Lecture Notes on Stochastic Optimization

Tim van Erven

October 24, 2017

1 Optimization with Gradient Descent

In the course we have seen many cases where we want to find parameters $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$ that minimize some function $F(\boldsymbol{\beta})$:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} F(\boldsymbol{\beta}).$$

For instance,

$$F(\boldsymbol{\beta}) = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \boldsymbol{X}_i^\top \boldsymbol{\beta})^2 \qquad \text{(least squares)}$$

$$F(\boldsymbol{\beta}) = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \boldsymbol{X}_i^\top \boldsymbol{\beta})^2 + \lambda \sum_{j=2}^{p} \beta_j^2 \qquad \text{(ridge regression)}$$

$$F(\boldsymbol{\beta}) = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \boldsymbol{X}_i^\top \boldsymbol{\beta})^2 + \lambda \sum_{j=2}^{p} |\beta_j| \qquad \text{(lasso)}$$

$$F(\boldsymbol{\beta}) = \frac{1}{N}\sum_{i=1}^{N}\log(1 + e^{-Y_i \boldsymbol{X}_i^\top \boldsymbol{\beta}}) \qquad \text{(logistic regression)}$$

N.B. I am dividing here by N, which I did not do in the lectures, but if we adjust λ appropriately then this makes no difference to the optimal parameters $\boldsymbol{\beta}^*$. These functions all have in common that they are *convex*. See Figure 1. Mathematically, this means that, for any two parameter vectors $\boldsymbol{\beta}_0$ and $\boldsymbol{\beta}_1$, the line between $F(\boldsymbol{\beta}_0)$ and $F(\boldsymbol{\beta}_1)$ lies above the function F. That is,

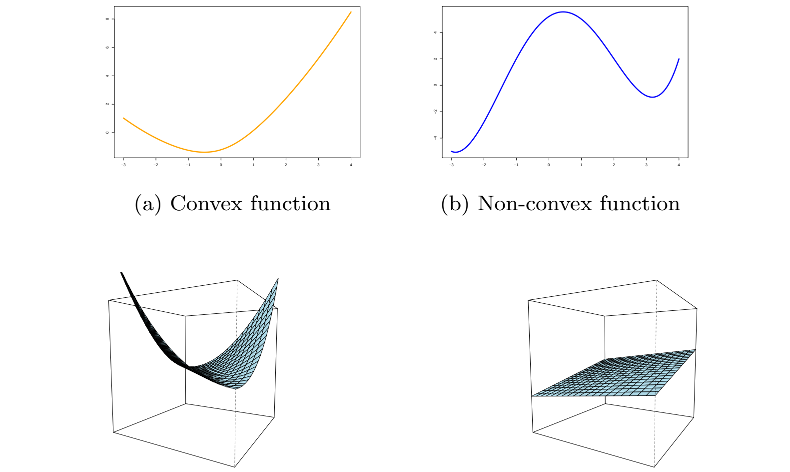
$$F\big((1-\lambda)\boldsymbol{\beta}_0 + \lambda\boldsymbol{\beta}_1\big) \leq (1-\lambda)F(\boldsymbol{\beta}_0) + \lambda F(\boldsymbol{\beta}_1) \qquad \text{for all } \lambda \in [0,1].$$

Practically, convexity means that we can find the minimum of F by starting at some parameters $\boldsymbol{\beta}_1$ and then making small improvements to the parameters that decrease the value of $F(\boldsymbol{\beta})$ until we reach the minimum. The most well-known method that does this is *gradient descent*, which makes improvements of the form

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta_t \nabla F(\boldsymbol{\beta}_t) \qquad \text{(gradient descent)}$$

where $\boldsymbol{\beta}_t$ are the current parameters after t steps of the algorithm, $\boldsymbol{\beta}_{t+1}$ are the new parameters, and $\nabla F(\boldsymbol{\beta}_t)$ is the *gradient* at $\boldsymbol{\beta}_t$. The gradient is the vector of partial derivatives

$$\nabla F(\boldsymbol{\beta}) = \begin{pmatrix} \frac{\partial}{\partial \beta_1} F(\boldsymbol{\beta}) \\ \vdots \\ \frac{\partial}{\partial \beta_p} F(\boldsymbol{\beta}) \end{pmatrix},$$



(a) Convex function

(b) Non-convex function

(c) Convex function of two parameters

(d) Another convex function of two parameters (this function is actually linear)

