Chapter 4

We will continue in the convex world and discuss alternatives to (projected) gradient descent. In this chapter, we will introduce *conditional gradient*, also known as Frank-Wolfe algorithm, for a more efficient convex constrained optimization. Frank-Wolfe algorithm got a lot of attention lately, because of the special structure of some important convex constraints, such as the ℓ_1-norm constraint—surrogate function for sparsity—and the nuclear norm constraint—surrogate function for low-rankness.

Conditional gradient (Frank-Wolfe) | ℓ_1-norm constraint | nuclear norm constraint

In this chapter, we will focus on the constrained case:

$$\begin{aligned} \min_{x\in\mathbb{R}^p} \quad & f(x) \\ \text{subject to} \quad & x \in \mathcal{C}. \end{aligned}$$

The discussion so far has focused on (projected) gradient descent, which can be easily motivated by the following set of motions:

- Under the assumption that f is a L-smooth function (*remember that this a rather mild assumption, that does not even imply convexity*), we know that we can upper bound f at a point x_t as follows:
$$f(x) \leq f(x_t) + \langle \nabla f(x_t),\, x - x_t\rangle + \tfrac{L}{2}\|x - x_t\|_2^2.$$
- This indicates that, *locally* and for given x_t, we can optimize our objective as:
$$\begin{aligned} \min_{x\in\mathbb{R}^p} \quad & f(x_t) + \langle \nabla f(x_t),\, x - x_t\rangle + \tfrac{L}{2}\|x - x_t\|_2^2 \\ \text{subject to} \quad & x \in \mathcal{C}. \end{aligned}$$
- Observe that the objective itself can be reformulated as:
$$\begin{aligned} &\min_x \left\{ f(x_t) + \langle \nabla f(x_t),\, x - x_t\rangle + \tfrac{L}{2}\|x - x_t\|_2^2 \right\} \\ &\propto \min_x \left\{ \langle \nabla f(x_t),\, x - x_t\rangle + \tfrac{L}{2}\|x - x_t\|_2^2 \right\} \\ &\propto \min_x \left\{ \tfrac{L}{2} \cdot \left\| x - \left(x_t - \tfrac{1}{L}\nabla f(x_t)\right)\right\|_2^2 \right\} \\ &\propto \min_x \left\{ \left\| x - \left(x_t - \tfrac{1}{L}\nabla f(x_t)\right)\right\|_2^2 \right\} \end{aligned}$$
- Thus, our problem becomes:
$$\begin{aligned} \min_{x\in\mathbb{R}^p} \quad & \left\| x - \left(x_t - \tfrac{1}{L}\nabla f(x_t)\right)\right\|_2^2 \\ \text{subject to} \quad & x \in \mathcal{C}. \end{aligned}$$
- If we denote $y := x_t - \frac{1}{L}\nabla f(x_t)$, then the above problem is just a projection onto $\mathcal{C}$!:
$$\begin{aligned} \min_{x\in\mathbb{R}^p} \quad & \|x - y\|_2^2 \\ \text{subject to} \quad & x \in \mathcal{C}, \end{aligned}$$
which by itself motivates the two-step iterative procedure:
$$x_{t+1} = \Pi_{\mathcal{C}}\left(x_t - \tfrac{1}{L}\nabla f(x_t)\right).$$

What the above motions dictate is that, by L-smoothness, we exploit the local quadratic approximations iteratively to minimize f, which by itself motivate the projected gradient descent motions (*the same arguments hold also for the non-projected gradient descent*). Key observation of the above reasoning is that, because of the quadratic form approximation and the $\|x_t - x\|_2^2$ term, we can complete the squares and generate the euclidean projection operation in the objective.

But is this the only way we can perform/define the projection? Also, what if we take a different order in the Taylor approximation of the objective?

