Review of Frank Wolfe and its variants

William Saint-Arnaud WILLIAM.ST-ARNAUD@UMONTREAL.CA
Elyes Lamouchi ELYESLAMOUCHI@GMAIL.COM
Frederic Boileau FREDERIC.BOILEAU@UMONTREAL.CA

Abstract

Due to the combinatorial nature of multilabel outputs, predicting structured data typically comes with an exponentially large number of constraints, which makes the problem inefficient or intractable in practice. There has been a lot of research focused on providing a solution to that issue. In the structured SVM setting, conditional gradient a.k.a Frank-Wolfe type algorithms have become a method of choice.

We start by presenting an earlier algorithm, namely the Dual Extragradient, then move on to the conditional gradient where we synthesize the recent advances, starting from the classical F-W to the more sophisticated variants, while motivating this with the problems each variant addresses. Then, we discuss the pitfalls of some variants and their intrinsic trade-offs. Finally we evaluate the performance of the methods proposed on synthetic data to see how reasonable are the assumptions (providing theoretical guarantees), and to get an idea whether each variant's trade-off is worth it.

1. Extragradient Algorithm for structured predictions

1.0.1. Structured output context

Consider the problem of learning in a structured output setting.
A popular method for this class of problems is using a probabilistic graphical model (PGM) to represent conditional independencies and learn a joint distribution over a large number of variables.
However, because of the combinatorial nature of the structured output space $\mathcal{Y}$, computing the partition function (which consists of summing over an exponential number of possible combinations of labels) is intractable.

$$Z_w(x) = \underbrace{\sum_{y \in \mathcal{Y}}}_{\text{exponential number of summands}} P_w(y|x)$$

Which makes estimating the maximum likelihood over a PGM unfeasible.

Large margin estimation addresses this problem, by formulating a problem which reduces the exponential number of constraints into a tractable one.
For a dataset $S = \{(\vec{x}_i, \vec{y}_i)\}_{i=1}^m$, where $\forall \vec{x}_i \in \mathcal{X}$, $\vec{y}_i$ is a multilabel output, we attempt to find the optimal parameter $\vec{w}$ of a linear classifier:

$$\vec{y}_i = _{\vec{y}'_i \in \mathcal{Y}} \ \vec{w}^T \vec{f}(\vec{x}_i, \vec{y}'_i) \tag{1}$$

With f being the feature map, we minimize the hinge loss $H_i(\vec{w})$,

$$\min_{\vec{w} \in \mathcal{W}} \sum_i \max_{\vec{y}'_i \in \mathcal{Y}_i} \underbrace{\left[\vec{w}^T \vec{f}_i(\vec{y}'_i) + l_i(\vec{y}'_i)\right] - \vec{w}^T \vec{f}_i(\vec{y}_i)}_{=H_i(\vec{w})} \tag{2}$$

By setting $\mathcal{W} = \{\vec{w} : ||\vec{w}||_2 \leq \gamma\}$, the parameter $\vec{w}$ is implicitly regularized. Moreover, since we are optimizing over $\vec{y}'_i$, we can drop the last term in equation (1), ending up with a loss-augmented inference problem inside the min function.

Assuming a Markov Network structure (undirected PGM) on the data, let $\vec{z}_i$'s be the node and marginalized edge potentials for the i^{th} node, $\vec{F}_i$ a vector of the feature mapping of all the possible labels for $\vec{y}_i$. and $\vec{c}_i$ be a vector of misclassification costs for the labels of $\vec{y}'_i$.

$$\min_{\vec{w} \in \mathcal{W}} \max_{\vec{z} \in \mathcal{Z}} \sum_i \left(\vec{w}^T \vec{F}_i \vec{z}_i + \vec{c}_i^T \vec{z}_i - \vec{w}^T \vec{f}_i(\vec{y}_i)\right) \tag{3}$$

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

Taking the dual, we have the following formulation,

$$\begin{aligned} \min_{\vec{w}\in\mathcal{W},(\vec{\lambda},\vec{\mu})\geq\vec{0}} \quad & \sum_i \left(\vec{b}_i^T\lambda_i + \mathbf{1}^T\vec{\mu}_i - \vec{w}^T\vec{f}_i(\vec{y}_i)\right) \\ \text{s.t.} \quad & \vec{F}_i^T\vec{w} + \vec{c}_i \leq \vec{A}_i^T\vec{\lambda}_i + \vec{\mu}_i \quad i = 1,\ldots,m \end{aligned} \tag{4}$$

The number of constraints is linear in the size of the data, hence learning w is now tractable.

1.0.2. Game interpretation

$$\mathcal{L}(\vec{w},\vec{z}) \triangleq \sum_i \vec{w}^T\vec{F}_i\vec{z}_i + \vec{c}_i^T - \vec{w}^T\vec{f}_i(\vec{y}_i) \tag{5}$$

The loss being bilinear in w and z, we can then imagine two players represented by $\vec{w}$ and $\vec{z}$ that play a zero-sum game. They perform updates using gradients of the objective w.r.t. their parameters. They then project the result to the set of feasible points given by the constraints imposed on the structure.

