Course Notes for EE227C (Spring 2018): Convex Optimization and Approximation

Instructor: Moritz Hardt

Email: hardt+ee227c@berkeley.edu

Graduate Instructor: Max Simchowitz

Email: msimchow+ee227c@berkeley.edu

March 5, 2018

10 Lecture 10: Stochastic optimization

The goal in stochastic optimization is to minimize functions of the form

$$f(x) = \mathop{\mathbb{E}}_{z\sim\mathcal{D}} g(x,z)$$

which have stochastic component given by a distribution $\mathcal{D}$. In the case where the distribution has finite support, the function can be written as

$$f(x) = \frac{1}{m}\sum_{i=1}^{m} f_i(x)\,.$$

To solve these kinds of problems, we examine the stochastic gradient descent method and some if its many applications.

10.1 The stochastic gradient method

Following Robbins-Monro [RM51], we define the stochastic gradient method as follows.

Definition 10.1 (Stochastic gradient method)**.** The stochastic gradient method starts from a point $x_0 \in \Omega$ and proceeds according to the update rule

$$x_{t+1} = x_t - \eta_t \nabla f_{i_t}(x_t)$$

where $i_t \in \{1,\dots,m\}$ is either selected at random at each step, or cycled through a random permutation of $\{1,\dots,m\}$.

Either of the two methods for selecting i_t above, lead to the fact that

$$\mathbb{E}\,\nabla f_{i_t}(x) = \nabla f(x)$$

This is also true when $f(x) = \mathbb{E}\,g(x,z)$ and at each step we update according to $\nabla g(x,z)$ for randomly drawn $z \sim \mathcal{D}$.

10.1.1 Sanity check

Let us check that on a simple problem that the stochastic gradient descent yields the optimum. Let $p_1, \ldots, p_m \in \mathbb{R}^n$, and define $f\colon \mathbb{R}^n \to \mathbb{R}_+$:

$$\forall x \in \mathbb{R}^n,\ f(x) = \frac{1}{2m}\sum_{i=1}^{m} \|x - p_i\|_2^2$$

Note that here $f_i(x) = \frac{1}{2}\|x - p_i\|_2^2$ and $\nabla f_i(x) = x - p_i$. Moreover,

$$x^* = \arg\min_{x\in\mathbb{R}^d} f(x) = \frac{1}{m}\sum_{i=1}^{m} p_i$$

Now, run SGM with $\eta_t = \frac{1}{t}$ in cyclic order i.e. $i_t = t$ and $x_0 = 0$:

$$\begin{aligned}
x_0 &= 0\\
x_1 &= 0 - \frac{1}{1}(0 - p_1) = p_1\\
x_2 &= p_1 - \frac{1}{2}(p_1 - p_2) = \frac{p_1 + p_2}{2}\\
&\vdots\\
x_n &= \frac{1}{m}\sum_{i=1}^{m} p_i = x^*
\end{aligned}$$

10.2 The Perceptron

The New York Times wrote in 1958 that the Perceptron [Ros58] was:

> *the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.*

So, let's see.

Definition 10.2 (Perceptron)**.** Given labeled points $((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathbb{R}^n \times \{-1, 1\})^m$, and and initial point $w_0 \in \mathbb{R}^n$, the Perceptron is the following algorithm. For $i_t \in \{1, \ldots, m\}$ selected uniformly at random,

$$w_{t+1} = w_t(1-\gamma) + \eta \begin{cases} y_{i_t} x_{i_t} & \text{if } y_{i_t}\langle w_t, x_{i_t}\rangle < 1\\ 0 & \text{otherwise} \end{cases}$$

Reverse-engineering the algorithm, the Perceptron is equivalent to running the SGM on the Support Vector Machine (SVM) objective function.

Definition 10.3 (SVM)**.** Given labeled points $((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathbb{R}^n \times \{-1, 1\})^m$, the SVM objective function is:

$$f(w) = \frac{1}{n} \sum_{i=1}^{m} \max(1 - y_i \langle w, x_i \rangle,\ 0) + \lambda \|w\|_2^2$$

The loss function $\max(1 - z,\ 0)$ is known as the Hinge Loss. The extra term $\lambda \|w\|_2^2$ is known as the regularization term.

10.3 Empirical risk minimization

We have two spaces of objects $\mathcal{X}$ and $\mathcal{Y}$, where we think of $\mathcal{X}$ as the space of *instances* or *examples*, and $\mathcal{Y}$ is a the set of *labels* or *classes*.

Our goal is to *learn* a function $h \colon \mathcal{X} \to \mathcal{Y}$ which outputs an object $y \in \mathcal{Y}$, given $x \in \mathcal{X}$. Assume there is a joint distribution $\mathcal{D}$ over the space $\mathcal{X} \times \mathcal{Y}$ and the training set consists of m instances $S = ((x_1, y_1), \ldots, (x_m, y_m))$ drawn i.i.d. from $\mathcal{D}$.

We also define a non-negative real-valued loss function $\ell(y', y)$ to measure the difference between the prediction y' and the true outcome y.

Definition 10.4. The *risk* of a function $h \colon \mathcal{X} \to \mathcal{Y}$ is defined as

$$R[h] = \mathbb{E}_{(x,y)\sim\mathcal{D}} \ell(h(x), y)$$

The ultimate goal of a learning algorithm is to find h^* among a class of functions $\mathcal{H}$ that minimizes $R[h]$:

$$h^* \in \arg\min_{h \in \mathcal{H}} R[h]$$

In general, the risk $R[h]$ cannot be computed because the joint distribution is unknown.