Conditional gradient. The conditional gradient method, also known as Frank-Wolfe algorithm, is based on two approximations compared to the discussion above:

- Instead of a local quadratic approximation, we approximate f locally with a linear function:
$$\begin{aligned} \min_{x\in\mathbb{R}^p} \quad & f(x_t) + \langle \nabla f(x_t),\, x - x_t\rangle \\ \text{subject to} \quad & x \in \mathcal{C}. \end{aligned}$$
Let s_t be the solution to this problem; then the direction to move to is $d_t = s_t - x_t$. Thus, using a descent iteration, we have
$$\begin{aligned} x_{t+1} &= x_t + \eta_t d_t = x_t + \eta_t(s_t - x_t) \\ &= (1 - \eta_t)x_t + \eta_t s_t. \end{aligned}$$
- Observe that the step where we find s_t is a type of project, but using the inner product, rather than the Euclidean norm.

The above summarize the conditional gradient algorithm. A standard way to set up the step size is:

$$\eta_t = \frac{2}{t+2};$$

i.e., using a decreasing step size.

Conditional gradient convergence analysis. Frank-Wolfe algorithm has similar convergence rate guarantees with projected gradient descent. In the following theorem, we will make the L-smoothness assumption:

Theorem 1. *Let function $f : \mathcal{C} \to \mathbb{R}$ be convex, L-smooth, and assume it attains its global minimum at a point $x^\star \in \mathcal{C}$. Then, Frank-Wolfe achieves*

$$f(x_t) - f(x^\star) \leq \frac{2LD^2}{t+2},$$

with step size

$$\eta_t = \frac{2}{t+2}.$$

Here, D is the diameter of $\mathcal{C}$: $D = \max_{x,y\in\mathcal{C}} \|x - y\|_2$.

Proof: By smoothness:

$$f(x_{t+1}) \leq f(x_t) + \langle \nabla f(x_t), x_{t+1} - x_t\rangle + \tfrac{L}{2}\|x_{t+1} - x_t\|^2.$$

Sequentially substituting the definition x_{t+1} in the above recursion:

$$\begin{aligned} f(x_{t+1}) &\leq f(x_t) + \langle \nabla f(x_t), (1-\eta_t)x_t + \eta_t s_t - x_t\rangle \\ &\quad + \tfrac{L}{2}\|x_{t+1} - x_t\|^2 \\ &= f(x_t) + \eta_t \langle \nabla f(x_t), s_t - x_t\rangle + \tfrac{L}{2}\|x_{t+1} - x_t\|^2 \\ &= f(x_t) + \eta_t \langle \nabla f(x_t), s_t - x_t\rangle + \tfrac{L}{2}\|(1-\eta_t)x_t + \eta_t s_t - x_t\|^2 \\ &= f(x_t) + \eta_t \langle \nabla f(x_t), s_t - x_t\rangle + \tfrac{\eta_t^2 L}{2}\|s_t - x_t\|^2 \end{aligned}$$

By the definition of D, observe that $\|s_t - x_t\|^2 \leq D^2$. The above inequality then becomes:

$$f(x_{t+1}) \leq f(x_t) + \eta_t \langle \nabla f(x_t), s_t - x_t\rangle + \tfrac{\eta_t^2 LD^2}{2}$$

Using convexity, we know that

$$f(x^\star) \geq f(x_t) + \langle \nabla f(x_t), x^\star - x_t \rangle \Rightarrow$$
$$\langle \nabla f(x_t), x^\star - x_t \rangle \leq f(x^\star) - f(x_t).$$

But we also know that s_t is the solution to the problem

$$\begin{aligned} \min_{x \in \mathbb{R}^p} \quad & f(x_t) + \langle \nabla f(x_t),\ x - x_t \rangle \\ \text{subject to} \quad & x \in \mathcal{C}. \end{aligned}$$

which is equivalent to the following problem:

$$\begin{aligned} \min_{x \in \mathbb{R}^p} \quad & \langle \nabla f(x_t), x \rangle \\ \text{subject to} \quad & x \in \mathcal{C}. \end{aligned}$$

This implies that:

$$\langle \nabla f(x_t), s_t \rangle \leq \langle \nabla f(x_t), x^\star \rangle$$

and thus:

$$\langle \nabla f(x_t), s_t - x_t \rangle \leq f(x^\star) - f(x_t).$$

Combining all the above in the main recursion (after subtracting $f(x^\star)$ on both sides):

$$f(x_{t+1}) - f(x^*) \leq (1 - \eta_t)\left(f(x_t) - f(x^\star)\right) + \tfrac{\eta_t^2 L D^2}{2}$$

We use induction in order to prove $f(x_t) - f(x^\star) \leq \frac{2LD^2}{t+2}$ based on above.

