A brief introduction to optimization

Notes for the lab on Mathematical Insights into Deep Learning
ADSI summer school 2018

Zaid Harchaoui, Vincent Roulet

July 2018

How to make the best decision ? This is what optimization is about: you have a set of d parameters $x_1, \dots, x_d$ whose values you want to take to minimize the cost of your decision, modeled by an *objective* function $f : \mathbb{R}^d \mapsto \mathbb{R}$[1]. This is formalized[2] as

$$\min_{x \in \mathbb{R}^d} f(x) \tag{1}$$

While for simple functions the minimizer can be computed analytically, this is often not possible for complex objectives f. We will then perform an iterative search: starting from an initial point, we will move towards a new point at each step to reduce the cost of our decision. This procedure defines an algorithm that a computer can perform.

Optimization problems (1) are then defined by:

- what information an algorithm can have access to at each step,
- the properties of the objective function f.

1 Classical optimization

1.1 Oracles

The information an algorithm can have access to is formalized by the black box model introduced by (Nemirovskii & Yudin 1983): at each point $x \in \mathbb{R}^d$ the algorithm can call an oracle that gives information on the function on that point. Generally this oracle gives an approximation of the function to some degree at that point: *zero order* oracle gives the value of the function $f(x)$, *first order* oracle gives the first derivative (through the gradient $\nabla f(x)$), *second order* oracle gives the second derivative (through the Hessian $\nabla^2 f(x)$) and so on... Deterministic optimization requires this information to be given exactly in contrast with stochastic optimization defined in next section.

Here we are interested in **first order oracles**, zero order being too less informative and second order being too costly for large dimension d.

1.2 Stopping criterion

Now we would like to use the information given by the oracle to assess if we have found an approximate minimizer of the problem and so that we can stop the algorithm. Recall that for a differentiable function f, a point x is a minimizer of f only if $\nabla f(x) = 0$, i.e. if it is a *stationary point*. However this is only a necessary condition, a sufficient condition involves the second order information[3]. In other words, *without further assumptions on the function f, convergence of a first order algorithm can only be stated to stationary points and not minimizers.*

[1] We assume the objective function to have at least one minimizer, formally $\arg\min_{x \in \mathbb{R}^d} f(x) \neq \emptyset$.

[2] More generally one can add constraints on the possible set of parameters for the task, modeled by $x \in \mathcal{C}$ instead of $x \in \mathbb{R}^d$, where $\mathcal{C} \subset \mathbb{R}^d$ is called the set of constraints.

[3] For a twice differentiable function f, if a point x is such that $\nabla^2 f(x) \succ 0$ and $\nabla f(x) = 0$ then it is a local minimizer of f.

Algorithm 1 Gradient descent

Input: Initial point x_0, step-sizes $(\gamma_k)_{k\in\mathbb{N}} \geq 0$, accuracy ϵ

repeat for $k = 0, \dots$

$$x_{k+1} = x_k - \gamma_k \nabla f(x_k)$$

until ϵ-near stationarity, i.e. $\|\nabla f(x_{k+1})\| \leq \epsilon$

Output: $\hat{x} := x_{k+1}$

Note that a natural class of functions to study is then the class of functions where the first order condition $\nabla f(x) = 0$ is also a sufficient condition for global minimization of the function. By requiring such class of functions to be stable by positive combinations and to contain the linear functions, we naturally get the set of convex functions (see (Nesterov 2013, Boyd & Vandenberghe 2004, Bertsekas & Scientific 2015) for more details).

1.3 Smooth functions

We can now precise properties that the objective f may satisfy to make the optimization problem reasonably computable. For the oracle to be relevant, we need the information it gives not too vary drastically around the requested point. For first order oracle this leads to the following definition of smooth functions[4].

Definition 1.1. *A differentiable function $f : \mathbb{R}^d \mapsto \mathbb{R}$ is L-smooth if its gradient is L-Lipschitz continuous, i.e.*

$$\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\|, \qquad \forall x, y \in \mathbb{R}^d.$$

This assumption ensures a quadratic upper bound on the function at each point as stated in the following Lemma (see e.g. (Nesterov 2013, Lemma 1.2.3)). Such simple upper bounds are at the basis of several optimization schemes: for example here minimizing it with respect to y amounts to a gradient step.

