This lecture covers screening rules and conditional gradient method.

## 23.1 Screening Rules

### Screening Rules

Screening rules try to determine the structure of the solution before actually solving the optimization problem. There are mainly two kinds of screening rules. One is for sparse problem, where screening rules try to determine which features will be inactive for the solution and throw out some features before even solving the problem. Another kind of screening rules tries to throw instances that will not affect the optimal solution, e.g., the instances that will not be support points in the SVM problem.

In some problems, screening rules can be used in combination with coordinate descent to further whittle down the active set. Screening rules themselves have amassed a sizeable literature recently. Here is an example, the SAFE rule to the lasso:\(^1\):

$$|X_i^T y| < \lambda - \|X_i\|_2 \|y\|_2 \frac{\lambda_{\text{max}} - \lambda}{\lambda_{\text{max}}} \quad \Rightarrow \quad \hat{\beta}_i = 0, \forall i = 1, \ldots, p$$

where \(\lambda_{\text{max}} = \|X^T y\|_\infty\), which is the smallest value of \(\lambda\) such that \(\hat{\beta} = 0\), and this can be checked by the KKT condition of the dual problem. The SAFE rule suggests that we can loop through each feature \(i\), and check it with the above rule. If the above inequality satisfies, we can discard \(X_i\) from the dataset, and solve a smaller lasso problem. This is called "SAFE" rule because this is deterministic, and if this happens, \(\hat{\beta}_i\) will always equal to 0.

Note that the SAFE rule is sufficient but not necessary. But it does give us a way of eliminating features apriori, without solving the lasso.

Why is the SAFE rule true? The construction of the SAFE rule comes from the lasso dual:

$$\max_{u \in \mathbb{R}^n} g(u) \quad \text{subject to} \quad \|X^T u\|_\infty \leq \lambda$$

where \(g(u) = \frac{1}{2} \|y\|_2^2 - \frac{1}{2} \|y - u\|_2^2\). Suppose that we are able to find a dual feasible point \(u_0\) (e.g., take \(u_0 = y \cdot \lambda / \lambda_{\text{max}}\)). Then \(\gamma = g(u_0)\) is a lower bound on the dual optimal value. Note that if we can find a feasible point close to optimality, we will get a better screening rule. If we have a feasible point far from optimality, we will have a loose screening rule. However, any feasible point will suffice. With the feasible point \(u_0\), the dual problem is equivalent to:

$$\max_{u \in \mathbb{R}^n} g(u) \quad \text{subject to} \quad \|X^T u\|_\infty \leq \lambda, g(u) \geq \gamma$$

\(^1\)El Ghaoui et al. (2010), "Safe feature elimination in sparse learning"
Now consider computing:

$$m_i = \max_{u \in \mathbb{R}^n} |X_i^T u| \quad \text{subject to} \quad g(u) \geq \gamma, \quad i = 1, \ldots, p$$

Then we would have:

$$m_i < \lambda \Rightarrow |X_i^T \hat{u}| < \lambda \Rightarrow \hat{\beta}_i = 0, \quad i = 1, \ldots, p$$

where the last implication comes from the KKT stationarity conditions of the primal problem, i.e. $\lambda \partial |\beta|_1 = X^T u$. The idea of the equations above is that suppose that we have $m_i < \lambda$, then we can guarantee that for all $u$ that satisfies the constraints of the equivalent dual problem 23.1, we have $|X_i^T u| < \lambda$, and further have $|X_i^T \hat{u}| < \lambda$. By the KKT stationarity condition of the primal problem, i.e., $\lambda \partial |\beta|_1 = X^T u$, we have $\hat{\beta}_i = 0$ from $|X_i^T \hat{u}| < \lambda$.

Figure 23.1 illustrates the basic idea of the SAFE rule. The feasible set of the dual problem is the shaded polytope. Two level sets of the dual function are shown, one corresponds to the optimal value and the other to the given lower bound $\gamma$. Constraints (A) and (B) (in green) are safely eliminated, but some inactive constraints (in red) are not. Here, $\theta_0$ corresponds to the unconstrained maximum of the dual function.