We measure the "goodness" of the parameters using the gap function $\mathcal{G}$:

$$\mathcal{G}(\vec{w},\vec{z}) \left[\max_{\vec{z}'\in\mathcal{Z}} \mathcal{L}(\vec{w},\vec{z}') - \mathcal{L}^*\right] + \left[\mathcal{L}^* - \min_{\vec{w}'\in\mathcal{W}} \mathcal{L}(\vec{w}',\vec{z})\right] \tag{6}$$

Where $\mathcal{L}^*$ gives the optimal value of the min-max problem. When we have a non-optimal point (i.e. not a saddle point), the gap is strictly positive. However, at an optimal point, the gap is exaclty equal to 0. Now the restricted gap is exactly the same but the min and max are computed over a set of parameters that are within a certain distance of the start point $(\hat{\vec{u}}_{\vec{w}}, \hat{\vec{u}}_{\vec{z}}) \in \mathcal{U}$:

$$\begin{aligned} \mathcal{G}_{D_{\vec{w}},D_{\vec{z}}}(\vec{w},\vec{z}) = & \max_{\vec{z}'\in\mathcal{Z}} [\mathcal{L}(\vec{w}',\vec{z}') : d(\vec{z},\vec{z}') \leq D_{\vec{z}}] \\ & - \left[\min_{\vec{w}'\in\mathcal{W}} \mathcal{L}(\vec{w}',\vec{z}) : d(\vec{w},\vec{w}') \leq D_{\vec{w}'}\right] \end{aligned} \tag{7}$$

The motivation for using this restricted gap function is that when starting "close" to an optimal point, we converge more rapidly to it.

1.0.3. Dual Extragradient algorithm

Considered the matrix formulation of the loss function,

$$\textit{Let} \quad \vec{F} = \begin{pmatrix} 0 & \vec{F}_1 & \ldots & \vec{F}_m \\ -\vec{F}_1^T & & & \\ \vdots & & \vec{0} & \\ -\vec{F}_m^T & & & \end{pmatrix}$$

Algorithm 1 Dual Extragradient

Initialize: Choose $\hat{\vec{u}} \in \mathcal{U}$, set $\vec{s}^{-1} = 0$.
for $t = 0$ to $t = \tau$ **do**
  $\vec{v} = \mathbf{\Pi}_{\mathcal{U}}(\hat{\vec{u}} + \eta\vec{s}^{t-1})$
  $\vec{u}^t = \mathbf{\Pi}_{\mathcal{U}}(\vec{v} - \eta(\vec{F}\vec{v} - \vec{a}))$
  $\vec{s}^t = \vec{s}^{t-1} - (\vec{F}\vec{u}^t - \vec{a})$
end for
return $\overline{\vec{u}^\tau} = \frac{1}{1+\tau}\sum_{t=0}^{\tau}\vec{u}^t$

$$\vec{u} = \begin{pmatrix} \vec{w} \\ \vec{z}_1 \\ \vdots \\ \vec{z}_m \end{pmatrix} \quad \textit{and} \quad \vec{a} = \begin{pmatrix} \sum_i \vec{f}_i(\vec{y}_i) \\ \vec{c}_1 \\ \vdots \\ \vec{c}_m \end{pmatrix}$$

$$\textit{such that} \begin{pmatrix} \nabla_{\vec{w}}\mathcal{L}(\vec{w},\vec{z}) \\ -\nabla_{\vec{z}_1}\mathcal{L}(\vec{w},\vec{z}) \\ \vdots \\ -\nabla_{\vec{z}_m}\mathcal{L}(\vec{w},\vec{z}) \end{pmatrix} = \vec{F}\vec{u} - \vec{a}$$

The Dual Extragradient (algorithm 1) has a lookahead step (the assignment in v) that serves to peform the actual gradient update u^t. The intuition behind this step is that given a function f with Lipschitz gradient, we have

$$f_D(\bar{u^n}) = \max_y \{\langle g(y), \bar{u^n} - y\rangle : d(\hat{u}, y) \leq D\}$$

where $\bar{u^n}$ is the weighted average over all the updates u^t up to iteration n and g corresponds to $\mathcal{L}(\vec{w},\vec{z})$.
When value of $f_D(\bar{u^n})$ gets close to 0, we have $g(y^*) \approx 0$ for y^* optimal, which means that we reached a saddle-point.
Nesterov then shows that this upper bound goes to 0, which proves the convergence to a saddle point.

1.0.4. Proximal step operator

The proximal step operator is defined as follows:

$$\mathcal{T}_\eta(\vec{u},\vec{s}) \triangleq \max_{\vec{u}\in\mathcal{U}} \left\{\langle\vec{s},\vec{u}' - \vec{u}\rangle - \frac{1}{\eta}d(\vec{u},\vec{u}') \leq D\right\} \tag{8}$$

Since $h(\vec{u})$ is strongly convex, we can find its convex conjugate $h^*(\vec{u}) = \max_{\vec{u}\in\mathcal{U}}[\langle\vec{s},\vec{u}\rangle - h(\vec{u})]$.
Following from the definition of strong convexity for h, we have,

$$h(\vec{u}') \geq h(\vec{u}) + \langle\nabla h(\vec{u}), \vec{u}' - \vec{u}\rangle + \frac{\sigma}{2}\|\vec{u}' - \vec{u}\|^2 \tag{9}$$

where σ is the strong convexity parameter. Moreover, let $d(\vec{u}',\vec{u})$ be an upper bound on the squared norm of $\vec{u}' - \vec{u}$. We get,

$$d(\vec{u}',\vec{u})h(\vec{u}') - h(\vec{u}) - \langle\nabla h(\vec{u}), \vec{u}' - \vec{u}\rangle \geq \frac{\sigma}{2}\|\vec{u}' - \vec{u}\|^2 \tag{10}$$

