Chapter 6

In our attempt to match the lower bounds for gradient descent in the previous chapter, we "cheated" by using information beyond the first-order gradient to achieve up to a *quadratic* convergence rate. But whether we can match the initial lower bounds by just using gradients remains open.

In this chapter, we will discuss one way to match these lower bounds using only gradient information, closing this gap. This is achieved with the notion of acceleration/momentum, where we will discuss the Heavy Ball method by Polyak and Nesterov's optimal methods.

Momentum | Heavy Ball method | Nesterov's acceleration | Adaptive restarts and noise in acceleration

We remind again of the limits of gradient descent-based methods under convex assumptions.

- For convex objective functions with Lipschitz continuous gradients, with constant L, we can prove that there exists an instance f such that first-order methods cannot be better than:

$$f(x_T) - f(x^\star) \geq \frac{3L\|x_0 - x^\star\|_2^2}{32(T+1)^2} = O\left(\tfrac{1}{T^2}\right).$$

  Under this assumption, and only using gradients, we cannot achieve better than the above.
- For convex objectives functions with both Lipschitz continuous gradients and strong convexity, a similar argument holds. I.e., there is a strongly convex function f such that gradient descent-based methods cannot be better than:

$$\|x_T - x^\star\|_2^2 \geq \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2T} \|x_0 - x^\star\|_2^2.$$

  where $\kappa = L/\mu > 1$. Here we observe that, while we have achieved the same convergence rate concerning the exponent—i.e., in both cases, we have c^T, for $c < 1$—in the lower bound case, we see $\sqrt{\kappa}$ instead of κ.

Gradient descent and acceleration. We will focus on two *multi-step* gradient descent methods: the Heavy Ball method and (one of) Nesterov's accelerated methods. These methods are called multi-step since they consider the history of points computed to prove convergence. In its most generic form (and abstractly denoting the algorithm as a function $\varphi(\cdot)$), these methods can be written as:

$$x_{t+1} = \varphi(x_t, x_{t-1}, \dots, x_{t-\ell}),$$

where ℓ here represents the time window in the past from which we take information to accelerate the process.

In a sense, gradient methods—and even second-order methods—are one-step methods with $\ell = 0$.

Heavy-ball method. We will start with the Heavy ball method, which the following recursion can describe:

$$x_{t+1} = \underbrace{x_t - \eta \nabla f(x_t)}_{\text{Gradient step}} + \underbrace{\beta(x_t - x_{t-1})}_{\text{Momentum step}}.$$

Here, x_t is the current estimate, η is the step size, similar to standard gradient descent, and β is the momentum parameter. Observe that, following the discussion above, this recursion belongs to the case:

$$x_{t+1} = \varphi(x_t, x_{t-1}).$$

What is the motivation for using such a method? A vital issue in gradient descent is pathological curvature. When curvature in different regions and directions is very different, for a fixed learning rate, gradient descent will make slow progress in one of either the high or low curvature regions/directions. For pathological curvature, we want to make smaller steps in regions of high curvature to dampen oscillations and make larger steps and accelerate in areas of low curvature.

Further, we will answer this question through some plots. See the following figures: instead of unnecessarily zig-zagging in the case of gradient descent updates, momentum uses past information to be "biased", thus achieving a more *direct* trajectory towards the (local or global) stationary point.

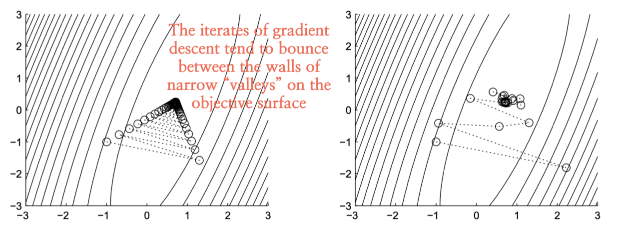

Fig. 38. Motivation for using acceleration in gradient descent. Borrowed from Boyd's and Vanderberghe book on "Convex optimization".

Some physical analogy inspires momentum: Consider we have a ball that moves along a curved surface (*that's why the method is called heavy-ball*). The motion of the ball in a potential field under the force of friction is described by a second-order differential equation:

$$\mu \cdot \tfrac{\partial^2 x(t)}{\partial t^2} = -\nabla f(x(t)) - b\tfrac{\partial x(t)}{\partial t}.$$

Observe that the intuition of the heavy-ball method comes from the continuous space, where gradient descent is known as gradient flow. (*The field that studies how we move from phenomena that happen in the continuous space to the discrete space is an active research area in optimization and machine learning*). One way to discretize the above continuous differential equation is to obtain:

$$\mu \cdot \tfrac{x_{t+\Delta t} - 2x_t + x_{t-\Delta t}}{\Delta t^2} = -\nabla f(x_t) - b\tfrac{x_t - x_{t-\Delta t}}{\Delta t},$$

which results in the following:

$$x_{t+\Delta t} = x_t - \tfrac{\Delta t^2}{\mu}\nabla f(x_t) + \left(1 - \tfrac{b\Delta t}{\mu}\right)(x_t - x_{t-\Delta t}).$$

This resembles the discrete Heavy-ball description above.

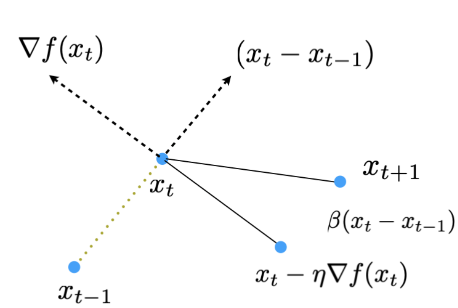

Fig. 39. Motions of the heavy-ball method. If the current gradient step is in the same direction as the previous step, then move a little further in that direction.

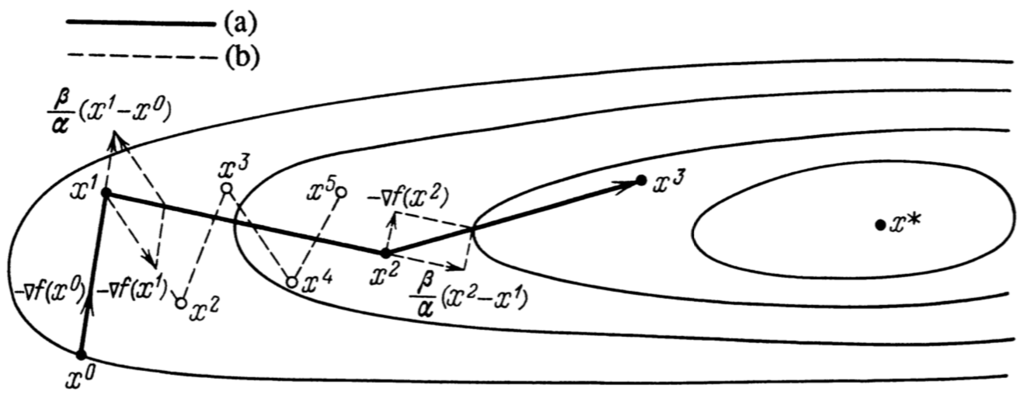

Fig. 40. Motivation for using acceleration in gradient descent. Borrowed from Polyak's book "Introduction to Optimization". (a) is Gradient descent, and (b) is the heavy-ball method.

Locally, at a point x_t, the Heavy ball method "makes decisions" according to the figure above.

But how does it perform in theory? Let us first assume that we use the heavy-ball method for convex functions f.