Figure 1: Examples of convex and non-convex functions

and $\nabla F(\boldsymbol{\beta}_t)$ is always pointing in the direction in which F would increase the most if we moved away from $\boldsymbol{\beta}_t$. Because we want to *decrease* F instead of increasing it, we take steps in the direction $-\nabla F(\boldsymbol{\beta}_t)$, which is the direction in which F decreases the most if we move away from $\boldsymbol{\beta}_t$. Finally, the number $\eta_t > 0$ in the definition of gradient descent is called the *step size*, because it controls the size of our steps. The best way to choose the step sizes η_t depends on properties of the function F, but common choices are $\eta_t = C$, $\eta_t = C/\sqrt{t}$ and $\eta_t = C/t$, where C is some constant. For example, if the function F is not only convex, but also γ-*smooth*, then choosing $\eta_t = \frac{1}{\gamma}$ guarantees that the difference between $F(\boldsymbol{\beta}_t)$ and $F(\boldsymbol{\beta}^*)$ (where $\boldsymbol{\beta}^*$ are the optimal parameters) decreases at a rate that is in the order of $O(1/t)$:

$$F(\boldsymbol{\beta}_t) - F(\boldsymbol{\beta}^*) \leq \frac{2\gamma \|\boldsymbol{\beta}_1 - \boldsymbol{\beta}^*\|^2}{t-1},$$

so the more steps we take (that is, the larger t), the closer we get to the optimum parameters [1]. (In case you are wondering, γ-smoothness means that the second derivative of F in any direction is at most γ.)

2 Stochastic Optimization

Every step of gradient descent requires computing the gradient of F. If F consists of a sum of many functions:

$$F(\boldsymbol{\beta}) = \frac{1}{N} \sum_{i=1}^{N} f_i(\boldsymbol{\beta}), \tag{1}$$

then this means we need to compute the gradient of each of these functions in every step of gradient descent, because

$$\nabla F(\boldsymbol{\beta}_t) = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\boldsymbol{\beta}_t),$$

which takes a lot of computation time when N is large.

Looking at the examples at the start of the previous section, we see that least squares and logistic regression are of the form (1), with $f_i(\boldsymbol{\beta}) = (Y_i - \boldsymbol{X}_i^\top \boldsymbol{\beta})^2$ and $f_i(\boldsymbol{\beta}) = \log(1 + e^{-Y_i \boldsymbol{X}_i^\top \boldsymbol{\beta}})$, respectively. But it turns out that ridge regression and the lasso also fall into this category, if we take $f_i(\boldsymbol{\beta}) = (Y_i - \boldsymbol{X}_i^\top \boldsymbol{\beta})^2 + \lambda \operatorname{pen}(\boldsymbol{\beta})$, where $\operatorname{pen}(\boldsymbol{\beta})$ is the ridge or lasso penalty. To see this, note that

$$\frac{1}{N}\sum_{i=1}^{N}\Big((Y_i - \boldsymbol{X}_i^\top \boldsymbol{\beta})^2 + \lambda \operatorname{pen}(\boldsymbol{\beta})\Big) = \frac{1}{N}\sum_{i=1}^{N}\Big((Y_i - \boldsymbol{X}_i^\top \boldsymbol{\beta})^2\Big) + \lambda \operatorname{pen}(\boldsymbol{\beta}).$$

When we need to minimize functions F of the form (1), *stochastic optimization* often provides savings in computation time. When combined with gradient descent, it works as follows:

Stochastic Gradient Descent The idea is that whenever we need to compute the gradient $\nabla F(\boldsymbol{\beta}_t)$ in gradient descent, we cheat a little bit and only compute an approximation. In every step t of the algorithm, we do this by randomly choosing one of the functions f_i and then using only the gradient $\nabla f_i(\boldsymbol{\beta}_t)$ instead of $\nabla F(\boldsymbol{\beta}_t)$. The resulting algorithm is called *stochastic gradient descent*, and each update step of the algorithm works as follows:

$$\text{Choose } i_t \text{ randomly from } \{1, 2, \ldots, N\}$$
$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta_t \nabla f_{i_t}(\boldsymbol{\beta}_t)$$

We see that each update step now only requires computing the gradient for a single function f_i instead of computing N gradients for each of the functions $f_1, \ldots, f_N$. We can therefore take N steps of stochastic gradient descent in the same computation time that we would need to take 1 step of ordinary gradient descent. Of course, we may now be concerned that $\nabla f_{i_t}(\boldsymbol{\beta}_t)$ may be a poor substitute for $\nabla F(\boldsymbol{\beta}_t)$, but it turns out that it is a reasonable estimate, because on average it has the right value:

$$\mathbb{E}_{i_t}[\nabla f_{i_t}(\boldsymbol{\beta}_t)] = \sum_{i=1}^{N} \frac{1}{N} \nabla f_{i_t}(\boldsymbol{\beta}_t) = \nabla F(\boldsymbol{\beta}_t).$$

In statistical terms, $\nabla f_{i_t}(\boldsymbol{\beta}_t)$ is an *unbiased estimator* for $\nabla F(\boldsymbol{\beta}_t)$.

References

[1] S. Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3–4):231–358, 2015.