`Base case` $t = 0$. Observe that the above hold for all t. For $t = 0$, we have $\eta_t = \frac{2}{0+2} = 1$. Hence:

$$\begin{aligned} f(x_1) - f(x^\star) &\leq (1 - \eta_0)(f(x_0) - f(x^\star)) + \tfrac{\eta_0^2 L D^2}{2} \\ &= (1-1)(f(x_0) - f(x^\star)) + \tfrac{LD^2}{2} \\ &= \tfrac{LD^2}{2} \\ &\leq \tfrac{2LD^2}{3} \end{aligned}$$

Thus, the induction hypothesis holds for our base case.

`Inductive step: from` t `to` $t+1$. Let us assume that for iteration count up to t the following holds: $f(x_t) - f(x^\star) \leq \frac{2LD^2}{t+2}$. We need to show that, under this assumption, it also holds for $t+1$.

By the main recursion we have:

$$\begin{aligned} f(x_{t+1}) - f(x^\star) &\leq (1 - \eta_t)\left(f(x_t) - f(x^\star)\right) + \tfrac{\eta_t^2 L D^2}{2} \\ &= \left(1 - \tfrac{2}{t+2}\right)\left(f(x_t) - f(x^\star)\right) + \tfrac{4}{2(t+2)^2} LD^2 \\ &\leq \left(1 - \tfrac{2}{t+2}\right) \cdot \tfrac{2LD^2}{t+2} + \tfrac{4}{2(t+2)^2} LD^2 \\ &= LD^2 \left(\tfrac{2t}{(t+2)^2} + \tfrac{2}{(t+2)^2}\right) \\ &= 2LD^2 \cdot \tfrac{t+1}{(t+2)^2} \\ &= 2LD^2 \cdot \tfrac{t+1}{t+2} \cdot \tfrac{1}{t+2} \\ &\leq 2LD^2 \cdot \tfrac{t+2}{t+3} \cdot \tfrac{1}{t+2} \\ &= 2LD^2 \tfrac{1}{t+3} \end{aligned}$$

Thus, the inequality also holds for the $t+1$ case.

What if f is also strongly convex. Intuition suggests that, by the time f is at the same time L-smooth and μ-strongly convex, we can achieve linear convergence rate. Unfortunately, this does not hold for Frank-Wolfe/conditional gradient method, when we make no assumptions about $\mathcal{C}$, other than convexity.

There is a negative result about this statement:

Claim 8. *Consider the following problem:*

$$\begin{aligned} \min_{x \in \mathbb{R}^p} \quad & \tfrac{1}{2} \langle Qx,\ x \rangle + \langle b, x \rangle \\ \text{subject to} \quad & x = Av,\ v_i \geq 0,\ \sum_{i=1}^{p} v_i = 1, \end{aligned}$$

for some Q, A matrices with appropriate dimensions, and for a vector b. Observe that the constraint set represents the convex hull of the columns of A.

Let $\{x_t\}$ denote the putative solutions obtained by running conditional gradient, with exact line search. Then, there is an initial point x_0 such that, for every $\varepsilon > 0$, we have:

$$f(x_t) - f(x^\star) \geq \tfrac{1}{t^{1+\varepsilon}}$$

In words, what this claim states is that *there are problem instances* for which we cannot improve upon the $O(1/t)$ convergence rate. Note though that this does not excludes the possibility that there is a specific function instance f and a specific constraint $\mathcal{C}$ for which we can provable attain linear convergence.

Where is conditional gradient useful. A quick comparison with projected gradient descent can be summarized as:

$$\overset{\text{(Proj. Gradient)}}{O\left(\tfrac{1}{T}\right)} \quad \text{vs} \quad \overset{\text{(Cond. Gradient)}}{O\left(\tfrac{1}{T}\right)}$$

Why then do we care about conditional gradient? The answer lies in the projection step and what is the computational complexity needed to complete that step. To clearly depict this, we will consider specific, but widely used, convex constraints $\mathcal{C}$.

