Homework 3
CS229T/STATS231 (Winter 2015–2016)

Please structure your writeups hierarchically: convey the overall plan before diving into details. You should justify with words why something's true (by algebra, convexity, etc.). There's no need to step through a long sequence of trivial algebraic operations. Be careful not to mix assumptions with things which are derived. Up to two additional points will awarded for especially well-organized and elegant solutions.

Due date: Wednesday, March 2 (at the beginning of class)

1. Kernels (*10 points*)

This problem will explore a number of kernels and non-kernels to get some more intuition for (i) what constitutes a valid kernel and (ii) what kind of functions we can implicitly define with kernels.

(**a**) Suppose k' is a valid kernel. Define the normalized kernel k based on k' as follows:

$$k(x, x') = \frac{k'(x, x')}{\sqrt{k'(x, x)k'(x', x')}}.$$

Is k a valid kernel? Justify your result either way.

(**b**) Suppose $k' : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ is a valid kernel, and let $\mathcal{F}$ be a finite collection of functions from $\mathcal{X}$ to $\mathcal{Y}$. Define k as follows:

$$k(x, x') = \sum_{f \in \mathcal{F}} \sum_{f' \in \mathcal{F}} k'(f(x), f'(x')).$$

Is k a valid kernel? Justify your result either way.

(**c**) Let $\mathcal{X} = [0, 1]$. Define:

$$k(x, x') = \min(x, x').$$

Show that k is a valid kernel. Hint: given n points $x_1, \ldots, x_n$, try to construct a n-dimensional multivariate Gaussian distribution with covariance matrix equal to the kernel matrix.

It turns out that k is the reproducing kernel for an RKHS over the space of functions differentiable almost everywhere:

$$\mathcal{H} = \{f : f(0) = 0 \text{ and } f' \in L^2([0, 1])\},$$

with $\langle f, g \rangle = \int_0^1 f'(x)g'(x)dx$. Show the reproducing property for k holds (that $\langle k(\cdot, x), f \rangle = f(x)$).

2. Varying step sizes (*10 points*)

Choosing step sizes properly is crucial in optimization and can have a dramatic impact on performance (as you probably saw in the previous problem). The version of online gradient descent (OGD) that we studied in class uses the same step size η on each iteration. Could we do better if we allowed the learning algorithm to change the step size? In this problem, we will devlop an analysis conducive to answering this question, and then use it to obtain bounds that inform us on how to choose a good step size schedule.

In this problem, we will develop a single analysis that lets us simultaneously explore two advantages of varying step sizes: (i) taking advantage of strong convexity to obtain better regret bounds, and (ii) setting the step size without having to know T in advance.

Let $S \subseteq \mathbb{R}^d$ be a convex set of experts, and let $f_1, \ldots, f_T$ be a sequence of loss functions. We assume each f_t is strongly convex; that is, there exists a $K_t \geq 0$ (called the strong convexity parameter) such that for all $w \in S$ and any subgradient $z_t \in \partial f_t(w)$, the following holds for all $u \in S$:

$$f_t(u) \geq f_t(w) + z_t \cdot (u - w) + \frac{K_t}{2}\|u - w\|_2^2. \tag{1}$$

Intuitively, f_t is lower bounded by not just the usual linearization $z_t \cdot (u - w)$, but an additional quadratic.

We will consider the following OGD update based on eager projection (as opposed to lazy projection that we discussed in class; make sure that you see the difference between the two):

$$w_{t+1} = \Pi_S\left(w_t - \eta_t z_t\right), \quad z_t \in \partial f_t(w_t), \tag{2}$$

where Π_S is the projection onto S and $w_1 = 0$. Note that the step size η_t depends on t. Let's figure out how to set it properly to obtain low regret. In the following, let $u \in S$ be any expert.

(**a**) Prove the following regret bound:

$$\text{Regret}(u) \leq \sum_{t=1}^{T}\left[\frac{1}{2\eta_t}\left(\|w_t - u\|_2^2 - \|w_{t+1} - u\|_2^2\right) + \frac{1}{2}\eta_t\|z_t\|_2^2 - \frac{K_t}{2}\|w_t - u\|_2^2\right]. \tag{3}$$

Remark: The bound has a form similar to the one presented in class. The first term in the summand measures the size of u (though the varying step size makes the expression more complex), and the second term measures the size of z_t. Finally, notice how strong convexity helps: the larger K_t is, the lower the regret.