Another dual argument shows that:

$$\max_{u \in \mathbb{R}^n} X_i^T u \quad \text{subject to} \quad g(u) \geq \gamma$$

$$= \min_{\mu > 0} -\gamma \mu + \frac{1}{\mu} \|\mu y - X_i\|^2$$

$$= \|X_i\|_2 \sqrt{\|y\|_2^2 - 2\gamma - X_i^T y}$$

where the last equality comes from directly computing the optimal $\mu^*$ from the second equation and then plugging back.

Thus $m_i$ is given the maximum of the above quantity over $\pm X_i$,

$$m_i = \|X_i\|_2 \sqrt{\|y\|_2^2 - 2\gamma + |X_i^T y|}, \quad i = 1, \ldots, p$$
Lastly, substitute \( \gamma = g(y \cdot \lambda / \lambda_{\text{max}}) \). Then \( m_i < \lambda \) is precisely the SAFE rule given previously.

### 23.2 Conditional Gradient Method Basic

Consider the constrained problem as follows

\[
\min_x f(x) \quad \text{subject to} \quad x \in C
\]

Here, \( f \) is convex and smooth, and \( C \) is convex. Recall that projected gradient descent is the only first order method that can be used to solve this constrained optimization problem. Projected gradient descent chooses an initial \( x^{(0)} \), repeats for \( k = 1, 2, 3 \ldots \)

\[
x^{(k)} = P_C(x^{(k-1)} - t_k \nabla f(x^{(k-1)}))
\]

where \( P_C \) is the projection operator onto the set \( C \). This was a special case of proximal gradient descent, motivated by a local quadratic expansion of \( f \):

\[
x^{(k)} = P_C \left( \arg \min_y \nabla f(x^{(k-1)})^T (y - x^{(k-1)}) + \frac{1}{2t} \| y - x^{(k-1)} \|^2 \right)
\]

The conditional gradient method, also known as the Frank-Wolfe method, uses a local linear expansion of \( f \). Write down the local linear expansion \( \hat{f}^{\text{lin}}(s) \) as follows:

\[
\hat{f}^{\text{lin}}(s) = f(x^{(k-1)}) + \nabla f(x^{(k-1)})^T (s - x^{(k-1)})
\]

This local linear expansion makes no sense for unconstrained optimization problem (term associated with \( s \) is linear and it leads \( \hat{f}^{\text{lin}}(s) \) to minus infinity); similarly, one also could not take a local linearization and project it onto the constrained sets for the same reason. Thus, Frank-Wolfe minimizes \( \hat{f}^{\text{lin}}(s) \) over all \( s \in C \).

\[
s^{(k-1)} = \arg \min_{s \in C} \hat{f}^{\text{lin}}(s)
\]

Then it moves a bit towards this minimizer, i.e. \( x^{(k)} = x^{(k-1)} + \gamma_k (s^{(k-1)} - x^{(k-1)}) \) with \( \gamma_k = \frac{2}{k+1}, k = 1, 2, 3. \)

There is no projection; update is solved directly over the constraint set \( C \). This process can be formally written as:

\[
s^{(k-1)} \in \arg \min_{s \in C} \nabla f(x^{(k-1)})^T s
\]

\[
x^{(k)} = (1 - \gamma_k) x^{(k-1)} + \gamma_k s^{(k-1)}
\]

The default choice for step sizes is \( \gamma_k = \frac{2}{k+1}, k = 1, 2, 3 \ldots \). For any choice \( 0 \leq \gamma_k \leq 1 \), we see that \( x^{(k)} \in C \) by convexity. This algorithm is described in Algorithm 1.