Theorem 6. *Consider the heavy-ball recursion, with step size η and momentum parameter β. Let f, the objective function, be convex, with L-Lipschitz continuous gradients. Further, assume that f is strongly convex with parameter μ, with a unique global minimum $x^\star$. Then, for step size and momentum parameters satisfying:*

$$\eta = \frac{4}{(\sqrt{\mu}+\sqrt{L})^2}, \text{ and } \beta = \left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2,$$

the heavy ball recursion gives an estimate x_T after T iterations, such that:

$$\|x_T - x^\star\|_2 \leq \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^T \|x_0 - x^\star\|_2.$$

Before we provide the proof, compare this with the lower bounds provided at the beginning of the chapter: *the Heavy-ball method achieves the lower bounds by just using the value of the estimates from the previous iteration!* I.e., we do not compute or store something extraordinarily large, such as keeping a long history of gradients or computing the Hessian.

Proof: In contrast to the gradient method, we will focus on the behavior of two consecutive distances, $\|x_{t+1} - x^\star\|_2$, $\|x_t - x^\star\|_2$:

$$\begin{aligned}
&\left\|\begin{bmatrix} x_{t+1} - x^\star \\ x_t - x^\star \end{bmatrix}\right\|_2 \\
&= \left\|\begin{bmatrix} x_t + \beta\,(x_t - x_{t-1}) - x^\star \\ x_t - x^\star \end{bmatrix} - \eta \begin{bmatrix} \nabla f\,(x_t) \\ 0 \end{bmatrix}\right\|_2 \\
&= \left\|\begin{bmatrix} (1+\beta)I & -\beta I \\ I & 0 \end{bmatrix} \cdot \begin{bmatrix} x_t - x^\star \\ x_{t-1} - x^\star \end{bmatrix} - \eta \begin{bmatrix} \nabla^2 f\,(z_t)\,(x_t - x^\star) \\ 0 \end{bmatrix}\right\|_2
\end{aligned}$$

For the last equality, we use the generalization of the *mean value theorem*, according to which, for a function $f : [\alpha, \beta] \to \mathbb{R}$, differentiable, there exists $\gamma \in (\alpha, \beta)$ such that:

$$f'(\gamma) = \frac{f(\beta)-f(\alpha)}{\beta-\alpha}.$$

This leads to the following equation for our case: $\nabla f(x_t) = \nabla^2 f(z_t)(x_t - x^\star)$, with z_t in the space between x_t and $x^\star$. (*To see this, consider the substitutions $f'(\cdot) \to \nabla^2 f(\cdot)$, $f(\cdot) \to \nabla f(\cdot)$, and the fact that $\nabla f(x^\star) = 0$.*) Continuing the above recursion, we have:

$$\begin{aligned}
&\left\|\begin{bmatrix} x_{t+1} - x^\star \\ x_t - x^\star \end{bmatrix}\right\|_2 \\
&= \left\|\begin{bmatrix} (1+\beta)I - \eta\nabla^2 f\,(z_t) & -\beta I \\ I & 0 \end{bmatrix} \cdot \begin{bmatrix} x_t - x^\star \\ x_{t-1} - x^\star \end{bmatrix}\right\|_2 \\
&\leq \left\|\begin{bmatrix} (1+\beta)I - \eta\nabla^2 f\,(z_t) & -\beta I \\ I & 0 \end{bmatrix}\right\|_2 \cdot \left\|\begin{bmatrix} x_t - x^\star \\ x_{t-1} - x^\star \end{bmatrix}\right\|_2
\end{aligned}$$

In the last step, we apply the Cauchy-Schwarz inequality.

Let us focus on the contraction matrix:

$$\left\|\begin{bmatrix} (1+\beta)I - \eta\nabla^2 f\,(z_t) & -\beta I \\ I & 0 \end{bmatrix}\right\|_2$$

We know that $\nabla^2 f(\cdot) \succ 0$ by strong convexity, and it has an eigenvalue decomposition:

$$\nabla^2 f\,(z_t) = U\Lambda U^\top,$$

where U is an orthonormal matrix, and Λ is a diagonal matrix, with the eigenvalues of $\nabla^2 f(\cdot)$ on its diagonal. Since $\nabla^2 f(\cdot) \succ 0$, observe that all the eigenvalues are positive. Let us denote the eigenvalues as λ_i. Then, for simplicity of our arguments, we will get the following equalities under proper assumptions:

$$\begin{aligned}
&\left\|\begin{bmatrix} (1+\beta)I - \eta\nabla^2 f\,(z_t) & -\beta I \\ I & 0 \end{bmatrix}\right\|_2 \\
&= \left\|\begin{bmatrix} U^\top & 0 \\ 0 & U^\top \end{bmatrix} \cdot \begin{bmatrix} (1+\beta)I - \eta U\Lambda U^\top & -\beta I \\ I & 0 \end{bmatrix} \cdot \begin{bmatrix} U & 0 \\ 0 & U \end{bmatrix}\right\|_2 \\
&= \left\|\begin{bmatrix} (1+\beta)U^\top IU - \eta U^\top U\Lambda U^\top U & -\beta U^\top IU \\ U^\top IU & 0 \end{bmatrix}\right\|_2 \\
&= \left\|\begin{bmatrix} (1+\beta)I - \eta\Lambda & -\beta I \\ I & 0 \end{bmatrix}\right\|_2 = \mathbf{T}
\end{aligned}$$

Let $\mathbf{T}$ represent the block diagonal matrix of the above expression. Now, let's come back to our entire expression:

$$\left\|\begin{bmatrix} x_{t+1} - x^\star \\ x_t - x^\star \end{bmatrix}\right\|_2 \leq \left\|\mathbf{T}^{\mathbf{t}}\right\|_2 \cdot \left\|\begin{bmatrix} x_t - x^\star \\ x_{t-1} - x^\star \end{bmatrix}\right\|_2$$

We want to bound $\left\|\mathbf{T^t}\right\|_2$ to have convergence. The spectrum of a diagonal matrix is the eigenvalues of the sub-matrices. Bounding $\mathbf{T^t}$ to the spectral radius, we can use the following fact:

$$\left\|T^i\right\|_2 \leq (\rho(\mathbf{T}) + \epsilon_i)^i$$

For some set of sequences of $\epsilon_i \geq 0$. Where $\rho(\mathbf{T})$ is the spectral radius of $\mathbf{T}$ (maximum magnitude of an eigenvalue of $\mathbf{T}$). Now, rewriting the problem to solve:

$$\max_i \begin{bmatrix} 1+\beta-\eta\lambda_i & -\beta \\ 1 & 0 \end{bmatrix} + \epsilon_i$$

The maximum value is equivalent to finding the maximum eigenvalue of many 2×2 matrices. We will drop the ϵ_i by setting it to 0. To reduce the expression to:

$$\max_i \begin{bmatrix} 1+\beta-\eta\lambda_i & -\beta \\ 1 & 0 \end{bmatrix}$$

To compute the eigenvalues of such matrices, we need to find the roots of the equation:

$$\xi^2 - (1+\beta-\eta\lambda_i)\xi + \beta = 0.$$

Observe that for $\beta \geq \left(1-\sqrt{\eta\lambda_i}\right)^2$, the roots of the characteristic equations are imaginary, and both have magnitude $\sqrt{\beta}$. By L-smoothness and strong convexity assumptions,

$$\left(1-\sqrt{\eta\lambda_i}\right)^2 \leq \max\left\{|1-\sqrt{\eta\mu}|^2,\ |1-\sqrt{\eta L}|^2\right\}.$$

Then, by letting $\beta = \max\left\{|1-\sqrt{\eta\mu}|^2,\ |1-\sqrt{\eta L}|^2\right\}$, we have:

$$\left\|\begin{bmatrix} (1+\beta)I - \eta\nabla^2 f(z_t) & -\beta I \\ I & 0 \end{bmatrix}\right\|_2 \leq \max\left\{|1-\sqrt{\eta\mu}|,\ |1-\sqrt{\eta L}|\right\}\cdot$$

for some auxiliary variable $\lambda \in (0,1]$.

