MILA Optimisation Crash Course **Winter 2024**

Lecture 6: March 20th

Lecturer: Damien Scieur *Scribe(s): Behnoush Khavari, Danilo Vucetic*

These are the scribe notes for the sixth lecture of the optimisation crash course at MILA organised by Quentin Bertrand, Damien Scieur, Lucas Maes, and Danilo Vucetic. The purpose of this course is to provide proofs for standard optimisation techniques in order to help practitioners better understand why their algorithms learn. Here is the course web page.

6.1 Recapitulation

Last week we saw stochastic gradient descent (SGD), how it is more efficient than gradient descent (GD), but is not guaranteed to converge. We defined the function to be optimised, $f(\cdot)$, as an empirical average over other functions, $f_i(\cdot)$, where each index i corresponded to a particular data point in a dataset. As such, the derivation corresponded closely to how we may optimise functions in machine learning problems.

6.2 Introduction

In this lecture we aim to answer the question: "how can we set the learning rate (*i.e.*, step size) given that we usually don't know the Lipshitz constant of the gradients, L, or the convexity constant, μ?" Remember, in GD, we set the learning rate to $\frac{1}{L}$, but L is difficult / impossible to estimate *a-priori*. The solutions we explore are called adaptive methods, since they adapt the learning rate over the course of an optimisation algorithm. Given some experience in implementing and training machine learning models on data, one usually encounters the issue of setting appropriate hyperparameters. With adaptive methods, the hope is that we can avoid searching for the best learning rate and learning rate schedule.

In this lecture we depart from our usual proof-based derivations and provide a few practical methods for setting adaptive learning rates. We will examine various line search methods and some others, starting from the most generic, and moving on to the most specialised. We define "generic" as requiring relatively little prior information on the learning problem.

6.3 Simple Line Search Methods

Before starting with the first adaptive method to adjust the learning rate, we warm up the discussion with some simple methods that naturally but probably less effectively do the adaptation.

First, we formulate our problem as this: gradient descent updates are given by this formula

$$x_{k+1} = x_k - \frac{1}{L}\nabla f(x_k) \tag{6.1}$$

and we generalize it to

$$x_{k+1} = x_k - h_k \nabla f(x_k) \tag{6.2}$$

and the question is what should be the value of h_k.

The simplest thing to do is to start with an arbitrary learning rate h_k at each step and verify if, after the gradient step, the value of the function decreases. If so, we continue to the next gradient step with the same learning rate h_k. Otherwise, we reduce the learning rate by some factor, *e.g.,* we divide it by two: $h_k \to \frac{h_k}{2}$ and check whether the reduced learning rate results in a descent of the function after the gradient step. We continue this way until convergence to a local minimum.

A second heuristic which uses some information about the function works as follows. Assuming that the function is smooth, which is equivalent to the function gradient being L-Lipschitz, we estimate the Lipschitz constant L by the second derivative of the function, *i.e.,* $L \approx \left\|\nabla^2 f(x_k)\right\|$.[1] This gives the following learning rate.

$$h_k = (\left\|\nabla^2 f(x_k)\right\| + \epsilon)^{-1}$$

where ϵ is some infinitesimal value added for numerical stability in case $\nabla^2 f(x_k) \to 0$. Also, note that in practice we estimate $\left\|\nabla^2 f(x_k)\right\|$ by $\frac{\|\nabla f(x_k) - \nabla f(x_{k-1})\|}{\|x_k - x_{k-1}\|}$ since Hessians are expensive to compute.

With these warm-up notes let's start with the first line search method which is called exact line search.