Lemma 1.2. *For a L-smooth function f at any point $x \in \mathbb{R}^d$, we have*

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|x - y\|^2, \qquad \text{for any } y \in \mathbb{R}^d$$

1.4 Gradient descent

To illustrate this formalism, consider the simple gradient descent algorithm for smooth functions given in Algorithm 1. At each point it goes along the descent direction given by the negative gradient with a given step-size. Several strategies exist for the step-size: constant step-size $\gamma_k = \gamma$, decreasing step-size, line-search ($\gamma_k = \arg\min_{\gamma \geq 0} f(x_k - \gamma \nabla f(x_k))$, approximate line-search like Armijo or Wolfe's linesearches (see (Nesterov 2013) for more details). We give the proof of convergence for constant step-size in the following proposition, intuitively, one should ensure a step such that the gradient information is still relevant.

Proposition 1.3. *Iterates produced by the gradient descent in Algorithm 1 with step-size $\gamma_k < 2/L$ satisfy*

$$\min_{0 \leq k \leq N} \|\nabla f(x_k)\| \leq \sqrt{\sum_{k=0}^{N} \frac{f(x_k) - f^*}{\omega_k}},$$

where $\omega_k = \gamma_k(1 - L\gamma_k/2)$ and $f^* = \lim_{k\to\infty} f(x_k)$.

[4] Throughout this report $\|\cdot\|$ denotes the Euclidean norm.

Proof. At each iteration $k \geq 0$, using Lemma 1.2,

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) + \langle \nabla f(x_k), x_{k+1} - x_k \rangle + \frac{L}{2} \|x_k - x_{k+1}\|^2 \\ &\leq f(x_k) - \gamma_k \|\nabla f(x_k)\|^2 + \frac{L\gamma_k^2}{2} \|\nabla f(x_k)\|^2 \\ &\leq f(x_k) - \omega_k \left\|\nabla f(x_k)^2\right\| \end{aligned}$$

That reads

$$\left\|\nabla f(x_k)^2\right\| \leq \frac{f(x_k) - f(x_{k+1})}{\omega_k}$$

By assumption, f is continuous and lower bounded ($\arg\min_x f(x) \neq 0$), so the sequence $f(x_k)$ converge to some f^*. Taking the minimum of the left hand side over $k = 1, \ldots, N$ and summing up this inequality gives the first result. The rest of the statements follow by taking special cases of the step-sizes. □

We deduce rates for different step-size strategies. Note that here a constant step-size ensures convergence, while it won't be the case in stochastic optimization below.

Corollary 1.4. *Iterates produced by the gradient descent in Algorithm 1 with step-size $\gamma_k = \gamma < 2/L$ satisfy*

$$\min_{0 \leq k \leq N} \|\nabla f(x_k)\| \leq \sqrt{\frac{f(x_0) - f^*}{\omega(N+1)}},$$

with $\omega = \gamma(1 - L\gamma/2)$. The optimal choice being $\gamma = 1/L$, for which we get

$$\min_{0 \leq k \leq N} \|\nabla f(x_k)\| \leq \sqrt{\frac{2L(f(x_0) - f^*)}{N+1}}.$$

From the above lemma, we understand that the optimization crucially depends on the smoothness of the function to get the optimal theoretical rate. In practice, if one does not know the smoothness parameters a backtracking line-search can be implemented as described in Algorithm 2. Proof of convergence of the gradient scheme with backtracking line-search is left as an exercise. The point is to count, from an initial guess $\hat{L}$ of the smoothness parameter, used for the initial step-size γ_0, how many computations of the backtracking line-search operations are needed and add them to the overall complexity. Notice here that such line-search is ensured as the criterion is known to be satisfied for a small enough step-size.