Now, by letting $\eta = \frac{4}{(\sqrt{\mu}+\sqrt{L})^2}$, we have:

$$\beta = \left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2, \text{ and } \max\left\{|1-\sqrt{\eta\mu}|,\ |1-\sqrt{\eta L}|\right\} = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}.$$

This leads finally to:

$$\left\|\begin{bmatrix} x_{t+1}-x^\star \\ x_t - x^\star \end{bmatrix}\right\|_2 \leq \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)\left\|\begin{bmatrix} x_t-x^\star \\ x_{t-1} - x^\star \end{bmatrix}\right\|_2.$$

Unfolding this recursion and focusing on the top row, we obtain:

$$\|x_T - x^\star\|_2 \leq \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^T \|x_0 - x^\star\|_2.$$

□

Thus, the heavy-ball method converges linearly, but, in Big-Oh notation and given that the factor κ is an important one, its iteration complexity is $O(\sqrt{\kappa}\log\frac{1}{\varepsilon})$, as compared to $O(\kappa\log\frac{1}{\varepsilon})$ of standard gradient descent. The corresponding `iPython Notebook` compares the convergence of gradient descent and the heavy ball method.

What about using the heavy-ball method for convex but just L-smooth functions? Can we still prove convergence or, even better, the acceleration? In our thus-far discussion on the heavy-ball method, we made the following assumptions, on top of convexity and L-smoothness:

- f is also strongly convex with parameter μ.
- f is twice differentiable.

There are some surprising results when we start dropping some of these assumptions. (*The research on these questions is still active; thus, if you find any results that disprove any of the statements below, please let me know.*) Zavriev and Kostyuk in [48] prove that the heavy-ball method trajectories converge to a stationary point, with sufficient conditions, when the function f is just L-smooth, but not necessarily convex.

It turns out that current state-of-the-art results for general L-smooth, *and convex* function f is the following theorem by Ghadimi, Feyzmahdavian, and Johansson [49].

Theorem 7. *Let f be a convex function with L-Lipschitz continuous gradients. Consider the heavy-ball recursion with momentum parameter and step size satisfying: $\beta \in [0,1)$, $\eta \in \left(0, \frac{2(1-\beta)}{L}\right)$. Then,*

$$f(\bar{x}_T) - f(x^\star) = O\left(\tfrac{1}{T}\right),$$

where $\bar{x}_T = \frac{1}{T+1}\sum_{t=0}^T x_t$.

Sketch of proof: The proof uses the following steps:

- Define $p_t = \frac{\beta}{1-\beta}(x_t - x_{t-1})$, which leads to heavy-ball recursion: $x_{t+1} + p_{t+1} = x_t + p_t - \frac{\eta}{1-\beta}\nabla f(x_t)$.
- Compute $\|x_{t+1} + p_{t+1} - x^\star\|_2^2$ by substituting the quantity $x_{t+1} + p_{t+1}$ and unrolling the square identity.
- Using standard L-smoothness identities, we get to:

$$\begin{aligned} &\tfrac{2\eta\lambda}{(1-\beta)}\sum_{t=0}^T \left(f(x_t) - f(x^\star)\right) \\ &+ \sum_{t=0}^T \left(\tfrac{2\eta\beta}{(1-\beta)^2}\left(f(x_t) - f(x^\star)\right) + \|x_{t+1}+p_{t+1}-x^\star\|^2\right) \\ &\leq \sum_{t=0}^T \left(\tfrac{2\eta\beta}{(1-\beta)^2}\left(f(x_{t-1}) - f(x^\star)\right) + \|x_t+p_t-x^\star\|^2\right) \end{aligned}$$

- This implies that:

$$\tfrac{2\eta\lambda}{(1-\beta)}\sum_{t=0}^T \left(f(x_t) - f(x^\star)\right) \leq \tfrac{2\eta\beta}{(1-\beta)^2}\left(f(x_0) - f(x^\star)\right) + \|x_0 - x^\star\|^2$$

- Given convexity of f, we have by Jensen's inequality that:

$$(T+1)f(\bar{x}_T) \leq \sum_{t=0}^T f(x_t).$$

- The above lead to:

$$\begin{aligned} &f(\bar{x}_T) - f(x^\star) \\ &\quad\leq \tfrac{1}{T+1}\left(\tfrac{\beta}{\lambda(1-\beta)}\left(f(x_0) - f(x^\star)\right) + \tfrac{1-\beta}{2\eta\lambda}\|x_0 - x^\star\|^2\right) \\ &\quad= O(\tfrac{1}{T}) \end{aligned}$$

□

The above result denotes that the average of all estimates drops with rate $O(frac1T)$; i.e., the current proof for heavy-ball is similar to that of the simple gradient descent method! One can use per-iteration specific values for η_t and β_t, which further leads to:

$$f(x_T) - f(x^\star) = O(\tfrac{1}{T}),$$

according to Ghadimi, Feyzmahdavian, and Johansson [49]. *However, there is still a gap between our current theory and the possibly achievable lower bounds!*

What is more interesting is the following fact: So far, we focused on the L-smoothness assumption; if we also assume strong convexity, but we drop the assumption that f is twice differentiable, there are cases where the heavy-ball method does not necessarily converge, even using Polyak's stability conditions!

Nesterov's accelerated method. In our discussion so far, for both theory and practice, we made the following choices:

- Practically, heavy-ball method satisfies the recursion $x_{t+1} = x_t - \eta \nabla f(x_t) + \beta(x_t - x_{t-1})$, where the gradient is computed at the current point x_t.
- Theoretically, the heavy-ball method was shown to achieve the lower bounds for the case of L-smooth and μ-strongly convex case.

Nesterov, in his seminal paper [50] in 1983, proved that a slightly different version of the heavy-ball method can achieve the lower bounds of $O(\frac{1}{T^2})$ for first-order methods under L-smoothness assumption; a result that is currently missing for the simple heavy-ball method.

First, let us describe Nesterov's proposal. The idea is based on the following observation: The Heavy-ball method

$$x_{t+1} = x_t - \eta \nabla f(x_t) + \beta(x_t - x_{t-1}),$$

can be equivalently written as a two-step procedure:

$$\widetilde{x}_t = x_t - \eta \nabla f(x_t)$$
$$x_{t+1} = \widetilde{x}_t + \beta(x_t - x_{t-1}).$$

In a way, in Heavy-ball, we end up to x_{t+1} after computing the gradient of f at x_t and performing the momentum step. But what if we compute the gradient at a point that looks *more similar* to the motions we perform, even after the gradient calculation in heavy-ball? This leads to Nesterov's suggestion where we compute:

$$\widetilde{x}_t = x_t - \eta \nabla f(x_t + \beta(x_t - x_{t-1}))$$
$$x_{t+1} = \widetilde{x}_t + \beta(x_t - x_{t-1}).$$

Locally, at a point x_t, the Nesterov's method "makes decisions" according to the following figure.