The intuition behind this is, we are moving less and less in the direction of the linearization minimizer as the algorithm proceeds. As shown in Figure 23.2, at a current position \( x \), the algorithm considers the linearization of the objective function, and moves towards a minimizer of this linear function.
**Algorithm 1:** Conditional Gradient Descent

Let \( x^{(0)} \in C \) for \( k = 1, \ldots, K \) do
- compute \( s^{(k-1)} \in \arg \min_{s \in C} \nabla f(x^{(k-1)})^T s \)
- update \( x^{(k)} = (1 - \gamma_k)x^{(k-1)} + \gamma_k s^{(k-1)} \), for \( \gamma_k = \frac{2}{k+2} \)

What happens when \( C = x : \|x\| \leq t \) for a norm \( \|\cdot\| \)? We could write the minimization of \( s \) as follows:

\[
\begin{align*}
    s & \in \arg \min_{\|s\| \leq t} \nabla f(x^{(k-1)})^T s \\
    & = -t \cdot (\arg \min_{\|s\| \leq 1} \nabla f(x^{(k-1)})^T s) \\
    & = -t \cdot \partial \|\nabla f(x^{(k-1)})\|_*
\end{align*}
\]

(Definition of Dual Norm)

Here, \( \|\cdot\|_* \) is the corresponding dual norm and \( \arg \min_{\|s\| \leq 1} \nabla f(x^{(k-1)})^T s \) is exactly the set of subgradients. In other words, if we know how to compute subgradients of the dual norm, then we can easily perform the Frank-Wolfe steps. It is worth mentioning that, the minimization of \( s \) has nothing to do with \( \nabla f(x^{(k-1)})^T \).

For Frank-Wolfe, it can often be simpler or cheaper than projection onto \( C = x : \|x\| \leq t \). It is often simpler or cheaper than the prox operator for \( \|\cdot\| \).

To sum up, conditional gradient method is an iterative first-order optimization algorithm for constrained convex optimization problems. It considers a linear approximation of the objective function, and moves less and less in the direction of linearization minimizer in each step.

### 23.3 Conditional Gradient Method Examples

In this section, we will concretize into some cases including regularization of \( l_1 \) norm, \( l_p \) norm and trace norm.

1. **\( l_1 \) regularization**

   \( l_1 \) norm is a quantity of interest since it gives out sparse solutions. The problem is expressed as:

   \[
   \min_x f(x) \quad \text{subject to} \quad \|x\|_1 \leq t
   \]
According to the previous section, all we need to compute Frank-Wolfe update is to look at the dual norm of $l_1$ norm, which is the infinity norm. So we have $s^{(k-1)} \in -t \partial \|\nabla f(x^{(k-1)})\|_{\infty}$. The problem now becomes how to compute the subgradient of $l_{\infty}$ norm.

Recall that for a $p$-dimensional vector $a$, $a_{\infty} = \max_i \{|a_1|, |a_2|, \ldots, |a_p|\}$, if the $i_{th}$ element is the largest, namely $|a_i| = \|a\|_{\infty}$, then $\text{sign}(a_i) \cdot e_i \in \partial \|a\|_{\infty}$, where $e_i$ is the unit vector, with all elements being 0 except the $i_{th}$ element being 1.

So the Frank-Wolfe update could be computed as:

$$i_{k-1} = \arg \max_{i=1,\ldots,p} |\nabla_i f(x^{(k-1)})|$$

$$x^k = (1 - \gamma_k)x^{k-1} - \gamma_k t \text{sign}(\nabla_{i_{k-1}} f(x^{k-1})) e_{i_{k-1}}$$

We can see that it looks very much like the greedy coordinate descent. Look at which coordinate direction will cause the biggest improvement and move along it.

Note that this is very cheap operation, just scanning through, finding the maximum and updating just one coordinate. This is much simpler than projection onto the $l_1$ ball, though both require $O(n)$ operations.

2. $l_p$ regularization

The problem is a generalization of $l_1$ norm:

$$\min_x f(x) \text{ subject to } \|x\|_p \leq t$$

for $1 \leq p \leq \infty$. We have $s^{(k-1)} \in -t \partial \|\nabla f(x^{(k-1)})\|_q$ where $p, q$ are dual, i.e., $1/p + 1/q = 1$. We can choose:

$$s_i^{k-1} = -\alpha \text{sign}(\nabla f_i(x^{k-1})) |\nabla f_i(x^{k-1})|^{p/q}, i = 1,\ldots,n$$

Where $\alpha$ is a constant such that $\|s^{k-1}\|_q = t$. Then Frank-Wolfe update is as usual. This is a lot simpler than projection onto the $l_p$ ball. Aside from special cases ($p = 1, 2, \infty$), these projections cannot be directly computed.

