IFT 6085 - Lecture 6
Nesterov's Momentum, Stochastic Gradient Descent

This version of the notes has not yet been thoroughly checked. Please report any bugs to the scribes or instructor.

Scribes:
Winter 2020: Manuela Girotti
Winter 2019: Yann Bouteiller & Remi Piche-Taillefer
Winter 2018: Charles Ashby & Gabriele Prato

Instructor: Ioannis Mitliagkas

1 Summary

This lecture covers the following elements of optimization theory:

- Failing case of Polyak's momentum
- Nesterov's momentum
- Stochastic gradient descent

Most of the lecture has been adapted from [1], [2], [3] and [4].

2 Failing case of Polyak's Momentum

In the previous lecture we presented Polyak's momentum algorithm (or heavy-ball method), in which the iteration step is given by:

$$x_{t+1} = x_t - \alpha\nabla f(x_t) + \beta(x_t - x_{t-1}), \quad \alpha > 0, \beta \in [0, 1] \tag{1}$$

This method is known to be robust and effective in many real-world applications. However, in a paper published in 2015, Lessard et al. [2] have shown that there exist strongly-convex and smooth functions for which, by choosing carefully the hyperparameters α and β and the initial condition x_0, the heavy-ball method fails to converge. We give here merely the intuition behind the malfunction of the algorithm as well as empirical results taken from the paper.

Let f be a piece-wise quadratic function whose gradient is:

$$\nabla f(x) = \begin{cases} 25x & x < 1 \\ x + 24 & 1 \leq x < 2 \\ 25x - 24 & x \geq 2 \end{cases} \tag{2}$$

By construction, $\forall\, x_1, x_2, \|\nabla f(x_1) - \nabla f(x_2)\| \leq 25\|x_1 - x_2\|$, therefore f is 25-smooth, and

$$\nabla^2 f(x) = \begin{cases} 25 & x < 1 \\ 1 & \leq x < 2 \\ 25 & x \geq 2 \end{cases} \tag{3}$$

implying $\nabla^2 f \geq 1 > 0$, therefore f is 1-strongly convex.

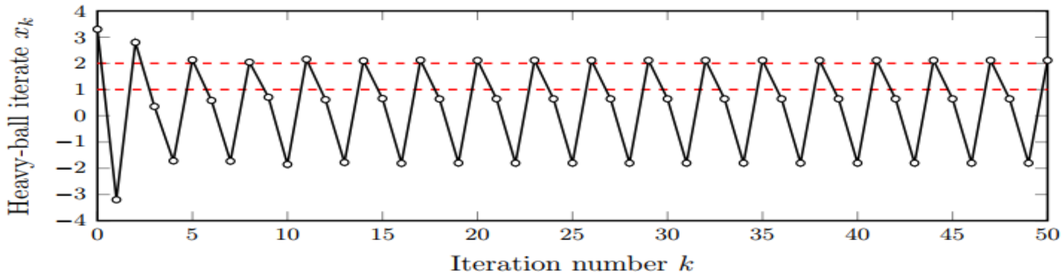

Figure 1: Value of x_k for the 50 first iterations of Polyak's momentum algorithm. The dashed red lines separate the pieces of $f(x)$. Taken from Lessard et al. [2].

In Figure 1 are displayed the first 50 iterates of Polyak's momentum algorithm applied to f, using $\alpha = \frac{1}{9}$, $\beta = \frac{4}{9}$ and $x_0 = 3.3$. The output values are bound to cycle through 3 points indefinitely.
We can see that despite the function f being 1-strongly convex and 25-smooth, the Polyak's GD map Φ_{P-GD} has a periodic point of period 3 and its flow is an attractive limit-cycle (see the proof in [2], Appendix B). In particular, the authors in [2] show that with the parameters α and β given as in Figure 1, there exists a sequence of iterates $\{x_t\}$ such that as $n \to \infty$

$$x_{t=3n} \to p \approx 0.65, \quad x_{t=3n+1} \to q \approx -1.80, \quad x_{t=3n+2} \to r \approx 2.12.$$

See Figure 2 for a visalization of the effect of the map Φ_{P-GD} on the cycle $\{p, q, r\}$.

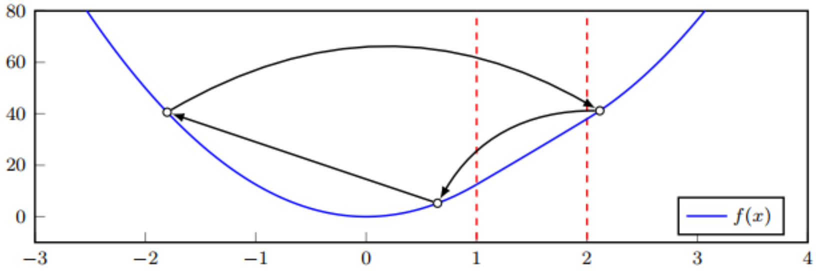

Figure 2: Illustration of the limit values of the failing case of Polyak's momentum algorithm, the dashed red lines separate the pieces of f. Image taken from Lessard et al. [2].

