COS 511: Theoretical Machine Learning

Homework #7 — Due:
Online learning and gradient descent — April 23, 2019

This entire problem set is concerned with a very general and powerful online learning setting called *online convex optimization*. (See the end of this problem set for brief but sufficient background on convex sets, convex functions, and projections.)

Let $\mathcal{A}$ be a convex subset of $\mathbb{R}^n$ representing the domain of interest (possibly all of $\mathbb{R}^n$). In this setting, on every round t, the learner chooses a vector $\mathbf{w}_t \in \mathcal{A}$. In response (and with knowledge of $\mathbf{w}_t$), the adversary chooses a convex, real-valued *function* $f_t : \mathcal{A} \to \mathbb{R}$. Thus, learning proceeds as follows:

for $t = 1, \ldots, T$:
  learner chooses $\mathbf{w}_t \in \mathcal{A}$
  adversary chooses a convex function $f_t : \mathcal{A} \to \mathbb{R}$

The learner's loss is then $f_t(\mathbf{w}_t)$ so that, intuitively, the learner is trying on each round to guess a vector $\mathbf{w}_t$ that is close to the minimum of the function f_t that will be chosen by the adversary. More specifically, the goal of learning is for the learner's cumulative loss to not be much worse than that of the best, fixed vector $\mathbf{u}$ for the same sequence of functions f_t. In other words, we want to guarantee that $\sum_t f_t(\mathbf{w}_t)$ is not much more than the minimum value (over $\mathbf{u}$) of $\sum_t f_t(\mathbf{u})$.

To solve this problem, we can often use an online version of gradient descent as follows: Initially, we let $\mathbf{w}_1 = \mathbf{0}$ (which we assume, without loss of generality, is included in $\mathcal{A}$). Then on each round t, we compute our new choice of $\mathbf{w}_{t+1}$ in two steps. First, we move slightly in the direction of the negative gradient, arriving at

$$\mathbf{w}'_{t+1} = \mathbf{w}_t - \eta \nabla f_t(\mathbf{w}_t),$$

where $\eta > 0$ is a parameter. Since it is possible that such a move will lead us outside the domain $\mathcal{A}$, in the second step of computing $\mathbf{w}_{t+1}$, we project back to $\mathcal{A}$, that is, we find the point in $\mathcal{A}$ that is closest to $\mathbf{w}'_{t+1}$. We write this operation as:

$$\mathbf{w}_{t+1} = \mathcal{P}_{\mathcal{A}}(\mathbf{w}'_{t+1}) = \mathcal{P}_{\mathcal{A}}(\mathbf{w}_t - \eta \nabla f_t(\mathbf{w}_t)).$$

(Throughout this homework, we assume that gradients $\nabla f(\mathbf{w})$ and projections $\mathcal{P}_{\mathcal{A}}(\mathbf{w})$ exist and are available as needed. Also, $||\cdot||$ always on this homework denotes the usual Euclidean norm $||\cdot||_2$, and all notions of distance are corespondingly with respect to this norm.)

Question 1. Suppose we run the online gradient descent algorithm described above. For this question, assume there exists $B > 0$ such that, for all t, $||\nabla f_t(\mathbf{w}_t)|| \leq B$.

a. [9] Let $\mathbf{u}$ be any fixed vector in $\mathcal{A}$, and let us define the potential function $\Phi_t = ||\mathbf{w}_t - \mathbf{u}||^2$. Prove that

$$\Phi_{t+1} - \Phi_t \leq \eta^2 B^2 - 2\eta(f_t(\mathbf{w}_t) - f_t(\mathbf{u})).$$

b. [5] Prove the following regret bound:

$$\sum_{t=1}^{T} f_t(\mathbf{w}_t) \leq \min_{\mathbf{u} \in \mathcal{A}} \left[\sum_{t=1}^{T} f_t(\mathbf{u}) + \frac{\eta B^2 T}{2} + \frac{||\mathbf{u}||^2}{2\eta} \right].$$

Question 2. *Instructions:* Please complete *either* the "basic" version *or* the "bonus" version of this question (but not both), as described below. Both versions are worth 6 "regular" homework points; the "bonus" version is also worth an additional 6 extra-credit points.