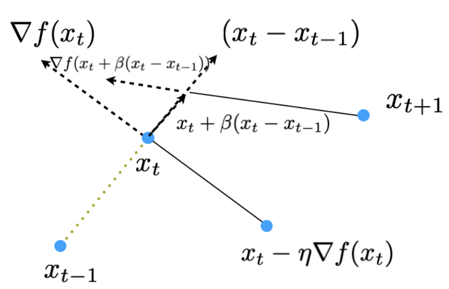

Fig. 41. Motions of Nesterov's accelerated method. If the current gradient step is in the same direction as the previous step, then move a little further in that direction. Compare this figure with previous Figure.

The above can be written in the following form, which is more recognizable as Nesterov's recursion:

$$x_{t+1} = y_t - \eta \nabla f(y_t)$$
$$y_{t+1} = x_{t+1} + \beta(x_{t+1} - x_t)$$

What was revolutionary is that Nesterov proposed specific, time-dependent values for β_t—that are simultaneously practical—which lead provably to acceleration! One such schedule for the momentum parameters β_t satisfies:

$$\theta_0 = 1, \; \theta_{t+1} = \frac{1+\sqrt{1+4\theta_t^2}}{2}, \; \beta_t = \frac{\theta_t - 1}{\theta_{t+1}}$$

Let us first consider the case where f is convex and L-smooth.

Theorem 8. *Let f be a convex function with L-Lipschitz continuous gradients. Then, Nesterov's recursion with β_t as defined above, and $\eta = \frac{1}{L}$ satisfies:*

$$f(x_T) - f(x^\star) \leq \frac{2L\|x_0 - x^\star\|_2^2}{T^2} = O\left(\frac{1}{T^2}\right).$$

I.e., Nesterov's accelerated method achieves the lower bound for the case of just L-smooth convex functions!

Further, for strongly convex functions with parameter μ, one can also show that, similarly to the heavy-ball method, it achieves the complexity $O\left(\sqrt{\kappa} \log \frac{1}{\varepsilon}\right)$; i.e., it also achieves the lower bound for the case of L-smooth and μ-strongly convex functions! (*We omit the proof and leave the discussion of acceleration in non-convex settings for later.*)

Interesting facts about acceleration. Closing this chapter, we will discuss two exciting facts using acceleration.

Set up: For the first one, we will need an optimal configuration for Nesterov's accelerated method when we know precisely the condition number of the convex problem, $\kappa = \frac{L}{\mu}$. The recursion satisfies the following:

$$x_{t+1} = y_t - \frac{1}{L} \nabla f(y_t)$$
$$y_{t+1} = x_{t+1} + \beta^\star(x_{t+1} - x_t),$$

where

$$\beta^\star = \frac{1 - \sqrt{\frac{\mu}{L}}}{1 + \sqrt{\frac{\mu}{L}}} = \frac{1 - \sqrt{\frac{1}{\kappa}}}{1 + \sqrt{\frac{1}{\kappa}}}$$

Let us define also $q^\star = \frac{1}{\kappa}$. The above recursion is optimal, and the proof is omitted; by optimal, we mean that there is a constant step size along with this momentum parameter that achieves the lower bounds. However, it requires the exact knowledge of the Lipschitz gradient continuity parameter L and strong convex parameter μ. Also, note that this selection is optimal, assuming convexity.

For the second one, we will assume that the gradient calculation step includes some noise. As before, we assume that the function satisfies Lipschitz gradient continuity. One natural way to think of this is to assume that we compute only a *noisy* version of the gradient:

$$\widetilde{\nabla} f(y_t) = \nabla f(y_t) + \xi.$$

We will need the following definition of inexact first-order oracle for the theory. In the noiseless case, we know that:

$$0 \leq f(y) - f(x) - \langle \nabla f(x), y - x \rangle \leq \tfrac{L}{2}\|x - y\|_2^2$$

Pictorially, at every point x, the function can be "sandwiched" between a tangent linear function, $\langle \nabla f(x), y - x \rangle$, and a parabola. For the inexact oracle, we will assume the same inequality holds with some slack $\delta > 0$:

$$0 \leq f(y) - f(x) - \langle \nabla f(x), y - x \rangle \leq \tfrac{L}{2}\|x - y\|_2^2 + \delta$$

Pictorially, it comes with the same illustration, except now there's some slack between the linear approximation and the parabola. Let us now describe these interesting phenomena. *Acceleration often leads to a non-decreasing sequence of function values.* It is common, when running an accelerated method, to have the appearance of ripples in the trace of the objective value; these are seemingly regular increases in the objective. The following figure is borrowed from [51] by O'Donoghue and Candes.

The function we are optimizing here is a simple quadratic function:

$$f(x) = \frac{1}{2}x^\top Ax,$$

where A is a positive definite matrix. First, observe that, in this case,

$$\min_x f(x)$$

has optimal solution $x^\star = 0$, and $f(x^\star) = 0$. Further, the Lipschitz gradient continuity parameter satisfies $L = \lambda_{\max}(A)$, and the strong convexity parameter satisfies $\mu = \lambda_{\min}(A)$.

Let's extract some information from the plot. The case where $q = 1$ leads to:

$$y_{t+1} = x_{t+1} + \frac{1-\sqrt{q}}{1+\sqrt{q}}(x_{t+1} - x_t) = x_{t+1}$$

and thus the accelerated version boils down to:

$$x_{t+1} = x_t - \frac{1}{L}\nabla f(x_t),$$

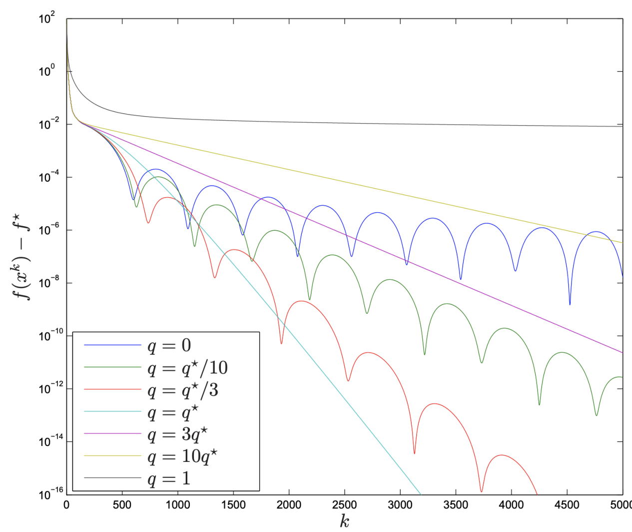

Fig. 42. Behavior of optimal's accelerated method for a convex function, where we do not set up the q parameter correctly (in other words, we only approximate the values L and μ).

the gradient descent method. Also, assuming that the momentum parameter takes values in $[0, 1]$, the maximum parameter case is when $q = 0$, where:

$$\beta = \frac{1-\sqrt{q}}{1+\sqrt{q}} = 1.$$

Ranging the value of β, we observe an interesting phenomenon. Starting with $q = 1$ (i.e., $\beta = 0$), we obtain the behavior of gradient descent, which from the figure shows the worst performance (in terms of iteration complexity). On the other end, for $q = 0$, we obtain the maximum β value that definitely "beats" gradient descent, but there are different values of β, between the values 0 and 1, that gives a better performance.