2.1 Food for thought

The result shown in [2] is certainly helpful in understanding the powerfulness and the limitations of Polyak's algorithm. However, it is worth pointing out that Polyak's momentum GD has guaranteed convergence for quadratic functions (and not piece-wise quadratic).

Let us look at it from a broader perspective. Consider the family of continuous functions $\{f_{\omega,\epsilon}\}$ with gradient

$$\nabla f_{\omega,\epsilon}(x) = \begin{cases} 25x & x < 1 \\ \omega x + 24 & 1 \leq x < 1 + \epsilon \\ 25x - 24 & x \geq \epsilon \end{cases} \tag{4}$$

This is a set of piece-wise continuous and piece-wise quadratic functions, where $\omega > 0$ represents the curvature of the function within a (small) window of size $\epsilon \geq 0$. The counterexample from [2] falls into this set with $\omega = \epsilon = 1$.

By construction, each function in this family is α-strongly convex and 25-Lipschitz (let's assume $\omega < 25$): therefore its condition number is $\kappa = \frac{25}{\omega}$, which can be arbitrarily high for very small values of ω.

On the other hand, $\forall\ \omega > 0$ fixed, $f_{\omega,\epsilon} \to f_0$ uniformly, where f_0 is the corresponding quadratic function with window-size $\epsilon = 0$, which clearly has condition number $\kappa = 1$.

Considering the Polyak's algorithm on this family of functions and their limit, we have optimal tuning for f_0 with step size $\alpha = \frac{1}{25}$ and momentum $\beta = 0$, while for $f_{\omega,\epsilon}$ the optimal tuning gives a value for the momentum

$$\beta = \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^2 \approx 1 \qquad \text{for large values of } \omega.$$

This mismatch between condition numbers of $\{f_{\omega,\epsilon}\}$ and their limit f_0, while a uniform convergence result holds, may suggest that for some families of functions Polyak's tuning is problematic. Otherwise said: there could be a better strategy for tuning the step size and the momentum parameter that can still yield convergence; finally, the quest for a better tuning could require a new innovative definition of condition number that would allow to extend convergence results to functions that are not simply quadratic. Indeed, numerical evidence shows that Polyak's momentum has still a lot of potential to be successfully applied to a broad class of functions.

3 Nesterov's Accelerated Gradient Descent

3.1 Motivation

In the last section, we saw that Polyak's momentum algorithm can fail to converge for some carefully built convex optimization problems. Leveraging the idea of momentum introduced by Polyak, Nesterov introduced a slightly altered update rule that has been shown to converge not only for quadratic functions, but for general convex functions. The main ideas of this demonstration are presented at the end of this section.

The Nesterov's Accelerated Gradient algorithm is described as follow by Sutskever et al. [5]:

Algorithm 1 Nesterov's Accelerated Gradient Descent

```
parameters: number of iterations $T$, step size $\alpha$, momentum $\beta$ and initial condition $x_0$.
initialize: $v_0 \leftarrow 0$
for $t = 0, \dots, T-1$ do
    $v_{t+1} \leftarrow \beta v_t - \alpha \nabla f(x_t + \beta v_t)$
    $x_{t+1} \leftarrow x_t + v_{t+1}$
end
return $x_T$
```

The quantity v_{t+1} can be thought of as the velocity at time t or as the displacement vector: it is calculated by applying the momentum β to the previous v_t displacement and subtracting the gradient step to $x_t + \beta v_t$, which is the point where the momentum term leads from x_t. Rewriting these two sequences as one yields:

$$x_{t+1} = x_t + \beta(x_t - x_{t-1}) - \alpha \nabla f(x_t + \beta(x_t - x_{t-1})) \tag{5}$$

Comparing equation (1) with equation (5), we can see that Polyak's method evaluates the gradient before adding momentum, whereas Nesterov's algorithm evaluates it after applying momentum, which intuitively brings us closer to the minimum x^*, as illustrated in Figure 3.

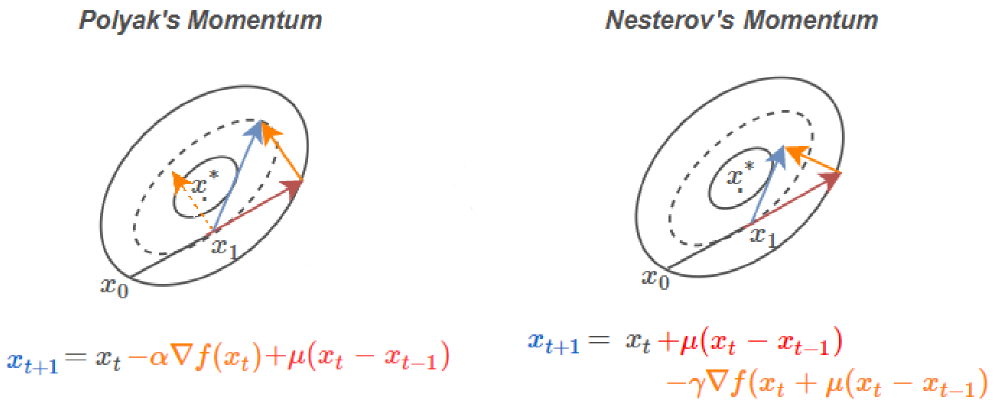
Polyak's Momentum
Nesterov's Momentum
x^*
x_1
x_0
$x_{t+1} = x_t - \alpha \nabla f(x_t) + \mu(x_t - x_{t-1})$
$x_{t+1} = x_t + \mu(x_t - x_{t-1})$
$- \gamma \nabla f(x_t + \mu(x_t - x_{t-1}))$

Figure 3: Comparison between Polyak's and Nesterov's momentum. The gradient descent step (orange arrow) is perpendicular to the level set before applying momentum to x_1 (red arrow) in Polyak's algorithm, whereas it is perpendicular to the level set after applying momentum to x_1 in Nesterov's algorithm.