The distance metric d is called the **Bregman divergence**.
For example, if we have $h(\vec{u}) = \frac{1}{2}\|\vec{u}\|_2^2$, the Bregman divergence becomes $d(\vec{u}', \vec{u}) = \frac{1}{2}\|\vec{u}' - \vec{u}'\|_2^2$.
We might wonder why we care about the Bregman divergence when the definition still includes the usual norm. After all, we still optimize the term $\langle \vec{s}, \vec{u}' - \vec{u}\rangle - \frac{1}{\eta} d(\vec{u}', \vec{u})$. This is because the conjugate h^* is differentiable at every point of its domain by the strong convexity of h.
Thus, it is easy to compute a projection in the usual fashion: we compute the derivative of the term inside the projection operator and set it to 0.
For matching, since the distance is not differentiable we need an alternative way. We provide the following steps to compute a projection:

$$\begin{aligned} &\vec{s} - \nabla_{\vec{u}'} d(\vec{u}', \vec{u}) = \vec{s} - \frac{1}{\eta} \nabla_{\vec{u}'} d(\vec{u}, \vec{u}') \\ &= \vec{s} - \frac{1}{\eta} \left[\nabla h(\vec{u}') - \nabla h(\vec{u})\right] \end{aligned} \tag{11}$$

By setting this equation to 0, it is possible to recover the optimal $\vec{y}'$ when, say, $h(\vec{u}) = \frac{1}{2}\|\vec{u}'\|^2$.

1.0.5. Convergence results

Theorem. The restricted gap function given by the Dual Extragradient algorithm satisfies,

$$\mathcal{G}_{D_{\vec{w}}, D_{\vec{z}}}(\overline{\vec{w}^\tau}, \overline{\vec{z}^\tau}) \leq \frac{(D_{\vec{w}} + D_{\vec{z}})\, L}{\tau + 1}. \tag{12}$$

Therefore we have a gap of $\mathcal{O}(\frac{1}{\epsilon})$, hence a sublinear convergence rate for the Dual Extragradient algorithm.

Proof sketch. In his proof on the convergence of the extragradient algorithm, Nesterov uses a function f_D instead of $\mathcal{G}_{D_{\vec{w}}, D_{\vec{z}}}$,

$$f_D(\vec{x}) = \max_{\vec{y} \in \mathcal{Q}} \{\langle g(\vec{y}), \vec{x} - \vec{y}\rangle : d(\vec{x}, \vec{y})\} \tag{13}$$

where $\mathcal{Q}$ is the set of parameters and g is a monotone operator. We can already see a link between the function f_D and the gap $\mathcal{G}_{D_{\vec{w}}, D_{\vec{z}}}$. This comes out as:

$$\begin{aligned} &\mathcal{G}_{D_{\vec{w}}, D_{\vec{z}}}(\vec{w}, \vec{z}) = \sum_i \vec{w}^T \vec{F}_i \vec{z}_i^* - (\vec{w}^*)^T \vec{F}_i \vec{z}_i \\ &- \sum_i (\vec{w}^T - (\vec{w}^*)^T) \vec{f}_i(\vec{y}_i) - \sum_i \vec{c}_i^T (\vec{z}_i - \vec{z}_i^*) \\ &= \sum_i (\vec{z}_i^*)^T \vec{F}_i^T \vec{w} - (\vec{w}^*)^T \vec{F}_i \vec{z}_i - \sum_i (\vec{f}_i(\vec{y}_i))^T (\vec{w} - \vec{w}^*) \\ &\qquad - \sum_i \vec{c}_i^T (\vec{z}_i - \vec{z}_i^*) \\ &= \sum_i (\vec{z}_i^*)^T \vec{F}_i^T (\vec{w} - \vec{w}^*) - (\vec{w}^*)^T \vec{F}_i (\vec{z}_i - \vec{z}_i^*) \\ &\qquad - \sum_i (\vec{f}_i(\vec{y}_i))^T (\vec{w} - \vec{w}^*) - \sum_i \vec{c}_i^T (\vec{z}_i - \vec{z}_i^*) \\ &= \begin{pmatrix} \sum_i \vec{F}_i \vec{z}_i^* \\ -\vec{F}_1^T \vec{w}^* \\ \vdots \\ -\vec{F}_m^T \vec{w}^* \end{pmatrix}^T \begin{pmatrix} \vec{w} - \vec{w}^* \\ \vec{z}_1 - \vec{z}_1^* \\ \vdots \\ \vec{z}_m - \vec{z}_m^* \end{pmatrix} \\ &\qquad - \begin{pmatrix} \sum_i \vec{f}_i(\vec{y}_i) \\ \vec{c}_1 \\ \vdots \\ \vec{c}_m \end{pmatrix}^T \begin{pmatrix} \vec{w} - \vec{w}^* \\ \vec{z}_1 - \vec{z}_1^* \\ \vdots \\ \vec{z}_m - \vec{z}_m^* \end{pmatrix} \end{aligned} \tag{14}$$

From this, we deduce that the function g from the definition of f_D corresponds to:

$$g(\vec{w}, \vec{z}) = \begin{pmatrix} \sum_i \vec{F}_i \vec{z}_i^* \\ -\vec{F}_i^T \vec{w}^* \\ \vdots \\ -\vec{F}_m^T \vec{w}^* \end{pmatrix} - \begin{pmatrix} \sum_i \vec{f}_i(\vec{y}_i) \\ \vec{c}_1 \\ \vdots \\ \vec{c}_m \end{pmatrix} = \vec{F}\vec{u}^* - \vec{a} \tag{15}$$

This function is constant and thus monotone as required by Nesterov's proof of the convergence of the algorithm. Its Lipschitz constant is,

$$L = \max_{\vec{u} \in \mathcal{U}} \|\vec{F}(\vec{u} - \vec{u}')\|_2 / \|\vec{u} - \vec{u}'\|_2 \leq \|\vec{F}\|_2$$

Moreover, if $\vec{w}, \vec{z}$ satisfy $\|\vec{w}\|_2 \leq D_{\vec{w}}$ and $\|\vec{z}\|_2 \leq D_{\vec{z}}$ then $\|(\vec{w}, \vec{z})\|_2 \leq D$ when $D = \sqrt{D_{\vec{w}}^2 + D_{\vec{z}}^2}$ since $(\vec{w}, 0) \perp (0, \vec{z})$.
It is then easy to see that $f_D \geq \mathcal{G}_{D_{\vec{w}}, D_{\vec{z}}}$. Thus $\mathcal{G}_{D_{\vec{w}}, D_{\vec{z}}}$ is upper bounded by the right-hand side of equation 12. We can also observe that the function $g(\vec{w}, \vec{z})$ is exactly the gradient of the objective $\mathcal{L}(\vec{w}, \vec{z})$ at the point $\vec{w}, \vec{z}$.

