10725/36725 Optimization
Homework 2

Due October 9, 2012 at beginning of class

Instructions: There are four questions in this assignment. Please submit your homework as 4 separate sets of pages with your name and userid on each set. For the last question which involves coding, please print out your code and graphs and attach them to the written part of your homework and email your code to submission10725f12@gmail.com. Refer to the course webpage for policies regarding collaboration, due dates, and extensions.

1 No Regrets About Taking Optimization? (Aaditya)

1.1 A Game Against An Adversary [2.5 points]

We shall deal with a fixed, closed, non-empty, convex set $S \subset \mathbb{R}^N$, and assume that its diameter is given to be the constant $D = \max_{x,y \in S} \|x - y\|_2$. Assume that F is the set of all convex functions with domain S such that every subgradient of the function is bounded by G everywhere in its domain, ie $\forall f \in F,\ \forall x \in S,\ \forall g_x \in \partial f(x), \|g_x\|_2 \leq G$.

We are going to play a game against an adversary that will last T rounds (T is fixed). In each round t ($1 \leq t \leq T$), you have to choose a point $x_t \in S$. Simultaneously and independently, nature chooses a function $f_t \in F$. After you choose x_t, nature's choice f_t is revealed completely to you, and you incur a penalty/loss in this round of $f_t(x_t)$.

Your aim when playing this game is to choose points x_t in order to minimize your total loss/penalty $\sum_{t=1}^{T} f_t(x_t)$. Clearly this is hard because you are forced to choose x_t before finding out f_t. To make the problem easier, we are going to instead minimize *regret*. At the end of the game, you have seen $f_1, ..., f_T$, and you can calculate the best point x^* that you could have chosen, when you look back in hindsight. This x^* can be defined as the one point that minimizes your total penalty, ie $x^* = \arg\min_{x \in S} \sum_{t=1}^{T} f_t(x)$.

The difference between your total loss and the loss incurred by the best fixed point is called regret, ie $R_T = \sum_{t=1}^{T} f_t(x_t) - \min_{x \in S} \sum_{t=1}^{T} f_t(x)$ (intuitively, at the end of the game, you think "damn, I used a fancy algorithm and picked $x_1, ..., x_T$ while I could have just simply picked $x^*, x^*, ..., x^*$ and I would've suffered a smaller loss by an amount of R_T").

Is Hindsight Easy? Argue that solving for x^* is a convex optimization problem.

1.2 Projected Subgradient Descent To The Rescue! [15 points]

Consider the following algorithm. Start at an arbitrary point $x_1 \in S$. After round t, on receiving f_t, calculate any subgradient $g_t \in \partial f_t(x_t)$. Take a step in the direction of negative subgradient g_t with constant step-size η. Since you might be outside the set, project back onto the set S and choose that as your next point x_{t+1}. So $x_{t+1} = \prod_S(x_t - \eta g_t)$ where $\prod_S(x) = \arg\min_{y \in S} \|x - y\|_2$ is the projection of x onto S (nearest point in S to x).

Is Projecting Easy? Argue that finding $\prod_S(x)$ is a convex optimization problem.

We are going to show that the regret grows sublinearly - specifically, if we play the game for T rounds, then our regret is only going to grow like $R_T = O(\sqrt{T})$. The proof will keep track of $\|x_t - x^*\|^2$ where distances will always be in L2-norm. We will use the shorthand $x'_{t+1} = x_t - \eta g_t$ to denote the unprojected point, so now, $x_{t+1} = \prod_S(x'_{t+1})$.

Projecting Helps! Argue that whatever x^* might be, the distance between x_{t+1} and x^* can only be smaller than the distance between x'_{t+1} and x^*.

To Iterate Is Human, To Recurse Divine! Use the above fact and the algorithm's update rule to derive a simple recursive inequality between $\|x_{t+1} - x^*\|^2$ and $\|x_t - x^*\|^2$. Rearrange the terms to get an upper bound on $g_t^\top(x_t - x^*)$.

Doesn't Convexity Rock? Use the definition of convexity to upper bound $f_t(x_t) - f_t(x^*)$ which is the "regret due to round t", in terms of η, G and $[\|x_t - x^*\|^2 - \|x_{t+1} - x^*\|^2]$.

The Telescope Makes It Clear! Sum the above upper bound over all time steps $t = 1, ..., T$ to derive an upper bound on the regret in terms of η, G, D, T.

Balancing Act Show how to set step size η so that the total regret is at most $O(GD\sqrt{T})$.

The field of online learning focuses on average per-round regret R_T/T. In this problem, we showed that projected subgradient descent achieves a $O(1/\sqrt{T})$ regret rate, and hence the average regret tends to zero as T tends to infinity. This is called a no-regret algorithm.

