direct handle on the suboptimality of $\boldsymbol{w}^{(k)}$ for the primal problem (3). Using $\boldsymbol{w}_{\boldsymbol{s}} := A\boldsymbol{s}$ and $\ell_{\boldsymbol{s}} := \boldsymbol{b}^T \boldsymbol{s}$, we observe that the gap is efficient to compute given the *primal* variables $\boldsymbol{w} := A\boldsymbol{\alpha}$ and $\ell := \boldsymbol{b}^T \boldsymbol{\alpha}$, which are maintained during the run of Algorithm 2. Therefore, we can use the duality gap $g(\boldsymbol{\alpha}^{(k)}) \leq \varepsilon$ as a proper stopping criterion.

Implementing the Line-Search. Because the objective of the structural SVM dual (4) is simply a quadratic function in $\boldsymbol{\alpha}$, the optimal step-size for any given candidate search point $\boldsymbol{s} \in \mathcal{M}$ can be obtained *analytically*. Namely, $\gamma_{LS} := \operatorname{argmin}_{\gamma \in [0,1]} f\big(\boldsymbol{\alpha} + \gamma(\boldsymbol{s} - \boldsymbol{\alpha})\big)$ is obtained by setting the derivative of this univariate quadratic function in γ to zero, which here (before restricting to $[0, 1]$) gives $\gamma_{opt} := \frac{\langle \boldsymbol{\alpha} - \boldsymbol{s}, \nabla f(\boldsymbol{\alpha}) \rangle}{\lambda \|A(\boldsymbol{\alpha} - \boldsymbol{s})\|^2} = \frac{g(\boldsymbol{\alpha})}{\lambda \|\boldsymbol{w} - \boldsymbol{w}_{\boldsymbol{s}}\|^2}$ (used in Algorithms 2 and 4).

Convergence Proof and Running Time. In the following, we write R for the maximal length of a difference feature vector, i.e. $R := \max_{i \in [n], \boldsymbol{y} \in \mathcal{Y}_i} \|\boldsymbol{\psi}_i(\boldsymbol{y})\|_2$, and we write the maximum error as $L_{\max} := \max_{i, \boldsymbol{y}} L_i(\boldsymbol{y})$. By bounding the curvature constant C_f for the dual SVM objective (4), we can now directly apply the known convergence results for the standard Frank-Wolfe algorithm to obtain the following *primal-dual* rate (proof in Appendix B.3):

Theorem 1. *Algorithm 2 obtains an ε-approximate solution to the structural SVM dual problem (4) and duality gap $g(\boldsymbol{\alpha}^{(k)}) \leq \varepsilon$ after at most $O\left(\frac{R^2}{\lambda \varepsilon}\right)$ iterations, where each iteration costs n oracle calls.*

Since we have proved that the duality gap is smaller than ε, this implies that the original SVM primal objective (3) is actually solved to accuracy ε as well.

Relationship with the Batch Subgradient Method in the Primal. Surprisingly, the batch Frank-Wolfe method (Algorithm 2) is equivalent to the batch subgradient method in the primal, though Frank-Wolfe allows a more clever choice of step-size, since line-search can be used in the dual. To see the equivalence, notice that a subgradient of (3) is given by $\boldsymbol{d}_{sub} = \lambda \boldsymbol{w} - \frac{1}{n} \sum_i \boldsymbol{\psi}_i(\boldsymbol{y}_i^*) = \lambda(\boldsymbol{w} - \boldsymbol{w}_{\boldsymbol{s}})$, where $\boldsymbol{y}_i^*$ and $\boldsymbol{w}_{\boldsymbol{s}}$ are as defined in Algorithm 2. Hence, for a step-size of β, the subgradient method update becomes $\boldsymbol{w}^{(k+1)} := \boldsymbol{w}^{(k)} - \beta \boldsymbol{d}_{sub} = \boldsymbol{w}^{(k)} - \beta\lambda(\boldsymbol{w}^{(k)} - \boldsymbol{w}_{\boldsymbol{s}}) = (1 - \beta\lambda)\boldsymbol{w}^{(k)} + \beta\lambda \boldsymbol{w}_{\boldsymbol{s}}$. Comparing this with Algorithm 2, we see that each Frank-Wolfe step on the dual problem (4) with step-size γ is equivalent to a batch subgradient step in the primal with a step-size of $\beta = \gamma/\lambda$, and thus our convergence results also apply to it. This seems to generalize the equivalence between Frank-Wolfe and the subgradient method for a quadratic objective with identity Hessian as observed by Bach et al. (2012, Section 4.1).

Relationship with Cutting Plane Algorithms. In each iteration, the cutting plane algorithm of Joachims et al. (2009) and the Frank-Wolfe method (Algorithm 2) solve the loss-augmented decoding problem for each datapoint, selecting the same new 'active' coordinates to add to the dual problem. The only difference is that instead of just moving towards the corner $\boldsymbol{s}$, as in classical Frank-Wolfe, the cutting plane algorithm re-optimizes over all the previously added 'active' dual variables (this task is a quadratic program). This shows that the method is exactly equivalent to the 'fully corrective' variant of Frank-Wolfe, which in each iteration re-optimizes over all previously visited corners (Clarkson, 2010; Shalev-Shwartz et al., 2010b). Note that the convergence results for the 'fully correc-

Algorithm 3 Block-Coordinate Frank-Wolfe Algorithm on Product Domain

Let $\boldsymbol{\alpha}^{(0)} \in \mathcal{M} = \mathcal{M}^{(1)} \times \ldots \times \mathcal{M}^{(n)}$
for $k = 0 \ldots K$ **do**
  Pick i at random in $\{1, \ldots, n\}$
  Find $\boldsymbol{s}_{(i)} := \underset{\boldsymbol{s}'_{(i)} \in \mathcal{M}^{(i)}}{\operatorname{argmin}} \left\langle \boldsymbol{s}'_{(i)}, \nabla_{(i)} f(\boldsymbol{\alpha}^{(k)}) \right\rangle$
  Let $\gamma := \frac{2n}{k+2n}$, or optimize γ by line-search
  Update $\boldsymbol{\alpha}_{(i)}^{(k+1)} := \boldsymbol{\alpha}_{(i)}^{(k)} + \gamma\big(\boldsymbol{s}_{(i)} - \boldsymbol{\alpha}_{(i)}^{(k)}\big)$

tive' variant directly follow from the ones for Frank-Wolfe (by inclusion), thus our convergence results apply to the cutting plane algorithm of Joachims et al. (2009), significantly simplifying its analysis.

5. Faster Block-Coordinate Frank-Wolfe

A major disadvantage of the standard Frank-Wolfe algorithm when applied to the structural SVM problem is that each iteration requires a full pass through the data, resulting in n calls to the maximization oracle. In this section, we present the main new contribution of the paper: a *block-coordinate* generalization of the Frank-Wolfe algorithm that maintains all appealing properties of Frank-Wolfe, but yields much cheaper iterations, requiring only one call to the maximization oracle in the context of structural SVMs. The new method is given in Algorithm 3, and applies to any constrained convex optimization problem of the form

$$\min_{\boldsymbol{\alpha} \in \mathcal{M}^{(1)} \times \ldots \times \mathcal{M}^{(n)}} f(\boldsymbol{\alpha}) \;, \tag{6}$$

where the domain has the structure of a Cartesian product $\mathcal{M} = \mathcal{M}^{(1)} \times \ldots \times \mathcal{M}^{(n)} \subseteq \mathbb{R}^m$ over $n \geq 1$ blocks. The main idea of the method is to perform cheaper update steps that only affect a single variable block $\mathcal{M}^{(i)}$, and not all of them simultaneously. This is motivated by coordinate descent methods, which have a very successful history when applied to large scale optimization. Here we assume that each factor $\mathcal{M}^{(i)} \subseteq \mathbb{R}^{m_i}$ is convex and *compact*, with $m = \sum_{i=1}^n m_i$. We will write $\boldsymbol{\alpha}_{(i)} \in \mathbb{R}^{m_i}$ for the i-th block of coordinates of a vector $\boldsymbol{\alpha} \in \mathbb{R}^m$. In each step, Algorithm 3 picks one of the n blocks uniformly at random, and leaves all other blocks unchanged. If there is only one block ($n = 1$), then Algorithm 3 becomes the standard Frank-Wolfe Algorithm 1. The algorithm can be interpreted as a simplification of Nesterov's 'huge-scale' uniform coordinate descent method (Nesterov, 2012, Section 4). Here, instead of computing a projection operator on a block (which is intractable for structural SVMs), we only need to solve one linear subproblem in each iteration, which for structural SVMs is equivalent to a call to the maximization oracle.

Algorithm 4 Block-Coordinate Primal-Dual Frank-Wolfe Algorithm for the Structural SVM

Let $\boldsymbol{w}^{(0)} := \boldsymbol{w}_i{}^{(0)} := \bar{\boldsymbol{w}}^{(0)} := \mathbf{0}, \quad \ell^{(0)} := \ell_i{}^{(0)} := 0$
for $k = 0 \ldots K$ **do**
  Pick i at random in $\{1, \ldots, n\}$
  Solve $\boldsymbol{y}_i^* := \underset{\boldsymbol{y} \in \mathcal{Y}_i}{\operatorname{argmax}} \; H_i(\boldsymbol{y}; \boldsymbol{w}^{(k)})$ cf. (2)
  Let $\boldsymbol{w}_{\boldsymbol{s}} := \frac{1}{\lambda n} \boldsymbol{\psi}_i(\boldsymbol{y}_i^*) \quad$ and $\quad \ell_{\boldsymbol{s}} := \frac{1}{n} L_i(\boldsymbol{y}_i^*)$
  Let $\gamma := \frac{\lambda(\boldsymbol{w}_i^{(k)} - \boldsymbol{w}_{\boldsymbol{s}})^T \boldsymbol{w}^{(k)} - \ell_i^{(k)} + \ell_{\boldsymbol{s}}}{\lambda \|\boldsymbol{w}_i^{(k)} - \boldsymbol{w}_{\boldsymbol{s}}\|^2}$ and clip to $[0, 1]$
  Update $\boldsymbol{w}_i{}^{(k+1)} := (1 - \gamma)\boldsymbol{w}_i{}^{(k)} + \gamma\, \boldsymbol{w}_{\boldsymbol{s}}$
    and $\ell_i{}^{(k+1)} := (1 - \gamma)\ell_i{}^{(k)} + \gamma\, \ell_{\boldsymbol{s}}$
  Update $\boldsymbol{w}^{(k+1)} := \boldsymbol{w}^{(k)} + \boldsymbol{w}_i{}^{(k+1)} - \boldsymbol{w}_i{}^{(k)}$
    and $\ell^{(k+1)} := \ell^{(k)} + \ell_i{}^{(k+1)} - \ell_i{}^{(k)}$
  (Optionally: Update $\bar{\boldsymbol{w}}^{(k+1)} := \frac{k}{k+2} \bar{\boldsymbol{w}}^{(k)} + \frac{2}{k+2} \boldsymbol{w}^{(k+1)}$)

Convergence Results. The following main theorem shows that after $O(1/\varepsilon)$ many iterations, Algorithm 3 obtains an ε-approximate solution to (6), and guaranteed ε-small duality gap (proof in Appendix C). Here the constant $C_f^\otimes := \sum_{i=1}^n C_f^{(i)}$ is the sum of the (partial) curvature constants of f with respect to the individual domain block $\mathcal{M}^{(i)}$. We discuss this Lipschitz assumption on the gradient in more details in Appendix A, where we compute the constant precisely for the structural SVM and obtain $C_f^\otimes = C_f / n$, where C_f is the classical Frank-Wolfe curvature.

Theorem 2. *For each $k \geq 0$, the iterate $\boldsymbol{\alpha}^{(k)}$ of Algorithm 3 (either using the predefined step-sizes, or using line-search) satisfies $\mathrm{E}\big[f(\boldsymbol{\alpha}^{(k)})\big] - f(\boldsymbol{\alpha}^*) \leq \frac{2n}{k+2n}\big(C_f^\otimes + h_0\big)$, where $\boldsymbol{\alpha}^* \in \mathcal{M}$ is a solution to problem (6), $h_0 := f(\boldsymbol{\alpha}^{(0)}) - f(\boldsymbol{\alpha}^*)$ is the initial error at the starting point of the algorithm, and the expectation is over the random choice of the block i in the steps of the algorithm.*

Furthermore, if Algorithm 3 is run for $K \geq 0$ iterations, then it has an iterate $\boldsymbol{\alpha}^{(\hat{k})}$, $0 \leq \hat{k} \leq K$, with duality gap bounded by $\mathrm{E}\big[g(\boldsymbol{\alpha}^{(\hat{k})})\big] \leq \frac{6n}{K+1}\big(C_f^\otimes + h_0\big)$.

Application to the Structural SVM. Algorithm 4 applies the block-coordinate Frank-Wolfe algorithm with line-search to the structural SVM dual problem (4), maintaining only the primal variables $\boldsymbol{w}$. We see that Algorithm 4 is equivalent to Algorithm 3, by observing that the corresponding primal updates become $\boldsymbol{w}_{\boldsymbol{s}} = A\boldsymbol{s}_{[i]}$ and $\ell_{\boldsymbol{s}} = \boldsymbol{b}^T \boldsymbol{s}_{[i]}$. Here $\boldsymbol{s}_{[i]}$ is the zero-padding of $\boldsymbol{s}_{(i)} := \mathbf{e}^{\boldsymbol{y}_i^*} \in \mathcal{M}^{(i)}$ so that $\boldsymbol{s}_{[i]} \in \mathcal{M}$. Note that Algorithm 4 has a primal parameter vector $\boldsymbol{w}_i$ ($= A\boldsymbol{\alpha}_{[i]}$) for each datapoint i, but that this does not significantly increase the storage cost of the algorithm since each $\boldsymbol{w}_i$ has a sparsity pattern that is the union of the corresponding $\boldsymbol{\psi}_i(\boldsymbol{y}_i^*)$ vectors. If the feature vectors are not sparse, it might be more efficient

to work directly in the dual instead (see the kernelized version below). The line-search is analogous to the batch Frank-Wolfe case discussed above, and formalized in Appendix B.4.

By applying Theorem 2 to the SVM case where $C_f^{\otimes} = C_f/n = 4R^2/\lambda n$ (in the worst case), we get that the number of iterations needed for our new block-wise Algorithm 4 to obtain a specific accuracy ε is the same as for the batch version in Algorithm 2 (under the assumption that the initial error h_0 is smaller than $4R^2/\lambda n$), even though each iteration takes n times fewer oracle calls. If $h_0 > 4R^2/\lambda n$, we can use the fact that Algorithm 4 is using line-search to get a weaker dependence on h_0 in the rate (Theorem C.4). We summarize the overall rate as follows (proof in Appendix B.3):

Theorem 3. *If $L_{max} \leq \frac{4R^2}{\lambda n}$ (so $h_0 \leq \frac{4R^2}{\lambda n}$), then Algorithm 4 obtains an ε-approximate solution to the structural SVM dual problem (4) and expected duality gap $\mathrm{E}[g(\boldsymbol{\alpha}^{(k)})] \leq \varepsilon$ after at most $O\left(\frac{R^2}{\lambda\varepsilon}\right)$ iterations, where each iteration costs a single oracle call.*

If $L_{max} > \frac{4R^2}{\lambda n}$, then it requires at most an additional (constant in ε) number of $O\left(n \log\left(\frac{\lambda n L_{max}}{R^2}\right)\right)$ steps to get the same error and duality gap guarantees.

In terms of ε, the $O(1/\varepsilon)$ convergence rate above is similar to existing stochastic subgradient and cutting-plane methods. However, unlike stochastic subgradient methods, the block-coordinate Frank-Wolfe method allows us to compute the optimal step-size at each iteration (while for an additional pass through the data we can evaluate the duality gap (5) to allow us to decide when to terminate the algorithm in practice). Further, unlike cutting-plane methods which require n oracle calls per iteration, this rate is achieved 'online', using only a single oracle call per iteration.

Approximate Subproblems and Decoding. Interestingly, we can show that all the convergence results presented in this paper also hold if only approximate minimizers of the linear subproblems are used instead of exact minimizers. If we are using an approximate oracle giving candidate directions $\boldsymbol{s}_{(i)}$ in Algorithm 3 (or $\boldsymbol{s}$ in Algorithm 1) with a *multiplicative* accuracy $\nu \in (0, 1]$ (with respect to the the duality gap (5) on the current block), then the above convergence bounds from Theorem 2 still apply. The only change is that the convergence is slowed by a factor of $1/\nu^2$. We prove this generalization in the Theorems of Appendix C. For structural SVMs, this significantly improves the applicability to large-scale problems, where *exact* decoding is often too costly but *approximate* loss-augmented decoding may be possible.

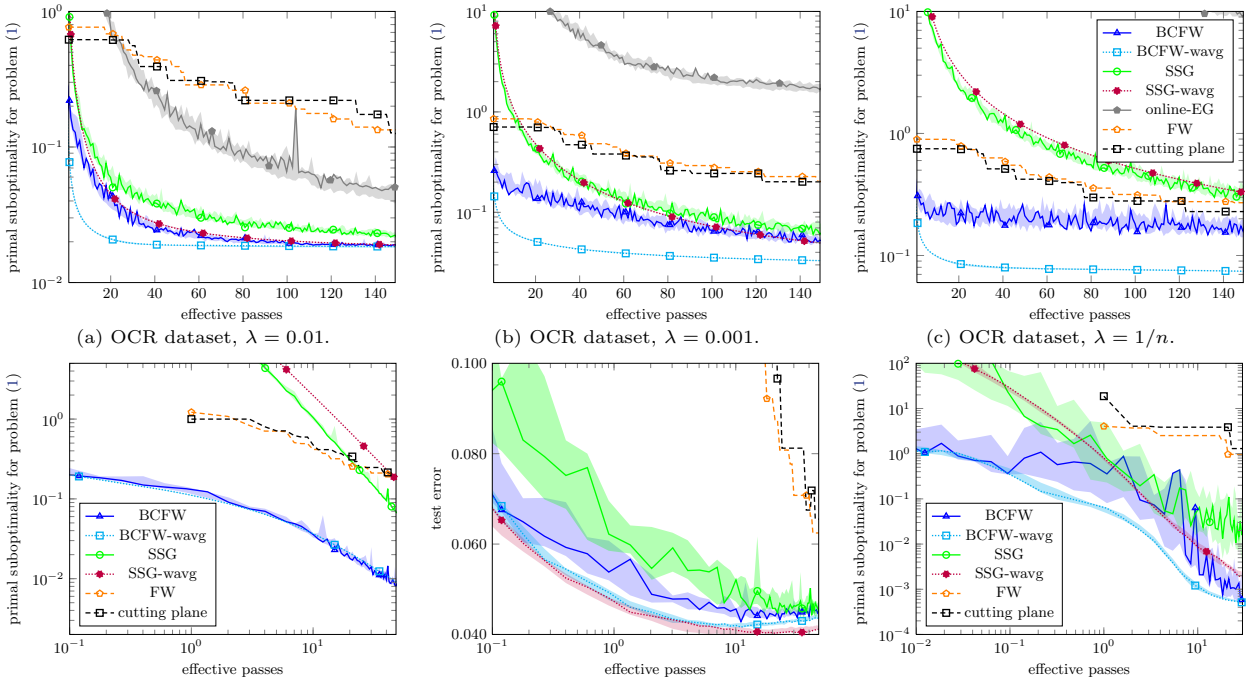
Kernelized Algorithms. Both Algorithms 2 and 4 can directly be used with kernels by maintaining the sparse dual variables $\boldsymbol{\alpha}^{(k)}$ instead of the primal variables $\boldsymbol{w}^{(k)}$. In this case, the classifier is only given implicitly as a sparse combination of the corresponding kernel functions, i.e. $\boldsymbol{w} = A\boldsymbol{\alpha}$. Using our Algorithm 4, we obtain the currently best known bound on the *number of support vectors*, i.e. a guaranteed ε-approximation with only $O(\frac{R^2}{\lambda\varepsilon})$ support vectors. In comparison, the standard cutting plane method (Joachims et al., 2009) adds n support vectors $\boldsymbol{\psi}_i(\boldsymbol{y}_i^*)$ at each iteration. More details on the kernelized variant of Algorithm 4 are discussed in Appendix B.5.