6.3.1 Exact Line Search

In this case, we want to optimize the step size such that the function is minimized as much as possible. As such, we solve the following minimization problem

$$h_k = \arg\min_h f(x_k - h \nabla f(x_k)) \tag{6.3}$$

This method, as is seen from its name can only work for function for which there exists an analytical solution for the above minimization problem. To see an example of how line search works for the exact case, consider the quadratic function

$$f(x) = \frac{\|Ax - b\|^2}{2} \tag{6.4}$$

The gradient is given by

$$\nabla f(x) = A^\intercal (Ax - b) \tag{6.5}$$

Then, Equation (6.3) becomes

$$\begin{aligned} h_k &= \arg\min_h f(x_k - h A^\intercal (Ax - b)) \\ &= \arg\min_h \frac{1}{2} \|Ax_k - hAA^\intercal (Ax - b) - b\|^2 \end{aligned} \tag{6.6}$$

To find the minimizer h for the above expression, we put its derivative w.r.t. h equal to zero, which gives some term multiplied by the term in the norm: $Ax_k - hAA^\intercal(Ax - b) - b$. Notice that by setting $h = (AA^\intercal)^{-1}$, the equation goes to zero.

Therefore, in the simple case of a quadratic function in one dimension, the exact line search gives us the minimum in a single gradient step. Another interesting case is the exact line search in two dimensions.

[1] It is quite nice to derive this approximation. It stems from the fact that smooth, twice differentiable, functions have the property $\langle u^\intercal \nabla^2 f(x), v\rangle \leq L \|u\| \|v\|$. With this, and the fact that an induced matrix norm takes the form of $\sup_{\|x\| \neq 0} \frac{\|Ax\|}{\|x\|}$, we can easily show that $\left\|\nabla^2 f(x)\right\| \leq L$! For details, see here, and here.

Consider the objective function as

$$f(x) = x^T Ax + b^T x \tag{6.7}$$

with $x, b \in \mathbb{R}^2$ and $A \in \mathbb{R}^{2\times 2}$. Also we consider A to be symmetric so that all the eigenvalues are real and more than that we require the real eigenvalues be positive, i.e., we consider A to be positive definite. This is in order for the minimization problem to have a unique minimum; in case A has a negative eigenvalue then the minimum of $f(x)$ will be $-\infty$, because we can always diagonalize a real symmetric matrix according to the formula $A = U\Lambda U^T$ with Λ being a diagonal matrix with the eigenvalues of A on its diagonal and U an orthogonal matrix with the corresponding eigenvectors of A as its columns. Then, the first term in (6.7) becomes

$$x^T Ax = x^T U\Lambda U^T x = (U^T x)^T \Lambda U^T x = x_t^T \Lambda x_t \tag{6.8}$$

where x_t stands for transformed x in the rotated coordinates through the orthogonal transformation by matrix U. Expanding the above relation (taking advantage of Λ being diagonal) we get

$$x^T Ax = x_t^T \Lambda x_t = \lambda_1 x_{t1}^2 + \lambda_2 x_{t2}^2 + \cdots + \lambda_n x_{tn}^2 \tag{6.9}$$

Now, since this second-order term in the function $f(x)$ is the dominant term if even only one of λ_i's is negative, the function $f(x)$ can reach the minimum value of $-\infty$ by going towards $x_{ti} \to \infty$. Therefore, for our minimization problem to be well-posed we require a positive definite matrix A.

The gradient is given by

$$\nabla_x f(x) = 2Ax + b \in \mathbb{R}^2 \tag{6.10}$$

Equation (6.3) becomes

$$h_k = \arg\min_h f(x_k - 2hAx_k - bh) \tag{6.11}$$

We have

$$x_{k+1} = x_k - h_k \nabla f(x_k) \tag{6.12}$$

with h_k now given in Equation (6.11). We define a function $g(h)$ as $g(h) := f(x_k - h\nabla f(x_k))$ which for the given function $f(x)$ becomes

$$g(h) = (x_k - h\nabla f(x_k))^T A(x_k - h\nabla f(x_k)) + b^T(x_k - h\nabla f(x_k))$$