The convergence rate (upper bound on the sub-optimality) for different classes of functions for gradient descent and Nesterov's accelerated gradient descent are compared in Table 1.

Table 1: Convergence rate for Gradient Descent & Nesterov's Accelerated Gradient

Class of Function	GD	NAG
Smooth	$\mathcal{O}(1/T)$	$\mathcal{O}(1/T^2)$
Smooth & Strongly-Convex	$\mathcal{O}\left(exp\left(-\frac{T}{\kappa}\right)\right)$	$\mathcal{O}\left(exp\left(-\frac{T}{\sqrt{\kappa}}\right)\right)$

Nesterov's Augmented Gradient in the case of smooth and strongly convex functions gives the acceleration that we had with Polyak's momentum for quadratic functions. This is great, because we get the guarantee for a more general class of functions. Recall that if f is μ-strongly convex and λ-smooth then $\kappa = \frac{\lambda}{\mu}$. Therefore, when $\lambda << \mu$ (i.e. some dimensions have very steep gradients), the acceleration becomes very significant.

3.2 General scheme for the Convergence of Nesterov's Accelerated Gradient Descent

In this section, we present the main items behind the proof of convergence for Nesterov's momentum algorithm for general convex functions. In order to achieve this proof, Nesterov uses an **estimate sequence**.

Definition 1 (Estimate Sequence; [3], Definition 2.2.1)**.** *A pair of sequences* $\{\phi_k(x)\}_{k=0}^{\infty}$ *and* $\{\lambda_k\}_{k=0}^{\infty}$ *where* $\lambda_k \geq 0$, *is called an Estimate Sequence of* f *if the following conditions hold:*

- $\lambda_k \to 0$ *when* $k \to \infty$
- $\forall\, x \in \mathbb{R}^n, \forall\, k \geq 0$, *the following holds:*

$$\phi_k(x) \leq (1 - \lambda_k) f(x) + \lambda_k \phi_0(x)$$

Figure 4 illustrates the way an estimate sequence aligns with the f function. The Definition above is suggesting that an estimate sequence has the property that the functions $\{\phi_k\}$ are pointwise "not too far above" the function f itself, for each $k \in \mathbb{N}$:

$$\phi_k(x) \leq f(x) + \lambda_k \left(\phi_0(x) - f(x)\right),$$

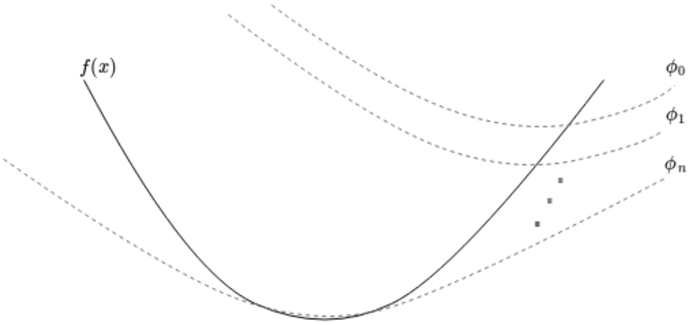

Figure 4: Example of an estimate sequence

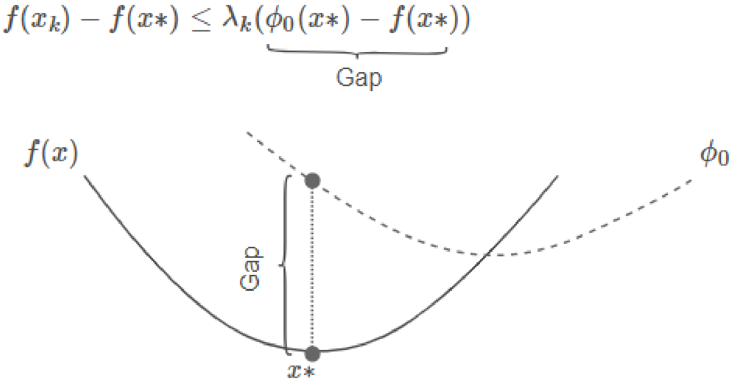
$f(x_k) - f(x*) \leq \lambda_k(\underbrace{\phi_0(x*) - f(x*)}_{\text{Gap}})$
$f(x)$
ϕ_0
Gap
$x*$

Figure 5: Illustration of Theorem 2.

since $\lambda_k \to 0$ and we can assume that the function f and ϕ_0 are sufficiently regular so that the difference $\phi_0(x) - f(x)$ is bounded in norm (at least pointwise). It also follows easily that $\forall\, x \in \mathbb{R}^n$

$$\lim_{k\to\infty} \phi_k(x) \leq f(x), \tag{6}$$

therefore, in the limit as $k \to \infty$, ϕ_k becomes a lower bound of f.