6. Experiments

We compare our novel Frank-Wolfe approach to existing algorithms for training structural SVMs on the OCR dataset ($n = 6251, d = 4028$) from Taskar et al. (2003) and the CoNLL dataset ($n = 8936, d = 1643026$) from Sang & Buchholz (2000). Both datasets are sequence labeling tasks, where the loss-augmented decoding problem can be solved exactly by the Viterbi algorithm. Our third application is a word alignment problem between sentences in different languages in the setting of Taskar et al. (2006) ($n = 5000, d = 82$). Here, the structured labels are bipartite matchings, for which computing marginals over labels as required by the methods of Collins et al. (2008); Zhang et al. (2011) is intractable, but loss-augmented decoding can be done efficiently by solving a min-cost flow problem.

We compare Algorithms 2 and 4, the batch Frank-Wolfe method (*FW*)[1] and our novel block-coordinate Frank-Wolfe method (*BCFW*), to the *cutting plane* algorithm implemented in SVMstruct (Joachims et al., 2009) with its default options, the online exponentiated gradient (*online-EG*) method of Collins et al. (2008), and the stochastic subgradient method (*SSG*) with step-size chosen as in the 'Pegasos' version of Shalev-Shwartz et al. (2010a). We also include the weighted average $\bar{\boldsymbol{w}}^{(k)} := \frac{2}{k(k+1)} \sum_{t=1}^{k} t\boldsymbol{w}^{(t)}$ of the iterates from *SSG* (called *SSG-wavg*) which was recently shown to converge at the faster rate of $O(1/k)$ instead of $O((\log k)/k)$ (Lacoste-Julien et al., 2012; Shamir & Zhang, 2013). Analogously, we average the iterates from *BCFW* the same way to obtain the *BCFW-wavg* method (implemented efficiently with the optional line in Algorithm 4), which also has a provable $O(1/k)$ convergence rate (Theorem C.3). The performance of the different algorithms according to several criteria is visualized in Figure 1. The results are discussed

[1] This is equivalent to the batch subgradient method with an adaptive step-size, as mentioned in Section 4.



(a) OCR dataset, $\lambda = 0.01$. (b) OCR dataset, $\lambda = 0.001$. (c) OCR dataset, $\lambda = 1/n$.

(d) CoNLL dataset, $\lambda = 1/n$. (e) Test error for $\lambda = 1/n$ on CoNLL. (f) Matching dataset, $\lambda = 0.001$.

Figure 1. The shaded areas for the stochastic methods (*BCFW*, *SSG* and *online-EG*) indicate the worst and best objective achieved in 10 randomized runs. The top row compares the suboptimality achieved by different solvers for different regularization parameters λ. For large λ (a), the stochastic algorithms (*BCFW* and *SSG*) perform considerably better than the batch solvers (*cutting plane* and *FW*). For a small λ (c), even the batch solvers achieve a lower objective earlier on than *SSG*. Our proposed *BCFW* algorithm achieves a low objective in both settings. (d) shows the convergence for CoNLL with the first passes in more details. Here *BCFW* already results in a low objective even after seeing only few datapoints. The advantage is less clear for the test error in (e) though, where *SSG-wavg* does surprisingly well. Finally, (f) compares the methods for the matching prediction task.

in the caption, while additional experiments can be found in Appendix F. In most of the experiments, the *BCFW-wavg* method dominates all competitors. The superiority is especially striking for the first few iterations, and when using a small regularization strength λ, which is often needed in practice. In term of test error, a peculiar observation is that the weighted average of the iterates seems to help both methods significantly: *SSG-wavg* sometimes slightly outperforms *BCFW-wavg* despite having the worst objective value amongst all methods. This phenomenon is worth further investigation.

7. Related Work

There has been substantial work on dual coordinate descent for SVMs, including the original sequential minimal optimization (SMO) algorithm. The SMO algorithm was generalized to structural SVMs (Taskar, 2004, Chapter 6), but its convergence rate scales badly with the size of the output space: it was estimated as $O(n|\mathcal{Y}|/\lambda\varepsilon)$ in Zhang et al. (2011). Further, this method requires an expectation oracle to work with its factored dual parameterization. As in our algorithm, Rousu et al. (2006) propose updating one training example at a time, but using multiple Frank-Wolfe updates to optimize along the subspace. However, they do not obtain any rate guarantees and their algorithm is less general because it again requires an expectation oracle. In the degenerate *binary* SVM case, our block-coordinate Frank-Wolfe algorithm is actually equivalent to the method of Hsieh et al. (2008), where because each datapoint has a unique dual variable, exact coordinate optimization can be accomplished by the line-search step of our algorithm. Hsieh et al. (2008) show a local linear convergence rate in the dual, and our results complement theirs by providing a global *primal* convergence guarantee for their algorithm of $O(1/\varepsilon)$. After our paper had appeared on arXiv, Shalev-Shwartz & Zhang (2012) have proposed a generalization of dual coordinate descent applicable to several regularized losses, including the structural SVM objective. Despite being motivated from a different perspective, a version of their algorithm (Option II of Figure 1) gives the exact same step-size and update direction as *BCFW* with line-search, and their Corol-

Table 1. Convergence rates given in the *number of calls to the oracles* for different optimization algorithms for the structural SVM objective (1) in the case of a Markov random field structure, to reach a specific accuracy ε measured for different types of gaps, in term of the number of training examples n, regularization parameter λ, size of the label space $|\mathcal{Y}|$, maximum feature norm $R := \max_{i,\boldsymbol{y}} \|\boldsymbol{\psi}_i(\boldsymbol{y})\|_2$ (some minor terms were ignored for succinctness). Table inspired from (Zhang et al., 2011). Notice that only stochastic subgradient and our proposed algorithm have rates independent of n.

Optimization algorithm	Online	Primal/Dual	Type of guarantee	Oracle type	# Oracle calls
dual extragradient (Taskar et al., 2006)	no	primal-'dual'	saddle point gap	Bregman projection	$O\left(\frac{nR\log\|\mathcal{Y}\|}{\lambda\varepsilon}\right)$
online exponentiated gradient (Collins et al., 2008)	yes	dual	expected dual error	expectation	$O\left(\frac{(n+\log\|\mathcal{Y}\|)R^2}{\lambda\varepsilon}\right)$
excessive gap reduction (Zhang et al., 2011)	no	primal-dual	duality gap	expectation	$O\left(nR\sqrt{\frac{\log\|\mathcal{Y}\|}{\lambda\varepsilon}}\right)$
BMRM (Teo et al., 2010)	no	primal	$\geq$primal error	maximization	$O\left(\frac{nR^2}{\lambda\varepsilon}\right)$
1-slack SVM-Struct (Joachims et al., 2009)	no	primal-dual	duality gap	maximization	$O\left(\frac{nR^2}{\lambda\varepsilon}\right)$
stochastic subgradient (Shalev-Shwartz et al., 2010a)	yes	primal	primal error w.h.p.	maximization	$\tilde{O}\left(\frac{R^2}{\lambda\varepsilon}\right)$
this paper: block-coordinate Frank-Wolfe	yes	primal-dual	expected duality gap	maximization	$O\left(\frac{R^2}{\lambda\varepsilon}\right)$ Thm. 3

lary 3 gives a similar convergence rate as our Theorem 3. Balamurugan et al. (2011) propose to approximately solve a quadratic problem on each example using *SMO*, but they do not provide any rate guarantees. The *online-EG* method implements a variant of dual coordinate descent, but it requires an expectation oracle and Collins et al. (2008) estimate its primal convergence at only $O\left(1/\varepsilon^2\right)$.

Besides coordinate descent methods, a variety of other algorithms have been proposed for structural SVMs. We summarize a few of the most popular in Table 1, with their convergence rates quoted in number of oracle calls to reach an accuracy of ε. However, we note that almost no guarantees are given for the optimization of structural SVMs with approximate oracles. A regret analysis in the context of online optimization was considered by Ratliff et al. (2007), but they do not analyze the effect of this on solving the optimization problem. The cutting plane algorithm of Tsochantaridis et al. (2005) was considered with approximate maximization by Finley & Joachims (2008), though the dependence of the running time on the the approximation error was left unclear. In contrast, we provide guarantees for batch subgradient, cutting plane, and block-coordinate Frank-Wolfe, for achieving an ε-approximate solution as long as the error of the oracle is appropriately bounded.

8. Discussion

This work proposes a novel randomized block-coordinate generalization of the classic Frank-Wolfe algorithm for optimization with block-separable constraints. Despite its potentially much lower iteration cost, the new algorithm achieves a similar convergence rate in the duality gap as the full Frank-Wolfe method. For the dual structural SVM optimization problem, it leads to a simple online algorithm that yields a solution to an issue that is notoriously difficult to address for stochastic algorithms: no step-size sequence needs to be tuned since the optimal step-size can be efficiently computed in closed-form. Further, at the cost of an additional pass through the data (which could be done alongside a full Frank-Wolfe iteration), it allows us to compute a duality gap guarantee that can be used to decide when to terminate the algorithm. Our experiments indicate that empirically it converges faster than other stochastic algorithms for the structural SVM problem, especially in the realistic setting where only a few passes through the data are possible.

Although our structural SVM experiments use an exact maximization oracle, the duality gap guarantees, the optimal step-size, and a computable bound on the duality gap are all still available when only an appropriate approximate maximization oracle is used. Finally, although the structural SVM problem is what motivated this work, we expect that the block-coordinate Frank-Wolfe algorithm may be useful for other problems in machine learning where a complex objective with block-separable constraints arises.

Acknowledgements. We thank Francis Bach, Bernd Gärtner and Ronny Luss for helpful discussions, and Robert Carnecky for the 3D illustration. MJ is supported by the ERC Project SIPA, and by the Swiss National Science Foundation. SLJ and MS are partly supported by the ERC (SIERRA-ERC-239993). SLJ is supported by a Research in Paris fellowship. MS is supported by a NSERC postdoctoral fellowship.

References

Bach, F., Lacoste-Julien, S., and Obozinski, G. On the equivalence between herding and conditional gradient algorithms. In *ICML*, 2012.

Balamurugan, P., Shevade, S., Sundararajan, S., and Keerthi, S. A sequential dual method for structural SVMs. In *SDM*, 2011.

Caetano, T.S., McAuley, J.J., Cheng, Li, Le, Q.V., and Smola, A.J. Learning graph matching. *IEEE PAMI*, 31 (6):1048–1058, 2009.

Clarkson, K. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Transactions on Algorithms*, 6(4):1–30, 2010.

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

Dunn, J.C. and Harshbarger, S. Conditional gradient algorithms with open loop step size rules. *Journal of Mathematical Analysis and Applications*, 62(2):432–444, 1978.

Finley, T. and Joachims, T. Training structural SVMs when exact inference is intractable. In *ICML*, 2008.

Frank, M. and Wolfe, P. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

Gärtner, B. and Jaggi, M. Coresets for polytope distance. *ACM Symposium on Computational Geometry*, 2009.

Hsieh, C., Chang, K., Lin, C., Keerthi, S., and Sundararajan, S. A dual coordinate descent method for large-scale linear SVM. In *ICML*, pp. 408–415, 2008.

Jaggi, M. *Sparse convex optimization methods for machine learning*. PhD thesis, ETH Zürich, 2011.

Jaggi, M. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *ICML*, 2013.

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

Lacoste-Julien, S., Schmidt, M., and Bach, F. A simpler approach to obtaining an O(1/t) convergence rate for the projected stochastic subgradient method. Technical Report 1212.2002v2 [cs.LG], arXiv, December 2012.

Mangasarian, O.L. Machine learning via polyhedral concave minimization. Technical Report 95-20, University of Wisconsin, 1995.

Nesterov, Yurii. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Ouyang, H. and Gray, A. Fast stochastic Frank-Wolfe algorithms for nonlinear SVMs. *SDM*, 2010.

Rakhlin, A., Shamir, O., and Sridharan, K. Making gradient descent optimal for strongly convex stochastic optimization. In *ICML*, 2012.

Ratliff, N., Bagnell, J. A., and Zinkevich, M. (Online) subgradient methods for structured prediction. In *AISTATS*, 2007.

Rousu, J., Saunders, C., Szedmak, S., and Shawe-Taylor, J. Kernel-based learning of hierarchical multilabel classification models. *JMLR*, 2006.

Sang, E.F.T.K. and Buchholz, S. Introduction to the CoNLL-2000 shared task: Chunking, 2000.

Shalev-Shwartz, S. and Zhang, T. Proximal stochastic dual coordinate ascent. Technical Report 1211.2717v1 [stat.ML], arXiv, November 2012.

Shalev-Shwartz, S., Singer, Y., Srebro, N., and Cotter, A. Pegasos: primal estimated sub-gradient solver for SVM. *Mathematical Programming*, 127(1), 2010a.

Shalev-Shwartz, S., Srebro, N., and Zhang, T. Trading accuracy for sparsity in optimization problems with sparsity constraints. *SIAM Journal on Optimization*, 20: 2807–2832, 2010b.

Shamir, O. and Zhang, T. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *ICML*, 2013.

Taskar, B. *Learning structured prediction models: A large margin approach*. PhD thesis, Stanford, 2004.

Taskar, B., Guestrin, C., and Koller, D. Max-margin Markov networks. In *NIPS*, 2003.

Taskar, B., Lacoste-Julien, S., and Jordan, M. I. Structured prediction, dual extragradient and Bregman projections. *JMLR*, 7:1627–1653, 2006.

Teo, C.H., Vishwanathan, S.V.N., Smola, A.J., and Le, Q.V. Bundle methods for regularized risk minimization. *JMLR*, 11:311–365, 2010.

Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484, 2005.

Zhang, X., Saha, A., and Vishwanathan, S. V. N. Accelerated training of max-margin Markov networks with kernels. In *ALT*, pp. 292–307. Springer, 2011.

Supplementary Material
Block-Coordinate Frank-Wolfe Optimization for Structural SVMs

Outline. In Appendix A, we discuss the curvature constants and compute them for the structural SVM problem. In Appendix B, we give additional details on applying the Frank-Wolfe algorithms to the structural SVM and provide proofs for Theorems 1 and 3. In the main Appendix C, we give a self-contained presentation and analysis of the new block-coordinate Frank-Wolfe method (Algorithm 3), and prove the main convergence Theorem 2. In Appendix D, the 'linearization'-duality gap is interpreted in terms of Fenchel duality. For completeness, we include a short derivation of the dual problem to the structural SVM in Appendix E. Finally, we present in Appendix F additional experimental results as well as more detailed information about the implementation.

A. The Curvature Constants C_f and $C_f^\otimes$

The Curvature Constant C_f. The *curvature constant* C_f is given by the maximum relative deviation of the objective function f from its linear approximations, over the domain $\mathcal{M}$ (Clarkson, 2010; Jaggi, 2013). Formally,

$$C_f := \sup_{\substack{\boldsymbol{x},\boldsymbol{s}\in\mathcal{M},\\ \gamma\in[0,1],\\ \boldsymbol{y}=\boldsymbol{x}+\gamma(\boldsymbol{s}-\boldsymbol{x})}} \frac{2}{\gamma^2}\left(f(\boldsymbol{y}) - f(\boldsymbol{x}) - \langle \boldsymbol{y}-\boldsymbol{x}, \nabla f(\boldsymbol{x})\rangle\right) \ . \tag{7}$$

The assumption of bounded C_f corresponds to a slightly weaker, affine invariant form of a *smoothness* assumption on f. It is known that C_f is upper bounded by the Lipschitz constant of the gradient ∇f times the squared diameter of $\mathcal{M}$, for any arbitrary choice of a norm (Jaggi, 2013, Lemma 8); but it can also be much smaller (in particular, when the dimension of the affine hull of $\mathcal{M}$ is smaller than the ambient space), so it is a more fundamental quantity in the analysis of the Frank-Wolfe algorithm than the Lipschitz constant of the gradient. As pointed out by Jaggi (2013, Section 2.4), C_f is invariant under affine transformations, as is the Frank-Wolfe algorithm.

The Product Curvature Constant $C_f^\otimes$. The curvature concept can be generalized to our setting of product domains $\mathcal{M} := \mathcal{M}^{(1)} \times \ldots \times \mathcal{M}^{(n)}$ as follows: over each individual coordinate block, the curvature is given by

$$C_f^{(i)} := \sup_{\substack{\boldsymbol{x}\in\mathcal{M},\, \boldsymbol{s}_{(i)}\in\mathcal{M}^{(i)},\\ \gamma\in[0,1],\\ \boldsymbol{y}=\boldsymbol{x}+\gamma(\boldsymbol{s}_{[i]}-\boldsymbol{x}_{[i]})}} \frac{2}{\gamma^2}\left(f(\boldsymbol{y}) - f(\boldsymbol{x}) - \langle \boldsymbol{y}_{(i)}-\boldsymbol{x}_{(i)}, \nabla_{(i)} f(\boldsymbol{x})\rangle\right) \ , \tag{8}$$

where the notation $\boldsymbol{x}_{[i]}$ refers to the zero-padding of $\boldsymbol{x}_{(i)}$ so that $\boldsymbol{x}_{[i]} \in \mathcal{M}$. By considering the Taylor expansion of f, it is not hard to see that also the 'partial' curvature $C_f^{(i)}$ is upper bounded by the Lipschitz constant of the partial gradient $\nabla_{(i)} f$ times the squared diameter of just one domain block $\mathcal{M}^{(i)}$. See also the proof of Lemma A.2 below.

We define the global *product curvature constant* as the sum of these curvatures for each block, i.e.

$$C_f^\otimes := \sum_{i=1}^n C_f^{(i)} \tag{9}$$

Observe that for the classical Frank-Wolfe case when $n = 1$, we recover the original curvature constant.

Computing the Curvature Constant C_f in the SVM Case.