Therefore, we instead minimize a proxy objective known as *empirical risk* and defined by averaging the loss function over the training set:

$$R_S[h] = \frac{1}{m} \sum_{i=1}^{m} \ell(h(x_i), y_i)$$

An empirical risk minimizer is any point $h^* \in \arg\min_{h \in \mathcal{H}} R_S[h]$.

The stochastic gradient method can be thought of as minimizing the risk directly, if each example is only used once. In cases where we make multiple passes over the training set, it is better to think of it as minimizing the empirical risk, which can give different solutions than minimizing the risk. We'll develop tools to relate risk and empirical risk in the next lecture.

10.4 Online Learning

An interesting variant of this learning setup is called *online learning*. It arises when we do not have a set of training data, but rather must make decisions one-by-one.

Taking advice from experts. Imagine we have access to the predictions of n experts. We start from an initial distribution over experts, given by weights $w_1 \in \Delta_n = \{w \in \mathbb{R}^n : \sum_i w_i = 1, w_i \geqslant 0\}$.

At each step $t = 1, \ldots, T$:

- we randomly choose an expert according to w_t
- nature deals us a loss function $f_t \in [-1, 1]^n$, specifying for each expert i the loss $f_t[i]$ incurred by the prediction of expert i at time t.
- we incur the expected loss $\mathbb{E}_{i \sim w_t} f_t[i] = \langle w_t, f_t \rangle$.
- we get to update our distribution to from w_t to w_{t+1}.

At the end of the day, we measure how well we performed relative to the best fixed distribution over experts in hindsight. This is called *regret*:

$$R = \sum_{t=1}^{T} \langle w_t, f_t \rangle - \min_{w \in \Delta_n} \sum_{t=1}^{T} \langle w, f_t \rangle$$

This is a relative benchmark. Small regret does not say that the loss is necessarily small. It only says that had we played the same strategy at all steps, we couldn't have done much better even with the benefit of hindsight.

10.5 Multiplicative weights update

Perhaps the most important online learning algorithm is the *multiplicative weights update*. Starting from the uniform distribution w_1, proceed according to the following simple update rule for $t > 1$,

$$v_t[i] = w_{t-1}[i] e^{-\eta f_t[i]} \qquad \text{(exponential weights update)}$$
$$w_t = v_t / (\textstyle\sum_i v_t[i]) \qquad \text{(normalize)}$$

The question is *how do we bound the regret* of the multiplicative weights update? We could do a direct analysis, but instead we'll relate multiplicative weights to gradient descent and use the convergence guarantees we already know.

10.6 Multiplicative weights as mirror descent

Recall that mirror descent requires a mirror map $\phi : \Omega \to R$ over a domain $\Omega \in \mathbb{R}^n$ where ϕ is strongly convex and continuously differentiable.

The associated projection is

$$\Pi_{\Omega}^{\phi}(y) = \arg\min_{x \in \Omega} \mathcal{D}_{\phi}(x, y)$$

where $\mathcal{D}_{\phi}(x, y)$ is Bregman divergence.

Definition 10.5. The Bregman divergence measures how good the first order approximation of the function ϕ is:

$$\mathcal{D}_{\phi}(x, y) = \phi(x) - \phi(y) - \nabla\phi(y)^{\intercal}(x - y)$$

The mirror descent update rule is:

$$\nabla\phi(y_{t+1}) = \nabla\phi(x_t) - \eta g_t$$
$$x_{t+1} = \Pi_{\Omega}^{\phi}(y_{t+1})$$

where $g_t \in \partial f(x_t)$. In the first homework, we proved the following results.

Theorem 10.6. *let $\|\cdot\|$ be arbitrary norm and suppose that ϕ is α-strongly convex **w.r.t.** $\|\cdot\|$ on Ω. Suppose that f_t is L-lipschitz **w.r.t.** $\|\cdot\|$, we have:*

$$\frac{1}{T}\sum_{t=1}^{T} f_t(x_t) \leqslant \frac{\mathcal{D}_{\phi}(x^*, x_0)}{T\eta} + \eta\frac{L^2}{2\alpha}$$

Multiplicative weights are an instance of the Mirror Descent where $\Phi(w) = \sum_{i=1}^{m} w_i \log(w_i)$ is the negative entropy function. We have that

$$\nabla\Phi(w) = 1 + \log(w),$$

where the logarithm is elementwise. The update rule in Mirror Descent becomes:

$$\nabla\Phi(v_{t+1}) = \nabla\Phi(w_t) - \eta_t f_t$$
$$\implies v_{t+1} = w_t e^{-\eta_t f_t}$$

Now comes the projection step. The Bregman divergence is, for all $(x, y) \in \Omega^2$:

$$D_{\Phi}(x, y) = \Phi(x) - \Phi(y) - \nabla\Phi(y)^{T}(x - y)\,.$$

If we write out this expression and simplify, we recover the well-known *relative entropy*. The projection

$$\Pi_{\Omega}^{\Phi}(y) = \arg\min_{x \in \Omega} D_{\Phi}(x, y)$$

turns out to just correspond to the normalization step in the update rule.

Concrete rate of convergence. To get a concrete rate of convergence from the preceeding theorem, we still need to determine what value of the strong convexity constant α we get in our setting. Here, we choose the norm to be the ℓ_∞-norm. It follows from Pinsker's inequality that Φ is $1/2$-strongly convex with respect to the ℓ_∞-norm. Moreover, in the ℓ_∞-norm all gradients are bounded by 1, since the loss ranges in $[1, 1]$. Finally, the relative entropy between the initial uniform distribution any any other distribution is at most $\log(n)$. Putting these facts together and balancing the step size η, we get the normalized regret bound

$$O\left(\sqrt{\frac{\log n}{T}}\right).$$

In particular, this shows that the normalized regret of the multiplicative update rule is vanishing over time.

References

[RM51] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.