Although the definition of estimate sequence sounds extremely artificial and lacking intuition, Nesterov showed that provided we manage to find such a sequence, then there is an automatic way to construct a sequence of iterates $\{x_k\}$ that successfully converges to the minimizer x^* with rate of convergence given by the rate of convergence of the sequence of $\{\lambda_k\}$ to 0.

Let us start with formalizing the last claim:

Theorem 2 ([3], Theorem 2.2.1)**.** *Given a μ-strongly convex, L-smooth function f on $\mathbb{R}^n$, an estimate sequence $(\{\phi_k(x)\}_{k=0}^{\infty}, \{\lambda_k\}_{k=0}^{\infty})$ of f and a corresponding sequence of iterates $\{x_k\}$ such that*

$$f(x_k) \leq \phi_k^* =: \min_x \phi_k(x), \qquad \forall\, k \in \mathbb{N},$$

then

$$f(x_k) - f(x^*) \leq \lambda_k \left[\phi_0(x^*) - f(x^*)\right].$$

Therefore, provided that Figure 5 presents a visualization of Theorem 2.

In [3] (Lemma 2.2.2 and Lemma 2.2.3), Nesterov already provided us with a recipe for an estimate sequence for a generic μ-strongly convex, L-smooth function f:

1. Pick a starting point x_0;
2. Set $\lambda_0 = 1$, $\phi_0(x) = f(x_0) + \frac{\mu}{2}\|x - x_0\|^2$.
3. Define recursively the rest of the sequence as

$$\lambda_{k+1} = (1 - \alpha_k)\lambda_k, \tag{7}$$

$$\phi_{k+1}(x) = (1 - \alpha_k)\phi_k(x) + \alpha_k \left(f(y_k) + \langle \nabla f(y_k), x - y_k \rangle + \frac{\mu}{2}\|x - y_k\|^2 \right), \tag{8}$$

   where $\{y_k\}_{k=0}^{\infty} \subseteq \mathbb{R}^n$ and $\{\alpha_k\}_{k=0}^{\infty} \subseteq \mathbb{R}$ are two arbitrary control sequences (as of now we only require $\{\alpha_k\}_{k=0}^{\infty}$ to be such that such that $\alpha_k \in (0, 1)$ and $\sum_{k=0}^{\infty} \alpha_k = +\infty$).

   Note that by construction, the family of functions $\{\phi_k\}$ have all the same curvature μ.

We can think of $\{y_k\}$ and $\{\alpha_k\}$ as a set of parameters that we can smartly pick (with tears; see [3], General Scheme 2.2.6) in order to guarantee that

$$\text{for each iteration step } k \in \mathbb{N},\ \exists\, x_k \in \mathbb{R}^n : \quad f(x^*) \leq f(x_k) \leq \min_x \phi_k(x) =: \phi_k^* \tag{9}$$

Note that only the value of the function at the iterate x_k is bounded above by ϕ_k, not the entire function f. Finally, ϕ_k becomes an increasingly good upper bound and although we lost the relaxation property on f (Nesterov's algorithm has momentum), the sequence $\{\phi_k^*\}$ is decreasing and forces the iterates $\{x_k\}$ to converge towards the minimizer x^*.

4 Stochastic Gradient Descent

4.1 Motivation

As we have seen in the previous lecture, Gradient Descent (GD) is not subject to variance since at every step we compute the average gradient using the whole dataset. The downside is that every step is very computationally expensive, $\mathcal{O}(nd)$ per iteration, where n is the number of samples in our dataset and d is the number of dimensions of x. Since we need $\mathcal{O}(d)$ iterations to converge, the total problem cost is about $\mathcal{O}(nd^2)$.

GD becomes impractical when dealing with large datasets. This is where Stochastic Gradient Descent (SGD) comes in. It is a modified version of GD which does not use the whole set of examples to compute the gradient at every step. In SGD we allow the direction towards which want to update the (weight) parameter $\mathbf{w}$ to be a random vector as long as its expected value at each iteration will equal the gradient.
By doing so, we can reduce computation all the way down to $\mathcal{O}(d)$ per iteration, instead of $\mathcal{O}(nd)$.

We consider a large class of functions for which SGD performs well, in particular we will not require the function f that we want to minimize to be differentiable (hence, the need of a "subgradient" ∂f, see Definition 4).
Let $f : \mathbb{R}^n \to \mathbb{R}$, $f = f(\mathbf{w})$:

Algorithm 2 Stochastic Gradient Descent

```
parameters: number of iterations T, step size $\eta$
initialize: $\mathbf{w}^{(1)} \leftarrow \mathbf{0}$
for $t = 1, 2, \ldots T$ do
    choose $\mathbf{v}_t$ at random from a distribution such that $\mathbb{E}[\mathbf{v}_t|\mathbf{w}^{(t)}] \in \partial f(\mathbf{w}^{(t)})$
    update $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta \mathbf{v}_t$
end
output $\bar{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}^{(t)}$
```

Remark 3. *It is important to stress that in general each random vector $\mathbf{v}_t$ is dependent on the previously generated vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_{t-1}\}$.*

The strategy to (randomly) choose the vector $\mathbf{v}_t$ could vary depending on the applications. In the ML world, we want to minimize the empirical risk (ERM method)

$$J(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_{\mathbf{w}}(x_i), y_i),$$

where $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ is the collection of (labelled) samples drawn from an unknown distribution, $f_{\mathbf{w}}$ is the predictor, ℓ is the loss function (we assume them to be differentiable).