Lemma A.1. *For the dual structural SVM objective function (4) over the domain $\mathcal{M} := \Delta_{|\mathcal{Y}_1|} \times \ldots \times \Delta_{|\mathcal{Y}_n|}$, the curvature constant C_f, as defined in (7), is upper bounded by*

$$C_f \leq \frac{4R^2}{\lambda} \ ,$$

where R is the maximal length of a difference feature vector, i.e. $R := \max_{i\in[n],\boldsymbol{y}\in\mathcal{Y}_i} \|\boldsymbol{\psi}_i(\boldsymbol{y})\|_2$.

Proof of Lemma A.1. If the objective function is twice differentiable, we can plug-in the second degree Taylor expansion of f into the above definition (7) of the curvature, see e.g. (Jaggi, 2011, Inequality (2.12)) or (Clarkson, 2010, Section 4.1). In our case, the gradient at $\boldsymbol{\alpha}$ is given by $\lambda A^T A\boldsymbol{\alpha} - \boldsymbol{b}$, so that the Hessian is $\lambda A^T A$, being a constant matrix independent of $\boldsymbol{\alpha}$. This gives the following upper bound[2] on C_f, which we can separate into two identical matrix-vector products with our matrix A:

$$\begin{aligned} C_f &\leq \sup_{\substack{\boldsymbol{x},\boldsymbol{y}\in\mathcal{M},\\ \boldsymbol{z}\in[\boldsymbol{x},\boldsymbol{y}]\subseteq\mathcal{M}}} (\boldsymbol{y}-\boldsymbol{x})^T \nabla^2 f(\boldsymbol{z})(\boldsymbol{y}-\boldsymbol{x}) \\ &= \lambda \cdot \sup_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{M}} (A(\boldsymbol{y}-\boldsymbol{x}))^T A(\boldsymbol{y}-\boldsymbol{x}) \\ &= \lambda \cdot \sup_{\boldsymbol{v},\boldsymbol{w}\in A\mathcal{M}} \|\boldsymbol{v}-\boldsymbol{w}\|_2^2 \ \leq \lambda \cdot \sup_{\boldsymbol{v}\in A\mathcal{M}} \|2\boldsymbol{v}\|_2^2 \end{aligned}$$

By definition of our compact domain $\mathcal{M}$, we have that each vector $\boldsymbol{v} \in A\mathcal{M}$ is precisely the sum of n vectors, each of these being a convex combination of the feature vectors for the possible labelings for datapoint i.

Therefore, the norm $\|\boldsymbol{v}\|_2$ is upper bounded by n times the longest column of the matrix A, or more formally $\|\boldsymbol{v}\|_2 \leq n\frac{1}{\lambda n}R$ with R being the longest[3] feature vector, i.e.

$$R := \max_{i\in[n],\boldsymbol{y}\in\mathcal{Y}_i} \|\boldsymbol{\psi}_i(\boldsymbol{y})\|_2 \ .$$

Altogether, we have obtained that the curvature C_f is upper bounded by $\frac{4R^2}{\lambda}$.

We also note that in the worst case, this bound is tight. For example, we can make $C_f = \frac{4R^2}{\lambda}$ by having for each datapoint i, two labelings which give opposite difference feature vectors ψ_i of the same maximal norm R. □

Computing the Product Curvature Constant $C_f^\otimes$ in the SVM Case.

Lemma A.2. *For the dual structural SVM objective function (4) over the domain $\mathcal{M} := \Delta_{|\mathcal{Y}_1|} \times \ldots \times \Delta_{|\mathcal{Y}_n|}$, the total curvature constant $C_f^\otimes$ on the product domain $\mathcal{M}$, as defined in (9), is upper bounded by*

$$C_f^\otimes \leq \frac{4R^2}{\lambda n}$$

where R is the maximal length of a difference feature vector, i.e. $R := \max_{i\in[n],\boldsymbol{y}\in\mathcal{Y}_i} \|\boldsymbol{\psi}_i(\boldsymbol{y})\|_2$.

Proof. We follow the same lines as in the above proof of Lemma A.1, but now applying the same bound to the block-wise definition (8) of the curvature on the i-th block. Here, the change from $\boldsymbol{x}$ to $\boldsymbol{y}$ is now restricted to only affect the coordinates in the i-th block $\mathcal{M}^{(i)}$. To simplify the notation, let $\mathcal{M}^{[i]}$ be $\mathcal{M}^{(i)}$ augmented with the zero domain for all the other blocks – i.e. the analog of $\boldsymbol{x}_{(i)} \in \mathcal{M}^{(i)}$ is $\boldsymbol{x}_{[i]} \in \mathcal{M}^{[i]}$. $\boldsymbol{x}_{(i)}$ is the i-th block of $\boldsymbol{x}$ whereas $\boldsymbol{x}_{[i]} \in \mathcal{M}$ is $\boldsymbol{x}_{(i)}$ padded with zeros for all the other blocks. We thus require that $\boldsymbol{y}-\boldsymbol{x} \in \mathcal{M}^{[i]}$ for a

[2] Because our function is a quadratic function, this is actually an equality.

[3] This choice of the radius R then gives $\frac{1}{\lambda n}R = \max_{i\in[n],\boldsymbol{y}\in\mathcal{Y}_i} \left\|\frac{1}{\lambda n}\boldsymbol{\psi}_i(\boldsymbol{y})\right\|_2 = \max_{i\in[n],\boldsymbol{y}\in\mathcal{Y}_i} \left\|A_{(i,\boldsymbol{y})}\right\|$.

valid change from $\boldsymbol{x}$ to $\boldsymbol{y}$. Again by the degree-two Taylor expansion, we obtain

$$\begin{aligned} C_f^{(i)} &\leq \sup_{\substack{\boldsymbol{x},\boldsymbol{y}\in\mathcal{M},\\ (\boldsymbol{y}-\boldsymbol{x})\in\mathcal{M}^{[i]}\\ \boldsymbol{z}\in[\boldsymbol{x},\boldsymbol{y}]\subseteq\mathcal{M}}} (\boldsymbol{y}-\boldsymbol{x})^T\nabla^2 f(\boldsymbol{z})(\boldsymbol{y}-\boldsymbol{x}) \\ &= \lambda\cdot \sup_{\substack{\boldsymbol{x},\boldsymbol{y}\in\mathcal{M}\\ (\boldsymbol{y}-\boldsymbol{x})\in\mathcal{M}^{[i]}}} (A(\boldsymbol{y}-\boldsymbol{x}))^T A(\boldsymbol{y}-\boldsymbol{x}) \\ &= \lambda\cdot \sup_{\boldsymbol{v},\boldsymbol{w}\in A\mathcal{M}^{(i)}} \|\boldsymbol{v}-\boldsymbol{w}\|_2^2 \;\leq \lambda\cdot \sup_{\boldsymbol{v}\in A\mathcal{M}^{(i)}} \|2\boldsymbol{v}\|_2^2 \end{aligned}$$

In other words, by definition of our compact domain $\mathcal{M}^{(i)} = \Delta_{|\mathcal{Y}_i|}$, we have that each vector $\boldsymbol{v} \in A\mathcal{M}^{(i)}$ is a convex combination of the feature vectors corresponding to the possible labelings for datapoint i. Therefore, the norm $\|\boldsymbol{v}\|_2$ is again upper bounded by the longest column of the matrix A, which means $\|\boldsymbol{v}\|_2 \leq \frac{1}{\lambda n} R$ with $R := \max_{i\in[n],\boldsymbol{y}\in\mathcal{Y}_i} \|\boldsymbol{\psi}_i(\boldsymbol{y})\|_2$. Summing up over the n blocks $\mathcal{M}^{(i)}$, we obtain that the product curvature $C_f^{\otimes}$ is upper bounded by $\frac{4R^2}{\lambda n}$.

For the same argument as at the end of the proof for Lemma A.1, this bound is actually tight in the worst case. □

B. More Details on the Algorithms for Structural SVMs

B.1. Equivalence of an Exact Frank-Wolfe Step and Loss-Augmented Decoding

To see that the proposed Algorithm 2 indeed exactly corresponds to the standard Frank-Wolfe Algorithm 1 applied to the SVM dual problem (4), we verify that the search direction $\boldsymbol{s}$ giving the update $\boldsymbol{w}_{\boldsymbol{s}} = A\boldsymbol{s}$ is in fact an exact Frank-Wolfe step, which can be seen as follows:

Lemma B.1. *The sparse vector $\boldsymbol{s} \in \mathbb{R}^n$ constructed in the inner for-loop of Algorithm 2 is an exact solution to $\boldsymbol{s} = \operatorname{argmin}_{\boldsymbol{s}'\in\mathcal{M}} \langle \boldsymbol{s}', \nabla f(\boldsymbol{\alpha}^{(k)})\rangle$ for optimization problem (4).*

Proof. Over the product domain $\mathcal{M} = \Delta_{|\mathcal{Y}_1|} \times \ldots \times \Delta_{|\mathcal{Y}_n|}$, the minimization $\min_{\boldsymbol{s}'\in\mathcal{M}} \langle \boldsymbol{s}', \nabla f(\boldsymbol{\alpha})\rangle$ decomposes as $\sum_i \min_{\boldsymbol{s}_i\in\Delta_{|\mathcal{Y}_i|}} \langle \boldsymbol{s}_i, \nabla_i f(\boldsymbol{\alpha})\rangle$. The minimization of a linear function over the simplex reduces to a search over its corners – in this case, it amounts for each i to find the minimal component of $-H_i(\boldsymbol{y};\boldsymbol{w})$ over $\boldsymbol{y} \in \mathcal{Y}_i$, i.e. solving the loss-augmented decoding problem as used in Algorithm 2 to construct the domain vertex $\boldsymbol{s}$. To see this, note that for our choice of primal variables $\boldsymbol{w} = A\boldsymbol{\alpha}$, the gradient of the dual objective, $\nabla f(\boldsymbol{\alpha}) = \lambda A^T A\boldsymbol{\alpha} - \boldsymbol{b}$, writes as $\lambda A^T\boldsymbol{w} - \boldsymbol{b}$. This vector is precisely the loss-augmented decoding function $-\frac{1}{n}H_i(\boldsymbol{y};\boldsymbol{w})$, for $i \in [n]$, $\boldsymbol{y} \in \mathcal{Y}_i$, as defined in (2). □

B.2. Relation between the Lagrange Duality Gap and the 'Linearization' Gap for the Structural SVM

We show here that the simple 'linearization' gap (5), evaluated on the structural SVM dual problem (4) is actually equivalent to the standard Lagrangian duality gap for the structural SVM primal objective (1) (these two duality gaps are not the same in general[4]). This is important for the duality gap convergence rate results of our Frank-Wolfe algorithms to be transferable as primal convergence rates on the original structural SVM objective (3), which is the one with statistical meaning (for example with generalization error bounds as given in Taskar et al. (2003)).

Proof. So consider the difference of our objective at $\boldsymbol{w} := A\boldsymbol{\alpha}$ in the primal problem (3), and the dual objective

[4]For example, the two gaps are different when evaluated on the dual of the conditional random field objective (see, for example, Collins et al. (2008) for the formulation), which does not have a Lipschitz continuous gradient.

at $\boldsymbol{\alpha}$ in problem (4) (in the maximization version). This difference is

$$\begin{aligned} g_{\text{Lagrange}}(\boldsymbol{w}, \boldsymbol{\alpha}) &= \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{1}{n}\sum_{i=1}^{n}\tilde{H}_i(\boldsymbol{w}) - \left(\boldsymbol{b}^T\boldsymbol{\alpha} - \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w}\right) \\ &= \lambda\boldsymbol{w}^T\boldsymbol{w} - \boldsymbol{b}^T\boldsymbol{\alpha} + \frac{1}{n}\sum_{i=1}^{n}\max_{\boldsymbol{y}\in\mathcal{Y}_i} H_i(\boldsymbol{y};\boldsymbol{w}) \ . \end{aligned}$$

Now recall that by the definition of A and $\boldsymbol{b}$, we have that $\frac{1}{n}H_i(\boldsymbol{y};\boldsymbol{w}) = (\boldsymbol{b} - \lambda A^T\boldsymbol{w})_{(i,\boldsymbol{y})} = (-\nabla f(\boldsymbol{\alpha}))_{(i,\boldsymbol{y})}$. By summing up over all points and re-using a similar argument as in Lemma B.1 above, we get that

$$\frac{1}{n}\sum_{i=1}^{n}\max_{\boldsymbol{y}\in\mathcal{Y}_i} H_i(\boldsymbol{y};\boldsymbol{w}) = \sum_{i=1}^{n}\max_{\boldsymbol{y}\in\mathcal{Y}_i}(-\nabla f(\boldsymbol{\alpha}))_{(i,\boldsymbol{y})} = \max_{\boldsymbol{s}'\in\mathcal{M}}\langle \boldsymbol{s}', -\nabla f(\boldsymbol{\alpha})\rangle \ ,$$

$$\begin{aligned} g_{\text{Lagrange}}(\boldsymbol{w}, \boldsymbol{\alpha}) &= (\lambda\boldsymbol{w}^T A - \boldsymbol{b}^T)\boldsymbol{\alpha} + \frac{1}{n}\sum_{i=1}^{n}\max_{\boldsymbol{y}\in\mathcal{Y}_i} H_i(\boldsymbol{y};\boldsymbol{w}) \\ &= \langle \nabla f(\boldsymbol{\alpha}), \boldsymbol{\alpha}\rangle + \max_{\boldsymbol{s}'\in\mathcal{M}}\langle -\boldsymbol{s}', \nabla f(\boldsymbol{\alpha})\rangle \ = \ \langle \boldsymbol{\alpha} - \boldsymbol{s}, \nabla f(\boldsymbol{\alpha})\rangle \ = \ g(\boldsymbol{\alpha}) \ , \end{aligned}$$

as defined in (5). □

B.3. Convergence Analysis

B.3.1. Convergence of the Batch Frank-Wolfe Algorithm 2 on the Structural SVM Dual

Theorem' 1. *Algorithm 2 obtains an ε-approximate solution to the structural SVM dual problem (4) and duality gap $g(\boldsymbol{\alpha}^{(k)}) \leq \varepsilon$ after at most $O\left(\frac{R^2}{\lambda\varepsilon}\right)$ iterations, where each iteration costs n oracle calls.*

Proof. We apply the known convergence results for the standard Frank-Wolfe Algorithm 1, as given e.g. in (Frank & Wolfe, 1956; Dunn & Harshbarger, 1978; Jaggi, 2013), or as given in the paragraph just after the proof of Theorem C.1: For each $k \geq 1$, the iterate $\boldsymbol{\alpha}^{(k)}$ of Algorithm 1 (either using the predefined step-sizes, or using line-search) satisfies $\mathrm{E}[f(\boldsymbol{\alpha}^{(k)})] - f(\boldsymbol{\alpha}^*) \leq \frac{2C_f}{k+2}$, where $\boldsymbol{\alpha}^* \in \mathcal{M}$ is an optimal solution to problem (4).

Furthermore, if Algorithm 1 is run for $K \geq 1$ iterations, then it has an iterate $\boldsymbol{\alpha}^{(\hat{k})}$, $1 \leq \hat{k} \leq K$, with duality gap bounded by $\mathrm{E}[g(\boldsymbol{\alpha}^{(\hat{k})})] \leq \frac{6C_f}{K+1}$. This was shown e.g. in (Jaggi, 2013) with slightly different constants, or also in our analysis presented below (see the paragraph after the generalized analysis provided in Theorem C.3, when the number of blocks n is set to one).

Now for the SVM problem and the equivalent Algorithm 2, the claim follows from the curvature bound $C_f \leq \frac{4R^2}{\lambda}$ for the dual structural SVM objective function (4) over the domain $\mathcal{M} := \Delta_{|\mathcal{Y}_1|} \times \ldots \times \Delta_{|\mathcal{Y}_n|}$, as given in the above Lemma A.1. □

B.3.2. Convergence of the Block-Coordinate Frank-Wolfe Algorithm 4 on the Structural SVM Dual

Theorem' 3. *If $L_{max} \leq \frac{4R^2}{\lambda n}$ (so $h_0 \leq \frac{4R^2}{\lambda n}$), then Algorithm 4 obtains an ε-approximate solution to the structural SVM dual problem (4) and expected duality gap $\mathrm{E}[g(\boldsymbol{\alpha}^{(k)})] \leq \varepsilon$ after at most $O\left(\frac{R^2}{\lambda\varepsilon}\right)$ iterations, where each iteration costs a single oracle call.*

If $L_{max} > \frac{4R^2}{\lambda n}$, then it requires at most an additional (constant in ε) number of $O\left(n\log\left(\frac{\lambda n L_{max}}{R^2}\right)\right)$ steps to get the same error and duality gap guarantees, whereas the predefined step-size variant will require an additional $O\left(\frac{nL_{max}}{\varepsilon}\right)$ steps.

Proof. Writing $h_0 = f(\boldsymbol{\alpha}^{(0)}) - f(\boldsymbol{\alpha}^*)$ for the error at the starting point used by the algorithm, the convergence Theorem 2 states that if $k \geq 0$ and $k \geq \frac{2n}{\varepsilon}(C_f^{\otimes} + h_0)$, then the expected error is $\mathrm{E}[f(\boldsymbol{\alpha}^{(k)})] - f(\boldsymbol{\alpha}^*) \leq \varepsilon$ and

analogously for the expected duality gap. The result then follows by plugging in the curvature bound $C_f^{\otimes} \leq \frac{4R^2}{\lambda n}$ for the dual structural SVM objective function (4) over the domain $\mathcal{M} := \Delta_{|\mathcal{Y}_1|} \times \ldots \times \Delta_{|\mathcal{Y}_n|}$, as detailed in Lemma A.2 (notice that it is n times smaller than the curvature C_f needed for the batch algorithm) and then bounding h_0. To bound h_0, we observe that by the choice of the starting point $\boldsymbol{\alpha}^{(0)}$ using only the observed labels, the initial error is bounded as $h_0 \leq g(\boldsymbol{\alpha}^{(0)}) = \boldsymbol{b}^T \boldsymbol{s} = \frac{1}{n} \sum_{i=1}^n \max_{\boldsymbol{y} \in \mathcal{Y}_i} L_i(\boldsymbol{y}) \leq L_{\max}$. Thus, if $L_{\max} \leq \frac{4R^2}{\lambda n}$, then we have $C_f^{\otimes} + h_0 \leq \frac{8R^2}{\lambda n}$, which proves the first part of the theorem.

In the case $L_{\max} > \frac{4R^2}{\lambda n}$, then the predefined step-size variant will require an additional $\frac{2nh_0}{\varepsilon} \leq \frac{2nL_{\max}}{\varepsilon}$ steps as we couldn't use the fact that $h_0 \leq C_f^{\otimes}$. For the line-search variant, on the other hand, we can use the improved convergence Theorem C.4, which shows that the algorithm require at most $k_0 \leq n \log(h_0 / C_f^{\otimes})$ steps to reach the condition $h_0 \leq C_f^{\otimes}$; once this condition is satisfied, we can simply re-use Theorem 2 with k redefined as $k - k_0$ to get the final convergence rates. We also point out that the statement of Theorem C.4 stays valid by replacing $C_f^{\otimes}$ with any $C_f^{\otimes\prime} \geq C_f^{\otimes}$ in it. So plugging in $C_f^{\otimes\prime} = \frac{R^2}{\lambda n}$ and the bound $h_0 \leq L_{\max}$ in the k_0 quantity gives back the number of additional steps mentioned in the second part of the theorem statement an ε-approximate solution. A similar argument can be made for the expected duality gap by using the improved convergence Theorem C.5, which simply adds the requirement $K \geq 5k_0$. □

We note that the condition $L_{\max} \leq \frac{4R^2}{\lambda n}$ is not necessarily too restrictive in the case of the structural SVM setup. In particular, the typical range of λ which is needed for a problem is around $O(1/n)$ – and so the condition becomes $L_{\max} \leq 4R^2$ which is typically satisfied when the loss function is normalized.

