Chapter 4

We will continue our journey within the convex world and discuss alternatives to (projected) gradient descent. In this chapter, we will introduce *conditional gradient*, also known as the Frank-Wolfe algorithm, for a more efficient convex constrained optimization. The Frank-Wolfe algorithm got a lot of attention, 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 thus far focuses 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 (*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 any given x_t, we can approximate our optimization problem by optimizing the surrogate objective:
$$\begin{aligned} \min_{x\in\mathbb{R}^p} \quad & f(x_t) + \langle \nabla f(x_t),\ x - x_t\rangle + \tfrac{L}{2}\|x - x_t\|_2^2 \\ \text{subject to} \quad & x \in \mathcal{C}. \end{aligned}$$
- Observe that the surrogate objective itself can be reformulated as:
$$\begin{aligned} & \min_x \left\{ f(x_t) + \langle \nabla f(x_t),\ x - x_t\rangle + \tfrac{L}{2}\|x - x_t\|_2^2 \right\} \\ \propto & \min_x \left\{ \langle \nabla f(x_t),\ x - x_t\rangle + \tfrac{L}{2}\|x - x_t\|_2^2 \right\} \\ \propto & \min_x \left\{ \tfrac{L}{2}\|x - x_t\|_2^2 + \langle \nabla f(x_t),\ x - x_t\rangle + \tfrac{1}{L}\|\nabla f(x_t)\|_2^2 \right\} \\ \propto & \min_x \left\{ \tfrac{L}{2} \cdot \left\| x - \left(x_t - \tfrac{1}{L}\nabla f(x_t)\right)\right\|_2^2 \right\} \\ \propto & \min_x \left\{ \left\| x - \left(x_t - \tfrac{1}{L}\nabla f(x_t)\right)\right\|_2^2 \right\} \end{aligned}$$
The symbol $\propto$ denotes that we can remove/add terms in the objective, without affecting the course of the optimization, since the removed/added terms are considered constants.
- 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 the Frank-Wolfe algorithm (see [39]), 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. To see how s_t is derived, it is actually the minimizer of the following optimization problem:
$$\begin{aligned} \min_{x\in\mathbb{R}^p} \quad & \langle \nabla f(x_t),\ x\rangle \\ \text{subject to} \quad & x \in \mathcal{C}. \end{aligned}$$
Given s_t, the direction to move to is $d_t = s_t - x_t$ (*remember that the direction is provided by the term $x - x_t$ in the Taylor approximation $f(x_t) + \langle \nabla f(x_t),\ x - x_t\rangle$*). 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 for finding s_t is a type of projection, but using the inner product, rather than the Euclidean norm. See the figure above for an illustration.
- Of course, we have not described how the projection for s_t is computed. This is because it depends on the set $\mathcal{C}$. More in the text below.

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 [40]. We will see below that this step size selection leads to convergence.

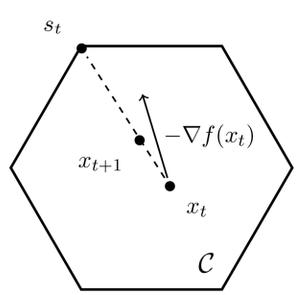

Fig. 32. Borrowed from appendix [5] - Illustration of conditional gradient descent

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

Theorem 2. *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 iterates achieve:*

$$f(x_{t+1}) - f(x^\star) \leq \frac{2LD^2}{t+2},$$

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:

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

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

$$\begin{aligned} f(x^\star) \geq f(x_t) + \langle \nabla f(x_t), x^\star - x_t \rangle \Rightarrow \\ \langle \nabla f(x_t), x^\star - x_t \rangle \leq f(x^\star) - f(x_t). \end{aligned}$$

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):

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

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:

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

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:

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

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

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

There is a negative result about this statement:

Claim 6. *Consider the following problem:*

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

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 exclude 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, given that the iteration complexity is similar to convex projected gradient descent? 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:

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

(Why we use the ℓ_1-norm will be apparent later on in Chapter 7)