3. trace norm regularization

The problem is an optimization of matrix:

$$\min_X f(X) \text{ subject to } \|X\|_{tr} \leq t$$

Trace norm is the sum of all singular values. The dual of trace norm is operator norm, which is the largest singular value. So $S^{k-1} \in -t \|\nabla f(X^{k-1})\|_{op}$. After computing the subgradient of operator norm from previous lectures, we can choose

$$S^{k-1} = -t uv^T$$

where $u, v$ are the leading left, right singular vectors of $\nabla f(X^{k-1})$. If we take SVD, then $u, v$ is the first column or row of $U$ and $V$, depending on the writing way. Then Frank-Wolfe update is as usual. It is also a lot more efficient than projection onto the trace norm ball which requires the whole SVD decomposition but here we only need to compute the leading left, right singular vectors. There are some methods such as power method that could solve it efficiently.
23.4 Constrained and Lagrange forms

Recall that the solution of the constrained problem:

\[ \min_x f(x) \quad \text{subject to} \quad ||x|| \leq t \]

are equivalent to those of the Lagrange problem

\[ \min_x f(x) + \lambda ||x|| \]

as we let the tuning parameters \( t \) and \( \lambda \) vary over \([0, \infty]\). Typically in statistics and ML problems, we would just solve whichever form is the easiest, over wide range of parameter values. So we should also compare the Frank-Wolfe updates under \( ||\cdot|| \) to the proximal operator of \( ||\cdot|| \).

- \( l_1 \) norm: Frank-Wolfe update scans for maximum of gradient; proximal operator soft-thresholds the gradient step. They both use \( O(n) \) flops. They didn’t make big difference but Frank-Wolfe is much straight forward while proximal operator needs special algorithm.

- \( l_p \) norm: Frank-Wolfe update computes raises each entry of gradient to power and sums, in \( O(n) \) flops; proximal operator not generally directly computable. They make a big difference.

- Trace norm: Frank-Wolfe update computes top left and right singular vectors of gradient; proximal operator soft-thresholds the gradient step, requiring a singular value decomposition.

Many other regularizers yield efficient Frank-Wolfe updates, e.g., special polyhedra or cone constraints, sum-of-norms (group-based) regularization, atomic norms. See Jaggi (2011).

Here, we make an actual comparison of projected and conditional gradient for constrained lasso problem, with \( n = 100 \) observations, \( p = 500 \) variables shown in Fig. 23.4.

So in practice, Frank-Wolfe has slower convergence than first order methods, even though it matches convergence rates of known first-order methods \( O(\frac{1}{k}) \); but in practice each iteration makes less progress in terms of converging to high accuracy. We also find from the graph Fig. 23.4 that conditional gradient is not necessarily a descent method. It could be seen in many examples. In practice, its convergence is between projected gradient method and subgradient method. Here, we fix step sizes here and line search would probably improve convergence.
23.5 Duality gap

Before we talk about the convergence rate, let’s talk about a very important property of Frank-Wolfe iterates. They admit a very natural duality gap, or actually a suboptimality gap.

Remember in Frank-Wolfe iterations, we do the following

\[
s = \arg\min_{s \in C} \nabla f(x)^T s = \arg\max_{s \in C} \nabla f(x)^T (x - s)
\]

as \(\nabla f(x)^T x\) is a constant in terms of \(s\).

If we ignore arg, then the max function is a very natural duality gap (truly, a suboptimality gap):

\[
\max_{s \in C} \nabla f(x_k - 1)^T (x_k - 1 - s)
\]

This is an upper bound on \(f(x_k - 1) - f^*\), namely \(f(x_k - 1) - f^* \leq \max_{s \in C} \nabla f(x_k - 1)^T (x_k - 1 - s)\). This is a very important property as we don’t need to specifically try to find the upper bound and we are computing it all along during Frank-Wolfe iterations.