which is quadratic and convex (in its variable h) and hence can be seen as

$$g(h) = ah^2 + dh + c \tag{6.13}$$

with $a = \nabla f(x_k)^T A \nabla f(x_k)$ and $d = -(b^T + 2x_k^T A)\nabla f(x_k) = -\nabla f(x_k)^T \nabla f(x_k)$. The minimum of $g(h)$ occurs at $h = -\frac{d}{2a}$. Therefore, the value of h at which $g(h)$ reaches its minimum, i.e., h_k is

$$h_k = \frac{\nabla f(x_k)^T \nabla f(x_k)}{2\nabla f(x_k)^T A \nabla f(x_k)} \tag{6.14}$$

with $\nabla f(x_k)$ given as in Equation (6.10).

It is important to note that h_k is only a scalar value which tells us the length that we progress along the gradient direction $\nabla f(x_k)$ at each point x_k, but interestingly we can also see that in this 2-dimensional case, each gradient step is perpendicular to the step before it. To show this, we find the inner product between two consecutive step vectors

$$\begin{aligned} x_{k+1} &= x_k - h_k \nabla f(x_k) \\ x_{k+2} &= x_{k+1} - h_{k+1} \nabla f(x_{k+1}) \end{aligned} \tag{6.15}$$

$$\langle x_{k+1} - x_k, x_{k+2} - x_{k+1} \rangle = h_k h_{k+1} \langle \nabla f(x_k), \nabla f(x_{k+1}) \rangle \tag{6.16}$$

Now, since h_k minimizes $g_k(h) := f(x_k - h\nabla f(x_k))$, we have

$$\frac{dg_k(h)}{dh}|_{h_k} = 0 \tag{6.17}$$

. Also, from the chain rule and the definition of $g_k(h)$ we have

$$\begin{aligned} g_k(h) &= f(x_k - h\nabla f(x_k)) \\ \rightarrow \frac{dg_k(h)}{dh}|_{h_k} &= \frac{df(x_k - h\nabla f(x_k))}{dh} \\ &= \frac{df(z)}{dz}\frac{dz}{dh} \\ &= \langle -\nabla f(x_k), \nabla f(x_k - \alpha_k \nabla f(x_k)) \rangle \\ &= \langle -\nabla f(x_k), \nabla f(x_{k+1}) \rangle \end{aligned} \tag{6.18}$$

From the above equations (6.17) and (6.18) we conclude

$$\langle x_{k+1} - x_k, x_{k+2} - x_{k+1} \rangle = 0 \tag{6.19}$$

The upshot is that in the two-dimensional case, we start by finding the direction of the gradient descent and continue until the function value decreases. The next step will be perpendicular to the earlier step and again we continue as far as the function value is decreasing and so on.

With this introduction to the exact line search method, we notice that the above two examples are very specific cases of simple functions in low dimensions, while most of the times we cannot solve the minimum step size equation analytically; hence in practice, we often use the inexact line search method in numerics.

6.3.2 Inexact Line Search

In this section, we present a few very commonly used line search algorithms on one-dimensional function $f(x)$. The following is a list of the algorithms that were quickly covered.

- Golden section: does not use gradients
- Bisection search: requires gradients
- Secant method: like bisection but uses secant lines, does not require gradients
- Newton's method: use gradient (or Hessian) to find descent direction.

We will only highlight bisection search in this section.

6.3.2.1 Bisection

For the one-dimensional case, this method works based on the following observation. If $f(x)$ has a unique minimum in the interval $x \in [a, b]$ and the minimum happens at some $x^* \in [a, b]$, then the slope of the function is negative on $[a, x^*)$, zero at x^* and positive on $(x^*, b]$. Hence, we can find the root of $\nabla f(x)$ to find the minimum x^*. For this, we iteratively half the given interval $[a, b]$, find the gradient value at the middle point $\frac{a+b}{2}$ and depending on this value being positive (negative) we update the value of b (a) by the value of the x at the middle. The stopping point of the algorithm is either as we reach some maximum number of iterations or when the difference between $\nabla f(x)$ and zero is smaller than some given ϵ.