More importantly, we observe these interesting ripples in the plots: the function values do not monotonically decrease as the iterations increase but rather follow a periodic pattern. However, despite this behavior, the function values decrease faster than plain gradient descent. Of course, as expected, the optimal performance—without any ripples—is achieved by $q^\star$.

Overall, slightly over- or under-estimating the optimal value q (or equivalently of κ) leads to a presumably severe detrimental effect on the rate of convergence of the algorithm. Note the clear difference between the cases where we underestimate ($q < q^\star$) and where we overestimate ($q > q^\star$): in the former, we observe this rippling behavior in the function traces, while in the latter, we observe the classical monotonic convergence.

To understand better what is happening during the ripples, we also provide the following plot from the same paper by O'Donoghue and Candes. The high momentum values cause the trajectory towards the optimum $x^\star$ to overshoot and oscillate around it. This causes a rippling in the function values along the trajectory as we get closer but then move further away from the optimum.

What about Nesterov's routines on selecting β_t? Someone would wonder "*what happens when we use the routine:*

$$\theta_0 = 1,\ \theta_{t+1} = \frac{1+\sqrt{1+4\theta_t^2}}{2},\ \beta_t = \frac{\theta_t - 1}{\theta_{t+1}}"$$

It turns out that, as the iterations increase, the β_t values keep growing towards the maximum value 1, as shown in the plot next. Thus, Nesterov's approach naturally often leads to a rippling behavior we observe in practice.

What could be a solution to this? (Adaptive) restarts of the momentum β procedure. One approach to avoid ripples is occasionally restarting the β_t computation procedure. E.g., one natural check we can make is to check at every new point whether the function value starts increasing; in that case, we can reset $\theta_{t+1} = 0$ and compute a new set of β's. But do these techniques work in practice? It turns out they do!

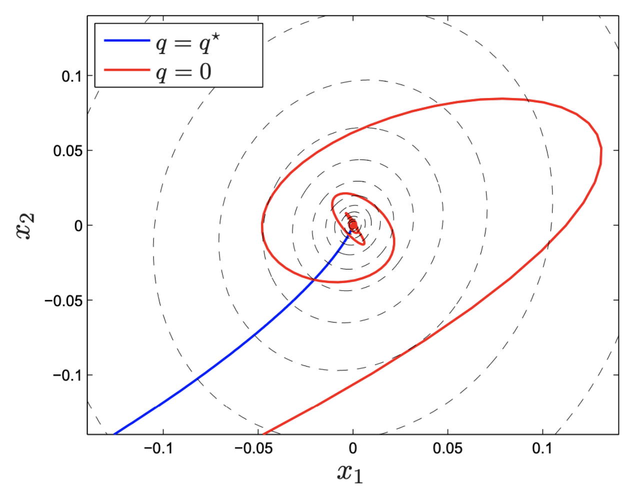

Fig. 43. Comparison of behavior between optimal $q^\star$ and maximum momentum parameter ($q = 0$) for a 2-dimensional toy example.

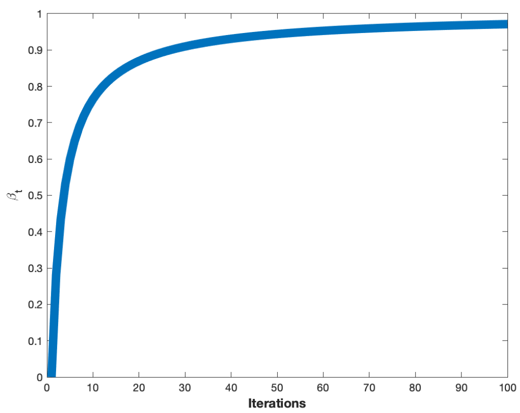

Fig. 44. β_t values w.r.t. number of iterations, according to the rule $\theta_0 = 1$, $\theta_{t+1} = \frac{1+\sqrt{1+4\theta_t^2}}{2}$, $\beta_t = \frac{\theta_t - 1}{\theta_{t+1}}$.

Behavior of acceleration under noisy settings. The point of this subsection is that simple GD is more noise-tolerant than accelerated methods. The noise tolerance corresponds to the case where we might not be able to compute the gradient exactly but have a rough approximation.

This statement is based on the work by Devolder, Glineur, and Nesterov [52]. The main idea is that, even if accelerated GD converges faster than the plain GD, it must also accumulate errors faster (linearly) with the number of iterations.

Let us consider a noisy version of the above experiment. In particular, instead of computing exactly $\nabla f(x) = Ax - b$ per iteration, we see $\nabla f(x) + \xi = Ax - b + \xi$ where ξ is a vector sampled from the n-dimensional normal distribution. Let us see how this performs in practice.

(*See ipython notebook.*)

But what can we say theoretically about this phenomenon? It turns out that what [52] shows is that, for an inexact first-order oracle that satisfies the Lipschitz gradient continuity with slack δ, we can hope for:

$$f(x_t) - \min_x f(x) \leq O\left(\tfrac{L}{t}\right) + \delta.$$

I.e., while we know that we decrease the error at a rate $O(\frac{1}{T})$, we cannot "beat" the fact that there is an error every step, and we cannot reduce the error more than within a δ radius around the optimum.

On the other hand, what acceleration probably gives us is the following:

$$f(x_t) - \min_x f(x) \leq O\left(\tfrac{L}{t^2}\right) + t \cdot \delta.$$

I.e., the same story holds but, at the same time, the error level that we want to "beat" increases with the number of iterations (i.e., $t_1\delta < t_2\delta$ for any $t_1 < t_2$). Thus, acceleration accumulates errors more quickly while converging faster in a noiseless setting. (*See ipython notebook.*)

—————— ∞ ——————

ODEs. Nesterov's methods can be represented as ordinary differential equations (ODEs). ODEs have long been connected with optimization. The connection between ODEs and numerical optimization is often made by having small step sizes so that the trajectory or solution path converges to a curve modeled by an ODE. An ODE can model Nesterov's algorithm in this manner. More precisely, the ODE representation of the first-order method is a second-order ODE:

$$\ddot{X} + \frac{3}{t}\dot{X} + \nabla f(X) = 0$$

for $t > 0$, with initial conditions $X(0) = x_0, \dot{X}(0) = 0$ where x_0 is the starting point in Nesterov's algorithm. $\dot{X} = \frac{dX}{dt}$ denotes the time derivative or velocity and $\ddot{X} = \frac{d^2X}{dt^2}$ denotes the acceleration. The time parameter in this ODE is related to the step size.

The Chebyshev Method. Here, we will continue our discussion at the end of Chapter 3. As a reminder, we consider the minimization problem of the function:

$$f(x) = \tfrac{1}{2}x^\top Qx - b^\top x + r,$$

where $x \in \mathbb{R}^p$, $Q \in \mathbb{R}^{p\times p}$ is a symmetric matrix, $b \in \mathbb{R}^p$ is a vector and r is a scalar. I.e.,

$$\min_{x\in\mathbb{R}^p} f(x).$$

Here, we follow the discussion in this chapter, where $\mu \cdot I \preceq Q \preceq L \cdot I$. Further, we know that $\nabla f(x) = Qx - b = Q(x - x^\star)$, assuming that $x^\star$ solves the problem and thus $Qx^\star = b$.