Algorithm 2 Gradient descent with backtracking line-search

Input: Initial point x_0, initial step-size $\gamma_0 \geq 0$, decreasing factor $\rho < 1$
  repeat for $k = 0, \ldots$
    $\tilde{\gamma} = \gamma_k$
    while $f(x - \tilde{\gamma}\nabla f(x)) > f(x) - \tilde{\gamma}/2 \|\nabla f(x)\|^2$ **do**
      $\tilde{\gamma} := \rho\tilde{\gamma}$
    end while
    $\gamma_k = \tilde{\gamma}$
    $x_{k+1} = x_k - \gamma_k \nabla f(x_k)$
  until ϵ-near stationarity, i.e. $\|\nabla f(x_{k+1})\| \leq \epsilon$
Output: $\hat{x} := x_{k+1}$

2 Stochastic optimization

2.1 Stochastic problems in machine learning

In (first order) stochastic optimization, the oracle outputs only a random estimate $\nabla\tilde{f}(x)$ of the gradient that we assume unbiased, i.e. $\mathbb{E}\nabla\tilde{f}(x) = \nabla f(x)$. If the queried point itself is a random variable (which is naturally the case for an algorithm that uses stochastic gradients), we assume the conditional expectation to be unbiased, i.e. $\mathbb{E}(g(x)|x) = \nabla f(x)$.

Algorithm 3 Stochastic gradient descent

Input: Initial point x_0, step-sizes $(\gamma_k)_{k\in\mathbb{N}} \geq 0$

for $k = 0, \dots, N$ **do**

$$x_{k+1} = x_k - \gamma_k \nabla \tilde{f}(x_k)$$

end for

Output: $\hat{x}$ chosen at random from $x_0, \dots, x_N$ with probability $\Pr(\hat{x} = x_k) = \gamma_k / \sum_{j=1}^N \gamma_j$

Such scenario arises naturally in machine learning problems where the objective is to minimize an expected loss ℓ of a predictor parametrized by x over data composed of inputs-outputs pairs $\xi = (a, b)$, which reads

$$\min_{x\in\mathbb{R}^d} f(x) = \mathbb{E}_\xi \, \ell(x, \xi). \tag{2}$$

When queried at x, the oracle can draw a sample ξ from the unknown distribution and output $\nabla\ell(x, \xi)$ that naturally satisfies the unbiased condition.

A second example is the minimization of the empirical loss, where data is given as a set of n training samples $\xi_1, \dots, \xi_n$. The objective is then a sum of simple functions, that reads

$$\min_{x\in\mathbb{R}^d} f(x) = \frac{1}{n} \sum_{i=1}^n \ell(x, \xi_i). \tag{3}$$

In this case a stochastic gradient can be obtained by selecting uniformly at random $i \in \{1, \dots, n\}$ and reporting $\nabla\ell(x, \xi_i)$.

Observe that the stochastic oracles in the two above cases are quite different. Consider the standard situation where one has access to a data set of i.i.d. samples $\xi_1, \dots, \xi_n$. Thus in the first case, where one wants to minimize the *expected loss*, one is limited to n queries to the oracle, that is to a *single pass* over the data (indeed one cannot ensure that the conditional expectations are correct if one uses twice a data point). On the contrary for the *empirical loss* one can do as many passes as one wishes.

2.2 Stochastic gradient descent

The main algorithm in this setting is the stochastic gradient descent presented in Algorithm 3. As for the deterministic case we need to assume smoothness of the function. We also need an assumption on the variance of the stochastic gradient, this measures once again its relevance and so the step-size that we can make. Namely we assume that for any $x \in \mathbb{R}^d$,

$$\mathbb{E}(\left\|\nabla f(x) - \nabla \tilde{f}(x)\right\|) \leq \sigma^2 \tag{4}$$

If x is itself a random variable we assume $\mathbb{E}(\left\|\nabla f(x) - \nabla \tilde{f}(x)\right\| |x) \leq \sigma^2$.