We can prove this simply by two lines. By the first-order condition for convexity at the point \(x_k - 1\).

\[
f(s) \geq f(x_k - 1) + \nabla f(x_k - 1)^T (s - x_k - 1)
\]

Then minimizing both sides over all \(s \in C\) yields

\[
f^* \geq f(x_k - 1) + \min_{s \in C} \nabla f(x_k - 1)^T (s - x_k - 1)
\]

If we rearrange it, move \(\min_{s \in C} \nabla f(x_k - 1)^T (s - x_k - 1)\) part to the left and it becomes

\[
\max_{s \in C} \nabla f(x_k - 1)^T (x_k - 1 - s)
\]

So it is exactly the upper bound of \(f(x_k - 1) - f^*\) as mentioned before.

Note that:

\[
\max_{s \in C} \nabla f(x_k - 1)^T (x_k - 1 - s) = \nabla f(x_k - 1)^T (x_k - 1 - s_k - 1)
\]

So this quantity comes directly from the Frank-Wolfe update and we don’t even need to specifically compute something.

Then why do we call it a duality gap? This could be something we can see from high levels from our work on duality. Well if we rewrite original problem as an unconstrained problem:

\[
\min_x f(x) + I_C(x)
\]

where \(I_C\) is the indicator function of \(C\). The dual problem is

\[
\max_u -f^*(u) - I^*_C(-u)
\]

where \(I^*_C\) is the support function of \(C\), expressed as \(I^*_C(x) = \max_{y \in C} y^T x\). It is the dual of the indicator function. According to Fenchel inequality \(f(x) + f^*(u) \geq x^T u\) mentioned in previous lectures, duality gap at \(x, u\) is

\[
f(x) + f^*(u) + I^*_C(-u) \geq x^T u + I^*_C(-u)
\]

If we evaluate it at \(x = x_k - 1, u = \nabla f(x_k - 1)\), it gives the claimed gap.
23.6 Convergence Analysis

The convergence analysis of Frank-Wolfe type algorithms crucially relies on measure of non-linearity of $f$ over the domain $C$. Following Jaggi\(^2\), we define the curvature constant of $f$ over $C$ as follows:

$$M = \max_{x,s,y \in C, y = (1 - \gamma)x + \gamma s} \frac{2}{\gamma^2} (f(y) - f(x) - \nabla f(x)^T(y - x))$$

Here, we restrict $\gamma \in [0, 1]$. For linear functions $f$, it holds that $M = 0$. A motivation to consider this quantity is, assuming we are moving from a current point $x$ towards a next point $y = x + \gamma(s - x)$ for any relative step size $\gamma \in [0, 1]$. The quantity $f(y) - f(x) - \nabla f(x)^T(y - x)$ is also called the Bregman divergence defined by $f$:

**Theorem 23.1** Conditional gradient method using fixed step sizes $\gamma_k = \frac{2}{k + 1}$, $k = 1, 2, 3...$ satisfies

$$f(x^{(k)}) - f^* \leq \frac{2M}{k + 2}$$

Here the number of iterations needed to achieve $f(x^{(k)}) - f^* \leq \epsilon$ is $O(\frac{1}{\epsilon})$. This convergence rate matches the known rate for projected gradient descent when $\nabla f$ is Lipschitz. We present a comparison of convergence rate of Frank-Wolfe and projected gradient descent.