As we implied in Chapter 3, what matters in first-order methods in quadratic function minimization is the following problem:

$$P_t^\star = \arg\min_{P:P(0)=1}\max_{Q\in\mathcal{Q}} \|P(Q)\|_2,$$

where in the case where $\mu \cdot I \preceq Q \preceq L \cdot I$ turns out to be:

$$P_t^\star = \arg\min_{P:P(0)=1}\max_{\lambda\in[\mu,L]} \|P(\lambda)\|_2.$$

It is known (out of the scope of this course) that polynomials that solve the above problem are the *Chebyshev polynomials of the first kind*; for more detailed discussion, please look into approximation theory results. Chebyshev polynomials of the first kind satisfy the following equations:

$$\begin{aligned}
&\mathcal{T}_0(x) = 1,\\
&\mathcal{T}_1(x) = x,\\
&\mathcal{T}_t(x) = 2x\mathcal{T}_{t-1}(x) - \mathcal{T}_{t-2}(x), \quad \text{for } t \geq 2.
\end{aligned}$$

The above expressions define just a recursion; i.e., these are the conditions a specific polynomial should satisfy (it is not a constructive argument, but an existential one). However, an explicit solution could be (out of the scope of this course):

$$\mathcal{T}_t(x) = \begin{cases} \cos(t\cdot \text{acos}(x)), & x \in [-1,1],\\ \cosh(t\cdot\text{acosh}(x)), & x > 1,\\ (-1)^t\cdot\cosh(t\cdot\text{acosh}(-x)), & x < 1.\end{cases}$$

It is a fact that this solution satisfies[11]:

$$\frac{\mathcal{T}_t}{2^{t-1}} = \arg\min_P \max_{\lambda\in[-1,1]} \|P(\lambda)\|_2,$$

that satisfies our original problem, but for a scaled version: instead of $\lambda \in [\mu, L]$, we have $\lambda \in [-1, 1]$. Simple linear mapping

[11] In fact, this polynomial satisfies the minimax property as a monic polynomial, i.e., a polynomial whose coefficient associated with the highest power is equal to one.

arguments from $[\mu, L]$ to $[-1, 1]$, lead to the shifted Chebyshev polynomials:

$$\mathcal{C}_t^{[\mu,L]}(x) = \frac{\mathcal{T}_t\left(\frac{2x-(L+\mu)}{L-\mu}\right)}{\mathcal{T}_t\left(\frac{-(L+\mu)}{L-\mu}\right)}$$

that optimize the original polynomial problem.

Going back to the original definition of Chebyshev polynomials, we get:

$$\begin{aligned}
&\mathcal{C}_0^{[\mu,L]}(x) = 1, \\
&\mathcal{C}_1^{[\mu,L]}(x) = 1 - \tfrac{2x}{L+\mu}, \\
&\mathcal{C}_t^{[\mu,L]}(x) = \tfrac{2\delta_t}{L-\mu} \cdot (L+\mu-2x)\mathcal{C}_{t-1}^{[\mu,L]}(x) \\
&\qquad\qquad + \left(1 + \tfrac{2\delta_t(L+\mu}{L-\mu}\right)\mathcal{C}_{t-2}^{[\mu,L]}(x), \quad \text{for } t \geq 2,
\end{aligned}$$

where $\delta_1 = \frac{L-\mu}{L+\mu}$ and:

$$\delta_t = \frac{1}{2\frac{L+\mu}{L-\mu} - \delta_{t-1}}, \quad \text{for } t \geq 2.$$

But how is this useful? Let us go back to the original gradient-based formulation from Chapter 3, where we had:

$$x_t - x^\star = P_t(Q) \cdot (x_0 - x^\star).$$

Here, $P_t(Q)$ is any polynomial that satisfies the constraints; thus, we can substitute this with the Chebyshev-based polynomial to get:

$$x_t - x^\star = \mathcal{C}_t^{[\mu,L]}(x) \cdot (x_0 - x^\star).$$

Using the definition of the polynomial $\mathcal{C}_t^{[\mu,L]}(x)$, we obtain the recursion:

$$\begin{aligned}
x_t - x^\star = &\tfrac{2\delta_t}{L-\mu} \cdot ((L+\mu) \cdot I - 2Q)(x_{t-1} - x^\star) \\
&+ \left(1 + \tfrac{2\delta_t(L+\mu}{L-\mu}\right)(x_{t-2} - x^\star)
\end{aligned}$$

Since the gradient for the quadratic functions satisfies: $\nabla f(x_t) = Q(x_t - x^\star)$, we finally obtain:

$$\begin{aligned}
x_t &= \tfrac{2\delta_t}{L-\mu} \cdot ((L+\mu) \cdot x_{t-1} - 2\nabla f(x_{t-1})) \\
&\qquad + \left(1 + \tfrac{2\delta_t(L+\mu}{L-\mu}\right)x_{t-2} \\
&= x_{t-1} - \tfrac{4\delta_t}{L-\mu}\nabla f(x_{t-1}) \\
&\qquad + \left(1 + \tfrac{2\delta_t(L+\mu}{L-\mu}\right)(x_{t-2} - x_{t-1})
\end{aligned}$$

Compare this expression with the Heavy-Ball's expression:

$$x_t = x_{t-1} - \eta\nabla f(x_{t-1}) + \beta(x_{t-1} - x_{t-2}).$$

There are some resemblances! In fact, assuming $t \to \infty$, one can solve the equation:

$$\delta_\infty = \frac{1}{2\frac{L+\mu}{L-\mu} - \delta_\infty}$$

to obtain a value for the $\delta_\infty = \frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}$ that leads to the following expression for Chebyshev method:

$$\begin{aligned}
x_t = x_{t-1} &- \tfrac{4}{(\sqrt{L}-\sqrt{\mu})^2}\nabla f(x_{t-1}) \\
&+ \tfrac{(\sqrt{L}-\sqrt{\mu})^2}{(\sqrt{L}+\sqrt{\mu})^2}(x_{t-1} - x_{t-2})
\end{aligned}$$

which is exactly the Polyak's Heavy-Ball method! (*Overall, the intersection of approximation theory, function/real analysis, and optimization is a fruitful research area with broad open questions.*)