Proposition 2.1. *Stochastic gradient descent with step-size $\gamma_k \leq 1/L$ outputs after N iterations a point $\hat{x}$ satisfying*

$$\mathbb{E}(\|\nabla f(\hat{x})\|^2) \leq \frac{f(x_0) - \mathbb{E}(f(x_{N+1}))}{\sum_{k=0}^N \gamma_k} + \frac{\sum_{k=0}^N L\sigma\gamma_k^2}{\sum_{k=0}^N \gamma_k}.$$

Proof. As for the gradient descent in the deterministic case, we have at iteration k,

$$f(x_{k+1}) \leq f(x_k) - \gamma_k \langle \nabla \tilde{f}(x_k), \nabla f(x_k) + \frac{L\gamma_k^2}{2} \left\|\nabla \tilde{f}(x_k)\right\|^2$$

Taking expectation on both sides conditioned to x_k, using the unbiased property of the stochastic gradient, we get

$$\mathbb{E}(f(x_{k+1})|x_k) \leq f(x_k) - \gamma_k \|\nabla f(x_k)\|^2 + \frac{L\gamma_k^2}{2} \mathbb{E}(\left\|\nabla \tilde{f}(x_k)\right\|^2 |x_k).$$

Using the bound on the variance of the stochastic gradient (4),

$$\mathbb{E}(f(x_{k+1})|x_k) \leq f(x_k) - \gamma_k(1 - \gamma_k L/2) \|\nabla f(x_k)\|^2 + \frac{L\sigma\gamma_k^2}{2}.$$

For $\gamma_k \leq 1/L$, taking full expectation on x_k, rearranging the terms,

$$\gamma_k \, \mathbb{E}(\|\nabla f(x_k)\|^2) \leq 2(\mathbb{E}(f(x_k)) - \mathbb{E}(f(x_{k+1}))) + L\sigma\gamma_k^2.$$

Summing up this inequality, and dividing by $\sum_{k=0}^N \gamma_k$, we get

$$\sum_{k=0}^N \frac{\gamma_k}{\sum_{j=0}^N \gamma_j} \mathbb{E}(\|\nabla f(x_k)\|^2) \leq \frac{f(x_0) - \mathbb{E}(f(x_{N+1}))}{\sum_{k=0}^N \gamma_k} + \frac{\sum_{k=0}^N L\sigma\gamma_k^2}{\sum_{k=0}^N \gamma_k}.$$

Denote $\hat{x}$ a point selected randomly among $x_1, \ldots, x_N$ with probability $\Pr(\hat{x} = x_k) = \frac{\gamma_k}{\sum_{j=0}^N \gamma_j}$ for $k = 0, \ldots, N$. Then

$$\mathbb{E}(\|\nabla f(\hat{x})\|^2) = \sum_{k=0}^N \frac{\gamma_k}{\sum_{j=0}^N \gamma_j} \mathbb{E}(\|\nabla f(x_k)\|^2).$$

Therefore we get

$$\mathbb{E}(\|\nabla f(\hat{x})\|^2) \leq \frac{f(x_0) - \mathbb{E}(f(x_{N+1}))}{\sum_{k=0}^N \gamma_k} + \frac{\sum_{k=0}^N L\sigma\gamma_k^2}{\sum_{k=0}^N \gamma_k}.$$

□

We deduce rates for different step-sizes strategies, notice that using a constant step-size does not ensure convergence, while a decreasing step-size ensures it.

Corollary 2.2. *Stochastic gradient descend with step-size $\gamma_k = 1/(L\sqrt{k+1})$ outputs after N iterations a point $\hat{x}$ satisfying,*

$$\mathbb{E}(\|\nabla f(\hat{x})\|^2) \leq \frac{L(f(x_0) - \mathbb{E}(f(x_{N+1}))) + \sigma(\log(N+1)+1)}{2(\sqrt{N+2}-1)}.$$

Stochastic gradient descent with constant step-size $\gamma_k = \gamma < 1/L$ outputs after N iterations a point $\hat{x}$ satisfying,

$$\mathbb{E}(\|\nabla f(\hat{x})\|^2) \leq \frac{f(x_0) - \mathbb{E}(f(x_{N+1}))}{\gamma N} + L\sigma\gamma.$$

References

Bertsekas, D. P. & Scientific, A. (2015), *Convex optimization algorithms*, Athena Scientific Belmont.

Boyd, S. & Vandenberghe, L. (2004), *Convex optimization*, Cambridge university press.

Nemirovskii, A. & Yudin, D. (1983), *Problem complexity and method efficiency in optimization*, Wiley.

Nesterov, Y. (2013), *Introductory lectures on convex optimization: A basic course*, Vol. 87, Springer Science & Business Media.