A multi-plane block-coordinate Frank-Wolfe algorithm for training structural SVMs with a costly max-oracle

Neel Shah, Vladimir Kolmogorov, Christoph H. Lampert
IST Austria (Institute of Science and Technology Austria), Am Campus 1, 3400 Klosterneuburg, Austria

Abstract Structural support vector machines (SSVMs) are amongst the best performing methods for structured computer vision tasks, such as semantic image segmentation or human pose estimation. Training SSVMs, however, is computationally costly, because it requires repeated calls to a structured prediction subroutine (called *max-oracle*), which has to solve an optimization problem itself, e.g. a graph cut.

In this work, we introduce a new algorithm for SSVM training that is more efficient than earlier techniques when the max-oracle is computationally expensive, as it is frequently the case in computer vision tasks. The main idea is to (i) combine the recent stochastic Block-Coordinate Frank-Wolfe algorithm [2] with efficient hyperplane caching, and (ii) use an automatic selection rule for deciding whether to call the exact max-oracle or to rely on an approximate one based on the cached hyperplanes.

We show experimentally that this strategy leads to faster convergence to the optimum with respect to the number of requires oracle calls, and that this translates into faster convergence with respect to the total runtime when the max-oracle is slow compared to the other steps of the algorithm.

A C++ implementation is provided at `www.ist.ac.at/~vnk`.

SSVM objective We now give some technical details. The task of *structured prediction* is to predict structured objects, $y \in \mathcal{Y}$, for given inputs, $x \in \mathcal{X}$. Structural support vector machines (SSVMs) offer a principled way for learning a structured prediction function, $h : \mathcal{X} \to \mathcal{Y}$, from training data in a maximum margin framework. We parameterize $h(x) = \arg\max_{y\in\mathcal{Y}} \langle w, \phi(x,y)\rangle$, where $\phi : \mathcal{X}\times\mathcal{Y} \to \mathbb{R}^d$ is a joint feature function of inputs and outputs, and $\langle\cdot,\cdot\rangle$ denotes the inner product in $\mathbb{R}^d$. The weight vector, w, is learned from a training set, $\{(x_1,y_1),\dots,(x_n,y_n)\}$, by solving the following convex optimization problem for some regularization parameter $\lambda \geq 0$:

$$\min_w \quad \frac{\lambda}{2}\|w\|^2 + \sum_{i=1}^n H_i(w). \tag{1}$$

Here $H_i(w)$ is the (scaled) *structured hinge loss* that is defined as

$$H_i(w) = \frac{1}{n}\max_{y\in\mathcal{Y}}\{\Delta(y_i,y) - \langle w, \phi(x_i,y_i) - \phi(x_i,y)\rangle\}, \tag{2}$$

where $\Delta : \mathcal{Y}\times\mathcal{Y}\to\mathbb{R}$ is a task-specific loss function, e.g. the Hamming loss for image segmentation tasks. It can be written more compactly as

$$H_i(w) = \max_{y\in\mathcal{Y}} \langle \varphi^{iy}, [w\ 1]\rangle \quad \text{with} \quad \varphi^{iy} = \frac{1}{n}[(\phi(x_i,y) - \phi(x_i,y_i))\ \ \Delta(y^i,y)], \tag{3}$$

where $[a\ b] \in \mathbb{R}^{d+1}$ is the concatenation of vector $a \in \mathbb{R}^d$ and scalar $b \in \mathbb{R}$.

Computing the value of $H_i(w)$, or the label that realizes this value, requires solving an optimization problem over the label set. We refer to the procedure to do so as the *max-oracle*, or just *oracle*.

Block-Coordinate Frank Wolfe (BCFW) [2] The algorithm solves the dual problem to (1). For each term $i \in [n]$ it maintains a vector of dual variables $\varphi^i \in \mathbb{R}^{d+1}$ which is a convex combination of vectors $\{\varphi^{iy} \mid y \in \mathcal{Y}\}$. It also maintains the sum $\varphi = \sum_{i=1}^n \varphi^i$. A basic step of BCFW is to pick an index $i \in [n]$ (e.g. at random), call the i-th oracle to get a subgradient $\hat\varphi^i$ of $H_i(w)$ (for a certain vector w computed from φ), and then update φ^i based on vectors φ^i, $\hat\varphi^i$ and φ. The dual objective is guaranteed to increase (or stay the same, if we are already at the maximum with respect to φ^i).