(**b**) Now assume that we have bounds on the norms of our experts and of our gradients: that $\|u\|_2 \leq B$ for all $u \in S$, and that $\|z_t\|_2 \leq L$ for all t.

Furthermore, assume that, when we pick varying step sizes, we will always pick them to satisfy $\frac{1}{\eta_t} \geq \frac{1}{\eta_{t-1}} + K_t$. Intuitively, the step size decreases at least as fast as what strong convexity dictates. (By convention, let $\eta_0 = \infty$.)

Note: if we had no strong convexity, i.e. if $K_t = 0$ for all t, then this assumption simply means that we require non-increasing step sizes.

Use part (a) to obtain the regret bound

$$\text{Regret} \leq 2B^2\left(\frac{1}{\eta_T} - \sum_{t=1}^{T} K_t\right) + \frac{1}{2}L^2\sum_{t=1}^{T}\eta_t. \tag{4}$$

(**c**) Let's use this result to bound regret in a setting where we have a simple strong convexity assumption and a simple schedule for our step sizes. Assume that all the loss functions are at least K-strongly convex, that is, $K_t \geq K > 0$ for all t. Show that if we set the step size according to $\eta_t = \frac{1}{\sum_{i=1}^{t} K_i}$, we obtain the following bound:

$$\text{Regret} \leq \frac{L^2(\log(T) + 1)}{2K}. \tag{5}$$

Remark: This shows that we get *logarithmic* regret for *all* strongly convex functions (not just quadratic functions), provided we set the step size to decay as $O(1/t)$. Note, however, that this bound does become increasingly bad as we lose strong convexity ($K \to 0$).

(**d**) Actually, not all hope is lost for our analysis when we lose strong convexity. Suppose that $K_t = 0$ (meaning that we make no assumptions on the f_t other than convexity). Show that if we set η_t to $\frac{B}{L\sqrt{t}}$ then we obtain the bound

$$\text{Regret} \leq 3BL\sqrt{T}. \tag{6}$$

Remark: This final result is significant because it shows that we can obtain a regret of $O(\sqrt{T})$ even without knowing T in advance.

(**e**) Let's see how our analysis guides our choice of step sizes when faced with a concrete problem. Assume that at each time step t, we get a feature vector $x_t \in \mathbb{R}^d$ for some $d \geq 2$. Suppose that $\|u\|_2 \leq B$ for all $u \in S$, $\|x_t\|_2 \leq C$, and $|y_t| \leq a$. We'll look at two fairly common loss functions:

1. Linear least-squares regression: $f_t(w) = (y_t - w \cdot x_t)^2$
2. Regularized SVM: $f_t(w) = \max\{0, 1 - y_t(w \cdot x_t)\} + \lambda\|w\|_2^2$.

How would you pick the step sizes for these two scenarios? Write your selections for the step size η_t (as a function of a, B, C, t) and give upper bounds on the resulting regret based on your calculations above. Please justify your answer.

3. Online Learning for Semidefinite Optimization (*10 points*)

A semidefinite program (SDP) is a kind of convex program with linear constraints (as in LP), plus a positive semidefinite constraint. SDPs are useful throughout several engineering settings, typically where modeling with matrices is involved (e.g. imaging, controls). For instance, in the machine learning world, they often show up in the context of clustering and of matrix completion problems.

In this problem we will show how to use online convex optimization to compute approximate solutions to semidefinite programs. In particular, consider the task of finding a positive semidefinite matrix X such that:

$$\begin{aligned} \text{tr}(A_i^\top X) &\geq b_i \quad i = 1, \ldots, m \\ \text{tr}(X) &\leq R, \end{aligned} \tag{7}$$

where without loss of generality we assume that the A_i are symmetric matrices. (Note that we are viewing this SDP as a feasibility problem for now, rather than directly as an optimization problem.)

Our algorithm to find such an X will reduce the problem to an instance of online convex optimization, where we will "play for" both nature and the learner. In the OCO game, we will consider a problem more relaxed than (7), have nature try to find feasible points X, and have the learner try in response to make the relaxed problem infeasible. (Note that nature is, for once, the one trying to solve the problem!)