B.4. Implementation

We comment on three practical implementation aspects of Algorithm 4 on large structural SVM problems:

Memory. For each datapoint i, our Algorithm 4 stores an additional vector $\boldsymbol{w}_i \in \mathbb{R}^d$ holding the contribution of its corresponding dual variables $\boldsymbol{\alpha}_{(i)}$ to the primal vector $\boldsymbol{w} = A\boldsymbol{\alpha}$, i.e. $\boldsymbol{w}_i = A\boldsymbol{\alpha}_{[i]}$, where $\boldsymbol{\alpha}_{[i]}$ is $\boldsymbol{\alpha}_{(i)}$ padded with zeros so that $\boldsymbol{\alpha}_{[i]} \in \mathbb{R}^m$ and $\boldsymbol{\alpha} = \sum_i \boldsymbol{\alpha}_{[i]}$. This means the algorithm needs more memory than the direct (or batch) Frank-Wolfe structural SVM Algorithm 2, but the additional memory can sometimes be bounded by a constant times the size of the input data itself. In particular, in the case that the feature vectors $\boldsymbol{\psi}_i(\boldsymbol{y})$ are sparse, we can sometimes get the same improvement in memory requirements for $\boldsymbol{w}_i$, since for fixed i, all vectors $\boldsymbol{\psi}_i(\boldsymbol{y})$ usually have the same sparsity pattern. On the other hand, if the feature vectors are not sparse, it might be more efficient to only work with the dual variables instead of the primal variables (see the kernelized version in Appendix B.5 for more details).

Duality Gap as a Stopping Criterion. Analogous as in the 'classical Frank-Wolfe' structural SVM Algorithm 2 explained in Section 4, we would again like to use the duality gap $g(\boldsymbol{\alpha}^{(k)}) \leq \varepsilon$ as the stopping criterion for the faster Algorithm 4. Unfortunately, since now in every step we only update a single one of the many blocks, such a single direction $\boldsymbol{s}_{(i)}$ will only determine the partial gap $g^{(i)}(\boldsymbol{\alpha}^{(k)})$ in the i-th block, but not the full information needed to know the total gap $g(\boldsymbol{\alpha}^{(k)})$. Instead, to compute the total gap, a single complete (batch) pass through all datapoints as in Algorithm 2 is necessary, to obtain a full linear minimizer $\boldsymbol{s} \in \mathcal{M}$. For efficiency reason, we could therefore compute the duality gap every say Nn iterations for some constant $N > 1$. Then stopping as soon as $g(\boldsymbol{\alpha}^{(k)}) = g(\boldsymbol{w}^{(k)}, \ell^{(k)}, \boldsymbol{w}_{\boldsymbol{s}}, \ell_{\boldsymbol{s}}) \leq \varepsilon$ will not affect our convergence results.

Line-Search. To compute the line-search step-size for Frank-Wolfe on the structural SVM, we recall that the analytic formula was given by $\gamma_{opt} := \frac{\langle \boldsymbol{\alpha} - \boldsymbol{s}, \nabla f(\boldsymbol{\alpha}) \rangle}{\lambda \|A(\boldsymbol{\alpha} - \boldsymbol{s})\|^2}$, and finally taking $\gamma_{LS} := \max\{0, \min\{1, \gamma_{opt}\}\}$. This is valid for any $\boldsymbol{s} \in \mathcal{M}$. For the block-coordinate Frank-Wolfe Algorithm 4, $\boldsymbol{s}$ is equal to $\boldsymbol{\alpha}$ for all blocks, except for the i-th block – this means that $\boldsymbol{\alpha} - \boldsymbol{s} = \boldsymbol{\alpha}_{[i]} - \boldsymbol{s}_{[i]}$, i.e. is zero everywhere except on the i-th block. By recalling that $\boldsymbol{w}_i = A\boldsymbol{\alpha}_{[i]}$ is the individual contribution to $\boldsymbol{w}$ from $\boldsymbol{\alpha}_{(i)}$ which is stored during the algorithm, we see that the denominator thus becomes $\lambda \|A(\boldsymbol{\alpha} - \boldsymbol{s})\|^2 = \lambda \|\boldsymbol{w}_i - w_{\boldsymbol{s}}\|^2$. The numerator is $\langle \boldsymbol{\alpha} - \boldsymbol{s}, \nabla f(\boldsymbol{\alpha}) \rangle = (\boldsymbol{\alpha} - \boldsymbol{s})^T (\lambda A^T A\boldsymbol{\alpha} - \boldsymbol{b}) = \lambda(\boldsymbol{w}_i - w_{\boldsymbol{s}})^T \boldsymbol{w} - \ell_i + \ell_{\boldsymbol{s}}$, where as before $\ell_i = \boldsymbol{b}^T \boldsymbol{\alpha}_{[i]}$ is maintained during Algorithm 4 and so the line-search step-size can be computed efficiently. We mention in passing that when $\boldsymbol{s}_{(i)}$ is the exact minimizer of the linear subproblem on $\mathcal{M}^{(i)}$, then the numerator is actually a duality

gap component $g^{(i)}(\boldsymbol{\alpha})$ as defined in (16) – the total duality gap then is $g(\boldsymbol{\alpha}) = \sum_i g^{(i)}(\boldsymbol{\alpha})$ which can only be computed if we do a batch pass over all the datapoints, as explained in the previous paragraph.

B.5. More details on the Kernelized Algorithm

Both Algorithms 2 and 4 can be used with kernels by explicitly maintaining the sparse dual variables $\boldsymbol{\alpha}^{(k)}$ instead of the primal variables $\boldsymbol{w}^{(k)}$. In this case, the classifier is only given implicitly as a sparse combination of the corresponding kernel functions, i.e. $\boldsymbol{w} = A\boldsymbol{\alpha}$, where $\boldsymbol{\psi}_i(\boldsymbol{y}) = k(\boldsymbol{x}_i, \boldsymbol{y}_i; \cdot, \cdot) - k(\boldsymbol{x}_i, \boldsymbol{y}; \cdot, \cdot)$ for a structured kernel $k : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}$. Note that the number of non-zero dual variables is upper-bounded by the number of iterations, and so the time to take dot products grows quadratically in the number of iterations.

Algorithm B.1 Kernelized Dual Block-Coordinate Frank-Wolfe for Structural SVM

Let $\boldsymbol{\alpha}^{(0)} := (\mathbf{e}^{\boldsymbol{y}_1}, \dots, \mathbf{e}^{\boldsymbol{y}_n}) \in \mathcal{M} = \Delta_{|\mathcal{Y}_1|} \times \dots \times \Delta_{|\mathcal{Y}_n|}$ and $\bar{\boldsymbol{\alpha}}^{(0)} = \boldsymbol{\alpha}^{(0)}$
for $k = 0 \dots K$ **do**
  Pick i uniformly at random in $\{1, \dots, n\}$
  Solve $\boldsymbol{y}_i^* := \operatorname{argmax}_{\boldsymbol{y} \in \mathcal{Y}_i} H_i(\boldsymbol{y}; A\boldsymbol{\alpha}^{(k)})$ *(solve the loss-augmented decoding problem (2))*
  $\boldsymbol{s}_{(i)} := \mathbf{e}^{\boldsymbol{y}_i^*} \in \mathcal{M}^{(i)}$ *(having only a single non-zero entry)*
  Let $\gamma := \frac{2n}{k+2n}$, or optimize γ by line-search
  Update $\boldsymbol{\alpha}_{(i)}^{(k+1)} := (1-\gamma)\boldsymbol{\alpha}_{(i)}^{(k)} + \gamma \boldsymbol{s}_{(i)}$
  (Optionally: Update $\bar{\boldsymbol{\alpha}}^{(k+1)} := \frac{k}{k+2}\bar{\boldsymbol{\alpha}}^{(k)} + \frac{2}{k+2}\boldsymbol{\alpha}^{(k+1)}$) *(maintain a weighted average of the iterates)*

To compute the line-search step-size, we simply re-use the same formula as in Algorithm 4, but reconstructing (implicitly) on the fly the missing quantities such as $\ell_i = \boldsymbol{b}^T \boldsymbol{\alpha}_{[i]}$, $\boldsymbol{w}_i = A\boldsymbol{\alpha}_{[i]}$ and $\boldsymbol{w}^{(k)} = A\boldsymbol{\alpha}^{(k)}$, and re-interpreting dot products such as $\boldsymbol{w}_i^T \boldsymbol{w}^{(k)}$ as the suitable sum of kernel evaluations (which has $O(k^2/n)$ terms, where k is the number of iterations since the beginning).

C. Analysis of the Block-Coordinate Frank-Wolfe Algorithm 3

This section gives a self-contained presentation and analysis of the new block-coordinate Frank-Wolfe optimization Algorithm 3. The main goal is to prove the convergence Theorem 2, which here is split into two parts, the *primal* convergence rate in Theorem C.1, and the *primal-dual* convergence rate in Theorem C.3. Finally, we will present a faster convergence result for the line-search variant in Theorem C.4 and Theorem C.5, which we have used in the convergence for the structural SVM case as presented above in Theorem 3.

Coordinate Descent Methods. Despite their simplicity and very early appearance in the literature, surprisingly few results were known on the convergence (and convergence rates in particular) of coordinate descent type methods. Recently, the interest in these methods has grown again due to their good scalability to very large scale problems as e.g. in machine learning, and also sparked new theoretical results such as (Nesterov, 2012).

Constrained Convex Optimization over Product Domains. We consider the general constrained convex optimization problem

$$\min_{\boldsymbol{x} \in \mathcal{M}} f(\boldsymbol{x}) \tag{10}$$

over a Cartesian product domain $\mathcal{M} = \mathcal{M}^{(1)} \times \dots \times \mathcal{M}^{(n)} \subseteq \mathbb{R}^m$, where each factor $\mathcal{M}^{(i)} \subseteq \mathbb{R}^{m_i}$ is convex and *compact*, and $\sum_{i=1}^n m_i = m$. We will write $\boldsymbol{x}_{(i)} \in \mathbb{R}^{m_i}$ for the i-th block of coordinates of a vector $\boldsymbol{x} \in \mathbb{R}^m$, and $\boldsymbol{x}_{[i]}$ for the padding of $\boldsymbol{x}_{(i)}$ with zeros so that $\boldsymbol{x}_{[i]} \in \mathbb{R}^m$.

Nesterov's 'Huge Scale' Coordinate Descent. If the objective function f is strongly smooth (i.e. has Lipschitz continuous partial gradients $\nabla_{(i)} f(\boldsymbol{x}) \in \mathbb{R}^{m_i}$), then the following algorithm converges[5] at a rate of $\frac{1}{k}$,

[5] By additionally assuming strong convexity of f w.r.t. the ℓ_1-norm (global on $\mathcal{M}$, not only on the individual factors), one can even get linear convergence rates, see again (Nesterov, 2012) and the follow-up paper (Richtárik & Takáč, 2011).

or more precisely $\frac{n}{k+n}$, as shown in (Nesterov, 2012, Section 4):

Algorithm C.1 Uniform Coordinate Descent Method, (Nesterov, 2012, Section 4)

Let $\boldsymbol{x}^{(0)} \in \mathcal{M}$
for $k = 0 \dots \infty$ **do**
  Pick i uniformly at random in $\{1, \dots, n\}$
  Compute $\boldsymbol{s}_{(i)} := \underset{\boldsymbol{s}_{(i)} \in \mathcal{M}^{(i)}}{\operatorname{argmin}} \left\langle \boldsymbol{s}_{(i)}, \nabla_{(i)} f(\boldsymbol{x}^{(k)}) \right\rangle + \frac{L_i}{2} \left\| \boldsymbol{s}_{(i)} - \boldsymbol{x}_{(i)} \right\|^2$
  Update $\boldsymbol{x}_{(i)}^{(k+1)} := \boldsymbol{x}_{(i)}^{(k)} + \left(\boldsymbol{s}_{(i)} - \boldsymbol{x}_{(i)}^{(k)} \right)$ *(only affecting the i-th coordinate block)*

Using Simpler Update Steps: Frank-Wolfe / Conditional Gradient Methods. In some large-scale applications, the above computation of the update direction $\boldsymbol{s}_{(i)}$ can be problematic, e.g. if the Lipschitz constants L_i are unknown, or —more importantly— if the domains $\mathcal{M}^{(i)}$ are such that the quadratic term makes the subproblem for $\boldsymbol{s}_{(i)}$ hard to solve.

The structural SVM is a nice example where this makes a big difference. Here, each domain block $\mathcal{M}^{(i)}$ is a simplex of exponentially many variables, but nevertheless the linear subproblem over one such factor (also known as loss-augmented decoding) is often relatively easy to solve.

We would therefore like to replace the above computation of $\boldsymbol{s}_{(i)}$ by a simpler one, as proposed in the following algorithm variant:

Algorithm C.2 Cheaper Coordinate Descent: Block-Coordinate Frank-Wolfe Algorithm

Let $\boldsymbol{x}^{(0)} \in \mathcal{M}$ and $\bar{\boldsymbol{x}}_w^{(0)} = \boldsymbol{x}^{(0)}$
for $k = 0 \dots \infty$ **do**
  Pick i uniformly at random in $\{1, \dots, n\}$
  Compute $\boldsymbol{s}_{(i)} := \underset{\boldsymbol{s}_{(i)} \in \mathcal{M}^{(i)}}{\operatorname{argmin}} \left\langle \boldsymbol{s}_{(i)}, \nabla_{(i)} f(\boldsymbol{x}^{(k)}) \right\rangle$
    (or alternatively, find $\boldsymbol{s}_{(i)}$ that solves this linear problem approximately, either up to an additive error (11) or up to a multiplicative error (12))
  Let $\gamma := \frac{2n}{k+2n}$, or perform line-search for the step-size: $\gamma := \underset{\gamma \in [0,1]}{\operatorname{argmin}} f\left(\boldsymbol{x}^{(k)} + \gamma \left(\boldsymbol{s}_{[i]} - \boldsymbol{x}_{[i]}^{(k)} \right) \right)$
  Update $\boldsymbol{x}_{(i)}^{(k+1)} := \boldsymbol{x}_{(i)}^{(k)} + \gamma \left(\boldsymbol{s}_{(i)} - \boldsymbol{x}_{(i)}^{(k)} \right)$ *(only affecting the i-th coordinate block)*
  (Optionally: Update $\bar{\boldsymbol{x}}_w^{(k+1)} := \frac{k}{k+2} \bar{\boldsymbol{x}}_w^{(k)} + \frac{2}{k+2} \boldsymbol{x}^{(k+1)}$) *(maintain a weighted average of the iterates)*

This natural coordinate descent type optimization method picks a single one of the n blocks uniformly at random, and in each step leaves all other blocks unchanged.

If there is only one factor ($n = 1$), then Algorithm C.2 becomes the standard Frank-Wolfe (or conditional gradient) algorithm, which is known to converge at a rate of $O(1/k)$ (Frank & Wolfe, 1956; Dunn & Harshbarger, 1978; Clarkson, 2010; Jaggi, 2013).

Using Approximate Linear Minimizers. If approximate linear minimizers are used internally in Algorithm C.2, then the necessary approximation quality for the candidate directions $\boldsymbol{s}_{(i)}$ is determined as follows (in either additive or multiplicative quality):

In the *additive* case, we choose a fixed additive error parameter $\delta \geq 0$ such that the candidate direction $\boldsymbol{s}_{(i)}$ satisfies

$$\left\langle \boldsymbol{s}_{(i)}, \nabla_{(i)} f(\boldsymbol{x}) \right\rangle \;\leq\; \min_{\boldsymbol{s}'_{(i)} \in \mathcal{M}^{(i)}} \left\langle \boldsymbol{s}'_{(i)}, \nabla_{(i)} f(\boldsymbol{x}) \right\rangle + \tfrac{1}{2} \delta \, \tilde{\gamma}_k \, C_f^{(i)} \;, \tag{11}$$

where $\tilde{\gamma}_k := \frac{2n}{k+2n}$ comes from the default step-size and is used for the convergence results to come. Note that if line-search is used to determine a different step-size, the candidate direction is still defined with respect to the default $\tilde{\gamma}_k$.

In the *multiplicative* case, we choose a fixed multiplicative error parameter $0 < \nu \leq 1$ such that the candidate directions $\boldsymbol{s}_{(i)}$ attain the current 'duality gap' on the i-th factor up to a *multiplicative* approximation error of ν, i.e.

$$\left\langle \boldsymbol{x} - \boldsymbol{s}_{(i)}, \nabla_{(i)} f(\boldsymbol{x}) \right\rangle \;\geq\; \nu \cdot \max_{\boldsymbol{s}'_{(i)} \in \mathcal{M}^{(i)}} \left\langle \boldsymbol{x} - \boldsymbol{s}'_{(i)}, \nabla_{(i)} f(\boldsymbol{x}) \right\rangle \;. \tag{12}$$

If a multiplicative approximate internal oracle is used together with the predefined step-size instead of doing line-search, then the step-size in Algorithm C.2 needs to be increased to $\gamma_k := \frac{2n}{\nu k + 2n}$ instead of the original $\frac{2n}{k+2n}$.

Both types of errors can be combined together with the following property for the candidate direction $\boldsymbol{s}_{(i)}$:

$$\left\langle \boldsymbol{x} - \boldsymbol{s}_{(i)}, \nabla_{(i)} f(\boldsymbol{x}) \right\rangle \;\geq\; \nu \cdot \max_{\boldsymbol{s}'_{(i)} \in \mathcal{M}^{(i)}} \left\langle \boldsymbol{x} - \boldsymbol{s}'_{(i)}, \nabla_{(i)} f(\boldsymbol{x}) \right\rangle - \tfrac{1}{2} \delta \, \tilde{\gamma}_k \, C_f^{(i)} \;, \tag{13}$$

where $\tilde{\gamma}_k := \frac{2n}{\nu k + 2n}$.