To provide some examples, consider the sparse linear regression problem (Lasso problem), as in:

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

In this ℓ_1-norm 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[10]:

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

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; e.g., see the constraint in the figure in first page of the chapter. We can prove that (one of) the solution(s) to this problem is to put all the "energy" on the component of the 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/coordinate vectors (e.g., e_1 is the p-th dimensional zero vector, except for the first coordinate being 1). Then, s_t is given by:

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

The first term -1 was chosen on purpose; if we had radius α, then we substitute -1 with $-\alpha$. 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 number of iterations these algorithms require is the same, the per iteration complexity of 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:

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

Just to provide an example, consider the least-squares version on matrix variables:

$$\min_{X\in\mathbb{R}^{p\times p}} \quad \tfrac{1}{2}\|\mathcal{A}(X) - b\|_2^2 \quad \text{subject to} \quad \|X\|_* \leq 1,$$

for some linear transformation mapping $\mathcal{A} : \mathbb{R}^{p\times p} \to \mathbb{R}^m$.

(Why we use the nuclear norm will be apparent later on in Chapter 8)

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:

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

[10] For those interested in understanding this part, the instructor suggests you to read the paper "Efficient projections onto the ℓ_1-ball for learning in high dimensions" [41]; it was included in the list of papers to review in previous homeworks.

and

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

(*The details how to compute the corresponding θ and ρ are left to the reader. Further, details about projections onto nuclear norm and low-rank recovery problems will be the task of Chapter 8.*)

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, for the Euclidean projection, 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 [42] and [43]), 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 substitute the first term -1 with $-\alpha$ if we had radius 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.

$$\text{———————} \quad \infty \quad \text{———————}$$

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:

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

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:

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

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:

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

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

See also [44].