Frank-Wolfe \hspace{2cm} Projected Gradient Descent

$$\frac{2 \text{diam}^2(C) \cdot L}{k + 2} \hspace{2cm} \frac{\|x^{(0)} - x^*\|^2}{2k}$$

One interesting part to look at is how the assumptions compare with each other? In fact, if $\nabla f$ is Lipschitz with constant $L$ then $M \leq \text{diam}^2 C \cdot L$, where

$$\text{diam}(C) = \max_{x, s \in C} \|x - s\|_2$$

To see this, since $\nabla f$ is Lipschitz with constant $L$, we have

$$f(y) - f(x) - \nabla f(x)^T(y - x) \leq \frac{L}{2}\|y - x\|^2_2$$

Maximizing over all $y = (1 - \gamma)x + \gamma s$, and multiplying by $\frac{2}{\gamma^2}$, we get the bound.

$$M \leq \max_{x, s, y \in C, y = (1 - \gamma)x + \gamma s} \frac{\gamma^2}{2} \frac{L}{2}\|y - x\|^2_2 = \max_{x, s \in C} L\|x - s\|^2_2 \leq \text{diam}^2 C \cdot L$$

Essentially, assuming a bounded curvature is no stronger than what we assumed for proximal gradient.

The key inequality used to prove the Frank-Wolfe convergence rate is:

$$f(x^{(k)}) \leq f(x^{(k-1)}) - \gamma_k g(x^{(k-1)}) + \frac{\gamma_k^2}{2} M$$

\(^2\text{http://m8j.net/math/revisited-FW.pdf}\)
Here \( g(x) = \max_{s \in C} \nabla f(x)^T (x - s) \) is the duality gap discussed earlier. The proof of the convergence rate depends on this on the improvement in each iteration, expressing the improvement in terms of the current duality gap. The rate follows from this inequality, using induction as follows.

**Proof:** Denote \( x^+ = x^{(k)}, x = x^{(k-1)}, s = s^{(k-1)}, \gamma = \gamma_k \). Then

\[
 f(x^+) = f(x + \gamma(s - x)) \\
 \leq f(x) + \gamma \nabla f(x)^T (s - x) + \frac{\gamma^2}{2} M \\
 = f(x) - \gamma g(x) + \frac{\gamma^2}{2} M
\]

Second line used definition of \( M \). The third line the definition of \( g \).

One important property of Frank-Wolfe is, it is fully invariant under affine transformations, i.e. its updates are affine invariant. Newton methods are affine invariant but gradient methods are usually not. Given nonsingular transformation \( A : \mathbb{R}^n \to \mathbb{R}^n \), define \( x = Ax' \), \( h(x') = f(Ax') \). Then Frank-Wolfe on \( h(x') \) proceeds as follows:

\[
 s' = \arg \min_{z \in A^{-1}C} \nabla h(x')^T z \\
 (x')^+ = (1 - \gamma)x' + \gamma s'
\]

Multiplying by \( A \) reveals precisely the same Frank-Wolfe update as would be performed on \( f(x) \), since . In fact, even the convergence analysis is affine invariant. Note than the curvature constant \( M \) of \( h \) is

\[
 M = \max_{x', y', s' \in A^{-1}C} \frac{\gamma^2}{2} \left( h(y') - h(x') - \nabla h(x')^T (y' - x') \right)
\]

matching that of \( f \), because \( \nabla h(c')^T (y' - x') = \nabla f(x)^T (y - x) \). That is, each iteration of Frank-Wolfe will remain exactly the same, and also the convergence rate with \( M \).

### 23.7 Inexact Updates

Jaggi\(^3\) also analyzes inexact Frank-Wolfe updates. That is, suppose we choose \( s^{(k-1)} \) so that:

\[
 \nabla f(x^{(k-1)})^T s^{(k-1)} \leq \min_{s \in C} \nabla f(x^{(k-1)})^T s + \frac{M\gamma_k}{2} \cdot \delta
\]

where \( \delta \geq 0 \) is our inaccuracy parameter. Note that the optimization error at step \( k \) is at most \( \frac{M\gamma_k}{2} \cdot \delta \). Since \( \gamma_k \to 0 \), the inexact update needs to be more and more exact as \( k \) proceeds. The same is true for proximal gradient algorithm, which requires the errors to vanish over the course of the algorithm. Then we basically attain the same convergence rate according to the following theorem:

**Theorem 23.2** Conditional gradient method using fixed step sizes \( \gamma_k = \frac{2}{k+1}, k = 1, 2, 3..., \) and inaccuracy parameter \( \delta \geq 0 \), satisfies

\[
 f(x^{(k)}) - f^* \leq \frac{2M}{k+1}(1 + \delta)
\]

\(^3\)M. Jaggi (2011), "Revisiting Frank-Wolfe: projection-free sparse convex optimization"
The proof for inexact updates is similar to the one for exact updates. We can plug in the $\delta$ term into the proof for exact updates, and then get the theorem above.