In this setting, SGD is set up in the following way: at every iteration t of the optimization method, we randomly draw an index i_t uniformly among the set of indices $\{1, \ldots, n\}$: $i_t \sim U(\{1, \ldots, n\})$. We then perform GD on the function $\ell(f_{\mathbf{w}}(x_{i_t}), y_{i_t})$, i.e. we set $\mathbf{v}_t = \nabla \ell(f_{\mathbf{w}}(x_{i_t}), y_{i_t})$ and we perform the weight update $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \mathbf{v}_t$.

Note that in expectation, the convergence is equivalent to GD, since $\mathbb{E}_{i_t \sim U(\{1,\ldots,n\})}[\mathbf{v}_t] = \nabla J(\mathbf{w})$, therefore, the expected iterate of SGD converges to the optimum. The downside is that stochasticity brings variance. Even if SGD is shown to converge, the variance can seriously handicap the convergence rate as we will see in Subsection 4.2. This is especially true the smaller the batch size is, as variance is inversely proportional to the number of examples used to compute the gradient at every step.

SGD's convergence rate for Lipschitz, convex functions is $\mathcal{O}(T^{-1/2})$, as we will see in Subsection 4.3, and $\mathcal{O}(T^{-1})$ for strongly convex. More iterations are needed to reach the same accuracy as GD, but the iterations are far cheaper.

4.2 Bias-Variance Trade-off

Recall from the last lecture that the iteration step at time t for Gradient Descent is given by $x_{t+1} = x_t - \gamma \nabla f(x_t)$ where $\nabla f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$. The variance from SGD comes from the fact that we do not take the average gradient over the whole dataset.

When training begins, the model weights x are more than likely very far from the optimal value $x* = \min_x f(x)$, therefore, most of our data points x_i, agree on the direction of the gradient. (see the high bias region in Figure 6)

However, when training goes on, most of the model weights are already close to the optimum, some are well tuned and others are not, this is where the variance has a significant impact on our convergence rate because almost all samples in our batch will produce gradients pointing in different directions, therefore, if we keep the same step size we will bounce around the optimum. This is called the noise ball effect.

To avoid bouncing around the minimum, we can use a decaying step size. The blue line in Figure 6 illustrates what happens when we reduce the variance by decaying the step size, another trick to reduce the variance would be to increase the batch size (in the limit we would get the same convergence rate as Gradient Descent).

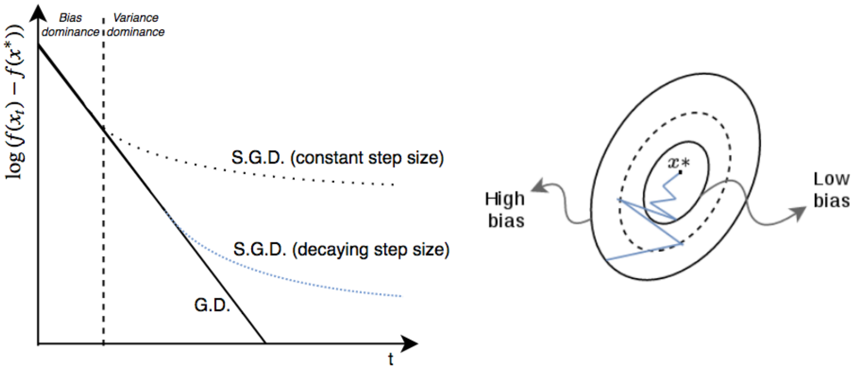

Figure 6: Convergence for Gradient Descent and Stochastic Gradient Descent. The blue line on the left plot illustrates what happens when we decay the step size. The red line shows what happens with constant step size, notice how it flattens out on the left plot.

4.3 Convergence rate for convex-Lipschitz-bounded functions

We provide here an upper bound on the suboptimality of the solution obtained with SGD with respect to the minimizer $\mathbf{w}^*$. Similar to the case of GD (see Theorem 1 from Lecture 2 of the course), we can show that SGD has convergence rate of the order of $\mathcal{O}(T^{-1/2})$ for convex, L-Lipschitz functions, where T is the number of iterates. However, since SGD is a random algorithm, we can only give this result in "expected" form.

Before stating the theorem, which will be valid for general non-differentiable, convex, Lipschitz function, we quickly recall the definition of the subgradient: the subgradient is a generalization of the gradient for non-differentiable functions. Similar to the gradient, a subgradient of a function f at a point $\mathbf{x}$ is the slope of a tangent line that bounds the function f at the point $\mathbf{x}$ from below. See Figure 7.