Appendix

1. J. Nocedal and S. Wright. Numerical optimization. Springer Science & Business Media, 2006.
2. Y. Nesterov. Introductory lectures on convex optimization: A basic course, volume 87. Springer Science & Business Media, 2013.
3. S. Boyd and L. Vandenberghe. Convex optimization. Cambridge university press, 2004.
4. D. Bertsekas. Convex optimization algorithms. Athena Scientific Belmont, 2015.
5. Sébastien Bubeck. Convex optimization: Algorithms and complexity. Foundations and Trends® in Machine Learning, 8(3-4):231–357, 2015.
6. S. Weisberg. Applied linear regression, volume 528. John Wiley & Sons, 2005.
7. T. Hastie, R. Tibshirani, and M. Wainwright. Statistical learning with sparsity: the lasso and generalizations. CRC press, 2015.
8. J. Friedman, T. Hastie, and R. Tibshirani. The elements of statistical learning, volume 1. Springer series in statistics New York, 2001.
9. M. Paris and J. Rehacek. Quantum state estimation, volume 649. Springer Science & Business Media, 2004.
10. M. Daskin. A maximum expected covering location model: formulation, properties and heuristic solution. Transportation science, 17(1):48–70, 1983.
11. I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. MIT press, 2016.
12. L. Trefethen and D. Bau III. Numerical linear algebra, volume 50. Siam, 1997.
13. G. Strang. Introduction to linear algebra, volume 3. Wellesley-Cambridge Press Wellesley, MA, 1993.
14. G. Golub. Cmatrix computations. The Johns Hopkins, 1996.
15. A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.
16. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.
17. S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems, pages 91–99, 2015.
18. T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, pages 3111–3119, 2013.
19. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.
20. Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 1243–1252. JMLR. org, 2017.
21. Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In Fifteenth annual conference of the international speech communication association, 2014.
22. Tom Sercu, Christian Puhrsch, Brian Kingsbury, and Yann LeCun. Very deep multilingual convolutional neural networks for LVCSR. In 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4955–4959. IEEE, 2016.
23. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. page arXiv:1706.03762, 2017.
24. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. page arXiv:1810.04805, 2018.
25. Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason J Corso, and Jianfeng Gao. Unified vision-language pre-training for image captioning and VQA. In AAAI, pages 13041–13049, 2020.
26. Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. arXiv preprint arXiv:2005.14165, 2020.
27. Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. arXiv preprint arXiv:1909.08053, 2019.
28. Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683, 2019.
29. Gary Marcus, Ernest Davis, and Scott Aaronson. A very preliminary analysis of DALL-E 2. arXiv preprint arXiv:2204.13807, 2022.
30. John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. Nature, 596(7873):583–589, 2021.
31. Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.
32. Or Sharir, Barak Peleg, and Yoav Shoham. The cost of training nlp models: A concise overview. arXiv preprint arXiv:2004.08900, 2020.
33. H. Karimi, J. Nutini, and M. Schmidt. Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 795–811. Springer, 2016.
34. Philip Wolfe. Convergence conditions for ascent methods. SIAM review, 11(2):226–235, 1969.
35. Larry Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. Pacific Journal of mathematics, 16(1):1–3, 1966.
36. Stephen Wright and Jorge Nocedal. Numerical optimization. Springer Science, 35(67-68):7, 1999.
37. B. Polyak. Introduction to optimization. Inc., Publications Division, New York, 1, 1987.
38. Stephen Boyd, Lin Xiao, and Almir Mutapcic. Subgradient methods. lecture notes of EE392o, Stanford University, Autumn Quarter, 2004:2004–2005, 2003.
39. Marguerite Frank, Philip Wolfe, et al. An algorithm for quadratic programming. Naval research logistics quarterly, 3(1-2):95–110, 1956.
40. M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In Proceedings of the 30th international conference on machine learning, number CONF, pages 427–435, 2013.
41. J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In Proceedings of the 25th international conference on Machine learning, pages 272–279, 2008.
42. Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. Computer, 42(8):30–37, 2009.
43. A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In Advances in neural information processing systems, pages 1257–1264, 2008.
44. T. Booth and J. Gubernatis. Improved criticality convergence via a modified Monte Carlo power iteration method. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.
45. Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. Quarterly of applied mathematics, 2(2):164–168, 1944.
46. Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. Journal of the society for Industrial and Applied Mathematics, 11(2):431–441, 1963.
47. Andreas Griewank. The modification of Newton's method for unconstrained optimization by bounding cubic terms. Technical report, Technical report NA/12, 1981.
48. Yurii Nesterov and Boris T Polyak. Cubic regularization of Newton method and its global performance. Mathematical Programming, 108(1):177–205, 2006.
49. Yurii Nesterov and Arkadii Nemirovskii. Interior-point polynomial algorithms in convex programming. SIAM, 1994.
50. Ulysse Marteau-Ferey, Francis Bach, and Alessandro Rudi. Globally convergent Newton methods for ill-conditioned generalized self-concordant losses. Advances in Neural Information Processing Systems, 32, 2019.
51. Quoc Tran-Dinh, Anastasios Kyrillidis, and Volkan Cevher. Composite self-concordant minimization. J. Mach. Learn. Res., 16(1):371–416, 2015.
52. Konstantin Mishchenko. Regularized Newton method with global $o(1/k^2)$ convergence. arXiv preprint arXiv:2112.02089, 2021.
53. S. Zavriev and F. Kostyuk. Heavy-ball method in nonconvex optimization problems. Computational Mathematics and Modeling, 4(4):336–341, 1993.
54. E. Ghadimi, H. Feyzmahdavian, and M. Johansson. Global convergence of the heavy-ball method for convex optimization. In 2015 European control conference (ECC), pages 310–315. IEEE, 2015.
55. Y. Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. In Dokl. akad. nauk Sssr, volume 269, pages 543–547, 1983.
56. B. O'Donoghue and E. Candes. Adaptive restart for accelerated gradient schemes. Foundations of computational mathematics, 15(3):715–732, 2015.
57. O. Devolder, F. Glineur, and Y. Nesterov. First-order methods of smooth convex optimization with inexact oracle. Mathematical Programming, 146(1-2):37–75, 2014.
58. L. Bottou, F. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. Siam Review, 60(2):223–311, 2018.