Two important variants of the conditional gradient method are:

- **Line search**: instead of fixing $\gamma_k = 2/(k + 1), k = 1, 2, 3, \ldots$, we use exact line search for the step size at each $k = 1, 2, \ldots$:
  \[
  \gamma_k = \arg \min_{\gamma \in [0, 1]} f(x^{(k-1)} + \gamma(s^{(k)-1} - x^{(k-1)}))
  \]

  Or we could use backtracking to search the step sizes.

- **Fully corrective**: we directly update by minimizing the function over all the points in the convex hull of the points that we have seen so far:
  \[
  x^{(k)} = \arg \min_y f(y) \text{ subject to } y \in \text{conv}\{x^{(0)}, s^{(0)}, ..., s^{(k-1)}\}
  \]

  The algorithm allows $s^{(0)}, \ldots, s^{(k-2)}$ in the previous steps help on the current step. Existing literatures show that the fully corrective version can make much better progress, but is also much harder to compute.

Both variants have the same $O(1/\epsilon)$ complexity, measured by the number of iterations.

### 23.8 Path Following

Consider the following norm constrained problem:

\[
\min_x f(x) \text{ subject to } \|x\| \leq t
\]

The Frank-Wolfe algorithm can be used for path following, i.e., given $\epsilon > 0$, we can produce an (approximate) solution path $\hat{x}(t), t \geq 0$ such that $f(\hat{x}(t)) - f(x^*(t)) \leq \epsilon$. Here is the algorithm. Beginning at $t_0 = 0$ and $x^*(0) = 0$, we fix parameters $\epsilon, m > 0$, then repeat for $k = 1, 2, 3, \ldots$:

- Calculate: $t_k = t_{k-1} + \frac{(1-1/m)\epsilon}{\|\nabla f(x(t_{k-1}))\|_*}$, and set $\hat{x}(t) = \hat{x}(t_{k-1})$ for all $t \in (t_{k-1}, t_k)$.

- Compute $\hat{x}(t)$ by running Frank-Wolfe at $t = t_k$, terminating when the duality gap is less than $\epsilon/m$.

This is a simplification of the strategy given in Giesen et al. (2012)\(^4\).

With this path following strategy, we are guaranteed that:

\[
f(\hat{x}(t)) - f(x^*(t)) \leq \epsilon \quad \forall t \in [0, t_k]
\]

i.e., we produce a (piecewise-constant) path with suboptimality gap uniformly bounded by $\epsilon$, over all $t$.

To see this, it helps to rewrite the Frank-Wolfe duality gap as:

\[
g_t(x) = \max_{\|s\| \leq t} \nabla f(x)^T(x - s)
\]

\[
= \nabla f(x)^T x + t \max_{\|s\| \leq 1} \nabla f(x)^T s
\]

\[
= \nabla f(x)^T x + t \|\nabla f(x)\|_*
\]

This is a linear function of $t$. Hence if $g_t(x) \leq \epsilon/m$, then we can increase $t$ until $t^+ = t + (1 - 1/m)\epsilon/\|\nabla f(x)\|_*$, because at this value, we have:

$$g_{t^+}(x) = \nabla f(x)^T x + t^+\|\nabla f(x)\|_*$$
$$= \nabla f(x)^T x + t\|\nabla f(x)\|_* + \epsilon - \epsilon/m$$
$$\leq \epsilon$$

i.e., the duality gap remains $\leq \epsilon$ for the same $x$, between $t$ and $t^+$. 