(*Applications and motivation are provided in the corresponding notebook.*)

ℓ_1-norm constraint: The convex set $\mathcal{C}$ in this case is:

$$\mathcal{C} = \{x \in \mathbb{R}^p \ \mid \ \|x\|_1 \leq 1\}$$

(*The discussion easily extends to the general case $\|x\|_1 \leq \alpha$ for some $\alpha > 0$, but we keep unit ball for simplicity.*) Then, our generic problem looks like:

$$\begin{aligned} \min_{x \in \mathbb{R}^p} \quad & f(x) \\ \text{subject to} \quad & \|x\|_1 \leq 1. \end{aligned}$$

In this particular case, if we were to perform the Euclidean norm projection:

$$\begin{aligned} \min_{x \in \mathbb{R}^p} \quad & \|x - y\|_2^2 \\ \text{subject to} \quad & \|x\|_1 \leq 1, \end{aligned}$$

this can be completed through the soft-thresholding operator, where:

$$\widehat{x} = \max\{y - \theta,\ 0\}$$

where

$$\theta = \tfrac{1}{\rho} \cdot \left(\sum_{i=1}^{\rho} y_{\sigma(i)} - 1\right),$$

and

$$\rho = \max\left\{ j \in [p] \mid y_{\sigma(j)} - \tfrac{1}{j} \cdot \left(\sum_{q=1}^{j} y_{\sigma(q)} - 1 \right) > 0 \right\}.$$

Here, $\sigma(\cdot)$ is a descending sorting index. In words, in order to compute the projection, we need to i) sort the input vector y in (usually) $O(p \log p)$ time, ii) compute quantities ρ and θ, and iii) apply the entrywise rule: $\widehat{x} = \max\{y - \theta,\ 0\}$, *per iteration.*

On the other hand, the conditional gradient "projection" step has the form:

$$\begin{aligned} \min_{x \in \mathbb{R}^p} \quad & \langle \nabla f(x_t), x \rangle \\ \text{subject to} \quad & \|x\|_1 \leq 1. \end{aligned}$$

It is easy to see that, if there were no constraints on x, the solution to this minimization is actually unbounded, and leads to $-\infty$ objective value. Under the $\mathcal{C}$, we restrict our search space within a bounded ℓ_1-norm with radius 1. We can prove that (one of) the solution(s) to this problem is to put all the "energy" on the component of the vector gradient $\nabla f(x_t)$ according to:

$$i^\star \in \underset{i\text{-th component}}{\operatorname{argmax}} \quad |\langle \nabla f(x_t), e_i \rangle|$$

where e_i are the basis vectors. Then, s_t is given by:

$$s_t = -1 \cdot \mathtt{sgn}\left(\langle \nabla f(x_t), e_{i^\star} \rangle \right) \cdot e_{i^\star}.$$

Here, we depict on purpose 1; if we had radius α, then we substitute 1 with α. In words, in order to compute the Frank-Wolfe "projection", we need to i) find the maximum component (in magnitude) of $\nabla f(x_t)$ in $O(p)$ time, and ii) compute s_t in constant time, *per iteration.*

Comparing the two approaches, and assuming that (under hidden constants in Big-Oh notation), the per iteration complexity is conditional gradient is lower than that of projected gradient descent. In conjunction with the fact that the asymptotic iteration complexity is the same, $O(\frac{1}{T})$, this means that we might prefer conditional gradient in practice.

Nuclear norm constraint: While the per iteration improvement in the example above seems negligible (we gain $O(\log p)$ per iteration), things get much more interesting in the matrix case. Problems such as matrix completion and matrix sensing take the general form (*we restrict our attention to square matrices just for clarity*):

$$\min_{X \in \mathbb{R}^{p \times p}} \quad f(X) \quad \text{subject to} \quad \|X\|_* \leq 1.$$

Here, the variable is a matrix in $p \times p$, $f(\cdot)$ is a matrix-valued function, and $\|X\|_*$ is the nuclear norm that has the closed-form expression:

$$\|X\|_* = \sum_{i=1}^{p} \sigma_i(X), \quad \text{where } \sigma_i(X) \text{ is the } i\text{-th singular value.}$$