Averaging the Iterates. In the above Algorithm C.2 we have also added an optional last line which maintains the following weighted average $\bar{\boldsymbol{x}}_w^{(k)}$ which is defined for $k \geq 1$ as

$$\bar{\boldsymbol{x}}_w^{(k)} := \frac{2}{k(k+1)} \sum_{t=1}^{k} t \, \boldsymbol{x}^{(t)} \;, \tag{14}$$

and by convention we also define $\bar{\boldsymbol{x}}_w^{(0)} := \boldsymbol{x}^{(0)}$. As our convergence analysis will show, the weighted average of the iterates can yield more robust duality gap convergence guarantees when the duality gap function g is convex in $\boldsymbol{x}$ (see Theorem C.3) – this is for example the case for quadratic functions such as in the structural SVM objective (4). We will also consider in our proofs a scheme which averages the last $(1-\mu)$-fraction of the iterates for some fixed $0 < \mu < 1$:

$$\bar{\boldsymbol{x}}_\mu^{(k)} := \frac{1}{k - \lceil \mu k \rceil + 1} \sum_{t=\lceil \mu k \rceil}^{k} \boldsymbol{x}^{(t)} \;. \tag{15}$$

This is what Rakhlin et al. (2012) calls $(1-\mu)$-*suffix averaging* and it appeared in the context of getting a stochastic subgradient method with $O(1/k)$ convergence rate for strongly convex functions instead of the standard $O((\log k)/k)$ rate that one can prove for the individual iterates $\boldsymbol{x}^{(k)}$. The problem with $(1-\mu)$-suffix averaging is that to implement it for a fixed μ (say $\mu = 0.5$) without storing a fraction of all the iterates, one needs to know when they will stop the algorithm. An alternative mentioned in Rakhlin et al. (2012) is to maintain a uniform average over rounds of exponentially increasing size (the so-called 'doubling trick'). This can give very good performance towards the end of the rounds as we will see in our additional experiments in Appendix F, but the performance varies widely towards the beginning of the rounds. This motivates the simpler and more robust weighted averaging scheme (14), which in the case of the stochastic subgradient method, was also recently proven to have $O(1/k)$ convergence rate by Lacoste-Julien et al. (2012)[6] and independently by Shamir & Zhang (2013), who called such schemes 'polynomial-decay averaging'.

Related Work. In contrast to the randomized choice of coordinate which we use here, the analysis of *cyclic* coordinate descent algorithms (going through the blocks sequentially) seems to be notoriously difficult, such that until today, no analysis proving a global convergence rate has been obtained as far as we know. Luo & Tseng (1992) has proven a *local* linear convergence rate for the strongly convex case.

For product domains, such a cyclic analogue of our Algorithm C.2 has already been proposed in Patriksson (1998), using a generalization of Frank-Wolfe iterations under the name 'cost approximation'. The analysis of Patriksson (1998) shows asymptotic convergence, but since the method goes through the blocks sequentially, no convergence rates could be proven so far.

[6] In this paper, they considered a $(k+1)$-weight instead of our k-weight, but similar rates can be proven for shifted versions. We motivate skipping the first iterate $\boldsymbol{x}^{(0)}$ in our weighted averaging scheme as sometimes bounds can be proven on the quality of $\boldsymbol{x}^{(1)}$ irrespective of $\boldsymbol{x}^{(0)}$ for Frank-Wolfe (see the paragraph after the proof of Theorem C.1 for example, looking at the $n = 1$ case).

C.1. Setup for Convergence Analysis

We review below the important concepts needed for analyzing the convergence of the block-coordinate Frank-Wolfe Algorithm C.2.

Decomposition of the Duality Gap. The product structure of our domain has a crucial effect on the duality gap, namely that it decomposes into a sum over the n components of the domain. The 'linearization' duality gap as defined in (5) (see also Jaggi (2013)) for any constrained convex problem of the above form (10), for a fixed feasible point $\boldsymbol{x} \in \mathcal{M}$, is given by

$$\begin{aligned} g(\boldsymbol{x}) :=& \max_{\boldsymbol{s} \in \mathcal{M}} \langle \boldsymbol{x} - \boldsymbol{s}, \nabla f(\boldsymbol{x}) \rangle \\ =& \sum_{i=1}^{n} \max_{\boldsymbol{s}_{(i)} \in \mathcal{M}^{(i)}} \left\langle \boldsymbol{x}_{(i)} - \boldsymbol{s}_{(i)}, \nabla_{(i)} f(\boldsymbol{x}) \right\rangle \\ =:& \sum_{i=1}^{n} g^{(i)}(\boldsymbol{x}) \ . \end{aligned} \tag{16}$$

Curvature. Also, the curvature can now be defined on the individual factors,

$$C_f^{(i)} := \sup_{\substack{\boldsymbol{x} \in \mathcal{M},\, \boldsymbol{s}_{(i)} \in \mathcal{M}^{(i)}, \\ \gamma \in [0,1], \\ \boldsymbol{y} = \boldsymbol{x} + \gamma(\boldsymbol{s}_{[i]} - \boldsymbol{x}_{[i]})}} \tfrac{2}{\gamma^2} \left(f(\boldsymbol{y}) - f(\boldsymbol{x}) - \langle \boldsymbol{y}_{(i)} - \boldsymbol{x}_{(i)}, \nabla_{(i)} f(\boldsymbol{x}) \rangle \right) \ . \tag{17}$$

We recall that the notation $\boldsymbol{x}_{[i]}$ and $\boldsymbol{x}_{(i)}$ is defined just below (10). We define the global product curvature as the sum of these curvatures for each block, i.e.

$$C_f^{\otimes} := \sum_{i=1}^{n} C_f^{(i)}. \tag{18}$$

C.2. Primal Convergence on Product Domains

The following main theorem shows that after $O\big(\frac{1}{\varepsilon}\big)$ many iterations, Algorithm C.2 obtains an ε-approximate solution.

Theorem C.1 (Primal Convergence)**.** *For each $k \geq 0$, the iterate $\boldsymbol{x}^{(k)}$ of the exact variant of Algorithm C.2 satisfies*

$$\mathrm{E}[f(\boldsymbol{x}^{(k)})] - f(\boldsymbol{x}^*) \leq \frac{2n}{k + 2n} \big(C_f^{\otimes} + f(\boldsymbol{x}^{(0)}) - f(\boldsymbol{x}^*) \big) \ ,$$

For the approximate variant of Algorithm C.2 with additive *approximation quality (11) for $\delta \geq 0$, it holds that*

$$\mathrm{E}[f(\boldsymbol{x}^{(k)})] - f(\boldsymbol{x}^*) \leq \frac{2n}{k + 2n} \big(C_f^{\otimes}(1 + \delta) + f(\boldsymbol{x}^{(0)}) - f(\boldsymbol{x}^*) \big) \ .$$

For the approximate variant of Algorithm C.2, with multiplicative *approximation quality (12) for $0 < \nu \leq 1$, it holds that*

$$\mathrm{E}[f(\boldsymbol{x}^{(k)})] - f(\boldsymbol{x}^*) \leq \frac{2n}{\nu k + 2n} \big(\frac{1}{\nu} C_f^{\otimes} + f(\boldsymbol{x}^{(0)}) - f(\boldsymbol{x}^*) \big) \ .$$

All convergence bounds hold both if the predefined step-sizes, or line-search is used in the algorithm. Here $\boldsymbol{x}^* \in \mathcal{M}$ is an optimal solution to problem (10), and the expectation is with respect to the random choice of blocks during the algorithm. (In other words all three algorithm variants deliver a solution of (expected) primal error at most ε after $O(\frac{1}{\varepsilon})$ many iterations.)

The proof of the above theorem on the convergence rate of the primal error crucially depends on the following Lemma C.2 on the improvement in each iteration.

Lemma C.2. *Let $\gamma \in [0,1]$ be an arbitrary fixed step-size. Moving only within the i-th block of the domain, we consider two variants of steps towards a direction $\boldsymbol{s}_{(i)} \in \mathcal{M}^{(i)}$: Let $\boldsymbol{x}_{\gamma}^{(k+1)} := \boldsymbol{x}(\gamma)$ be the point obtained by moving towards $\boldsymbol{s}_{(i)}$ using step-size γ, and let $\boldsymbol{x}_{LS}^{(k+1)} := \boldsymbol{x}(\gamma_{LS})$ be the corresponding point obtained by line-search, i.e. $\gamma_{LS} := \operatorname{argmin}_{\bar{\gamma}\in[0,1]} f\left(\boldsymbol{x}(\bar{\gamma})\right)$. Here for convenience we have used the notation $\boldsymbol{x}(\bar{\gamma}) := \boldsymbol{x}^{(k)} + \bar{\gamma}\big(\boldsymbol{s}_{[i]} - \boldsymbol{x}_{[i]}^{(k)}\big)$ for $\bar{\gamma} \in [0,1]$.*

If for each i the candidate direction $\boldsymbol{s}_{(i)}$ satisfies the additive *approximation quality (11) for $\delta \geq 0$ and some fixed $\tilde{\gamma}_k$, then in expectation over the random choice of the block i and conditioned on $\boldsymbol{x}^{(k)}$, it holds that*

$$\mathrm{E}\left[f(\boldsymbol{x}_{LS}^{(k+1)}) \,|\, \boldsymbol{x}^{(k)}\right] \leq \mathrm{E}\left[f(\boldsymbol{x}_{\gamma}^{(k+1)}) \,|\, \boldsymbol{x}^{(k)}\right] \;\leq\; f(\boldsymbol{x}^{(k)}) - \frac{\gamma}{n} g(\boldsymbol{x}^{(k)}) + \frac{1}{2n}(\gamma^2 + \delta\tilde{\gamma}_k\gamma) C_f^{\otimes} \;.$$

On the other hand, if $\boldsymbol{s}_{(i)}$ attains the duality gap $g^{(i)}(\boldsymbol{x})$ on the i-th block up to a multiplicative *approximation quality (12) for $0 < \nu \leq 1$, then*

$$\mathrm{E}\left[f(\boldsymbol{x}_{LS}^{(k+1)}) \,|\, \boldsymbol{x}^{(k)}\right] \leq \mathrm{E}\left[f(\boldsymbol{x}_{\gamma}^{(k+1)}) \,|\, \boldsymbol{x}^{(k)}\right] \;\leq\; f(\boldsymbol{x}^{(k)}) - \frac{\gamma}{n} \nu\, g(\boldsymbol{x}^{(k)}) + \frac{\gamma^2}{2n} C_f^{\otimes} \;.$$

All expectations are taken over the random choice of the block i and conditioned on $\boldsymbol{x}^{(k)}$.

Proof. We write $\boldsymbol{x} := \boldsymbol{x}^{(k)}$, $\boldsymbol{y} := \boldsymbol{x}_{\gamma}^{(k+1)} = \boldsymbol{x} + \gamma(\boldsymbol{s}_{[i]} - \boldsymbol{x}_{[i]})$, with $\boldsymbol{x}_{[i]}$ and $\boldsymbol{s}_{[i]}$ being zero everywhere except in their i-th block. We also write $d_{\boldsymbol{x}} := \nabla_{(i)} f(\boldsymbol{x})$ to simplify the notation. From the definition (17) of the curvature constant $C_f^{(i)}$ of our convex function f over the factor $\mathcal{M}^{(i)}$, we have

$$\begin{aligned} f(\boldsymbol{y}) = \;& f(\boldsymbol{x} + \gamma(\boldsymbol{s}_{[i]} - \boldsymbol{x}_{[i]})) \\ \leq \;& f(\boldsymbol{x}) + \gamma\langle \boldsymbol{s}_{(i)} - \boldsymbol{x}_{(i)}, d_{\boldsymbol{x}}\rangle + \tfrac{\gamma^2}{2} C_f^{(i)} \;. \end{aligned}$$

Now we use that by (11), the choice of $\boldsymbol{s}_{(i)}$ with $\left\langle \boldsymbol{s}_{(i)}, \nabla_{(i)} f(\boldsymbol{x})\right\rangle \leq \min_{\boldsymbol{s}'_{(i)} \in \mathcal{M}^{(i)}} \left\langle \boldsymbol{s}'_{(i)}, \nabla_{(i)} f(\boldsymbol{x})\right\rangle + \frac{1}{2}\delta\tilde{\gamma}_k C_f^{(i)}$ is a good descent direction for the linear approximation to f at $\boldsymbol{x}$, on the i-th factor $\mathcal{M}^{(i)}$, giving

$$\langle \boldsymbol{s}_{(i)} - \boldsymbol{x}_{(i)}, d_{\boldsymbol{x}}\rangle \leq -g^{(i)}(\boldsymbol{x}) + \tfrac{\delta\tilde{\gamma}_k}{2} C_f^{(i)} \;, \tag{19}$$

by the definition (16) of the duality gap. Altogether, we have obtained

$$\begin{aligned} f(\boldsymbol{y}) \leq \;& f(\boldsymbol{x}) + \gamma(-g^{(i)}(\boldsymbol{x}) + \tfrac{\delta\tilde{\gamma}_k}{2} C_f^{(i)}) + \tfrac{\gamma^2}{2} C_f^{(i)} \\ = \;& f(\boldsymbol{x}) - \gamma g^{(i)}(\boldsymbol{x}) + \tfrac{1}{2}(\gamma^2 + \delta\tilde{\gamma}_k\gamma) C_f^{(i)} \;. \end{aligned}$$

Using that the line-search by definition must lead to an objective value at least as good as the one at the fixed γ, we therefore have shown the inequality

$$f(\boldsymbol{x}_{LS}^{(k+1)}) \leq f(\boldsymbol{x}_{\gamma}^{(k+1)}) \;\leq\; f(\boldsymbol{x}^{(k)}) - \gamma g^{(i)}(\boldsymbol{x}^{(k)}) + \tfrac{1}{2}(\gamma^2 + \delta\tilde{\gamma}_k\gamma) C_f^{(i)} \;.$$

Finally the claimed bound on the expected improvement directly follows by taking the expectation: With respect to the (uniformly) random choice of the block i, the expected value of the gap $g^{(i)}(\boldsymbol{x}^{(k)})$ corresponding to the picked i is exactly $\frac{1}{n} g(\boldsymbol{x}^{(k)})$. Also, the expected curvature of the i-th factor is $\frac{1}{n} C_f^{\otimes}$.

The proof for the case of multiplicative approximation follows completely analogously, using $\langle \boldsymbol{s}_{(i)} - \boldsymbol{x}_{(i)}, d_{\boldsymbol{x}}\rangle \leq -\nu\, g^{(i)}(\boldsymbol{x})$, which then gives a step improvement of $f(\boldsymbol{y}) \leq f(\boldsymbol{x}) - \gamma\nu g^{(i)}(\boldsymbol{x}) + \frac{\gamma^2}{2} C_f^{(i)}$. □

Having Lemma C.2 at hand, we will now prove our above primal convergence Theorem C.1 using similar ideas as for general domains, such as in Jaggi (2013).

Proof of Theorem C.1. We first prove the theorem for the approximate variant of Algorithm C.2 with multiplicative approximation quality (12) of $0 < \nu \leq 1$ – the exact variant of the algorithm is simply the special case

$\nu = 1$. From the above Lemma C.2, we know that for every inner step of Algorithm C.2 and conditioned on $\boldsymbol{x}^{(k)}$, we have that $\mathrm{E}[f(\boldsymbol{x}_\gamma^{(k+1)}) \,|\, \boldsymbol{x}^{(k)}] \leq f(\boldsymbol{x}^{(k)}) - \frac{\gamma\nu}{n} g(\boldsymbol{x}^{(k)}) + \frac{\gamma^2}{2n} C_f^\otimes$, where the expectation is over the random choice of the block i (this bound holds independently whether line-search is used or not). Writing $h(\boldsymbol{x}) := f(\boldsymbol{x}) - f(\boldsymbol{x}^*)$ for the (unknown) primal error at any point $\boldsymbol{x}$, this reads as

$$\begin{aligned} \mathrm{E}[h(\boldsymbol{x}_\gamma^{(k+1)}) \,|\, \boldsymbol{x}^{(k)}] \leq & \quad h(\boldsymbol{x}^{(k)}) - \tfrac{\gamma\nu}{n} g(\boldsymbol{x}^{(k)}) + \tfrac{\gamma^2}{2n} C_f^\otimes \\ \leq & \quad h(\boldsymbol{x}^{(k)}) - \tfrac{\gamma\nu}{n} h(\boldsymbol{x}^{(k)}) + \tfrac{\gamma^2}{2n} C_f^\otimes \\ = & \quad (1 - \tfrac{\gamma\nu}{n}) h(\boldsymbol{x}^{(k)}) + \tfrac{\gamma^2}{2n} C_f^\otimes, \end{aligned} \tag{20}$$

where in the second line, we have used *weak duality* $h(\boldsymbol{x}) \leq g(\boldsymbol{x})$ (which follows directly from the definition of the duality gap, together with convexity of f). The inequality (20) is conditioned on $\boldsymbol{x}^{(k)}$, which is a random quantity given the previous random choices of blocks to update. We get a deterministic inequality by taking the expectation of both sides with respect to the random choice of previous blocks, yielding:

$$\mathrm{E}[h(\boldsymbol{x}_\gamma^{(k+1)})] \leq \quad (1 - \tfrac{\gamma\nu}{n}) \, \mathrm{E}[h(\boldsymbol{x}^{(k)})] + \tfrac{\gamma^2}{2n} C_f^\otimes. \tag{21}$$

We observe that the resulting inequality (21) with $\nu = 1$ is of the same form as the one appearing in the standard Frank-Wolfe primal convergence proof such as in Jaggi (2013), though with a crucial difference of the $1/n$ factor (and that we are now working with the expected values $\mathrm{E}[h(\boldsymbol{x}^{(k)})]$ instead of the original $h(\boldsymbol{x}^{(k)})$). We will thus follow a similar induction argument over k, but we will see that the $1/n$ factor will yield a slightly different induction base case (which for $n = 1$ can be analyzed separately to obtain a better bound). To simplify the notation, let $h_k := \mathrm{E}[h(\boldsymbol{x}^{(k)})]$.

By induction, we are now going to prove that

$$h_k \leq \frac{2nC}{\nu k + 2n} \qquad \text{for} \quad k \geq 0 \; .$$

for the choice of constant $C := \frac{1}{\nu} C_f^\otimes + h_0$.

The *base-case* $k = 0$ follows immediately from the definition of C, given that $C \geq h_0$.

Now we consider the *induction step* for $k \geq 0$. Here the bound (21) for the particular choice of step-size $\gamma_k := \frac{2n}{\nu k + 2n} \in [0, 1]$ given by Algorithm C.2 gives us (the same bound also holds for the line-search variant, given that the corresponding objective value $f(\boldsymbol{x}_{\text{Line-Search}}^{(k+1)}) \leq f(\boldsymbol{x}_\gamma^{(k+1)})$ only improves):

$$\begin{aligned} h_{k+1} \leq & \quad (1 - \tfrac{\gamma_k \nu}{n}) h_k + (\gamma_k)^2 \tfrac{C\nu}{2n} \\ = & \quad (1 - \tfrac{2\nu}{\nu k + 2n}) h_k + (\tfrac{2n}{\nu k + 2n})^2 \tfrac{C\nu}{2n} \\ \leq & \quad (1 - \tfrac{2\nu}{\nu k + 2n}) \tfrac{2nC}{\nu k + 2n} + (\tfrac{1}{\nu k + 2n})^2 2nC\nu \; , \end{aligned}$$

where in the first line we have used that $C_f^\otimes \leq C\nu$, and in the last inequality we have plugged in the induction hypothesis for h_k. Simply rearranging the terms gives

$$\begin{aligned} h_{k+1} \leq & \quad \tfrac{2nC}{\nu k + 2n} \left(1 - \tfrac{2\nu}{\nu k + 2n} + \tfrac{\nu}{\nu k + 2n} \right) \\ = & \quad \tfrac{2nC}{\nu k + 2n} \tfrac{\nu k + 2n - \nu}{\nu k + 2n} \\ \leq & \quad \tfrac{2nC}{\nu k + 2n} \tfrac{\nu k + 2n}{\nu k + 2n + \nu} \\ = & \quad \tfrac{2nC}{\nu(k+1) + 2n} \; , \end{aligned}$$

which is our claimed bound for $k \geq 0$.