Basic version: [6] Suppose we are interested in minimizing a convex function $g : \mathcal{A} \to \mathbb{R}$ over its domain $\mathcal{A}$. Assume that $||\nabla g(\mathbf{w})|| \leq B$ and $||\mathbf{w}|| \leq R$ for all $\mathbf{w} \in \mathcal{A}$. To minimize g, we can use the online gradient descent algorithm described above with f_t set to g on every round. Let

$$\mathbf{v} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t.$$

For an appropriate setting of η (which you should state explicitly, and which can be a function of T, B and R), apply Question 1 to prove that

$$g(\mathbf{v}) \leq \min_{\mathbf{u}\in\mathcal{A}} g(\mathbf{u}) + \frac{BR}{\sqrt{T}}.$$

Bonus version: [6+6] Suppose we are interested in minimizing a convex function G over some domain $\mathcal{A}$, where G is itself an average of convex functions; that is, G is of the form

$$G(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} g_i(\mathbf{w}),$$

where each function $g_i : \mathcal{A} \to \mathbb{R}$ is convex. For instance, in linear regression, given $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$, we might want to find $\mathbf{w}$ to minimize $\frac{1}{m}\sum_{i=1}^{m}(\mathbf{w} \cdot \mathbf{x}_i - y_i)^2$, and might further wish to constrain $\mathbf{w}$ to have norm at most R. In this case, we can simply let $g_i(\mathbf{w}) = (\mathbf{w}\cdot\mathbf{x}_i - y_i)^2$, and choose $\mathcal{A}$ to be the ball of radius R about the origin. (This is just meant for illustration; there is nothing that needs to be done with regard to this particular example.)

Returning to the general case, assume that $||\nabla g_i(\mathbf{w})|| \leq B$ and that $||\mathbf{w}|| \leq R$ for all $\mathbf{w} \in \mathcal{A}$ and for all of the g_i's. To solve this minimization problem, instead of applying gradient descent to G, it is often computationally cheaper to repeatedly choose a single function g_i at random and to apply gradient descent just to that one function. This leads to the following stochastic gradient descent algorithm:

$\mathbf{w}_1 = \mathbf{0}$
for $t = 1, \ldots, T$:
  choose i_t uniformly at random from $\{1, \ldots, m\}$
  $\mathbf{w}_{t+1} = \mathcal{P}_{\mathcal{A}}(\mathbf{w}_t - \eta\nabla g_{i_t}(\mathbf{w}_t))$
output $\mathbf{v} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t$

For an appropriate setting of η (which you should state explicitly, and which can be a function of T, B and R), apply Question 1 to prove that

$$\mathrm{E}\left[G(\mathbf{v})\right] \leq \min_{\mathbf{u}\in\mathcal{A}} G(\mathbf{u}) + \frac{BR}{\sqrt{T}}$$

where expectation is over the random choice of indices i_t on each round.

(Note that the "basic" version above is exactly a special case of this algorithm and result, as can be seen by setting $m = 1$ and $G = g_1 = g$, so that $i_t = 1$ on every round.)

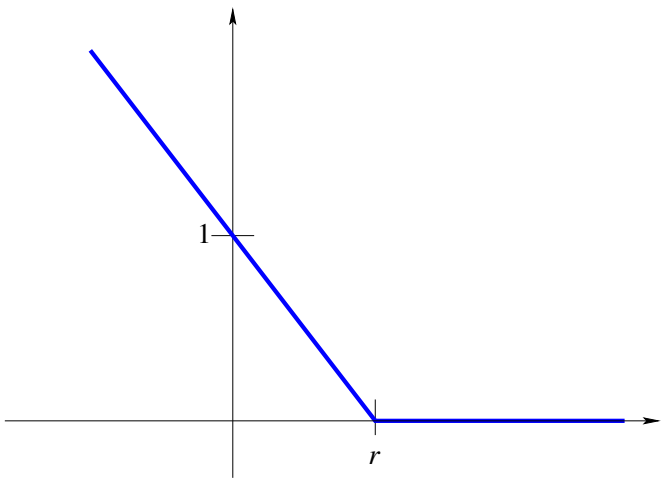

Figure 1: The function $\ell_r(z)$.