If we were to perform the Euclidean norm projection:

$$\begin{aligned} \min_{X \in \mathbb{R}^{p \times p}} \quad & \|X - Y\|_F^2 \\ \text{subject to} \quad & \|X\|_* \leq 1, \end{aligned}$$

this can be completed, again, through the soft-thresholding operator over matrices, where instead of "soft-thresholing" elements of a vector, we now soft-threshold the vector of singular values:

$$y \equiv \left[\sigma_1(X),\ \sigma_2(X),\ \ldots,\ \sigma_p(X) \right]$$

and

$$\widehat{y} = \max\{y - \theta,\ 0\}.$$

(*The details how to compute the corresponding θ and ρ are left to the reader.*)

The gist of this description is that, in order to compute the projection of X onto the nuclear ball, we need to compute the *full singular value decomposition* of the input matrix X. This further implies that we need to i) compute a SVD in $O(p^3)$ time, ii) perform soft-thresholding and apply the entrywise rule: $\widehat{x} = \max\{y - \theta,\ 0\}$ in $O(p)$, *per iteration.* In real applications (see Netflix recommendation system), the size of these matrices could be in several hundred thousands, if not millions; thus affording a cubic computational complexity per iteration, is often infeasible.

On the other hand, the conditional gradient "projection" step has the form:

$$\begin{aligned} \min_{x \in \mathbb{R}^p} \quad & \langle \nabla f(X_t), X \rangle \equiv \operatorname{Tr}\left(\nabla f(X_t)^\top X \right) \\ \text{subject to} \quad & \|X\|_* \leq 1. \end{aligned}$$

Let the SVD of the matrix $\nabla f(X_t)$ be:

$$\nabla f(X_t) = U \Sigma V^\top,$$

where, without loss of generality, we have sorted Σ (and the corresponding U, V components) so that it contains the singular values in descending order. Denote $\sigma_1(X) \equiv \Sigma_{1,1},\ u_1 \equiv U_{:,1},\ v_1 \equiv V_{:,1}$. Then, the Frank-Wolfe projection is equivalent to the solution:

$$S_t = -1 \cdot u_1 v_1^\top$$

Similarly to the vector case, we depict on purpose 1; if we had radius α, then we substitute 1 with α. In words, in order to compute the Frank-Wolfe "projection", we need to i) find the maximum singular value-vector pair of $\nabla f(X_t)$, and ii) compute S_t in $O(p^2)$, *per iteration. The key difference between this step and the Euclidean projection is that in this case we care only about the top singular value-vector pair, while in Euclidean projection step, we need all the singular value-vector pairs.*

A simple way to put it is that, assuming that in practice the SVD (either partial or full) is computed in approximation through iterative methods—such as the Power Iteration method or the Lanczos algorithm—the Frank-Wolfe projection is $\times O(p)$ faster, as it can be roughly be computed in $O(p^2)$ complexity. Thus, the main intuition behind using conditional gradient—as opposed to standard projected gradient descent algorithms— is that, per iteration, we require the best 1-sparse or rank-1 approximation of the gradient to proceed, as opposed to full ℓ_1-norm constrained or nuclear norm constrained approximation in the former case. For sparsity, a quick analysis shows that this saves a logarithmic factor per iteration, while in the case of low-rankness, we can save up to $\times O(p)$ per iteration, as we are interested in the rank-1 approximation of the gradient, instead of a full rank soft-thresholding projection.

———— ∞ ————

As an interlude, we will study the *power iteration method*. For simplicity, we will consider here $A \in \mathbb{R}^{p\times p}$ is a diagonalizable matrix (e.g., a real symmetric matrix); thus, our focus here is on the eigenvalue decomposition of A, where:

$$A = U\Lambda U^\top, \quad U \in \mathbb{R}^{p\times p}, \Lambda \in \mathbb{R}^{p\times p},$$

where the columns of U are orthonormal and represent the eigenvectors, and Λ is a diagonal matrix, with the eigenvalues, λ_i, on its diagonal. We will make the assumption the eigenvalues are sorted such that

$$|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_p|.$$

Pay attention that we assume that there is a gap between $|\lambda_1|$ and $|\lambda_2|$. In that case, λ_1 is considered the dominant eigenvalue of the matrix.