The analogous claim for Algorithm C.2 using the approximate linear primitive with *additive* approximation quality (11) with $\tilde{\gamma}_k = \frac{2n}{\nu k + 2n}$ follows from exactly the same argument, by replacing every occurrence of $C_f^\otimes$ in the proof here by $C_f^\otimes (1 + \delta)$ instead (compare to Lemma C.2 also – note that $\gamma = \tilde{\gamma}_k$ here). Note moreover that one can combine easily both a multiplicative approximation with an additive one as in (13), and modify the convergence statement accordingly. □

Domains Without Product Structure: $n = 1$. Our above convergence result also holds for the case of the standard Frank-Wolfe algorithm, when no product structure on the domain is assumed, i.e. for the case $n = 1$. In this case, the constant in the convergence can even be improved for the variant of the algorithm without a multiplicative approximation ($\nu = 1$), since the additive term given by h_0, i.e. the error at the starting point, can be removed. This is because already after the first step, we obtain a bound for h_1 which is independent of h_0. More precisely, plugging $\gamma_0 := 1$ and $\nu = 1$ in the bound (21) when $n = 1$ gives $h_1 \leq 0 + C_f^{\otimes}(1 + \delta) \leq C$. Using $k = 1$ as the base case for the same induction proof as above, we obtain that for $n = 1$:

$$h_k \leq \frac{2}{k+2} C_f^{\otimes}(1+\delta) \quad \text{for all} \quad k \geq 1 \ ,$$

which matches the convergence rate given in Jaggi (2013). Note that in the traditional Frank-Wolfe setting, i.e. $n = 1$, our defined curvature constant becomes $C_f^{\otimes} = C_f$.

Dependence on h_0. We note that the only use of including h_0 in the constant $C = \nu^{-1} C_f^{\otimes} + h_0$ was to satisfy the base case in the induction proof, at $k = 0$. If from the structure of the problem we can get a guarantee that $h_0 \leq \nu^{-1} C_f^{\otimes}$, then the smaller constant $C' = \nu^{-1} C_f^{\otimes}$ will satisfy the base case and the whole proof will go through with it, without needing the extra h_0 factor. See also Theorem C.4 for a better convergence result with a weaker dependence on h_0 in the case where the line-search is used.

C.3. Obtaining Small Duality Gap

The following theorem shows that after $O\left(\frac{1}{\varepsilon}\right)$ many iterations, Algorithm C.2 will have visited a solution with ε-small duality gap in expectation. Because the block-coordinate Frank-Wolfe algorithm is only looking at one block at a time, it doesn't know what is its current true duality gap without doing a full (batch) pass over all blocks. Without monitoring this quantity, the algorithm could miss which iterate had a low duality gap. This is why, if one is interested in having a good duality gap (such as in the structural SVM application), then the averaging schemes considered in (14) and (15) become interesting: the following theorem also says that the bound hold for *each* of the averaged iterates, *if the duality gap function g is convex*, which is the case for example when f is a quadratic function.[7]

Theorem C.3 (Primal-Dual Convergence)**.** *For each $K \geq 0$, the variants of Algorithm C.2 (either using the predefined step-sizes, or using line-search) will yield at least one iterate $\boldsymbol{x}^{(\hat{k})}$ with $\hat{k} \leq K$ with expected duality gap bounded by*

$$\mathrm{E}\left[g(\boldsymbol{x}^{(\hat{k})})\right] \leq \beta \frac{2n}{\nu(K+1)} C \ ,$$

where $\beta = 3$ and $C = \nu^{-1} C_f^{\otimes}(1+\delta) + f(\boldsymbol{x}^{(0)}) - f(\boldsymbol{x}^*)$. $\delta \geq 0$ and $0 < \nu \leq 1$ are the approximation quality parameters as defined in (13) – use $\delta = 0$ and $\nu = 1$ for the exact variant.

Moreover, if the duality gap g is a convex function of $\boldsymbol{x}$, then the above bound also holds both for $\mathrm{E}\left[g(\bar{\boldsymbol{x}}_w^{(K)})\right]$ and $\mathrm{E}\left[g(\bar{\boldsymbol{x}}_{0.5}^{(K)})\right]$ for each $K \geq 0$, where $\bar{\boldsymbol{x}}_w^{(K)}$ is the weighted average of the iterates as defined in (14) and $\bar{\boldsymbol{x}}_{0.5}^{(K)}$ is the 0.5-suffix average of the iterates as defined in (15) with $\mu = 0.5$.

Proof. To simplify notation, we will again denote the expected primal error and expected duality gap for any iteration $k \geq 0$ in the algorithm by $h_k := \mathrm{E}[h(\boldsymbol{x}^{(k)})] := \mathrm{E}[f(\boldsymbol{x}^{(k)}) - f(\boldsymbol{x}^*)]$ and $g_k := \mathrm{E}[g(\boldsymbol{x}^{(k)})]$ respectively.

The proof starts again by using the crucial improvement Lemma C.2 with $\gamma = \gamma_k := \frac{2n}{\nu k + 2n}$ to cover both variants of Algorithm C.2 at the same time. As in the beginning of the proof of Theorem C.1, we take the expectation with respect to $\boldsymbol{x}^{(k)}$ in Lemma C.2 and subtract $f(\boldsymbol{x}^*)$ to get that for each $k \geq 0$ (for the general approximate variant of the algorithm):

$$\begin{aligned} h_{k+1} \leq \ & h_k - \tfrac{1}{n}\gamma_k \nu \, g_k + \tfrac{1}{2n}({\gamma_k}^2 + \delta\tilde{\gamma}_k\gamma_k) C_f^{\otimes} \\ = \ & h_k - \tfrac{1}{n}\gamma_k \nu \, g_k + \tfrac{1}{2n}{\gamma_k}^2 C_f^{\otimes}(1+\delta) \ , \end{aligned}$$

[7] To see that g is convex when f is quadratic, we refer to the equivalence between the gap $g(\boldsymbol{x})$ and the Fenchel duality $p(\boldsymbol{x}) - d(\nabla f(\boldsymbol{x})))$ as shown in Appendix D. The dual function $d(\cdot)$ is concave, so if $\nabla f(\boldsymbol{x}))$ is an affine function of $\boldsymbol{x}$ (which is the case for a quadratic function), then d will be a concave function of $\boldsymbol{x}$, implying that $g(\boldsymbol{x}) = p(\boldsymbol{x}) - d(\nabla f(\boldsymbol{x})))$ is convex in $\boldsymbol{x}$, since the primal function p is convex.

since $\tilde{\gamma}_k \leq \gamma_k$. By isolating g_k and using the fact that $C \geq \nu^{-1} C_f^{\otimes}(1+\delta)$, we get the crucial inequality for the expected duality gap:

$$g_k \leq \frac{n}{\nu\gamma_k}(h_k - h_{k+1}) + \gamma_k \frac{C}{2} \ . \tag{22}$$

The general proof idea to get an handle on g_k is to take a convex combination over multiple k's of the inequality (22), to obtain a new upper bound. Because a convex combination of numbers is upper bounded by its maximum, we know that the new bound has to upper bound at least one of the g_k's (this gives the existence $\hat{k}$ part of the theorem). Moreover, if g is convex, we can also obtain an upper bound for the expected duality gap of the same convex combination of the iterates.

So let $\{w_k\}_{k=0}^K$ be a set of non-negative weights, and let $\rho_k := w_k / S_K$, where $S_K := \sum_{k=0}^K w_k$. Taking the convex combination of inequality (22) with coefficient ρ_k, we get

$$\begin{aligned}
\sum_{k=0}^K \rho_k g_k &\leq \frac{n}{\nu} \sum_{k=0}^K \rho_k \left(\frac{h_k}{\gamma_k} - \frac{h_{k+1}}{\gamma_k} \right) + \sum_{k=0}^K \rho_k \gamma_k \frac{C}{2} \\
&= \frac{n}{\nu} \left(h_0 \frac{\rho_0}{\gamma_0} - h_{K+1} \frac{\rho_K}{\gamma_K} \right) + \frac{n}{\nu} \sum_{k=0}^{K-1} h_{k+1} \left(\frac{\rho_{k+1}}{\gamma_{k+1}} - \frac{\rho_k}{\gamma_k} \right) + \sum_{k=0}^K \rho_k \gamma_k \frac{C}{2} \\
&\leq \frac{n}{\nu} h_0 \frac{\rho_0}{\gamma_0} + \frac{n}{\nu} \sum_{k=0}^{K-1} h_{k+1} \left(\frac{\rho_{k+1}}{\gamma_{k+1}} - \frac{\rho_k}{\gamma_k} \right) + \sum_{k=0}^K \rho_k \gamma_k \frac{C}{2} \ ,
\end{aligned} \tag{23}$$

using $h_{K+1} \geq 0$. Inequality (23) can be seen as a master inequality to derive various bounds on g_k. In particular, if we define $\bar{\boldsymbol{x}} := \sum_{k=0}^K \rho_k \boldsymbol{x}^{(k)}$ and we suppose that g is convex (which is the case for example when f is a quadratic function), then we have $\mathrm{E}[g(\bar{\boldsymbol{x}})] \leq \sum_{k=0}^K \rho_k g_k$ by convexity and linearity of the expectation.

Weighted-averaging case. We first consider the weights $w_k = k$ which appear in the definition of the weighted average of the iterates $\bar{\boldsymbol{x}}_w^{(K)}$ in (14) and suppose $K \geq 1$. In this case, we have $\rho_k = k/S_K$ where $S_K = K(K+1)/2$. With the predefined step-size $\gamma_k = 2n/(\nu k + 2n)$, we then have

$$\begin{aligned}
\frac{\rho_{k+1}}{\gamma_{k+1}} - \frac{\rho_k}{\gamma_k} &= \frac{1}{2nS_K} \left((k+1)(\nu(k+1)+2n) - k(\nu k + 2n) \right) \\
&= \frac{\nu(2k+1)+2n}{2nS_K} \ .
\end{aligned}$$

Plugging this in the master inequality (23) as well as using the convergence rate $h_k \leq \frac{2nC}{\nu k + 2n}$ from Theorem C.1, we obtain

$$\begin{aligned}
\sum_{k=0}^K \rho_k g_k &\leq \frac{n}{\nu S_K} \left[0 + \sum_{k=0}^{K-1} \frac{2nC}{\nu(k+1)+2n} \frac{\nu(2k+1)+2n}{2n} \right] + \sum_{k=0}^K \frac{2nk}{\nu k + 2n} \frac{C}{2S_K} \\
&\leq \frac{nC}{\nu S_K} \left[2 \sum_{k=0}^{K-1} 1 + \sum_{k=1}^K 1 \right] \\
&= \frac{2nC}{\nu(K+1)} \cdot 3.
\end{aligned}$$

Hence we have proven the bound with $\beta = 3$ for $K \geq 1$. For $K = 0$, the master inequality (23) becomes

$$g_0 \leq \frac{n}{\nu} h_0 + \frac{1}{2} C \leq \frac{nC}{\nu} \left(1 + \frac{1}{2n} \right)$$

since $h_0 \leq C$ and $\nu \leq 1$. Given that $n \geq 1$, we see that the bound also holds for $K = 0$.

Suffix-averaging case. For the proof of convergence of the 0.5-suffix averaging of the iterates $\bar{\boldsymbol{x}}_{0.5}^{(K)}$, we refer the reader to the proof of Theorem C.5 which can be re-used for this case (see the last paragraph of the proof to explain how). □

Domains Without Product Structure: $n = 1$. As we mentioned after the proof of the primal convergence Theorem C.1, we note that if $n = 1$, then we can replace C in the statement of Theorem C.3 by $C_f^{\otimes}(1 + \delta)$ for $K \geq 1$ when $\nu = 1$, as then we can ensure that $h_1 \leq C$ which is all what was needed for the primal convergence induction. Again, $C_f^{\otimes} = C_f$ when $n = 1$.

C.4. An Improved Convergence Analysis for the Line-Search Case

C.4.1. Improved Primal Convergence for Line-Search

If line-search is used, we can improve the convergence results of Theorem C.1 by showing a weaker dependence on the starting condition h_0 thanks to faster progress in the starting phase of the first few iterations:

Theorem C.4 (Improved Primal Convergence for Line-Search)**.** *For each $k \geq k_0$, the iterate $\boldsymbol{x}^{(k)}$ of the line-search variant of Algorithm C.2 (where the linear subproblem is solved with a multiplicative approximation quality (12) of $0 < \nu \leq 1$) satisfies*

$$\mathrm{E}\left[f(\boldsymbol{x}^{(k)})\right] - f(\boldsymbol{x}^*) \;\leq\; \frac{1}{\nu}\frac{2nC_f^{\otimes}}{\nu(k-k_0)+2n} \tag{24}$$

where $k_0 := \max\left\{0, \left\lceil \log\left(\frac{2\nu h(\boldsymbol{x}^{(0)})}{C_f^{\otimes}}\right) \Big/ (-\log \xi_n) \right\rceil\right\}$ is the number of steps required to guarantee that $\mathrm{E}\left[f(\boldsymbol{x}^{(k)})\right] - f(\boldsymbol{x}^*) \leq \nu^{-1}C_f^{\otimes}$, with $\boldsymbol{x}^* \in \mathcal{M}$ being an optimal solution to problem (10), and $h(\boldsymbol{x}^{(0)}) := f(\boldsymbol{x}^{(0)}) - f(\boldsymbol{x}^*)$ is the primal error at the starting point, and $\xi_n := 1 - \frac{\nu}{n} < 1$ is the geometric decrease rate of the primal error in the first phase while $k < k_0$ — i.e. $\mathrm{E}\left[f(\boldsymbol{x}^{(k)})\right] - f(\boldsymbol{x}^*) \leq (\xi_n)^k\, h(\boldsymbol{x}^{(0)}) + C_f^{\otimes}/2\nu$ for $k < k_0$.

If the linear subproblem is solved with an additive approximation quality (11) of $\delta \geq 0$ instead, then replace all appearances of $C_f^{\otimes}$ above with $C_f^{\otimes}(1 + \delta)$.

Proof. For the line-search case, the expected improvement guaranteed by Lemma C.2 for the multiplicative approximation variant of Algorithm C.2, in expectation as in (21), is valid for *any* choice of $\gamma \in [0, 1]$:

$$\mathrm{E}\left[h(\boldsymbol{x}_{LS}^{(k+1)})\right] \leq \quad (1 - \tfrac{\nu\gamma}{n})\,\mathrm{E}\left[h(\boldsymbol{x}^{(k)})\right] + \tfrac{\gamma^2}{2n}C_f^{\otimes}\ . \tag{25}$$

Because the bound (25) holds for any γ, we are free to choose the one which minimizes it subject to $\gamma \in [0, 1]$, that is $\gamma^* := \min\left\{1, \frac{\nu h_k}{C_f^{\otimes}}\right\}$, where we have again used the identification $h_k := \mathrm{E}\left[h(\boldsymbol{x}_{LS}^{(k)})\right]$. Now we distinguish two cases:

If $\gamma^* = 1$, then $\nu h_k \geq C_f^{\otimes}$. By unrolling the inequality (25) recursively to the beginning and using $\gamma = 1$ at each step, we get:

$$\begin{aligned}
h_{k+1} &\leq \left(1 - \tfrac{\nu}{n}\right) h_k + \tfrac{1}{2n}C_f^{\otimes} \\
&\leq \left(1 - \tfrac{\nu}{n}\right)^{k+1} h_0 + \tfrac{1}{2n}C_f^{\otimes} \textstyle\sum_{t=0}^{k} \left(1 - \tfrac{\nu}{n}\right)^t \\
&\leq \left(1 - \tfrac{\nu}{n}\right)^{k+1} h_0 + \tfrac{1}{2n}C_f^{\otimes} \textstyle\sum_{t=0}^{\infty} \left(1 - \tfrac{\nu}{n}\right)^t \\
&= \left(1 - \tfrac{\nu}{n}\right)^{k+1} h_0 + \tfrac{1}{2n}C_f^{\otimes} \left(\tfrac{1}{1-(1-\nu/n)}\right) \\
&= \left(1 - \tfrac{\nu}{n}\right)^{k+1} h_0 + \tfrac{1}{2\nu}C_f^{\otimes}\ .
\end{aligned}$$

We thus have a geometric decrease with rate $\xi_n := 1 - \frac{\nu}{n}$ in this phase. We then get $h_k \leq \nu^{-1}C_f^{\otimes}$ as soon as $(\xi_n)^k h_0 \leq C_f^{\otimes}/2\nu$, i.e. when $k \geq \log_{1/\xi_n}(2\nu h_0/C_f^{\otimes}) = \log(2\nu h_0/C_f^{\otimes})/-\log(1 - \frac{\nu}{n})$. We thus have obtained a logarithmic bound on the number of steps that fall into the first regime case here, i.e. where h_k is still 'large'. Here it is crucial to note that the primal error h_k is always decreasing in each step, due to the line-search, so once we leave this regime of $h_k \geq \nu^{-1}C_f^{\otimes}$, then we will never enter it again in subsequent steps.

On the other hand, as soon as we reach a step k (e.g. when $k = k_0$) such that $\gamma^* < 1$ or equivalently $h_k < \nu^{-1}C_f^{\otimes}$, then we are always in the second phase where $\gamma^* = \frac{\nu h_k}{C_f^{\otimes}}$. Plugging this value of γ^* in (25) yields the recurrence bound:

$$h_{k+1} \leq h_k - \frac{1}{\zeta}h_k^2 \quad \forall k \geq k_0 \tag{26}$$

where $\zeta := \frac{2nC_f^\otimes}{\nu^2}$, with the initial condition $h_{k_0} \leq \frac{C_f^\otimes}{\nu} = \frac{\nu\zeta}{2n}$. This is a standard recurrence inequality which appeared for example in Joachims et al. (2009, Theorem 5, see their Equation (23)) or in the appendix of Teo et al. (2007). We can solve the recurrence (26) by following the argument of Teo et al. (2007), where it was pointed out that since h_k is monotonically decreasing, we can upper bound h_k by the solution to the corresponding differential equations $h'(t) = -h^2(t)/\zeta$, with initial condition $h(k_0) = h_{k_0}$. Integrating both sides, we get the solution $h(t) = \frac{\zeta}{t-k_0+\zeta/h_{k_0}}$. Plugging in the value for h_{k_0} and since $h_k \leq h(k)$, we thus get the bound:

$$h_k \leq \frac{1}{\nu}\frac{2nC_f^\otimes}{\nu(k-k_0)+2n} \quad \forall k \geq k_0, \tag{27}$$

which completes the proof for the multiplicative approximation variant.