Question 3. Consider a setting like the one studied for the perceptron algorithm in which, on each round t, the adversary chooses an example $\mathbf{x}_t \in \mathbb{R}^n$ and a label $y_t \in \{-1, +1\}$ (which typically is equal to a linear threshold function of $\mathbf{x}_t$), and the goal is for the learner to predict y_t correctly as often as possible. In this question, we will see how online gradient descent can be applied to solve this problem, and how the perceptron algorithm and its analysis can be viewed as a special case of the results of Question 1.

First, for $r > 0$, let $\ell_r : \mathbb{R} \to \mathbb{R}$ be the piecewise linear function shown in Figure 1 and defined by:

$$\ell_r(z) = \begin{cases} 1 - z/r & \text{if } z \leq r \\ 0 & \text{else.} \end{cases}$$

Let $\mathcal{A} = \mathbb{R}^n$. On each round t, online gradient descent selects a vector $\mathbf{w}_t$, and the adversary selects $(\mathbf{x}_t, y_t)$, where we assume as usual that $||\mathbf{x}_t|| \leq 1$. Let $\hat{y}_t = \text{sign}(\mathbf{w}_t \cdot \mathbf{x}_t)$. In the context of the perceptron algorithm, our goal would be to minimize the number of mistakes, that is, the number of times that $\hat{y}_t \neq y_t$. However, to apply the online gradient descent algorithm to this problem, we somehow must construct a function f_t for use by that algorithm. To do this, we define f_t as follows: If $\hat{y}_t \neq y_t$ then we set $f_t(\mathbf{w}) = \ell_r(y_t(\mathbf{w} \cdot \mathbf{x}_t))$ for all $\mathbf{w}$; otherwise, we choose $f_t(\mathbf{w}) = 0$ for all $\mathbf{w}$. With this choice of f_t, the online gradient descent algorithm proceeds as usual to compute the next vector $\mathbf{w}_{t+1}$. (Technically, ℓ_r', the first derivative of ℓ_r, is not defined at r; however, for the purposes of this problem, it is okay to use $\ell_r'(r) = 0$.)

a. [5] For the choice of f_t given above, prove that the resulting instantiation of online gradient descent is exactly equivalent to the perceptron algorithm. More specifically, if the adversary chooses the same sequence of examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$, prove that the perceptron algorithm and this version of online gradient descent will produce the identical sequence of predictions $\hat{y}_t$. Your argument should hold for *all* fixed choices of $\eta > 0$ and $r > 0$.

b. [10] As usual for studying perceptron, suppose that there exist $\delta > 0$ and a vector $\mathbf{u} \in \mathbb{R}^n$ (neither of which are known to the learner) such that $||\mathbf{u}|| = 1$ and $y_t(\mathbf{u} \cdot \mathbf{x}_t) \geq \delta$ for all t. Show how the result of Question 1 yields the same mistake bound for perceptron as was previously proved in class. That is, using the result of Question 1 applied to the version of online gradient descent given above (and for an appropriate choice of $\eta > 0$ and $r > 0$), prove that the number of rounds on which $\hat{y}_t \neq y_t$ is at most $1/\delta^2$.