This work: Multi-Plane BCFW (MP-BCFW) We observe that the computational efforts in BCFW can be very unbalanced: there is only $\Theta(d)$ amount of work per each (potentially very slow) oracle call. Also, BCFW acts wastefully by discarding the plane $\hat\varphi^i$ after completing the step for the

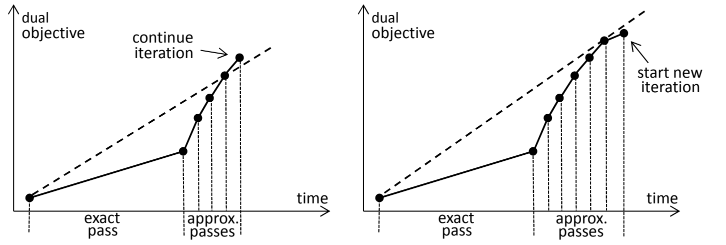

Figure 1: After each approximate pass, we compare the relative progress (increase in objective per time = slope of the last black line segment) to the relative progress of the complete iteration so far (slope of the dashed line). If the former is higher (left), we make another approximate pass. Otherwise, we start a new iteration by performing an exact pass (right).

term H_i, even though it required an expensive call to the max-oracle to obtain $\hat\varphi^i$. We propose to retain some of these planes, maintaining a *working set* $\mathcal{W}_i$ for each $i \in [n]$. Inactive planes are removed from $\mathcal{W}_i$ after a certain number of iterations. This is similar to [1], where a working set was used in a cutting-plane framework. However, we use set $\mathcal{W}_i$ in a different way. We define function $\tilde H_i(w) = \max_{\tilde\varphi^i\in\mathcal{W}_i}\langle\tilde\varphi^i, [w\ 1]\rangle$, which we view as an approximation of $H_i(w)$. We now perform *exact passes* (where we go through all $i \in [n]$ in a random order and call the "exact" oracle for H_i) and *approximate passes* (where we call an "approximate" oracle for $\tilde H_i$ instead).

We propose a geometrically motivated criterion described in Fig. 1 to decide "on the fly" which pass to perform next. Intuitively, it tries to maximize the increase of the dual objective per unit of time, This gives a easy-to-use method whose default settings work well for a range of different scenarios (e.g. both fast and slow oracles). Some results are shown below.

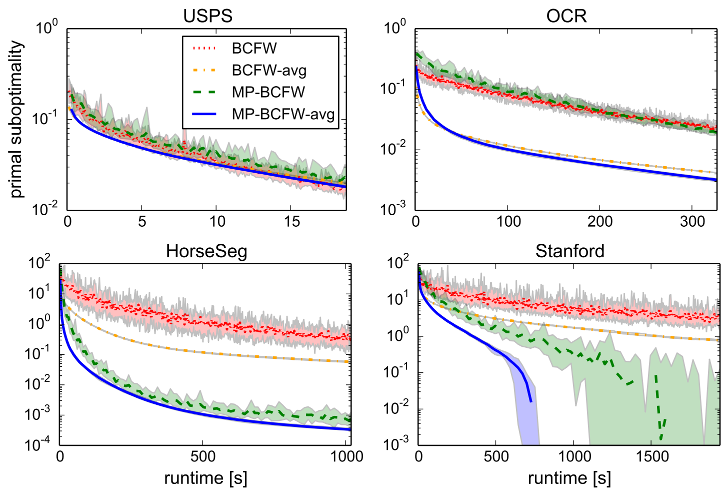

Figure 2: Shaded areas indicate minimum and maximum values over 10 repeats. When the max-oracle is fast (USPS and OCR), the multi-plane algorithms (MP-BCFW, MP-BCFW-avg) behave similarly to their single-plane counterparts (BCFW, BCFW-avg) due to the automatic parameter adjustment. When the max-oracle is computationally costly (HorseSeg, Stanford) the multi-plane variants converge substantially faster.

[1] T. Joachims, T. Finley, and C. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 1:27–59, 2009.

[2] S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher. Block-coordinate Frank-Wolfe optimization for structural SVMs. In *ICML*, 2014.

This is an extended abstract. The full paper is available at the Computer Vision Foundation webpage: `http://www.cv-foundation.org/openaccess/CVPR2015.py`