For the *additive approximation* variant, the inequality (25) with $\gamma = 1$ in Lemma C.2 becomes:

$$\begin{aligned} h_{k+1} \leq & \left(1-\tfrac{\nu}{n}\right) h_k + \tfrac{1}{2n}(1+\delta\tilde{\gamma}_k)C_f^\otimes \\ \leq & \left(1-\tfrac{\nu}{n}\right) h_k + \tfrac{1}{2n}(1+\delta)C_f^\otimes \ , \end{aligned}$$

since $\tilde{\gamma}_k \leq 1$. By unrolling this inequality as before, we get the geometric rate of decrease in the initial phase by using $\gamma = 1$ until $k = k_0$ where we can ensure that $h_{k_0} \leq C_f^\otimes(1+\delta)/\nu$. We then finish the proof by re-using the induction proof from Theorem C.1, but with Equation (24) as the induction hypothesis, replacing $C_f^\otimes$ with $C_f^\otimes(1+\delta)$. The base case at $k = k_0$ is satisfied by the definition of k_0. For the induction step, we use $\gamma_k = \frac{2n}{\nu(k-k_0)+2n}$ (note that because we use line-search, we are free to use any γ we want in the inequality from Lemma C.2), and use the crucial fact that $\tilde{\gamma}_k = \frac{2n}{\nu k+2n} \leq \gamma_k$ to get a similar argument as in Theorem C.1. □

Number of Iterations. We now make some observations in the case of $\delta = 0$ (for simplicity). Note that since for $n > 0.5$ and $-\log\left(1-\frac{\nu}{n}\right) > \frac{\nu}{n}$ for the natural logarithm, we get that $k_0 \leq \left\lceil \frac{n}{\nu}\log\left(\frac{2\nu h(\boldsymbol{x}^{(0)})}{C_f^\otimes}\right)\right\rceil$ and so unless the structure of our problem can guarantee that $h(\boldsymbol{x}^{(0)}) \leq C_f^\otimes/\nu$, we get a linear number of steps in n required to reach the second phase, but the dependence is logarithmic in $h(\boldsymbol{x}^{(0)})$ – instead of linear in $h(\boldsymbol{x}^{(0)})$ as given by our previous convergence Theorem C.1 for the fixed step-size variant (in the fixed step-size variant, we would need $k_0 = \left\lceil 2n\frac{h(\boldsymbol{x}^{(0)})}{C_f^\otimes}\right\rceil$ steps to guarantee $h_{k_0} \leq C_f^\otimes/\nu$). Therefore, for the line-search variant of our Algorithm C.2, we have obtained guaranteed ε-small error after

$$\left\lceil \frac{n}{\nu}\log\left(\frac{2\nu h(\boldsymbol{x}^{(0)})}{C_f^\otimes}\right)\right\rceil + \left\lceil \frac{2nC_f^\otimes}{\nu^2\,\varepsilon}\right\rceil$$

iterations.

Effect of Line-Search. It is also interesting to point out that even though we were using the optimal step-size in the second phase of the above proof (which yielded the recurrence (26)), the second phase bound is not better than what we could have obtained by using a fixed step-size schedule of $\frac{2n}{\nu(k-k_0)+2n}$ and following the same induction proof line as in the previous Theorem C.1 (using the base case $h_{k_0} \leq C_f^\otimes/\nu$ and so we could let $C := \nu^{-1}C_f^\otimes$). This thus means that the advantage of the line-search over the fixed step-size schedule only appears in knowing when to switch from a step-size of 1 (in the first phase, when $h_k \geq \nu^{-1}C_f^\otimes$) to a step-size of $\frac{2n}{\nu(k-k_0)+2n}$ (in the second phase), which unless we know the value of $f(\boldsymbol{x}^*)$, we cannot know in general. In the standard Frank-Wolfe case where $n = 1$ and $\nu = 1$, there is no difference in the rates for line-search or fixed step-size schedule as in this case we know $h_1 \leq C_f^\otimes$ as explained at the end of the proof of Theorem C.1. This also suggests that if $k_0 > n$, it might be more worthwhile in theory to first do one batch Frank-Wolfe step to ensure that $h_1 \leq C_f^\otimes$, and then proceed with the block-coordinate Frank-Wolfe algorithm afterwards.

C.4.2. Improved Primal-Dual Convergence for Line-Search

Using the improved primal convergence theorem for line-search, we can also get a better rate for the expected duality gap (getting rid of the dependence of h_0 in the constant C):

Theorem C.5 (Improved Primal-Dual Convergence for Line-Search)**.** *Let k_0 be defined as in Theorem C.4. For each $K \geq 5k_0$, the line-search variant of Algorithm C.2 will yield at least one iterate $\boldsymbol{x}^{(\hat{k})}$ with $\hat{k} \leq K$ with expected duality gap bounded by*

$$\mathrm{E}\left[g(\boldsymbol{x}^{(\hat{k})})\right] \leq \beta \frac{2n}{\nu(K+2)} C \ ,$$

where $\beta = 3$ and $C = \nu^{-1} C_f^{\otimes}(1+\delta)$. $\delta \geq 0$ and $0 < \nu \leq 1$ are the approximation parameters as defined in (13) – use $\delta = 0$ and $\nu = 1$ for the exact variant.

Moreover, if the duality gap g is a convex function of $\boldsymbol{x}$, then the above bound also holds for $\mathrm{E}\left[g(\bar{\boldsymbol{x}}_{0.5}^{(K)})\right]$ for each $K \geq 5k_0$, where $\bar{\boldsymbol{x}}_{0.5}^{(K)}$ is the 0.5-suffix average of the iterates as defined in (15) with $\mu = 0.5$.

Proof. We follow a similar argument as in the proof of Theorem C.3, but making use of the better primal convergence Theorem C.4 as well as using the 0.5-suffix average for the master inequality (23). Let $K \geq 5k_0$ be given. Let $\gamma_k := \frac{2n}{\nu(k-k_0)+2n}$ for $k \geq k_0$. Note then that $\tilde{\gamma}_k = \frac{2n}{\nu k+2n} \leq \gamma_k$ and so the gap inequality (22) appearing in the proof of Theorem C.3 is valid for this γ_k (because we are considering the line-search variant of Algorithm C.2, we are free to choose any $\gamma \in [0,1]$ in Lemma C.2). This means that the master inequality (23) is also valid here with $C = \nu^{-1} C_f^{\otimes}(1+\delta)$.

We consider the weights which appear in the definition of the 0.5-suffix average of iterates $\bar{\boldsymbol{x}}_{0.5}^{(K)}$ given in (15), i.e. the average of the iterates $\boldsymbol{x}^{(k)}$ from $k = K_s := \lceil 0.5K \rceil$ to $k = K$. We thus have $\rho_k = 1/S_K$ for $K_s \leq k \leq K$ and $\rho_k = 0$ otherwise, where $S_K = K - \lceil 0.5K \rceil + 1$. Notice that $K_s \geq k_0$ by assumption.

With these choices of ρ_k and γ_k, the master inequality (23) becomes

$$\begin{aligned}
\sum_{k=0}^{K} \rho_k g_k &\leq \frac{n}{\nu S_K}\left[\frac{h_{K_s}}{\gamma_{K_s}} + \sum_{k=K_s}^{K-1} h_{k+1}\left(\frac{1}{\gamma_{k+1}} - \frac{1}{\gamma_k}\right)\right] + \sum_{k=K_s}^{K} \gamma_k \frac{C}{2S_K} \\
&\leq \frac{n}{\nu S_K}\left[C + \sum_{k=K_s}^{K-1} \frac{2nC}{\nu(k+1-k_0)+2n}\left(\frac{\nu}{2n}\right)\right] + \sum_{k=K_s}^{K} \frac{2n}{\nu(k-k_0)+2n}\frac{C}{2S_K} \\
&= \frac{nC}{\nu S_K}\left[1 + \sum_{k=K_s}^{K-1} \frac{1}{k+1-k_0+2n/\nu} + \sum_{k=K_s}^{K} \frac{1}{k-k_0+2n/\nu}\right] \\
&\leq \frac{nC}{\nu S_K}\left[1 + 2\sum_{k=K_s}^{K} \frac{1}{k-k_0+2n/\nu}\right] \\
&\leq \frac{2nC}{\nu(K+2)}\left[1 + 2\sum_{k=K_s}^{K} \frac{1}{k-k_0+2n/\nu}\right],
\end{aligned} \tag{28}$$

where in the second line we used the faster convergence rate $h_k \leq \frac{2nC}{\nu(k-k_0)+2n}$ from Theorem C.4, given that $K_s \geq k_0$. In the last line, we used $S_K \leq 0.5K + 1$. The rest of the proof simply amounts to get an upper bound of $\beta = 3$ on the term between brackets in (28), thus concluding that $\sum_{k=0}^{K} \rho_k g_k \leq \beta \frac{2nC}{\nu(K+2)}$. Then following a similar argument as in Theorem C.3, this will imply that there exists some $g_{\hat{k}}$ similarly upper bounded (the existence part of the theorem); and that if g is convex, we have that $\mathrm{E}\left[g(\bar{\boldsymbol{x}}_{0.5}^{(K)})\right]$ is also similarly upper bounded.

We can upper bound the summand term in (28) by using the fact that for any non-negative decreasing integrable function f, we have $\sum_{k=K_s}^{K} f(k) \leq \int_{K_s-1}^{K} f(t)dt$. Let $a_n := k_0 - 2n/\nu$. Using $f(k) := \frac{1}{k-a_n}$, we have that

$$\begin{aligned}
\sum_{k=K_s}^{K} \frac{1}{k-a_n} &\leq \int_{K_s-1}^{K} \frac{1}{t-a_n} dt = \left[\log(t-a_n)\right]_{t=K_s-1}^{t=K} \\
&= \log\frac{K-a_n}{K_s-1-a_n} \leq \log\frac{K-a_n}{0.5K-1-a_n} =: b(K),
\end{aligned}$$

where we used $K_s \geq 0.5K$. We want to show that $b(K) \leq 1$ for $K \geq 5k_0$ to conclude that $\beta = 3$ works as a bound in (28) and thus completing the proof. By looking at the sign of the derivative of $b(K)$, we can see that it is an increasing function of K if $a_n \leq -2$ i.e. if $2n/\nu \geq k_0 + 2$ (which is always the case if $k_0 = 0$ as $n \geq 1$), and a strictly decreasing function of K otherwise. In the case where $b(K)$ is increasing, we have $b(K) \leq \lim_{K \mapsto \infty} b(K) = \log(2) < 1$. In the case where $b(K)$ is decreasing, we upper bound it by letting K take its minimal value from the theorem, namely $K \geq 5k_0$. From the definition of a_n, we then get that $b(5k_0) = \log \frac{4k_0 + 2n/\nu}{1.5k_0 - 1 + 2n/\nu}$, which is an increasing function of k_0 as long as $2n/\nu \geq 2$ (which is indeed always the case). So letting $k_0 \to \infty$, we get that $b(5k_0) \leq \log(4/1.5) \approx 0.98 < 1$, thus completing the proof.

We finally note that statement for $\mathrm{E}\left[g(\bar{\boldsymbol{x}}_{0.5}^{(K)})\right]$ in Theorem C.3 can be proven using the same argument as above, but with $k_0 = 0$ and $C = \nu^{-1} C_f^{\otimes}(1 + \delta) + h_0$ and using the original primal convergence bound on h_k in Theorem C.1 instead. This will work for both predefined step-size or the line search variants — the only place where we used the line-search in the above proof was to use the different primal convergence result as well as shifted-by-k_0 step-sizes γ_k (which reduce to the standard step-sizes when $k_0 = 0$). □

We note that we cannot fully get rid of the dependence on h_0 for the convergence rate of the expected duality gap of the weighted averaged scheme because we average over $k < k_0$, a regime where the primal error depends on h_0. With a more refined analysis for the weighted average with line-search scheme though, we note that one can replace the $h_0 \frac{n}{K}$ dependence in the bound with a $h_0 (\frac{n}{K})^2$ one, i.e. a quadratic speed-up to forget the initial conditions when line-search is used.

We also note that a bound of $O(1/K)$ can be derived similarly for $\mathrm{E}\left[g(\bar{\boldsymbol{x}}_{\mu}^{(K)})\right]$ for $0 < \mu < 1$ — namely using the C as in Theorem C.3 and $\beta = \beta_\mu := (1 - \mu)^{-1}(0.5 - \log \mu)$ (notice that $\beta_\mu = \infty$ if $\mu = 0$ or $\mu = 1$). This result is similar as the one for the stochastic subgradient method and where the $O(1/K)$ rate was derived by Rakhlin et al. (2012) for the $(1 - \mu)$-suffix averaging scheme — this provided a motivation for the scheme as the authors proved that the full averaging scheme has $\Omega((\log K)/K)$ rate in the worst case. If we use $\mu = 0$ (i.e. we average from the beginning), then the sum in (28) becomes $O(\log K)$, yielding $O((\log K)/K)$ for the expected gap.

D. Equivalence of the 'Linearization'-Duality Gap to a Special Case of Fenchel Duality

For our used constrained optimization framework, the notion of the simple duality gap was crucial. Consider a general constrained optimization problem

$$\min_{\boldsymbol{x} \in \mathcal{M}} f(\boldsymbol{x}) \ , \tag{29}$$

where the domain (or feasible set) $\mathcal{M} \subseteq \mathcal{X}$ is an arbitrary compact subset of a Euclidean space $\mathcal{X}$. We assume that the objective function f is convex, but not necessarily differentiable.

In this case, the general 'linearization' duality gap (5) as proposed by (Jaggi, 2013) is given by

$$g(\boldsymbol{x}; d_{\boldsymbol{x}}) = \mathbf{I}^*_{\mathcal{M}}(-d_{\boldsymbol{x}}) + \langle \boldsymbol{x}, d_{\boldsymbol{x}} \rangle \ . \tag{30}$$

Here $d_{\boldsymbol{x}}$ is an arbitrary subgradient to f at the candidate position $\boldsymbol{x}$, and $\mathbf{I}^*_{\mathcal{M}}(\boldsymbol{y}) := \sup_{\boldsymbol{s} \in \mathcal{M}} \langle \boldsymbol{s}, \boldsymbol{y} \rangle$ is the *support function* of the set $\mathcal{M}$.

Convexity of f implies that the linearization $f(\boldsymbol{x}) + \langle \boldsymbol{s} - \boldsymbol{x}, d_{\boldsymbol{x}} \rangle$ always lies below the graph of the function f, as illustrated by the figure in Section 3. This immediately gives the crucial property of the duality gap (30), as being a *certificate* for the current approximation quality, i.e. upper-bounding the (unknown) error $g(\boldsymbol{x}) \geq f(\boldsymbol{x}) - f(\boldsymbol{x}^*)$, where $\boldsymbol{x}^*$ is some optimal solution.

Note that for differentiable functions f, the gradient is the unique subgradient at $\boldsymbol{x}$, therefore the duality gap equals $g(\boldsymbol{x}) := g(\boldsymbol{x}; \nabla f(\boldsymbol{x}))$ as we defined in (5).

Fenchel Duality. Here we will additionally explain how the duality gap (30) can also be interpreted as a special case of standard Fenchel convex duality.

We consider the equivalent formulation of our constrained problem (29), given by

$$\min_{\boldsymbol{x}\in\mathcal{X}} \; f(\boldsymbol{x}) + \mathbf{I}_{\mathcal{M}}(\boldsymbol{x}) \; .$$

Here the *set indicator function* $\mathbf{I}_{\mathcal{M}}$ of a subset $\mathcal{M} \subseteq \mathcal{X}$ is defined as $\mathbf{I}_{\mathcal{M}}(\boldsymbol{x}) := 0$ for $\boldsymbol{x} \in \mathcal{M}$ and $\mathbf{I}_{\mathcal{M}}(\boldsymbol{x}) := +\infty$ for $\boldsymbol{x} \notin \mathcal{M}$.

The *Fenchel conjugate* function f^* of a function f is given by $f^*(\boldsymbol{y}) := \sup_{\boldsymbol{x}\in\mathcal{X}} \langle \boldsymbol{x}, \boldsymbol{y} \rangle - f(\boldsymbol{x})$.

For example, observe that the Fenchel conjugate of a set indicator function $\mathbf{I}_{\mathcal{M}}(.)$ is given by its support function $\mathbf{I}^*_{\mathcal{M}}(.)$.

From the above definition of the conjugate, the *Fenchel-Young inequality* $f(\boldsymbol{x}) + f^*(\boldsymbol{y}) \geq \langle \boldsymbol{x}, \boldsymbol{y} \rangle \; \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ follows directly.

Now we consider the *Fenchel dual problem* of minimizing $p(\boldsymbol{x}) := f(\boldsymbol{x}) + \mathbf{I}_{\mathcal{M}}(\boldsymbol{x})$, which is defined as to maximize $d(\boldsymbol{y}) := -f^*(\boldsymbol{y}) - \mathbf{I}^*_{\mathcal{M}}(-\boldsymbol{y})$. By the Fenchel-Young inequality, and assuming that $\boldsymbol{x} \in \mathcal{M}$, we have that $\forall \boldsymbol{y} \in \mathcal{X}$,

$$\begin{aligned} p(\boldsymbol{x}) - d(\boldsymbol{y}) &= f(\boldsymbol{x}) - (-f^*(\boldsymbol{y}) - \mathbf{I}^*_{\mathcal{M}}(-\boldsymbol{y})) \\ &\geq \langle \boldsymbol{x}, \boldsymbol{y} \rangle + \mathbf{I}^*_{\mathcal{M}}(-\boldsymbol{y}) \\ &= g(\boldsymbol{x}; \boldsymbol{y}) \; . \end{aligned}$$

Furthermore, this inequality becomes an equality if and only if $\boldsymbol{y}$ is chosen as a subgradient to f at $\boldsymbol{x}$, that is if $\boldsymbol{y} := -d_{\boldsymbol{x}}$. The last fact follows from the known equivalent characterization of the subdifferential in terms of the Fenchel conjugate: $\partial f(\boldsymbol{x}) := \{\boldsymbol{y} \in \mathcal{X} \mid f(\boldsymbol{x}) + f^*(\boldsymbol{y}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle\}$. For a more detailed explanation of Fenchel duality, we refer the reader to the standard literature, e.g. (Borwein & Lewis, 2006, Theorem 3.3.5).

To summarize, we have obtained that the simpler 'linearization' duality gap $g(\boldsymbol{x}; d_{\boldsymbol{x}})$ as given in (30) is indeed the difference of the current objective to the Fenchel dual problem, when being restricted to the particular choice of the dual variable $\boldsymbol{y}$ being a subgradient at the current position $\boldsymbol{x}$.