6.4 Advanced Line Search Methods

This section covers more advanced line search procedures. We start with generic procedures which require less prior knowledge of your problem, but might take longer to converge, and continue to increasingly specialised methods. Each method follows a similar structure whereby a step size is computed, then a condition is checked to see whether the function value has decreased sufficiently (unless the step size is known to be optimal).

6.4.1 Wolfe Line Search

In Wolfe line search we are interested in finding a descent direction, d_k,[2] and a step size, h_k. Our problem may be stated as follows.

$$h_k = \arg\min_h f(x_k - hd_k) \tag{6.20}$$

In order to accept a step size, it must satisfy the following conditions. Note that we introduce constants c_1, c_2 such that $0 < c_1 < c_2 < 1$.

- The sufficient descent condition: $f(x_k - hd_k) \leq f(x_k) - c_1 h d_k^\intercal \nabla f(x_k)$ (easy to satisfy with small h_k)
- The weak (or strong) Wolfe condition: $d_k^\intercal \nabla f(x_k + hd_k) \leq c_2 d_k^\intercal \nabla f(x_k)$ (ensures that h_k is not too small)

The smaller c_1, the easier it is for the first condition to be satisfied. Usually set c_2 to be larger so that the function decreases by non-trivial amounts. Note that Wolfe line search is hard to use with stochastic gradients, since the conditions are hard to check.

6.4.2 Backtracking Line Search

According to Damien backtracking line search is a "lazy way to do GD." Here, we assume that our function is L-smooth, and that we are using gradient descent as the update rule. The basic idea of the method is to successively update an approximation of L, the smoothness constant. As such, we have a sort of "local" approximation of L. The descent condition is $f(x_{k+1}) \leq f(x_k) - \frac{1}{2\mu_k} \|\nabla f(x_k)\|^2$, where μ_k is our local approximation of L.

The search proceeds as follows, with an initial $\mu_0 = 1$.

- Update the parameters: $x_k - \frac{1}{2\mu_k} \nabla f(x_k)$
- If the descent condition is met, set $\mu_{k+1} = \mu_k$, and go back to the first step.
- If the descent condition is not met, then the gradient is too big, and we need to decrease the step size, so set $\mu_k \leftarrow 2\mu_k$ and try again until the condition is met.

We can also be optimistic in setting μ_k if we instead set $\mu_{k+1} = \mu_k/2$, in the hopes that we can keep the step size larger for longer. With this approach, the total number of function evaluations is in $O(T + log(2L/\mu_0))$.

[2] we may be accustomed to gradients as our descent direction thanks to GD, SGD, etc., but we can choose something else.

6.4.3 Polyak Step Size

Must know $f^\star$ ahead of time (*e.g.,* a loss of zero), and works primarily for subgradient methods. Assume that $f(x)$ is convex but non-smooth. As such, f obeys the property that $f(y) - f(x) \geq \nabla f(x)^\intercal (y - x)$. Then, by solving the following for h, we find the optimal step size.

$$\begin{aligned}
\|x_{k+1} - x^\star\| &= \|x_k - x^\star - h\nabla f(x_k)\|^2 \\
&= \|x_k - x^\star\|^2 + h^2 \|\nabla f(x_k)\|^2 - 2h\langle \nabla f(x_k), x_k - x^\star \rangle \\
&\leq \|x_k - x^\star\|^2 + h^2 \|\nabla f(x_k)\|^2 - 2h\left(f(x_k) - f(x^\star)\right)
\end{aligned}$$

We want to minimise the right hand side with respect to h, so by taking the gradient, we can trivially find

$$h_k = \frac{f(x_k) - f(x^\star)}{\|\nabla f(x_k)\|^2} \tag{6.21}$$

What's nice about this method is that it minimises the hidden constant in subgradient descent, giving the best possible convergence rate for that method.