Definition 4. *Given a convex function $f : S \to \mathbb{R}$, with $S \subseteq \mathbb{R}^n$ a convex set, a vector $\mathbf{v} \in R^n$ is a* **subgradient** *of f at the point $\mathbf{x}$ if*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \mathbf{y} - \mathbf{x}, \mathbf{v} \rangle, \qquad \forall\, \mathbf{y} \in S.$$

The set of all subgradients of f at the point $\mathbf{x}$ is called **differential set** *and it is denoted $\partial f(\mathbf{x})$.*

It is clear that in case of differentiable functions the subgradient consists of only one element, which is indeed the gradient of the function ($\partial f(\mathbf{x}) = \{\nabla f(\mathbf{x})\}$).

The following theorem gives a bound on the expected value of the suboptimality gap:

Theorem 5 ([4], Theorem 14.8)**.** *Fix $B, \rho > 0$. Given $f : \mathbb{R}^n \to \mathbb{R}^n$ a convex function with $\mathbf{w}^* \in \operatorname{argmin}_{\mathbf{w}:\|\mathbf{w}\|<B} f(\mathbf{w})$, consider a SGD algorithm with step size $\eta = \frac{B}{\rho\sqrt{T}}$ and such that $\|\mathbf{v}_t\| \leq \rho$ with probability 1. Then,*

$$\mathbb{E}\left[f(\bar{\mathbf{w}})\right] - f(\mathbf{w}^*) \leq \frac{B\rho}{\sqrt{T}} \tag{10}$$

We highlight the following remark:

1. The constraint that the minimizer needs to belong to a ball of radius B centered at the origin can be relaxed.
2. The bounded-ness condition on the subgradient $\mathbf{v}_t$ of f is equivalent to a Lipschitz-continuity assumption. It is indeed possible to prove that a convex function is ρ-Lipschitz over its domain S if and only if $\forall\, \mathbf{x} \in S$, $\|\mathbf{v}\| \leq \rho$ $\forall\, \mathbf{v} \in \partial f(\mathbf{x})$ (see [4], Lemma 14.7).

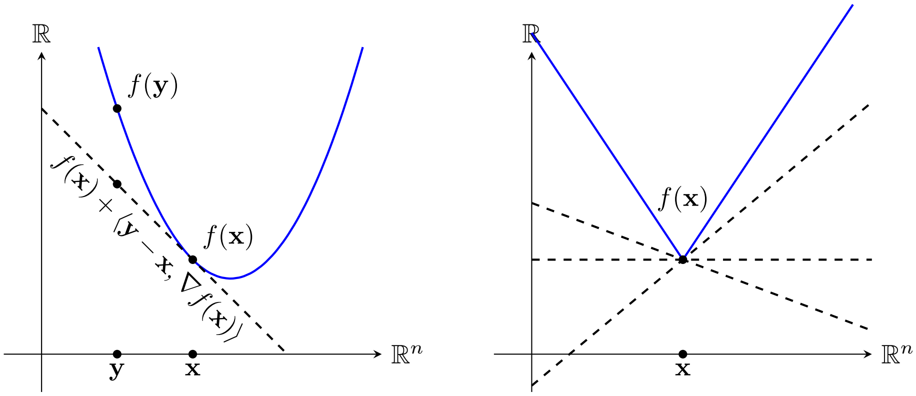

Figure 7: Left: for a convex, differentiable function, the gradient identifies the tangent line that bounds the function f from below. Right: illustration of several subgradients of a nondifferentiable convex function.

3. From the theorem above, it follows that given a fixed $\epsilon > 0$,

$$\text{if } T \geq \frac{B^2\rho^2}{\epsilon^2} \qquad \Rightarrow \qquad \mathbb{E}\left[f(\bar{\mathbf{w}})\right] - f(\mathbf{w}^*) \leq \epsilon,$$

i.e. in order to achieve a good approximation (on average) of our minimum $f(\mathbf{w}^*)$ with tolerance ϵ, one needs to run the SGD algorithm a number T of iteration such that $T \geq \dfrac{B^2\rho^2}{\epsilon^2}$.

Proof. The proof is given in [4]. We rewrite it here entirely, detailing all the passages. We first introduce the notation $\mathbf{v}_{1:t}$ to denote the sequence of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_t$. The proof can be broken down in three steps.

Step 1. Recall that $\bar{\mathbf{w}} := \frac{1}{T}\sum_1^T \mathbf{w}^{(t)}$ (average over the iterates). Therefore, since f is convex, we can apply Jensen inequality to obtain

$$\begin{aligned} f(\bar{\mathbf{w}}) - f(\mathbf{w}^*) &= f\left(\frac{1}{T}\sum_1^T \mathbf{w}^{(t)}\right) - f(\mathbf{w}^*) \\ &\leq \frac{1}{T}\sum_{t=1}^T f(\mathbf{w}^{(t)}) - f(\mathbf{w}^*) = \frac{1}{T}\sum_{t=1}^T \left[f(\mathbf{w}^{(t)}) - f(\mathbf{w}^*)\right] \end{aligned} \tag{11}$$