E. Derivation of the n-Slack Structural SVM Dual

Proof of the dual of the n-Slack-Formulation. See also Collins et al. (2008). For a self-contained explanation of Lagrange duality we refer the reader to Boyd & Vandenberghe (2004, Section 5). The Lagrangian of (1) is

$$L(\boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \frac{\lambda}{2}\langle \boldsymbol{w}, \boldsymbol{w} \rangle + \frac{1}{n}\sum_{i=1}^{n} \xi_i + \sum_{i\in[n],\, \boldsymbol{y}\in\mathcal{Y}_i} \frac{1}{n}\alpha_i(\boldsymbol{y}) \left(-\xi_i + \langle \boldsymbol{w}, -\boldsymbol{\psi}_i(\boldsymbol{y}) \rangle + L_i(\boldsymbol{y})\right),$$

where $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_n) \in \mathbb{R}^{|\mathcal{Y}_1|} \times \cdots \times \mathbb{R}^{|\mathcal{Y}_n|} = \mathbb{R}^m$ are the corresponding (non-negative) Lagrange multipliers. Here we have re-scaled the multipliers (dual variables) by a constant of $\frac{1}{n}$, corresponding to multiplying the corresponding original primal constraint by $\frac{1}{n}$ on both sides, which does not change the optimization problem.

Since the objective as well as the constraints are continuously differentiable with respect to $(\boldsymbol{w}, \xi)$, the Lagrangian L will attain its finite minimum over $\boldsymbol{\alpha}$ when $\nabla_{(\boldsymbol{w},\xi)} L(\boldsymbol{w}, \xi, \boldsymbol{\alpha}) = 0$. Making this saddle-point condition explicit results in a simplified Lagrange dual problem, which is also known as the Wolfe dual. In our case, this condition from differentiating w.r.t. $\boldsymbol{w}$ is

$$\lambda \boldsymbol{w} = \sum_{i\in[n],\, \boldsymbol{y}\in\mathcal{Y}_i} \frac{1}{n}\alpha_i(\boldsymbol{y})\boldsymbol{\psi}_i(\boldsymbol{y}) \; . \tag{31}$$

And differentiating with respect to ξ_i and setting the derivatives to zero gives[8]

$$\sum_{\boldsymbol{y}\in\mathcal{Y}_i} \alpha_i(\boldsymbol{y}) = 1 \quad \forall i \in [n] \; .$$

[8]Note that because the Lagrangian is linear in ξ_i, if this condition is not satisfied, the minimization of the Lagrangian in ξ_i yield $-\infty$ and so these points can be excluded.

Plugging this condition and the expression (31) for $\boldsymbol{w}$ back into the Lagrangian, we obtain the Lagrange dual problem

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\frac{\lambda}{2}\left\| \sum_{i\in[n],\, \boldsymbol{y}\in\mathcal{Y}_i} \alpha_i(\boldsymbol{y}) \frac{\boldsymbol{\psi}_i(\boldsymbol{y})}{\lambda n} \right\|^2 + \sum_{i\in[n],\, \boldsymbol{y}\in\mathcal{Y}_i} \alpha_i(\boldsymbol{y}) \frac{L_i(\boldsymbol{y})}{n} \\ \text{s.t.} \quad & \sum_{\boldsymbol{y}\in\mathcal{Y}} \alpha_i(\boldsymbol{y}) = 1 \quad \forall i \in [n], \\ & \text{and} \quad \alpha_i(\boldsymbol{y}) \geq 0 \quad \forall i \in [n],\, \forall \boldsymbol{y} \in \mathcal{Y}_i \ , \end{aligned}$$

which is exactly the negative of the quadratic program claimed in (4). □

F. Additional Experiments

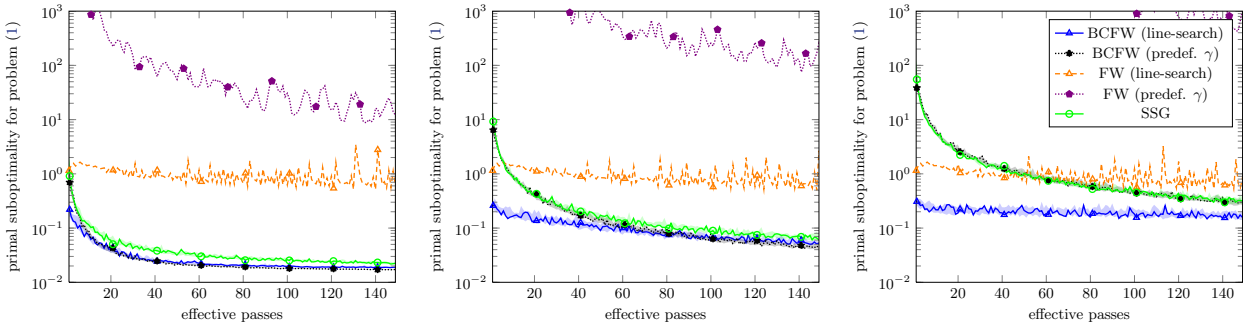
Complementing the results presented in Figure 1 in Section 6 of the main paper, here we provide additional experimental results as well as give more information about the experimental setup used.

For the Frank-Wolfe methods, Figure 2 presents results on OCR comparing setting the step-size by line-search against the simpler predefined step-size scheme of $\gamma_k = 2n/(k+2n)$. There, *BCFW* with predefined step-sizes does similarly as *SSG*, indicating that most of the improvement of *BCFW* with line-search over *SSG* is coming from the optimal step-size choice (and not from the Frank-Wolfe formulation on the dual). We also see that *BCFW* with predefined step-sizes can even do worse than batch Frank-Wolfe with line-search in the early iterations for small values of λ.

Figure 3 and Figure 4 show additional results of the stochastic solvers for several values of λ on the OCR and CoNLL datasets. Here we also include the (uniformly) averaged stochastic subgradient method (*SSG-avg*), which starts averaging at the beginning; as well as the 0.5-suffix averaging versions of both *SSG* and *BCFW* (*SSG-tavg* and *BCFW-tavg* respectively), implemented using the 'doubling trick' as described just after Equation (15) in Appendix C. The 'doubling trick' uniformly averages all iterates since the last iteration which was a power of 2, and was described by Rakhlin et al. (2012), with experiments for *SSG* in Lacoste-Julien et al. (2012). In our experiments, *BCFW-tavg* sometimes slightly outperforms the weighted average scheme *BCFW-wavg*, but its performance fluctuates more widely, which is why we recommend the *BCFW-wavg*, as mentioned in the main text. In our experiments, the objective value of *SSG-avg* is always worse than the other stochastic methods (apart *online-EG*), which is why it was excluded from the main text. *Online-EG* performed substantially worse than the other stochastic solvers for the OCR dataset, and is therefore not included in the comparison for the other datasets.[9]

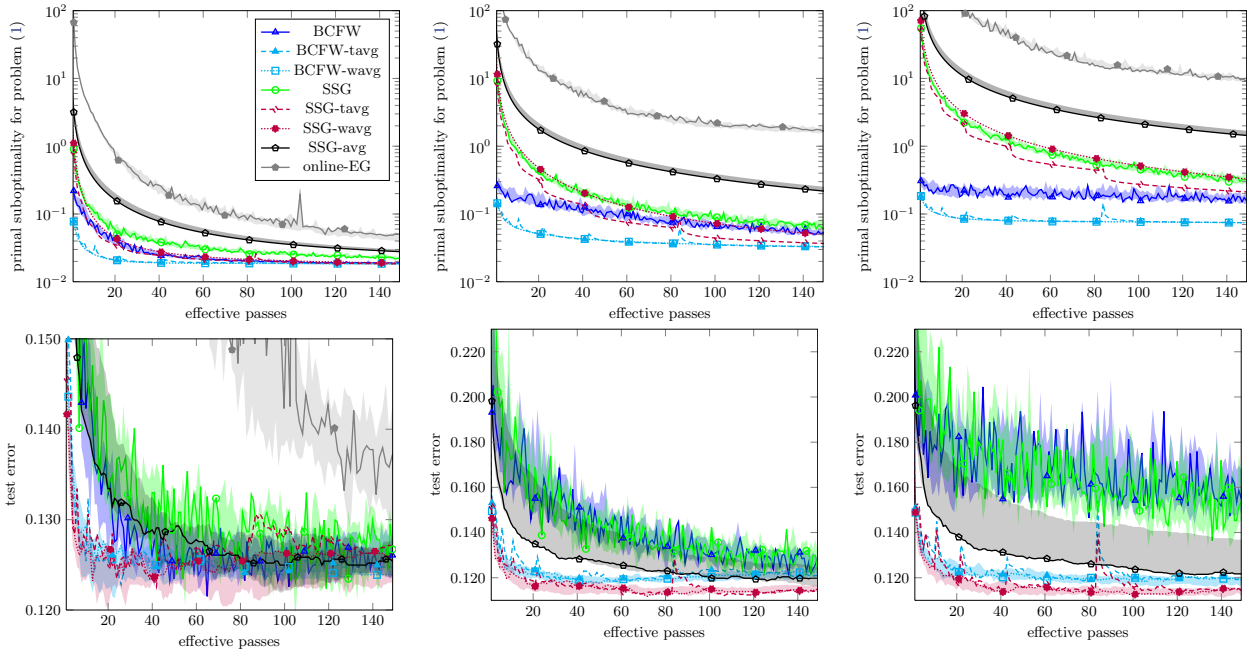
Finally, Figure 5 presents additional results for the matching application from Taskar et al. (2006).

[9]The worse performance of the online exponentiated gradient method could be explained by the fact that it uses a log-parameterization of the dual variables and so its iterates are forced to be in the *interior* of the probability simplex, whereas we know that the optimal solution for the structural SVM objective lies at the *boundary* of the domain and thus these parameters need to go to infinity.



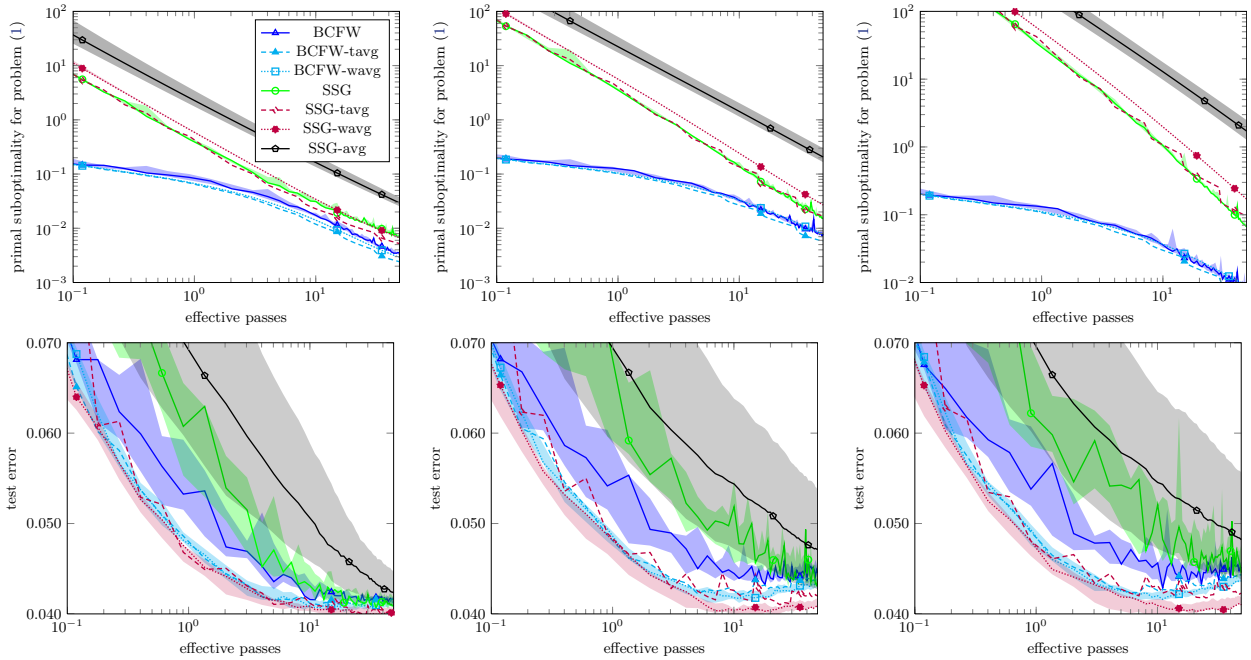
(a) $\lambda = 0.01$. (b) $\lambda = 0.001$. (c) $\lambda = 1/n = 0.00016$.

Figure 2. Convergence of the Frank-Wolfe algorithms on the OCR dataset, depending on the choice of the step-size. We compare the line-search variants as used in Algorithms 2 and 4, versus the simpler predefined step-sizes $\gamma := \frac{2}{k+2}$ (and $\gamma := \frac{2n}{k+2n}$ in the block-coordinate case respectively). See also the original optimization Algorithms 1 and 3.



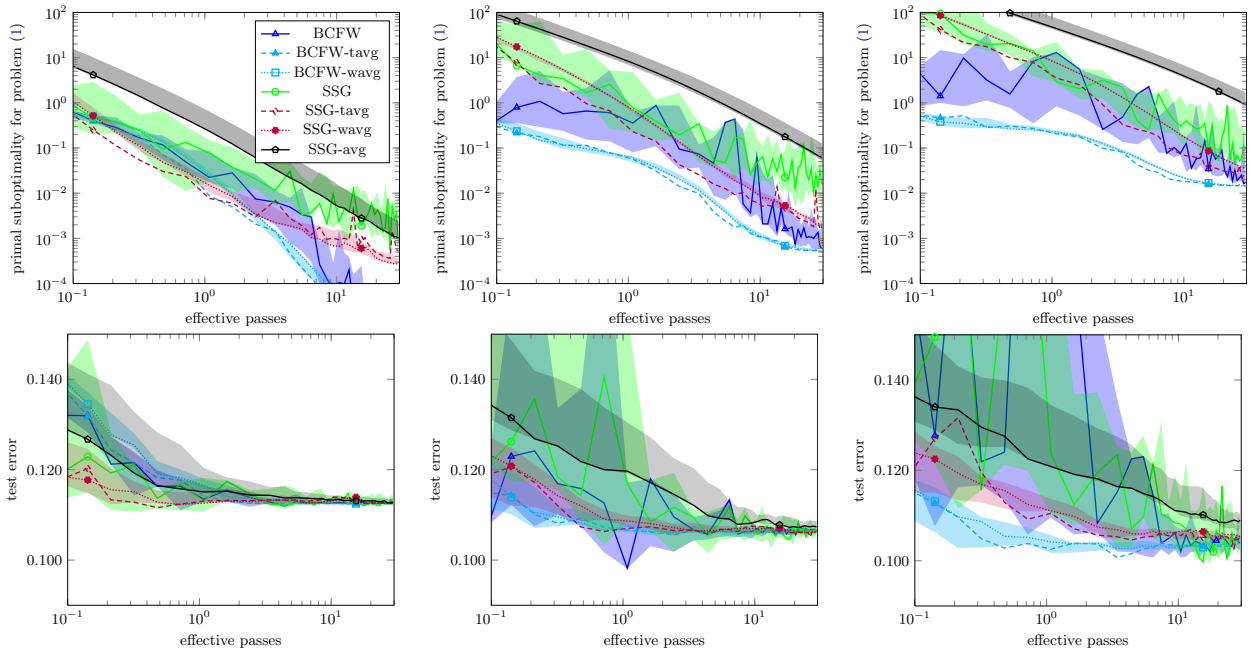
(a) $\lambda = 0.01$. (b) $\lambda = 0.001$. (c) $\lambda = 1/n = 0.00016$.

Figure 3. Convergence (top) and test error (bottom) of the stochastic solvers on the OCR dataset. While the block-coordinate Frank-Wolfe algorithm generally achieves the best objective, the averaging versions of the stochastic algorithms achieve a lower test error. While this interesting observation should be subject to further investigation, this could probably be due to the fact that these methods implicitly perform model averaging, which seems to lead to improved generalization performance. One can see some kind of 'overfitting' for example in the $\lambda = 0.001$ case, where *BCFW-wavg* reaches early a low test error which then starts to increase (and after running it for thousands of iterations, it does seem to converge to a parameter with higher test error than seen in the early iterations).



(a) $\lambda = 0.01$. (b) $\lambda = 0.001$. (c) $\lambda = 1/n = 0.000112$.

Figure 4. Convergence (top) and test error (bottom) of the stochastic solvers on the CoNLL dataset, using a logarithmic x-axis to focus on the early iterations.



(a) $\lambda = 0.01$. (b) $\lambda = 0.001$. (c) $\lambda = 1/n = 0.0002$.

Figure 5. Convergence (top) and test error (bottom) of the stochastic solvers on the Matching dataset.

More Information about Implementation. We note that since the value of the true optimum is unknown, the primal suboptimality for each experiment was measured as the difference to the highest dual objective seen for the corresponding regularization parameter (amongst all methods). Moreover, the *lower envelope* of the obtained primal objective values was drawn in Figure 1 for the batch methods (cutting plane and Frank-Wolfe), given that these methods can efficiently keep track of the best parameter seen so far.

The *online-EG* method used the same adaptive step-size scheme as described in Collins et al. (2008) and with the parameters from their code `egstra-0.2` available online.[10] Each datapoint has their own step-size, initialized at 0.5. Backtracking line-search is used, where the step-size is halved until the objective is decreased (or a maximum number of halvings has been reached: 2 for the first pass through the data; 5 otherwise). After each line-search, the step-size is multiplied by 1.05. We note that each evaluation of the objective requires a new call to the (expectation) oracle, and we count these extra calls in the computation of the effective number of passes appearing on the x-axis of the plots. Unlike all the other methods which initialize $\boldsymbol{w}^{(0)} = \mathbf{0}$, *online-EG* initially sets the dual variables $\boldsymbol{\alpha}_{(i)}^{(0)}$ to a uniform distribution, which yields a problem-dependent initialization $\boldsymbol{w}^{(0)}$.

For *SSG*, we used the same step-size as in the 'Pegasos' version of Shalev-Shwartz et al. (2010a): $\gamma_k := \frac{1}{\lambda(k+1)}$.

For the *cutting plane* method, we use the version 1.1 of the `svm-struct-matlab` MATLAB wrapper code from Vedaldi (2011) with its default options.

The test error for the OCR and CoNLL tasks is the normalized Hamming distance on the sequences.

For the matching prediction task, we use the same setting from Taskar et al. (2006), with 5,000 training examples and 347 Gold test examples. During training, an asymmetric Hamming loss is used where the precision error cost is 1 while the recall error cost is 3. For testing, error is the 'alignment error rate', as defined in Taskar et al. (2006).

Supplementary References

Borwein, J. and Lewis, A. *Convex analysis and nonlinear optimization: theory and examples*. 2006.

Boyd, S. and Vandenberghe, L. *Convex optimization*. 2004.

Luo, Z Q and Tseng, P. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35, 1992.

Patriksson, M. Decomposition methods for differentiable optimization problems over cartesian product sets. *Computational Optimization and Applications*, 9(1):5–42, 1998.

Richtárik, P. and Takáč, M. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. Technical Report 1107.2848v1 [math.OC], arXiv, 2011.

Teo, C.H., Smola, A.J., Vishwanathan, SVN, and Le, Q.V. A scalable modular convex solver for regularized risk minimization. *ACM SIGKDD*, pp. 727–736, 2007.

Vedaldi, A. A MATLAB wrapper of SVM$^{\text{struct}}$. `http://www.vlfeat.org/~vedaldi/code/svm-struct-matlab.html`, 2011.

[10] http://groups.csail.mit.edu/nlp/egstra/