1.0.6. Non-Euclidean setting

The main problem with the Euclidean projection operator is that for many problems, it is hard to compute. Indeed for

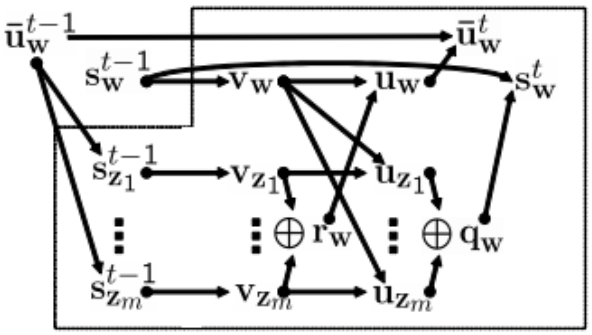

Figure 1. Dependency diagram for memory-efficient dual extragradient. The dotted box represents the computations of an iteration of the algorithm.

min-cut, we need to compute the partition function first, which is #P-complete. Thus, the authors of the paper introduced the Bregman operator, which computes the projection using the Bregman divergence which is much easier to compute. We can see this with l_1 regularization. Computing a projection using the l_1 distance is hard since it is not differentiable.
However, using the negative entropy, we find that the Bregman divergence corresponds to the KL divergence, which is differentiable. Hence we can get the parameter that minimizes it.

1.0.7. MEMORY-EFFICIENT TWEAK

In the dual extragradient algorithm, both a vector s^t and a vector $\bar{u^t}$ are maintained. However, we can observe that the s_t's can be found using the running average $\bar{u^t}$ since $s^t = -(t+1)\sum_{i=0}^{t}(F\bar{u^t} - a)$. Hence we only have to store the vector $\bar{u^t}$.
Moreover, when $|\mathcal{Z}| \gg |\mathcal{W}|$ since $\bar{u^t} = \{\bar{u^t}_w, \bar{u^t}_z\}$ and we only care about the part that corresponds to w, $\bar{u^t}_z$ is maintained implicitly by storing a vector of size $|\mathcal{W}|$ (although we now need to store s_w^t). It can be reconstructed using $\bar{u^t}_w$, therefore saving more memory cost.
Figure 1 illustrates the various dependencies.

2. From classical Frank-Wolfe to more sophisticated variants

2.1. Classical Frank-Wolfe

Consider the problem of minimizing a convex objective function f over the convex hull of its domain $\mathcal{M} = conv(\mathcal{A})$.
By minimizing the first-order approximation of the objective function, the Frank-Wolfe algorithms takes a convex combination of the immediate iterate with the previous one. Consider a linear minimization oracle,

$$LMO_{\mathcal{A}}(\nabla f(x_t)) \in \textit{argmin}_{s\prime\in\mathcal{A}}\langle s\prime, \nabla f(x_t)\rangle$$

Starting with an active set of an initial feasible point $S^0 = \{x_0\}$, the Frank-Wolfe algorithm adds an "atom" $s_t = LMO_{\mathcal{A}}(\nabla f(x_t))$, to the active set in a convex combination with its elements while maintaining this combination sparse.

2.1.1. CONVERGENCE RESULTS

We define the duality gap

$$g(\alpha^k) = \max_{s\in\mathcal{M}}\langle \alpha^k - s, \nabla f(\alpha^k)\rangle$$

By first order convexity of the objective, we have

$$f(s) \geq f(\alpha^k) + \langle \alpha^k - s, \nabla f(\alpha^k)\rangle$$
$$\implies g(\alpha^k) = -\min_{s\in\mathcal{M}}\langle \alpha^k - s, \nabla f(\alpha^k)\rangle \geq f(\alpha^k) - f^*$$

We can thus see that the duality gap gives us a computable optimality guarantee.

Definition. The curvature constant C_f is given by the maximum relative deviation of the objective function f from its linear aaporximations, over the domain $\mathcal{M}$,

$$C_f = \sup_{\substack{x,s\in\mathcal{M}\\ \gamma\in[0,1], y=x+\gamma(s-x)}} \frac{2}{\gamma^2}\Big(f(y) - f(x) - \langle y - x, \nabla f(x)\rangle\Big)$$

Intuitively, the curvature constant can be seen as a measure of how flat the objective function is. For example, if the objective is linear, say $f(x) = ax + b$ and $x \in [e, f]$ then $\nabla f(x) = a$ and the curvature constant is zero:

$$C_f = \frac{2}{\gamma^2}\Big(ay + b - ax - b + (-ay + ax)\Big) = 0$$

Moreover $s = \textit{argmin}_{s\in[e,f]}\langle s, a\rangle = \frac{e}{a}$. Hence we reach the minimum in one F-W step.
Thus, we can observe that for flatter functions, that is with smaller curvature constants, Frank-Wolfe should converge faster.

Theorem. The duality gap obtained in the t^{th} iteraton of the Frank-Wolfe algorithm satisfies

$$g(x_t) \leq 2\beta\frac{C_f}{t+2}(1+\delta)$$

Where $\beta = \frac{27}{8}$ and δ is the approximation error tolerated in the LMO.

