Lecture #7: Optimization in Learning

Instructor: Aditya Bhaskara *Scribe: Tharindu Rusira*

CS 5966/6966: Theory of Machine Learning

February 1st, 2017

Abstract

We saw the importance of empirical risk minimization in learning. In many settings, the ERM problem is phrased as convex optimization, and solved using the vast literature on the topic. In this and the next few lectures, we will cover some of the basics. Today, we discuss the basics of convexity, approximating loss functions by convex functions, and introduce the gradient descent algorithm.

1 Introduction

So far in the course, we have seen ways to prove that the empirical risk minimizer (ERM) also generalizes well. Now we focus on the problem of computing the ERM, given a hypothesis class $\mathcal{H}$. I.e., given a sample S of points from $\mathcal{D}$, and their labels f, we wish to find a hypothesis $h \in \mathcal{H}$ that minimizes the empirical loss (risk):

$$\text{argmin}_{h\in\mathcal{H}} L_S(h), \text{ where } L_S(h) := \sum_{x\in S} \mathbb{1}[h(x) \neq f(x)].$$

Consider the simple example of linear classifiers. Here $\mathcal{H}$ consists of all hypotheses of the form $h(x) = \text{sign}(w^T x)$. One of the earliest and most elegant algorithms for finding a linear classifier is the so-called *perceptron algorithm*. It works as follows.

A linear classifier for n dimensional vectors x is simply a function defined by weights $w_1, \ldots, w_n$, where $h(x) = sign(w_1x_1 + w_2x_2 + ... + w_nx_n)$.

Input: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ..., (x^{(m)}, y^{(m)})$
Goal: find a weight vector w s.t. $\text{sign}(\langle w, x^{(i)}\rangle) = y^{(i)}$ for all i
$w \leftarrow \mathbf{0} \in \mathbb{R}^d$
while a misclassified point $x^{(i)}$ is found,
  $w^{(t+1)} = w^{(t)} + y_i x^i$

In the 'well-classifiable' case, i.e., when there exists a w with $\|w\|_2 = 1$ s.t. $y^{(i)} \cdot \langle w, x^{(i)}\rangle \geq \eta \|x^{(i)}\|_2$ for some $\eta > 0$, it can be shown that the algorithm converges in $O\left(\frac{1}{\eta^2}\right)$ iterations.

What if the data is not perfectly classifiable? In this case, the algorithm may end up in cycles, and never converge. Furthermore, when there is no perfect classifier (i.e., in an agnostic setup), finding the hypothesis with the least number of mistakes (i.e., that minimizes the loss function) is NP hard. The problem is that the loss function $\ell(h, x) = \mathbb{1}[h(x) \neq f(x)]$ is too *discrete*. This motivates us to consider different loss functions that can be minimized easily, and can act as a proxy for the number of mistakes.

Three natural candidates are the following:

1. squared loss, where the loss of the ith point is defined as $(y^{(i)} - \langle w, x^{(i)} \rangle)^2$.
2. ℓ_1 loss, where the loss is $|y^{(i)} - \langle w, x^{(i)} \rangle|$.
3. logistic regression, where the loss is defined using the sigmoid. Formally, the loss of the ith point is $\log\left(\frac{1}{1+e^{-y_i \langle w, x^{(i)} \rangle}}\right)$.

Using a *convex* loss function is often preferred, as minimizing convex functions over convex domains is an efficiently solvable problem. This will be the subject of the next few lectures of the course.

2 Convex sets and functions

2.1 definition (Convex set). A set $K \subseteq \mathbb{R}^d$ is said to be a convex set if $\forall x, y \in K$, the segment joining x, y is fully contained in K.

2.2 definition (Convex function). A function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be a convex function over a convex set D, if for all $t \in (0, 1)$, we have

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y), \ \forall x, y \in D. \tag{1}$$

In real analysis, you may have seen convexity defined in terms of the second derivative – $f : \mathbb{R} \to \mathbb{R}$ iff $f''(x) \geq 0 \ \forall x$. The two definitions are equivalent when f is twice differentiable. The definition (1) makes sense for all continuous f.

We now quickly recall basic notions from multivariate calculus.

2.3 definition. *Partial derivative* of $f : \mathbb{R}^d \to \mathbb{R}$ w.r.t. x_i is defined as,

$$\frac{\partial f(x)}{\partial x_i} = \lim_{\delta \to 0} \frac{f(x + \delta e_i) - f(x)}{\delta}$$

where e_i is the i-th basis in $\mathbb{R}^d$.