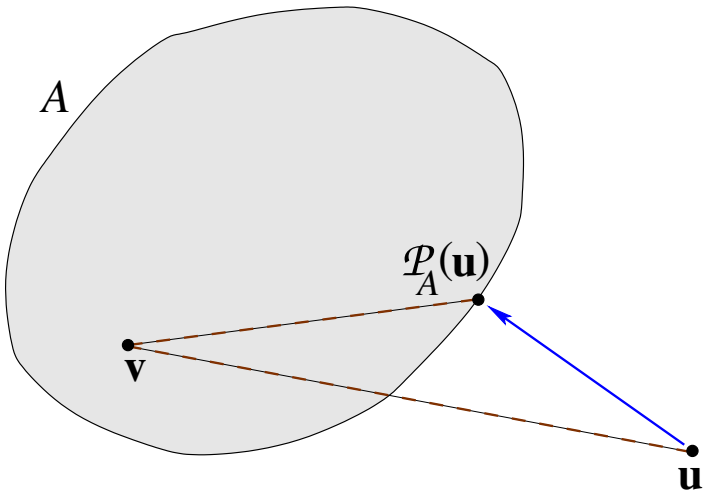

Figure 2: The projection of a point $\mathbf{u}$ onto a convex set $\mathcal{A}$. Every point $\mathbf{v} \in \mathcal{A}$ must be at least as close to $\mathcal{P}_{\mathcal{A}}(\mathbf{u})$ as it is to $\mathbf{u}$.

Appendix: Brief background on convex sets, convex functions, and projections.

(You can make use of any of the facts stated in this section without proof.)

A set $\mathcal{A} \subseteq \mathbb{R}^n$ is a *convex set* if for all $\mathbf{u}, \mathbf{v} \in \mathcal{A}$, and for all $p \in [0,1]$, the point $p\mathbf{u} + (1-p)\mathbf{v}$ is also in $\mathcal{A}$. For such a convex set $\mathcal{A}$, we say that $f : \mathcal{A} \to \mathbb{R}$ is a *convex function* if for all $\mathbf{u}, \mathbf{v} \in \mathcal{A}$, and for all $p \in [0,1]$,

$$f(p\mathbf{u} + (1-p)\mathbf{v}) \;\leq\; pf(\mathbf{u}) + (1-p)f(\mathbf{v}).$$

For instance, the functions $1 - 2x$, x^2, e^x, and $-\ln x$ are all convex on their respective domains.

The property of convexity is closed under various natural operations. For instance, the sum of two or more convex functions is convex, as is the composition of a convex function with a linear function.

A convex function $f : \mathcal{A} \to \mathbb{R}$ must lie entirely above any tangent hyperplane at any point $\mathbf{x}_0$. This means

$$f(\mathbf{x}) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0)$$

for all $\mathbf{x} \in \mathcal{A}$ (assuming the gradient $\nabla f(\mathbf{x}_0)$ exists).

Jensen's inequality states that if $f : \mathcal{A} \to \mathbb{R}$ is convex, and $\mathbf{X}$ is any real-valued random variable taking values in the convex set $\mathcal{A}$, then

$$f(\mathrm{E}[\mathbf{X}]) \leq \mathrm{E}[f(\mathbf{X})].$$

Let $\mathcal{A} \subseteq \mathbb{R}^n$ be convex. For any point $\mathbf{u} \in \mathbb{R}^n$, we define the *projection* of $\mathbf{u}$ onto $\mathcal{A}$, denoted $\mathcal{P}_{\mathcal{A}}(\mathbf{u})$, to be that point in $\mathcal{A}$ that is closest to $\mathbf{u}$. That is,

$$\mathcal{P}_{\mathcal{A}}(\mathbf{u}) = \arg\min_{\mathbf{x} \in \mathcal{A}} ||\mathbf{x} - \mathbf{u}||.$$

Naturally, if $\mathbf{u}$ is already in $\mathcal{A}$, then $\mathcal{P}_{\mathcal{A}}(\mathbf{u}) = \mathbf{u}$. On this homework, we assume we are always dealing with a set $\mathcal{A}$ for which the projection is guaranteed to exist (which basically means that it needs to be closed and nonempty). It is a fact that every point in $\mathcal{A}$ will be at least as close to the projection of $\mathbf{u}$ onto $\mathcal{A}$ as it was to the original point $\mathbf{u}$. That is, for all $\mathbf{v} \in \mathcal{A}$,

$$||\mathbf{v} - \mathcal{P}_{\mathcal{A}}(\mathbf{u})|| \leq ||\mathbf{v} - \mathbf{u}||.$$

See Figure 2.