Definition. A function f has Lipschitz continuous gradient if:

$$\forall x, y \in dom(f), \exists L > 0 \quad \textit{such that}$$
$$||\nabla f(x) - \nabla f(y)||^2 \leq L||x - y||^2$$

Theorem. If a convex function f on C has Lipschitz gradient, i.e $||\nabla f(x) - \nabla f(y)||_p \leq L_q||x - y||_p, \quad \forall x, y \in C$, then

$$C_f \leq L_q.\text{diam}_p^2(C)$$

Proof. f has Lipschitz gradient therefore by the fundamental descent lemma we have,

$$f(y) - f(x) - \langle y - x, \nabla f(x)\rangle \leq \frac{L_q}{2}||y - x||_p^2$$
$$C_f \leq \max_{\substack{y=(1-\gamma)x+\gamma s \\ x,s,y\in C}} \frac{2}{\gamma^2}\frac{L_q}{2}\underbrace{||y - x||_p^2}_{=\gamma^2||x-s||_p^2}$$
$$C_f \leq L_q \underbrace{\max_{x,s\in C}||x - s||_p^2}_{\triangleq diam_p^2(C)} \qquad \square.$$

Therefore, assuming $\delta = 0$, we get the following optimality certificate

$$f(\alpha^k) - f^* \leq g(\alpha^k) \leq 2\beta\frac{C_f}{t+2} \leq 2\beta\frac{L_q.\text{diam}_p^2(C)}{t+2}$$

Thus, we see that the Frank-Wolfe algorithm has a sublinear convergence rate.

2.1.2. Optimality in terms of sparsity of the iterates

Lemma. For $f(x) = ||x||_2^2$ and $1 \leq k \leq n$, it holds that

$$\min_{\substack{x\in_n \\ card(x)\leq k}} f(x) = \frac{1}{k}, \quad \textit{and}$$
$$g(x) \geq \frac{2}{k} \quad \forall x \in \Delta_n \quad \textit{s.t.} \quad card(x) \leq k.$$

By the first equality we have, for any vector x s.t. $card(x) = k$, we get $g(x) \leq \frac{1}{k} - \frac{1}{n}$. Thus, combining the upper and lower bound, we have that the sparsity (number of used atoms) by the Frank-Wolfe algorithm is worst case optimal.

2.2. Away steps Frank-Wolfe

When the minimizer of the objective function lies at the boundary of the domain, after a number of iterations, the duality gap starts to stagnate.
As a result of the strong dependency of the immediate iterate on previously accumulated atoms in the active set, as it approaches to boundary, the F-W algorithm starts to zig-zag around the descent direction as can be seen in figure 2.

Algorithm 2 Frank-Wolfe

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

Algorithm 3 Away-steps Frank-Wolfe

Let $x_0 \in \mathcal{A}$ and $S_0 := \{x_0\}$
for $t = 0...T$ **do**
  Let $s_t := LMO_{\mathcal{A}}(\nabla f(x_t))$ and $d_t^{FW} := s_t - x_t$
  Let $v_t \in \underset{v\in S_t}{\textit{argmax}}\langle \nabla f(x_t), v\rangle$ and $d_t^A := x_t - v_t$
  if $g_t^{FW} = \langle -\nabla f(x_t), d_t^{FW}\rangle \leq \epsilon$ **then return** x_t
  if $\langle -\nabla f(x_t), d_t^{FW}\rangle \geq \langle -\nabla f(x_t), d_t^A\rangle$ **then**
    $d_t = d_t^{FW}$ and $\gamma^{max} = 1$
  else
    $d_t = d_t^A$ and $\gamma^{max} = \frac{\alpha_{v_t}^{(t)}}{1-\alpha_{v_t}^{(t)}}$
  Line-search: $\gamma_t \in \underset{\gamma}{argmin} f(x_t + \gamma d_t)$
  Update $x_t = x_t + \gamma_t d_t$
end for

To address this issue an improved variant of F-W named **Away-steps Frank-Wolfe** adds the possibility of moving away (by removing a fraction of) a maximizer of the $LMO_{\mathcal{S}_\sqcup}$ in the active set. While this slows down each iteration it should be noted that the added step is easier than $LMO_{\mathcal{A}}$ given that we maximize over a subset of $\mathcal{A}$. Furthermore, given that this variant converges linearly, the algorithm progresses in a fewer number of iterations in the descent direction, making it much faster than the original F-W.

2.3. Block Coordinate Frank Wolfe

2.3.1. Structured SVM context

Given a training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ where $y \in \mathcal{Y}$ is a multi-label output, and a feature map $\phi : \mathcal{X} \times \mathcal{Y} \longrightarrow \mathbb{R}$,

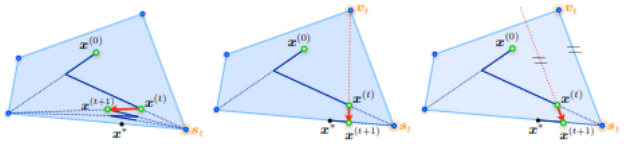

Figure 2. (left) The FW algorithm zig-zags when the solution x lies on the boundary. (middle) Adding the possibility of an away step attenuates this problem. (right) As an alternative, a pairwise FW step.

which encodes a similarity measure between $\mathcal{X}$ and $\mathcal{Y}$, such that if y_i is the ground truth (target) for an input x_i, then

$$\forall y \in \mathcal{Y} \backslash \{y_i\} \quad \textit{we have} \quad \psi_i(y) = \phi(x_i, y_i) - \phi(x_i, y) > 0$$

The aim is to construct an accurate linear classifyer, $h_w(x) = \underset{y \in \mathcal{Y}(x)}{\textit{argmax}} \langle w, \phi(x, y) \rangle$.
To learn w, consider the task loss $L : \mathcal{Y} \times \mathcal{Y} \longrightarrow \mathbb{R}_+$, where $L(y, y\prime) = 0 \Longleftrightarrow y = y\prime$.
The n-slack formulation of the problem would be,

$$\begin{aligned} &\underset{w,\xi}{\textit{max}} \quad \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{i=1}^{n} \varepsilon_i \\ &\textit{s.t.} \quad \langle w, \psi_i(y) \rangle \geq L(y_i, y) - \varepsilon_i, \quad \forall i, \forall y \in \mathcal{Y}(x) = \mathcal{Y}_i \end{aligned}$$

Problems: (1) The zero-one loss is not differentiable and (2) we have an exponential number of constraints.
Solutions: (1) Minimizing an upper bound to the task loss gives us a worst case guarantee.