2.4 definition. *Gradient* of function f is defined as

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_d} \end{pmatrix}$$

The first order "Taylor approximation" for f can be directly obtained from the above definitions.

$$f(x + h) \approx f(x) + \langle h, \nabla f(x) \rangle, \text{ for all } h, x \in \mathbb{R}^d. \tag{2}$$

Another important property of a convex function (which is indeed crucial to the analysis of gradient descent that we will see later) is the following:

$$f(y) - f(x) \geq \langle y - x, \nabla f(x) \rangle \qquad \text{for all } x, y \tag{3}$$

Geometrically, this inequality says that a convex function always "lies above" the tangent to the function at any point. Indeed, convex functions are sometimes defined using the *envelope* of such tangent planes.

Next, we define some other useful notions about multivariate functions.

2.5 definition (Lipschitz property). A function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be *ρ-Lipschitz* if $|f(x) - f(y)| \leq \rho \|x - y\| \ \forall x, y$.

2.6 DEFINITION. A function f is said to be $\beta-$smooth if ∇f is $\beta-$Lipschitz, i.e., $\|\nabla f(x) - \nabla f(y)\| \leq \beta\|x - y\|$.

3 Minimizing convex functions

Let f be a convex function defined over a convex set $D \subseteq \mathbb{R}^d$. We will be interested in the minimization problem: $\min_{x \in D} f(x)$.

Most of the known algorithms for *general* convex optimization work in an iterative way – they start with any point in D, and try to move to a *nearby* point with a smaller value of f. They repeat this process until no "local improvement" is possible (this is essentially *local search*). The nice thing about convex functions is that such a procedure is guaranteed to converge to the (global) minimum! (I.e., any local minimum is a global minimum – this is a nice exercise for those who have not seen it earlier.)

Now, which direction should we move in order to obtain a local improvement? From the first order Taylor approximation of f, we know,

$$f(x + h) \approx f(x) + \langle h, \nabla f(x) \rangle \text{ for "smal" } h.$$

For small h, if we need to have $f(x + h) < f(x)$, we must make sure that $\langle h, \nabla f(x) \rangle < 0$, i.e., we should "move" along a direction whose inner product with the gradient is negative. The easiest choice is the negative of the gradient! I.e., we can set $h = -\eta \nabla f(x)$ with $\nabla > 0$ which implies $f(x + h) - f(x) \approx -\eta \|\nabla f(x)\|^2 < 0$.

*η is called **step size** in this context. It is a parameter that should be chosen apropriately, depending on f.*

This is the idea behind the popular algorithm – *gradient descent*. We will describe and analyze gradient descent over the next few lectures.

4 Gradient descent

Formalizing the discussion above, the gradient descent procedure is the following:

set $x^{(0)} \leftarrow$ any point in D
for $t = 0 : T - 1$
  $x^{(t+1)} = x^{(t)} - \eta \nabla f(x^{(t)})$
return $x^{(T)}$

Does this procedure always converge to the optimum? How do we choose η and T? What if $x^{(t+1)}$ defined as above is not in D (i.e., we lose feasibility)?

These are all questions that we will study. The choice of η is an important concern during implementation – if it's chosen to be too small, then the algorithm typically converges to the minimum, but it could be too slow. If η is large, then the algorithm may oscillate, and even diverge. In general, choosing a varying η is also possible, and sometimes works well.

The other issue is the loss of feasibility, which occurs when $x^{(t+1)}$ as defined above ceases to be feasible (in D). In this case, a general solution is to set $x^{(t+1)}$ to be the projection of $x^{(t)} - \eta \nabla f(x^{(t)})$ onto D. All the analyses we will see in the coming lectures continue to hold under such a projection.

*A **projection** of a point x onto a convex set D is defined to be the closest point to x in D. For* simple *sets D, such a projection is often easy to compute (e.g., by normalizing, etc.).*

5 Analysis of Gradient Descent

We now study the gradient descent algorithm above, under the two assumptions:

1. f is ρ-Lipschitz, and
2. The starting point $x^{(0)}$ is at a distance $\leq B$ from the optimum x^*.

In the next lecture, we will show the following bound:

$$\frac{1}{T}\sum_{t=1}^{T}(f(x^{(t)}) - f(x^*)) \leq \frac{B^2}{2\eta T} + \frac{\rho^2\eta}{2}. \tag{4}$$

This bound can be used to determine the setting of η and T. The LHS is the *average difference* between the function values of the current iterate and the optimum. We note that this analysis does not really imply that $x^{(t)} \to x^*$. Indeed, there could be a "region" around the optimum in which the function could be relatively flat, and it would take many iterations for $x^{(t)}$ to really get close to x^*. However, if all we care about is the function value (as is true in nearly all applications), the bound above is good enough.

In order to have the RHS being ϵ, an easy way to set the parameters is to choose $\eta = \epsilon/\rho^2$, and $T = B^2/\epsilon\eta$. If we think of B, ρ as constants, this means that the number of iterations should be $O(1/\epsilon^2)$.