In words, the power iteration method approximates the extremal eigenvalues of the matrix, that is, the eigenvalues having largest and smallest module, as well as their associated eigenvectors. Power iteration is simple and can be described by the following two-step procedure:

$$\begin{aligned}\widetilde{q}_{t+1} &= Aq_t \\ q_{t+1} &= \frac{\widetilde{q}_{t+1}}{\|\widetilde{q}_{t+1}\|_2}\end{aligned}$$

Observe that:

- The algorithm requires an initial vector $q_0 \in \mathbb{R}^p$.
- The algorithm has no step sizes.
- The algorithm solves a non-convex problem, due to the second step (it is a projection step on $\|\cdot\|_2 = 1$).
- The algorithm requires mostly matrix-vector multiplications, which makes it efficient in practice.

Let us analyze the convergence properties of the power iteration method. Unfolding the recursion, we observe that:

$$q_{t+1} = \frac{A^t q_0}{\|A^t q_0\|_2}.$$

This relation explains the role played by the powers of the input matrix A. For A in $\mathbb{R}^{p\times p}$, its eigenvectors $u_1,\ u_2,\ \ldots,\ u_p$ form a basis in $\mathbb{R}^p$. This means that the initial vector q_0 can be represented as:

$$q_0 = \sum_{i=1}^{p} \alpha_i u_i, \quad \alpha_i \in \mathbb{R}.$$

Moreover, by definition of the eigenvectors, we have:

$$Au_i = \lambda_i u_i, \quad \forall i.$$

The above lead to an equivalent representation of $A^t q_0$ as:

$$\begin{aligned}A^t q_0 = A^t \cdot \sum_{i=1}^{p} \alpha_i u_i &= \sum_{i=1}^{p} \alpha_i A^t u_i \\ &= \sum_{i=1}^{p} \alpha_i \lambda_i^t u_i \\ &= \alpha_1\lambda_1 \cdot \left(u_1 + \sum_{j=2}^{p} \tfrac{\alpha_j}{\alpha_1} \cdot \left(\tfrac{\lambda_j}{\lambda_1}\right)^t u_j\right).\end{aligned}$$

Using this representation we can show the following lemma.

Lemma 6. *Assume $A \in \mathbb{R}^{p\times p}$ is a diagonalizable matrix, with eigenvalues $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_p|$. Suppose that $\alpha_1 \neq 0$; then, there exists a constant $c > 0$ such that:*

$$\|z_t - u_1\|_2 \leq c \cdot \left|\tfrac{\lambda_2}{\lambda_1}\right|^t, \quad t \geq 1,$$

where z_t is a scaled version of q_t:

$$z_t = \frac{q_t \cdot \|A^t q_0\|_2}{\alpha_1 \lambda_1^t} = u_1 + \sum_{j=2}^{p} \tfrac{\alpha_j}{\alpha_1} \cdot \left(\tfrac{\lambda_j}{\lambda_1}\right)^t u_j.$$

Proof: It is easy to see that:

$$\begin{aligned}\|z_t - u_1\|_2 &= \left\| u_1 + \sum_{j=2}^{p} \tfrac{\alpha_j}{\alpha_1} \cdot \left(\tfrac{\lambda_j}{\lambda_1}\right)^t u_j - u_1 \right\|_2 \\ &= \left\| \sum_{j=2}^{p} \tfrac{\alpha_j}{\alpha_1} \cdot \left(\tfrac{\lambda_j}{\lambda_1}\right)^t u_j \right\|_2 \\ &\leq \left(\sum_{j=2}^{p} \left(\tfrac{\alpha_j}{\alpha_1}\right)^2 \cdot \left(\tfrac{\lambda_j}{\lambda_1}\right)^{2t} \right)^{1/2} \\ &\leq \left|\tfrac{\lambda_2}{\lambda_1}\right|^t \cdot \left(\sum_{j=2}^{p} \left(\tfrac{\alpha_j}{\alpha_1}\right)^2 \right)^{1/2} \\ &= c \cdot \left|\tfrac{\lambda_2}{\lambda_1}\right|^t\end{aligned}$$

for $c := \left(\sum_{j=2}^{p} \left(\tfrac{\alpha_j}{\alpha_1}\right)^2\right)^{1/2}$. □