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. Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In Neural networks: Tricks of the trade, pages 9–50. Springer, 2002.
16. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision, pages 1026–1034, 2015.
17. Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. Advances in neural information processing systems, 31, 2018.
18. 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.
19. 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.
20. 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.
21. 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.
22. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.
23. 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.
24. 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.
25. 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.
26. 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.
27. 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.
28. 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.
29. 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.
30. 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.
31. 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.
32. Gary Marcus, Ernest Davis, and Scott Aaronson. A very preliminary analysis of DALL-E 2. arXiv preprint arXiv:2204.13807, 2022.
33. 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.
34. 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.
35. Or Sharir, Barak Peleg, and Yoav Shoham. The cost of training nlp models: A concise overview. arXiv preprint arXiv:2004.08900, 2020.
36. 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.
37. Philip Wolfe. Convergence conditions for ascent methods. SIAM review, 11(2):226–235, 1969.
38. Larry Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. Pacific Journal of mathematics, 16(1):1–3, 1966.
39. Stephen Wright and Jorge Nocedal. Numerical optimization. Springer Science, 35(67-68):7, 1999.
40. B. Polyak. Introduction to optimization. Inc., Publications Division, New York, 1, 1987.
41. Stephen Boyd, Lin Xiao, and Almir Mutapcic. Subgradient methods. lecture notes of EE392o, Stanford University, Autumn Quarter, 2004:2004–2005, 2003.
42. Marguerite Frank, Philip Wolfe, et al. An algorithm for quadratic programming. Naval research logistics quarterly, 3(1-2):95–110, 1956.
43. 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.
44. 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.
45. Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. Computer, (8):30–37, 2009.
46. A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In Advances in neural information processing systems, pages 1257–1264, 2008.
47. 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.
48. S. Zavriev and F. Kostyuk. Heavy-ball method in nonconvex optimization problems. Computational Mathematics and Modeling, 4(4):336–341, 1993.
49. 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.
50. Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(\frac{1}{k^2})$. In Soviet Mathematics Doklady, volume 27, pages 372–376, 1983.
51. B. O'Donoghue and E. Candes. Adaptive restart for accelerated gradient schemes. Foundations of computational mathematics, 15(3):715–732, 2015.
52. 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.
53. L. Bottou, F. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. Siam Review, 60(2):223–311, 2018.
54. S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. SIAM review, 43(1):129–159, 2001.
55. R. Tibshirani. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological), 58(1):267–288, 1996.
56. P. Hoff. Lasso, fractional norm and structured sparse estimation using a Hadamard product parametrization. Computational Statistics & Data Analysis, 115:186–198, 2017.
57. S. Becker, J. Bobin, and E. Candès. NESTA: A fast and accurate first-order method for sparse recovery. SIAM Journal on Imaging Sciences, 4(1):1–39, 2011.
58. T. Blumensath and M. Davies. Iterative hard thresholding for compressed sensing. Applied and computational harmonic analysis, 27(3):265–274, 2009.
59. D. Needell and J. Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. Applied and computational harmonic analysis, 26(3):301–321, 2009.
60. S. Foucart. Hard thresholding pursuit: an algorithm for compressive sensing. SIAM Journal on Numerical Analysis, 49(6):2543–2563, 2011.
61. J. Tanner and K. Wei. Normalized iterative hard thresholding for matrix completion. SIAM Journal on Scientific Computing, 35(5):S104–S125, 2013.
62. K. Wei. Fast iterative hard thresholding for compressed sensing. IEEE Signal processing letters, 22(5):593–597, 2014.
63. Rajiv Khanna and Anastasios Kyrillidis. Iht dies hard: Provable accelerated iterative hard thresholding. In International Conference on Artificial Intelligence and Statistics, pages 188–198. PMLR, 2018.
64. Jeffrey D Blanchard and Jared Tanner. GPU accelerated greedy algorithms for compressed sensing. Mathematical Programming Computation, 5(3):267–304, 2013.
65. A. Kyrillidis, G. Puy, and V. Cevher. Hard thresholding with norm constraints. In 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3645–3648. Ieee, 2012.
66. A. Kyrillidis and V. Cevher. Recipes on hard thresholding methods. In Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2011 4th IEEE International Workshop on, pages 353–356. IEEE, 2011.