Consider the **max oracle**, $\tilde{H} = \underset{y \in \mathcal{Y}_i}{\textit{max}} \underbrace{L_i(y) - \langle w, \psi_i(y) \rangle}_{=H_i(y,w) \quad \textit{the hinge loss}}$.
(2) The exponential number of constraints are replaced by n piecewise linear ones.

Proposition. The max oracle is a convex upper bound to the task loss.
Proof. The maximum of two convex (linear) functions is convex, and

$$\begin{aligned} L(y_i, h_w(x_i)) &\leq L(y_i, h_w(x_i)) + \underbrace{\langle w, \psi_i(y) \rangle}_{\geq 0 \textit{ by definition}} \\ &\leq \underset{y \in \mathcal{Y}_i}{\textit{max}} L_i(y) - \langle w, \psi_i(y) \rangle \end{aligned}$$

Thus learning w amounts to the unconstrained problem,

$$\underset{w}{\textit{max}} \quad \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{i=1}^{n} \tilde{H}_i(w)$$

2.3.2. BCFW VARIANT IN THE STRUCTURED SVM SETTING

Due to the exponential number of dual variables in the structured SVM setting, classical algorithms, like projected gradient are intractable.
Stochastic subgradient methods, on the other hand, achieve a sublinear convergence rate while only requiring a single call to the maximization oracle every step. They are nonetheless very sensitive to the sequence of stepsizes and it is unclear when to terminate the iterations.

Frank-Wolfe methods address these problems by giving an adaptive stepsize $\gamma = \frac{2}{k+2}$ and a computable duality gap while still retaining a sublinear convergence rate. Moreover, despite the exponential number of constraints, the algorithm has sparse iterates alleviating the memory issues which come with the exponential number of dual variables.

Algorithm 4 Batch Primal-Dual Frank-Wolfe

Let $\alpha \in \mathcal{M}$
Let $w^0 = 0, l^0 = 0$
for $k = 0, \ldots, K$ **do**
  for $i = 1, \ldots, n$ **do**
    Solve $y_i^* = \underset{y_i \in \mathcal{Y}_i}{\max} H_i(y, w^k)$//
    Let $w_s = \sum_{i=1}^{n} \frac{1}{n\lambda}\psi_i(y_i^*)$, and $l_s = \frac{1}{n}\sum_{i=1}^{n} L_i(y_i^*)$
    Let $\gamma = \frac{\lambda(w^k - w_s)^T w^k - l^k + l_s}{\lambda||w^k - w_s||^2}$, and clip to $[0, 1]$
    Update $w^{k+1} = (1-\gamma)w^k + \gamma w_s$, and $l^{k+1} = (1-\gamma)l^k + \gamma l_s$
  end for
end for

Note. The main idea here, is that the linear subproblem in Frank-Wolfe and the loss augmented decoding of the structured SVM are equivalent.

Proof of the equivalence. The objective function being differentiable and convex, if we are at a point α such that $f(\alpha)$ is minimized along each coordinate axis, then α is a global minimizer. Therefore,

$$\underset{s \in \mathcal{M}}{\textit{min}} \langle s, \nabla f(\alpha) \rangle = \sum_i \underset{s_i \in \Delta_{|\mathcal{Y}_i|}}{\textit{min}} \langle s_i, \nabla_i f(\alpha) \rangle$$

Moreover, with

$$\begin{aligned} &w = A\alpha, A = \Big[\frac{1}{n\lambda}\psi_1(y) ... \frac{1}{n\lambda}\psi_{\sum_i |\mathcal{Y}_i|}(y)\Big] \\ &\textit{and} \quad b = \Big(\frac{1}{n}L_i(y)\Big)_{i \in [n], y \in \mathcal{Y}_i} \end{aligned}$$

The gradient of the dual would be,

$$\begin{aligned} &\nabla f(\alpha) = \nabla\Big[\frac{\lambda}{2}||A\alpha||^2 - b^T\alpha\Big] = \lambda A^T A\alpha - b \\ &= \lambda A^T w - b = \frac{1}{n}H_i(y, w) \\ &\underset{y_i \in \mathcal{Y}_i}{\textit{max}} \quad \tilde{H}_i = -\underset{y_i \in \mathcal{Y}_i}{\textit{min}} \quad \tilde{H}_i = \underset{y_i \in \mathcal{Y}_i}{\textit{min}} \quad L_i - \langle w, \psi_i \rangle \\ &= \underset{s_i \in \Delta_{|\mathcal{Y}_i|}}{\textit{min}} \langle s_i, \nabla_i f(\alpha) \rangle \end{aligned}$$

Thus we can see that, if $n =$ size of the training data, one Frank-Wolfe step is equivalent to n calls to the maximization oracle. Unlike stochastic subgradient and stochastic methods in general, classical Frank-Wolfe requires one