Applying expectation on both sides, yields

$$\mathbb{E}_{\mathbf{v}_{1:T}}\left[f(\bar{\mathbf{w}}) - f(\mathbf{w}^*)\right] \leq \mathbb{E}_{\mathbf{v}_{1:T}}\left[\frac{1}{T}\sum_{t=1}^T \left[f(\mathbf{w}^{(t)}) - f(\mathbf{w}^*)\right]\right]. \tag{12}$$

Step 2. We want to prove that

$$\mathbb{E}_{\mathbf{v}_{1:T}}\left[\frac{1}{T}\sum_{t=1}^T \left[f(\mathbf{w}^{(t)}) - f(\mathbf{w}^*)\right]\right] \leq \mathbb{E}_{\mathbf{v}_{1:T}}\left[\frac{1}{T}\sum_{i=1}^T \langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle\right] \tag{13}$$

We will start from the right-hand side of the inequality: using linearity, we have

$$\mathbb{E}_{\mathbf{v}_{1:T}}\left[\frac{1}{T}\sum_{i=1}^T \langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle\right] = \frac{1}{T}\sum_{i=1}^T \mathbb{E}_{\mathbf{v}_{1:T}}\left[\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle\right] = \frac{1}{T}\sum_{i=1}^T \mathbb{E}_{\mathbf{v}_{1:t}}\left[\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle\right], \tag{14}$$

where the last equality follows from the fact that each random vector $\mathbf{v}_t$ (and each random iterate $\mathbf{w}^{(t)}$) is independent on the subsequent vectors $\{\mathbf{v}_{t+1}, \ldots, \mathbf{v}_T\}$.

We recall the *law of total expectation*: for any two random variables X, Y, we have $\mathbb{E}_X[f(X)] = \mathbb{E}_Y[\mathbb{E}_X[f(X)|Y]]$ for any function f. In particular, we set $X = \mathbf{v}_{1:t}$ and $Y = \mathbf{v}_{1:t-1}$: $\forall\, t = 1, \ldots, T$

$$\mathbb{E}_{\mathbf{v}_{1:t}}\left[\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle\right] = \mathbb{E}_{\mathbf{v}_{1:t-1}}\left[\mathbb{E}_{\mathbf{v}_{1:t}}\left[\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle\right]|\mathbf{v}_{1:t-1}\right]; \tag{15}$$

Notice that once $\mathbf{v}_{1:t-1}$ is known, $\mathbf{w}^{(t)}$ is not random anymore (it is actually uniquely determined according to the update rule 2). Therefore,

$$\begin{aligned}\mathbb{E}_{\mathbf{v}_{1:t-1}}\left[\mathbb{E}_{\mathbf{v}_{1:t}}\left[\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle\right]|\mathbf{v}_{1:t-1}\right] &= \mathbb{E}_{\mathbf{v}_{1:t-1}}\left[\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbb{E}_{\mathbf{v}_t}\left[\mathbf{v}_t\,|\mathbf{v}_{1:t-1}\right]\rangle\right] \\ &= \mathbb{E}_{\mathbf{v}_{1:t-1}}\left[\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbb{E}_{\mathbf{v}_t}[\mathbf{v}_t|\mathbf{w}^{(t)}]\rangle\right] \end{aligned} \tag{16}$$

where the last equality is using the fact that $\mathbf{v}_{1:t-1} \leftrightarrow \mathbf{w}^{(t)} \leftrightarrow \mathbf{v}_{1:t}$ is a Markov chain and the value of $\mathbf{w}^{(t)}$ is uniquely determined once the value of $\mathbf{v}_{1:t-1}$ is known, therefore $\mathbb{E}_{\mathbf{v}_t}[\mathbf{v}_t|\mathbf{w}^{(t)}] = \mathbb{E}_{\mathbf{v}_t}[\mathbf{v}_t|\mathbf{w}^{(t)}, \mathbf{v}_{1:t-1}] = \mathbb{E}_{\mathbf{v}_t}[\mathbf{v}_t|\mathbf{v}_{1:t-1}]$.
By construction $\mathbb{E}_{\mathbf{v}_t}\left[\mathbf{v}_t\,\middle|\mathbf{w}^{(t)}\right] \in \partial f(\mathbf{w}^{(t)})$ and by definition of the subgradient,

$$\mathbb{E}_{\mathbf{v}_{1:t-1}}\left[\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbb{E}_{\mathbf{v}_t}[\mathbf{v}_t|\mathbf{w}^{(t)}]\rangle\right] \geq \mathbb{E}_{\mathbf{v}_{1:t-1}}\left[f(\mathbf{w}^{(t)}) - f(\mathbf{w}^*)\right] = \mathbb{E}_{\mathbf{v}_{1:T}}\left[f(\mathbf{w}^{(t)}) - f(\mathbf{w}^*)\right]. \tag{17}$$

Collecting all the calculations (15), (16) and (17), we have

$$\mathbb{E}_{\mathbf{v}_{1:t}}\left[\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle\right] \geq \mathbb{E}_{\mathbf{v}_{1:T}}\left[f(\mathbf{w}^{(t)}) - f(\mathbf{w}^*)\right]; \tag{18}$$

using linearity of the expected value, we finally sum over all $t = 1, \ldots, T$ and divide by T to obtain the inequality (13).