67. X. Zhang, Y. Yu, L. Wang, and Q. Gu. Learning one-hidden-layer ReLU networks via gradient descent. In The 22nd International Conference on Artificial Intelligence and Statistics, pages 1524–1534, 2019.
68. Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. IEEE Transactions on information theory, 52(2):489–509, 2006.
69. Joachim Dahl, Lieven Vandenberghe, and Vwani Roychowdhury. Covariance selection for nonchordal graphs via chordal embedding. Optimization Methods & Software, 23(4):501–520, 2008.
70. Joseph B Altepeter, Daniel FV James, and Paul G Kwiat. 4 qubit quantum state tomography. In Quantum state estimation, pages 113–145. Springer, 2004.
71. Jens Eisert, Dominik Hangleiter, Nathan Walk, Ingo Roth, Damian Markham, Rhea Parekh, Ulysse Chabaud, and Elham Kashefi. Quantum certification and benchmarking. arXiv preprint arXiv:1910.06343, 2019.
72. Masoud Mohseni, AT Rezakhani, and DA Lidar. Quantum-process tomography: Resource analysis of different strategies. Physical Review A, 77(3):032322, 2008.
73. D. Gross, Y.-K. Liu, S. Flammia, S. Becker, and J. Eisert. Quantum state tomography via compressed sensing. Physical review letters, 105(15):150401, 2010.
74. Y.-K. Liu. Universal low-rank matrix recovery from Pauli measurements. In Advances in Neural Information Processing Systems, pages 1638–1646, 2011.
75. K Vogel and H Risken. Determination of quasiprobability distributions in terms of probability distributions for the rotated quadrature phase. Physical Review A, 40(5):2847, 1989.
76. Miroslav Ježek, Jaromír Fiurášek, and Zdeněk Hradil. Quantum inference of states and processes. Physical Review A, 68(1):012305, 2003.
77. Konrad Banaszek, Marcus Cramer, and David Gross. Focus on quantum tomography. New Journal of Physics, 15(12):125020, 2013.
78. A. Kalev, R. Kosut, and I. Deutsch. Quantum tomography protocols with positivity are compressed sensing protocols. Nature partner journals (npj) Quantum Information, 1:15018, 2015.
79. Giacomo Torlai, Guglielmo Mazzola, Juan Carrasquilla, Matthias Troyer, Roger Melko, and Giuseppe Carleo. Neural-network quantum state tomography. Nat. Phys., 14:447–450, May 2018.
80. Matthew JS Beach, Isaac De Vlugt, Anna Golubeva, Patrick Huembeli, Bohdan Kulchytskyy, Xiuzhe Luo, Roger G Melko, Ejaaz Merali, and Giacomo Torlai. Qucumber: wavefunction reconstruction with neural networks. SciPost Physics, 7(1):009, 2019.
81. Giacomo Torlai and Roger Melko. Machine-learning quantum states in the NISQ era. Annual Review of Condensed Matter Physics, 11, 2019.
82. M. Cramer, M. B. Plenio, S. T. Flammia, R. Somma, D. Gross, S. D. Bartlett, O. Landon-Cardinal, D. Poulin, and Y.-K. Liu. Efficient quantum state tomography. Nat. Comm., 1:149, 2010.
83. BP Lanyon, C Maier, Milan Holzäpfel, Tillmann Baumgratz, C Hempel, P Jurcevic, Ish Dhand, AS Buyskikh, AJ Daley, Marcus Cramer, et al. Efficient tomography of a quantum many-body system. Nature Physics, 13(12):1158–1162, 2017.
84. D. Gonçalves, M. Gomes-Ruggiero, and C. Lavor. A projected gradient method for optimization over density matrices. Optimization Methods and Software, 31(2):328–341, 2016.
85. E. Bolduc, G. Knee, E. Gauger, and J. Leach. Projected gradient descent algorithms for quantum state tomography. npj Quantum Information, 3(1):44, 2017.
86. Jiangwei Shang, Zhengyun Zhang, and Hui Khoon Ng. Superfast maximum-likelihood reconstruction for quantum tomography. Phys. Rev. A, 95:062336, Jun 2017.
87. Zhilin Hu, Kezhi Li, Shuang Cong, and Yaru Tang. Reconstructing pure 14-qubit quantum states in three hours using compressive sensing. IFAC-PapersOnLine, 52(11):188 – 193, 2019. 5th IFAC Conference on Intelligent Control and Automation Sciences ICONS 2019.
88. Zhibo Hou, Han-Sen Zhong, Ye Tian, Daoyi Dong, Bo Qi, Li Li, Yuanlong Wang, Franco Nori, Guo-Yong Xiang, Chuan-Feng Li, et al. Full reconstruction of a 14-qubit state within four hours. New Journal of Physics, 18(8):083036, 2016.
89. C. Riofrío, D. Gross, S.T. Flammia, T. Monz, D. Nigg, R. Blatt, and J. Eisert. Experimental quantum compressed sensing for a seven-qubit system. Nature Communications, 8, 2017.
90. Martin Kliesch, Richard Kueng, Jens Eisert, and David Gross. Guaranteed recovery of quantum processes from few measurements. Quantum, 3:171, 2019.
91. S. Flammia, D. Gross, Y.-K. Liu, and J. Eisert. Quantum tomography via compressed sensing: Error bounds, sample complexity and efficient estimators. New Journal of Physics, 14(9):095022, 2012.
92. A. Kyrillidis, A. Kalev, D. Park, S. Bhojanapalli, C. Caramanis, and S. Sanghavi. Provable quantum state tomography via non-convex methods. npj Quantum Information, 4(36), 2018.
93. B. Recht, M. Fazel, and P. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. SIAM review, 52(3):471–501, 2010.
94. N. Srebro, J. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. In Advances in neural information processing systems, pages 1329–1336, 2004.
95. J. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In Proceedings of the 22nd international conference on Machine learning, pages 713–719. ACM, 2005.
96. D. DeCoste. Collaborative prediction using ensembles of maximum margin matrix factorizations. In Proceedings of the 23rd international conference on Machine learning, pages 249–256. ACM, 2006.
97. J. Bennett and S. Lanning. The Netflix prize. In Proceedings of KDD cup and workshop, volume 2007, page 35, 2007.
98. M. Jaggi and M. Sulovsk. A simple algorithm for nuclear norm regularized problems. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), pages 471–478, 2010.
99. R. Keshavan. Efficient algorithms for collaborative filtering. PhD thesis, Stanford University, 2012.
100. R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In Proceedings of the 22nd international conference on World Wide Web, pages 13–24. International World Wide Web Conferences Steering Committee, 2013.
101. K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain. Sparse local embeddings for extreme multi-label classification. In Advances in Neural Information Processing Systems, pages 730–738, 2015.
102. G. Carneiro, A. Chan, P. Moreno, and N. Vasconcelos. Supervised learning of semantic classes for image annotation and retrieval. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 29(3):394–410, 2007.
103. A. Makadia, V. Pavlovic, and S. Kumar. A new baseline for image annotation. In Computer Vision–ECCV 2008, pages 316–329. Springer, 2008.
104. C. Wang, S. Yan, L. Zhang, and H.-J. Zhang. Multi-label sparse coding for automatic image annotation. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 1643–1650. IEEE, 2009.
105. J. Weston, S. Bengio, and N. Usunier. WSABIE: Scaling up to large vocabulary image annotation. In IJCAI, volume 11, pages 2764–2770, 2011.
106. Andrew I. Schein, Lawrence K. Saul, and Lyle H. Ungar. A generalized linear model for principal component analysis of binary data. In AISTATS, 2003.
107. K.-Y. Chiang, C.-J. Hsieh, N. Natarajan, I. Dhillon, and A. Tewari. Prediction and clustering in signed networks: A local to global perspective. The Journal of Machine Learning Research, 15(1):1177–1213, 2014.
108. C. Johnson. Logistic matrix factorization for implicit feedback data. Advances in Neural Information Processing Systems, 27, 2014.
109. Koen Verstrepen. Collaborative Filtering with Binary, Positive-only Data. PhD thesis, University of Antwerpen, 2015.
110. N. Gupta and S. Singh. Collectively embedding multi-relational data for predicting user preferences. arXiv preprint arXiv:1504.06165, 2015.
111. Y. Liu, M. Wu, C. Miao, P. Zhao, and X.-L. Li. Neighborhood regularized logistic matrix factorization for drug-target interaction prediction. PLoS Computational Biology, 12(2):e1004760, 2016.
112. S. Aaronson. The learnability of quantum states. In Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, volume 463, pages 3089–3114. The Royal Society, 2007.
113. E. Candes, Y. Eldar, T. Strohmer, and V. Voroninski. Phase retrieval via matrix completion. SIAM Review, 57(2):225–251, 2015.
114. I. Waldspurger, A. d'Aspremont, and S. Mallat. Phase recovery, MaxCut and complex semidefinite programming. Mathematical Programming, 149(1-2):47–81, 2015.
115. P. Biswas, T.-C. Liang, K.-C. Toh, Y. Ye, and T.-C. Wang. Semidefinite programming approaches for sensor network localization with noisy distance measurements. IEEE transactions on automation science and engineering, 3(4):360, 2006.
116. K. Weinberger, F. Sha, Q. Zhu, and L. Saul. Graph Laplacian regularization for large-scale semidefinite programming. In Advances in Neural Information Processing Systems, pages 1489–1496, 2007.
117. F. Lu, S. Keles, S. Wright, and G. Wahba. Framework for kernel regularization with application to protein clustering. Proceedings of the National Academy of Sciences of the United States of America, 102(35):12332–12337, 2005.
118. H. Andrews and C. Patterson III. Singular value decomposition (SVD) image coding. Communications, IEEE Transactions on, 24(4):425–432, 1976.
119. M. Fazel, H. Hindi, and S. Boyd. Rank minimization and applications in system theory. In American Control Conference, 2004. Proceedings of the 2004, volume 4, pages 3273–3278. IEEE, 2004.
120. E. Candès and B. Recht. Exact matrix completion via convex optimization. Foundations of Computational mathematics, 9(6):717–772, 2009.
121. P. Jain, R. Meka, and I. Dhillon. Guaranteed rank minimization via singular value projection. In Advances in Neural Information Processing Systems, pages 937–945, 2010.
122. S. Becker, V. Cevher, and A. Kyrillidis. Randomized low-memory singular value projection. In 10th International Conference on Sampling Theory and Applications (Sampta), 2013.
123. L. Balzano, R. Nowak, and B. Recht. Online identification and tracking of subspaces from highly incomplete information. In Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on, pages 704–711. IEEE, 2010.
124. K. Lee and Y. Bresler. ADMiRA: Atomic decomposition for minimum rank approximation. Information Theory, IEEE Transactions on, 56(9):4402–4416, 2010.
125. A. Kyrillidis and V. Cevher. Matrix recipes for hard thresholding methods. Journal of mathematical imaging and vision, 48(2):235–265, 2014.
126. Z. Lin, M. Chen, and Y. Ma. The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices. arXiv preprint arXiv:1009.5055, 2010.
127. S. Becker, E. Candès, and M. Grant. Templates for convex cone problems with applications to sparse signal recovery. Mathematical Programming Computation, 3(3):165–218, 2011.
128. J. Cai, E. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. SIAM Journal on Optimization, 20(4):1956–1982, 2010.
129. Y. Chen, S. Bhojanapalli, S. Sanghavi, and R. Ward. Coherent matrix completion. In Proceedings of The 31st International Conference on Machine Learning, pages 674–682, 2014.

130. A. Yurtsever, Q. Tran-Dinh, and V. Cevher. A universal primal-dual convex optimization framework. In Advances in Neural Information Processing Systems 28, pages 3132–3140. 2015.

131. F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6):386–408, 1958.

132. Robin M. Schmidt, Frank Schneider, and Philipp Hennig. Descending through a crowded valley - benchmarking deep learning optimizers. CoRR, abs/2007.01547, 2020.

133. John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res., 12(null):2121–2159, jul 2011.

134. Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc' aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc Le, and Andrew Ng. Large scale distributed deep networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 25. Curran Associates, Inc., 2012.

135. Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.