Algorithm 5 Block-Coordinate Frank-Wolfe

Let $w^0 = w_i^0 = \overline{w}^0 = 0, l^0 = l_i^0 = 0$
for $k = 0...K$ **do**
  Pick i at random in $\{1, ..., n\}$
  Solve $y_i^* = \underset{y_i \in \mathcal{Y}_i}{max} \quad H_i(y, w^k)$
  Let $w_s = \frac{1}{n\lambda}\psi_i(y_i^*)$, and $l_s = \frac{1}{n}L_i(y_i^*)$
  Let $\gamma = \frac{\lambda(w_i^k - w_s)^T w^k - l_i^k + l_s}{\lambda ||w_i^k - w_s||^2}$, and clip to $[0, 1]$
  Update $w_i^{k+1} = (1-\gamma)w_i^k + \gamma w_s$, and $l_i^{k+1} = (1-\gamma)l_i^k + \gamma l_s$
  Update $w^{k+1} = w^k + w_i^{k+1} - w_i^k$, and $l_i^{k+1} = (1-\gamma)l_i^k + \gamma l_s$
end for

call for each training example at each iteration. For large datasets, this can get unpractical.
Hence the stochastic variant of Frank Wolfe, **Block Coordinate Frank Wolfe (BCFW)**.

Theorem. Given a convex, differentiable objective $f : \mathcal{M}^1 \times ... \times \mathcal{M}^n \to \mathbb{R}$, where $\forall i \in \{1..n\}$, each factor $\mathcal{M}^i \subseteq \mathbb{R}^n$ is convex and compact, if we are at a point x such that $f(x)$ is minimized along each coordinate axis, then x is a global minimum.

As in coordinate descent, we minimize the objective function one coordinate (block) at a time. At each iteration, BCFW picks the i^{th} block (from n) uniformly at random and updates the i^{th} coordinate of the corresponding weight, by calling the maximization oracle on the chosen block.

2.3.3. Convergence Results

Definition. Over each coordinate block $\mathcal{M}^i$, let the curvature be given by,

$$C_f^{(i)} = \underset{\substack{x \in \mathcal{M}, s_i \in \mathcal{M}^i \\ y = x + \gamma(s_{[i]} - x_{[i]}) \\ \gamma \in [0,1]}}{sup} \frac{2}{\gamma^2}\Big(f(y) - f(x) - \langle y_i - x_i, \nabla_i f(x)\rangle\Big)$$

Where $x_{[i]}$ refers to the zero-padding of i^{th} coordinate of x. And let the global *product curvature constant* be,

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

Theorem. For the dual structural SVM objective function over the domain $\mathcal{M} = \Delta_{|\mathcal{Y}_1|} \times ... \times \Delta_{|\mathcal{Y}_n|}$, the total curvature constant $C_f^{\otimes}$, on the product domain $\mathcal{M}$, is upper bounded by,

$$C_f^{\otimes} \leq \frac{4R^2}{\lambda n} \quad \textit{where} \quad R = \underset{i \in [n], y \in \mathcal{Y}_i}{max} ||\psi_i(y)||_2$$

Proof. By the second order convexity condition on f at y, we have

$$\begin{aligned} f(y) &\leq f(x) + \langle y_i - x_i, \nabla_i f(x)\rangle \\ &\quad + (y-x)^T \nabla^2 f(x)(y-x) \\ f(y) - f(x) - \langle y_i - x_i, \nabla_i f(x)\rangle &\\ &\leq (y-x)^T \nabla^2 f(x)(y-x) \end{aligned}$$

$$C_f^{(i)} \leq \underset{\substack{x \in \mathcal{M}, s_i \in \mathcal{M}^i \\ y = x + \gamma(s_{[i]} - x_{[i]}) \\ \gamma \in [0,1]}}{sup} \Big(f(y) - f(x) - \langle y_i - x_i, \nabla_i f(x)\rangle\Big)$$

$$\leq \underset{\substack{x,y \in \mathcal{M}, (y-x) \in \mathcal{M}^{[i]} \\ z \in [x,y] \subseteq \mathcal{M}}}{sup} (y-x)^T \nabla^2 f(z)(y-x)$$

Moreover
$$\underset{\substack{x,y \in \mathcal{M}, (y-x) \in \mathcal{M}^{[i]} \\ z \in [x,y] \subseteq \mathcal{M}}}{sup} (y-x)^T \nabla^2 f(z)(y-x)$$

$$= \lambda \underset{x,y \in \mathcal{M}, (y-x) \in \mathcal{M}^{[i]}}{sup} (A(y-x))^T \nabla^2 f(z)(A(y-x))$$

$$C_f^{(i)} \leq \lambda \underset{v,w \in A\mathcal{M}^{(i)}}{sup} ||v - w||_2^2 \leq \lambda \underset{v \in A\mathcal{M}^{(i)}}{sup} ||2v||_2^2$$

Where $\forall v \in A\mathcal{M}^{(i)}$, v is a convex combination of the feature vectors corresponding to the possible labelings for the i^{th} example of the training data, such that $||v||_2 \leq$ *the longest column of* $A = \frac{1}{n\lambda}R$. Therefore,

$$C_f^{\otimes} = \sum_{i=1}^{n} C_f^{(i)} \leq 4\lambda \sum_{i=1}^{n} \Big(\frac{1}{n\lambda}R\Big)^2 = \frac{4}{n\lambda}R^2 \quad \square$$

First, we observe that the curvature constant for BCFW is n times smaller than that of batch Frank Wolfe which is $\leq \frac{4}{\lambda}R^2$. Hence the n times faster convergence rate of BCFW.