Step 3. Wrapping up all the inequalities, we have

$$\mathbb{E}_{\mathbf{v}_{1:T}}\left[f(\bar{\mathbf{w}}) - f(\mathbf{w}^*)\right] \leq \mathbb{E}_{\mathbf{v}_{1:T}}\left[\frac{1}{T}\sum_{t=1}^{T}\left[f(\mathbf{w}^{(t)}) - f(\mathbf{w}^*)\right]\right] \leq \mathbb{E}_{\mathbf{v}_{1:T}}\left[\frac{1}{T}\sum_{i=1}^{T}\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle\right] \overset{\star}{\leq} \frac{B\rho}{\sqrt{T}} \tag{19}$$

We'll now prove the last inequality ($\star$) (see also [4], Lemma 14.1).
$\forall\, t = 1, \ldots, T$, we have

$$\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle = \frac{1}{2\eta}\cdot 2\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \eta\mathbf{v}_t\rangle \tag{20}$$

and we recognize here the double product of a perfect square:

$$\|\mathbf{w}^{(t)} - \mathbf{w}^* - \eta\mathbf{v}_t\|^2 = \|\mathbf{w}^{(t)} - \mathbf{w}^*\|^2 - 2\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \eta\mathbf{v}_t\rangle + \|\eta\mathbf{v}_t\|^2; \tag{21}$$

Therefore,

$$\begin{aligned}\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle &= \frac{1}{2\eta}\left[\|\mathbf{w}^{(t)} - \mathbf{w}^*\|^2 + \|\eta\mathbf{v}_t\|^2 - \|\mathbf{w}^{(t)} - \mathbf{w}^* - \eta\mathbf{v}_t\|^2\right] \\ &= \frac{1}{2\eta}\left[\|\mathbf{w}^{(t)} - \mathbf{w}^*\|^2 + \eta^2\|\mathbf{v}_t\|^2 - \|\mathbf{w}^{(t+1)} - \mathbf{w}^*\|^2\right]\end{aligned} \tag{22}$$

where we used the update rule 2. Summing over $t = 1, \ldots, T$ and averaging, we have

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle &= \frac{1}{2\eta T}\sum_{t=1}^{T}\left[\|\mathbf{w}^{(t)} - \mathbf{w}^*\|^2 + \eta^2\|\mathbf{v}_t\|^2 - \|\mathbf{w}^{(t+1)} - \mathbf{w}^*\|^2\right] \\ &= \frac{1}{2\eta T}\sum_{t=1}^{T}\left[\|\mathbf{w}^{(t)} - \mathbf{w}^*\|^2 - \|\mathbf{w}^{(t+1)} - \mathbf{w}^*\|^2\right] + \frac{1}{2\eta T}\sum_{t=1}^{T}\eta^2\|\mathbf{v}_t\|^2 \\ &= \frac{1}{2\eta T}\left[\|\mathbf{w}^{(1)} - \mathbf{w}^*\|^2 - \|\mathbf{w}^{(T+1)} - \mathbf{w}^*\|^2\right] + \frac{\eta}{2T}\sum_{t=1}^{T}\|\mathbf{v}_t\|^2\end{aligned} \tag{23}$$

(the first sum over the $\mathbf{w}^{(t)}$'s is a telescoping sum).
Finally,

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T}\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle &\leq \frac{1}{2\eta T}\|\mathbf{w}^{(1)} - \mathbf{w}^*\|^2 + \frac{\eta}{2T}\sum_{t=1}^{T}\|\mathbf{v}_t\|^2 && (\|\mathbf{w}^{(T+1)} - \mathbf{w}^*\|^2 \geq 0) \\
&= \frac{1}{2\eta T}\|\mathbf{w}^*\|^2 + \frac{\eta}{2T}\sum_{t=1}^{T}\|\mathbf{v}_t\|^2 && (\text{we initialized } \mathbf{w}^{(1)} = \mathbf{0}) \\
&\leq \frac{1}{2\eta T}\|\mathbf{w}^*\|^2 + \frac{\eta\rho^2}{2} && (\|\mathbf{v}_t\| \leq \rho \text{ with probability one}) \\
&\leq \frac{B^2}{2\eta T} + \frac{\eta\rho^2}{2} && (\mathbf{w}^* \in \{\mathbf{w} : \|\mathbf{w}\| < B\}) \\
&\leq \frac{B\rho}{2\sqrt{T}} + \frac{B\rho}{2\sqrt{T}} = \frac{B\rho}{\sqrt{T}} && (\eta = \frac{B}{\rho\sqrt{T}}) \qquad (24)
\end{aligned}$$

□

References

[1] S. Bubeck. Convex Optimization: Algorithms and Complexity. *ArXiv e-prints*, Nov. 2015.

[2] L. Lessard, B. Recht, and A. Packard. Analysis and Design of Optimization Algorithms via Integral Quadratic Constraints. *ArXiv e-prints*, Aug. 2014.

[3] Y. Nesterov. Introductory lecture on convex programming. *ISBN*, 1998.

[4] B.-D. S. Shalev-Shwartz S. Understanding machine learning: From theory to algorithms. *Cambridge*, 2014.

[5] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. *JMLR*, 2013.