The relaxed problem. For a matrix $X \succeq 0$ and distribution $w \in \Delta_m$ over constraints, define a loss function as follows:

$$\ell(X, w) = \sum_{i=1}^{m} w_i(\text{tr}(A_i^\top X) - b_i). \tag{8}$$

Intuitively, this is the expected margin – under the distribution w – by which the constraints are satisfied.

Each iteration $t = 1, \ldots, T$, the learner first chooses a distribution over the constraints $w^{(t)} \in \Delta_m$. Nature then chooses:

$$X_t = \underset{\substack{X \succeq 0 \\ \text{tr}(X) \leq R}}{\text{argmax}} \ \ell(X, w^{(t)}). \tag{9}$$

Remember that nature can look at the learner's prediction $w^{(t)}$. As suggested by our choice of X_t, our usual notion of the OCO loss function is $f_t(w) \stackrel{\text{def}}{=} \ell(X_t, w)$.

Note that using X_t corresponds to choosing the matrix that maximizes the loss of $w^{(t)}$. We can intuitively think of the $w^{(t)}$ as something like Lagrange multipliers and the X_t as approximately feasible points (in some sense, X_t is chosen to satisfy the constraints, weighted by $w^{(t)}$, by the largest possible margin). By minimizing the loss, the learner tries to shift probability mass on constraints that are violated.

(**a**) First, let's characterize nature's moves. That is, we want to know what happens when nature solves the maximization problem (9). Specifically, show that X_t must equal either 0 or $Ruu^\top$, where u is the (unit-norm) eigenvector of $\sum_{i=1}^m w_i^{(t)} A_i$ with largest eigenvalue.

(**b**) Next, we will need a technical lemma to help us bound quantities of the form $\operatorname{tr}(X^\top Y)$. This is analogous to how – in class and in previous problems – when using OCO to minimize loss functions with *vector*- rather than matrix-valued arguments, we simplified the analysis by upper-bounding vector dot products with norms.

Show that, for positive semidefinite matrices X, $\operatorname{tr}(X^\top X) \leq \operatorname{tr}(X)^2$. Also show that, for all matrices X and Y, $\operatorname{tr}(X^\top Y) \leq \sqrt{\operatorname{tr}(X^\top X)\operatorname{tr}(Y^\top Y)}$.

(**c**) Now we need to decide how the learner plays and analyze the learner's regret. The learner will use the exponentiated gradient (EG) algorithm covered in class.

Assume that $\operatorname{tr}(A_i^\top A_i) \leq L^2$ and $|b_i| \leq B$ for all i. Show that if we use the update

$$w_i^{(t+1)} \propto w_i^{(t)} \exp\left(-\eta(\operatorname{tr}(A_i^\top X_t) - b_i)\right), \quad i = 1, \ldots, m, \tag{10}$$

then

$$\operatorname{Regret}(u) \stackrel{\text{def}}{=} \sum_{t=1}^T [\ell(X_t, w^{(t)}) - \ell(X_t, u)] \tag{11}$$

$$\leq \frac{\log(m)}{\eta} + \frac{\eta T}{2}(LR + B)^2. \tag{12}$$

Show that for the appropriate choice of η, this implies:

$$\sum_{t=1}^T \ell(X_t, u) \geq \sum_{t=1}^T \ell(X_t, w^{(t)}) - (RL + B)\sqrt{2\log(m)T}. \tag{13}$$

(**d**) Finally, we want to transform the $\{X_t\}$ into a solution to the original SDP feasibility problem (7), and prove that this solution is good.

Assume that the original semidefinite program is feasible. Let $X^* = \frac{1}{T}\sum_{t=1}^T X_t$. Show that $\operatorname{tr}(X^*) \leq R$, $X^* \succeq 0$, and, for all i,

$$\operatorname{tr}(A_i^\top X^*) \geq b_i - \frac{(RL + B)\sqrt{2\log(m)}}{\sqrt{T}}. \tag{14}$$

In other words, X^* is an approximately feasible solution to the semidefinite program. (The extent to which it is infeasible inversely with the square root of the iteration count.)

4. Feedback (*0 points*)

These questions are only to help us calibrate and improve the assignments, and the responses will not impact your grade.

(**a**) How many hours did it take you to do this assignment?

(**b**) On a scale of 1 to 10, how useful was this assignment (1 was not useful, 10 was very useful)?