2.3.4. Tightening the bound

Definition. Let $||.||$ be a norm on $\mathbb{R}^n$. The associated dual norm, denoted $||.||_*$ is defined as,

$$||z||_* = \underset{z}{sup}\{z^T x | \quad ||x|| \leq 1\}$$

We denote the dual norm of l_p by l_q. For $p = 2$ we have $q = 2$ and for $p = 1$, $q = \infty$. *Problem.* For $p = q = 2$ we get $diam_2^2(C) = 2n$, and the Lipschitz constant L_q is the largest eigenvalue of the hessian.

$$\lambda A^T A = \frac{1}{n^2\lambda}\Big(\langle \psi_i(y) - \psi_j(y\prime)\rangle\Big)_{(i,y),(j,y\prime)}$$

And say $\langle \psi_i(y) - \psi_j(y\prime)\rangle \approx 1$ *for a lot of outputs, we get:*

$$1^T 1 \approx 1 \underbrace{diam_2(C)}_{=\sqrt{2n}}$$

Hence the largest eigenvalue the hessian, and therefore the Lipschitz constant, can scale with the dimension of

$A^T A$, i.e exponentially with the size of the training data, rendering the bound above very loose, and thus of little practical use.

Solution. Taking $p = 1$ and therefore $q = \infty$, we get $Ldiam^2(C) \approx \frac{4}{\lambda} R^2$.

Combined with the the convergence results above, we get a sublinear convergence rate for BCFW. And although subgradient methods converge at the same rate, BCFW presents an adaptive stepsize and an indication as to when to terminate, making it a more practical alternative.

3. Randomized Away-step Frank-Wolfe

A crucial assumption in constructing the BCFW is whether the domain is block-separable. While this is true in the context of the structured SVM, this leaves out important cases such as l_1 constrained optimization (e.g. lasso type problems).
Moreover, while being an improvement on the classical variant by being $n =$ *size of the data* times cheaper per iteration, BCFW still converges at a sublinear rate unlike the Away-step FW.

The Randomized Away-steps Frank-Wolfe (RAWF) finds a compromise between the two variants. By subsampling a $\eta \in (0, 1]$ portion of the domain $\mathcal{A}$ in the LMO and adding an away step at each iteration, we get a linear convergence rate with cheaper oracle calls than that of the original F-W.

3.0.1. Convergence results

Definition. Let the *away curvature* C_f^A and the *geometric strong convexity* constants be, respectively

$$C_f^A = \sup_{\substack{x,s,v \in \mathcal{M} \\ y = x + \gamma(s-x) \\ \gamma \in [0,1]}} \frac{2}{\gamma^2} \Big(f(y) - f(x) - \gamma \langle \nabla f(x), s - v \rangle \Big)$$

$$\mu_f^A = \inf_{x \in M} \inf_{\substack{x^* \in \mathcal{M} \\ \langle \nabla f(x), x^* - x \rangle < 0}} \frac{2}{\gamma^A(x, x^*)^2} B_f(x, x^*)$$

$$where \quad \gamma^A(x, x^*) = \frac{\langle -\nabla f(x), x^* - x \rangle}{\langle -\nabla f(x), s_f(x) - v_f(x) \rangle}$$

And $s_f, v_f(x)$ are the FW atom and away atom respectively, starting from x.

Theorem. Consider the set $\mathcal{M} = conv(\mathcal{A})$, with $\mathcal{A}$ a finite set of extreme atoms, adter T iterations of RAFW, we have the following convergence rate

$$E\big[f(x_{T+1})\big] - f^* \leq \Big(f(x_0) - f^* \Big) . \Big(1 - \eta^2 \rho_f \Big)^{max\{0, \lfloor \frac{T-s}{2} \rfloor\}}$$

Algorithm 6 Randomized Away-steps Frank-Wolfe

Let $x_0 = \sum_{v \in \mathcal{A}} \alpha_v^{(0)}$ with $s = |S_0|$, a subsampling parameter $1 \leq p \leq |\mathcal{A}|$.
for $t = 0 ... T$ **do**
  Get $\mathcal{A}_t$ by sampling $min\{p, |\mathcal{A} \setminus S_t|\}$ elements uniformly from $|\mathcal{A} \setminus S_t|$
  Compute $s_t = LMO(\nabla f(x), S_t \bigcup \mathcal{A}_t)$
  Let $d_t^{FW} = s_t - x_t$ **RFW step**
  Compute $v_t = LMO(-\nabla f(x), S_t)$
  Let $d_t^A = x_t - v_t$ **Away step**
  if $\langle -\nabla f(x_t), d_t^{FW} \rangle \geq \langle -\nabla f(x_t), d_t^A \rangle$ **then**
    $d_t = d_t^{FW}$ and $\gamma^{max} = 1$
  else
    $d_t = d_t^A$ and $\gamma^{max} = \frac{\alpha_{v_t}^{(t)}}{1 - \alpha_{v_t}^{(t)}}$
  Let $x_{t+1} = x_t + \gamma_t d_t$
  Let $S_{t+1} = \{v \in \mathcal{A} \quad s.t. \quad \alpha_{v_t}^{(t)} > 0\}$
end for

With $\rho_f = \frac{\mu_f^A}{4 C_f^A}, \quad \eta \frac{p}{|\mathcal{A}|} \quad and \quad s = |S_0|$.

Proof sketch. First we upper-bound $h_t = f(x_t) - f^*$ by the pairwise dual gap $\tilde{g}_t = \langle \tilde{s}_t - v_t \rangle$, then we lower bound the progress $h_t - h_{t+1}$ by using the away curvature constant in similar way to the proof in (Lacoste-Julien Jaggi, 2015, Theorem 8).□

With the above theorem, we get

$$\lim_{t \to \infty} \frac{Ef(x_{t+1}) - f^*}{Ef(x_t) - f^*} \in \big(0, 1\big)$$

Thus proving a linear convergence rate for the Randomized Away-steps Frank-Wolfe.