1.3 Stochastic First Order Oracle Model [12.5 points]

In convex optimization theory, we often come across the black-box oracle model of computation. In this model, there is an oracle which has access to a hidden unknown function f, and we get to query that oracle at points x_t in the domain. A zeroth order oracle, when queried at x_t, will give you only the function value $f(x_t)$. A first order oracle gives you both $f(x_t)$ and any subgradient $g_t \in \partial f(x_t)$ (similarly, the second order oracle also provides a Hessian).

In the algorithm of section 1.2, note that we didn't really need to know the function f_t to calculate our next move (we only need it to calculate our regret). In fact, given only any one subgradient $g_t \in \partial f_t(x_t)$ we could have run the same algorithm, and this proof shows that we would've been sure that whatever the functions were, our average regret would grow like $O(1/\sqrt{T})$. So, we say that this algorithm achieves a regret rate of $1/\sqrt{T}$ in the first order oracle model of computation. In online learning, this is also called the full information model, where you get the entire function f_t (and hence its gradient) rather than the bandit setting where you only get to know $f_t(x_t)$ but not the function f_t.

We will show a reduction from a regret algorithm to traditional convex optimization as seen in class, ie, any algorithm that achieves a regret bound $\frac{\sum_{t=1}^{T} f_t(x_t)}{T} - f_t(x^*) = O(1/\sqrt{T})$ can be used as an algorithm to achieve a convex optimization error of $O(1/\sqrt{T})$. For this, assume that we run the same algorithm from the previous section, but with the additional knowledge that since this is a convex optimization problem (and not an online game with an adversary), we have $f_t = f$ for all time steps t.

From Regret To Error Let $x_1, ..., x_T$ be the T points chosen by our algorithm. Define $\bar{x}_T = \frac{x_1 + ... + x_T}{T}$. Explain why the following equations hold :

$$f(\bar{x}_T) \leq \frac{f(x_1) + ... + f(x_T)}{T} \tag{1}$$

$$f(\bar{x}_T) - f(x^*) = O(1/\sqrt{T}) \tag{2}$$

Stochastic convex optimization deals with the setting where a noiseless subgradient or function value may not be available (or is too costly to compute exactly). A stochastic first

order oracle captures this setting by not answering a query at x_t exactly with $(f(x_t), g_t)$, but instead with $(\hat{f}(x_t), \hat{g}_t)$ satisfying $\mathbb{E}[\hat{f}(x_t)] = f(x_t)$, and $\mathbb{E}[\hat{g}_t] = g_t \in \partial f(x_t)$ with $\|\hat{g}_t\| \leq G$. Here, the expectation is with respect to any randomness of the oracle for this particular query (for example, when you query at x_t, it may add Gaussian noise to the true values before returning them to you; alternatively, if f is a sum of a loss function over a very large number of training examples, it may compute an approximate subgradient at a point by only evaluating it over a random subset of examples). We will show that the exact same algorithm works with noisy subgradients by deriving a bound that looks like $\mathbb{E}f(\bar{x}_T) - f(x^*) = O(1/\sqrt{T})$, where the expectation is over all the randomness of the oracle during all rounds.

Great Expectations Let $\mathbb{E}_{t-1}$ denote taking expectation of the randomness of round t *conditioned* on all randomness till round $t-1$. Justify why the following equations hold:

$$\mathbb{E}_{t-1}[\hat{g}_t] = g_t \tag{3}$$

$$f(x_t) - f(x^*) \leq \mathbb{E}_{t-1}[\hat{g}_t]^\top (x_t - x^*) \tag{4}$$

$$\mathbb{E}[f(x_t)] - f(x^*) \leq \mathbb{E}[\hat{g}_t]^\top (x_t - x^*) \tag{5}$$

The rest of the proof should carry through just like the noiseless regret case (but you don't need to show that it does), finally proving that projected subgradient descent achieves a $O(1/\sqrt{T})$ rate for any general convex function over any general convex set in the black-box stochastic first order oracle model of optimization.

2 Subgradient Smörgåsbord (Shiva)

2.1 Normal-er

Let $C \subset \mathbb{R}^n$ be a possibly non-convex set. During our discussion of subgradients, we said that g is normal to C at x when $g^T(x - x') \geq 0$ for all $x' \in C$. The set of all such vectors constitutes the normal cone $\mathcal{N}_C(x)$. Some examples are drawn in the slides. Let's consider some special normal vectors, but in reverse: first geometry, then subgradients. Let the projection $\Pi_C(x)$ map x to the set of points of C closest to x. If g satisfies $x \in \Pi_C(x + \tau g)$ for some $\tau > 0$, then g is called p-normal of C at x. For nice C, this distinction lacks a difference.

(a) [4 points] Prove that for convex C every normal is p-normal. (Write the projection as an optimization, then consider optimality conditions.)

g is called a p-subgradient of f at x if there exist $\rho > 0$ and $\delta > 0$ such that for all x' that are δ-close to x,

$$f(x') \geq f(x) + g^T(x' - x) - \frac{\rho}{2}||x' - x||^2$$

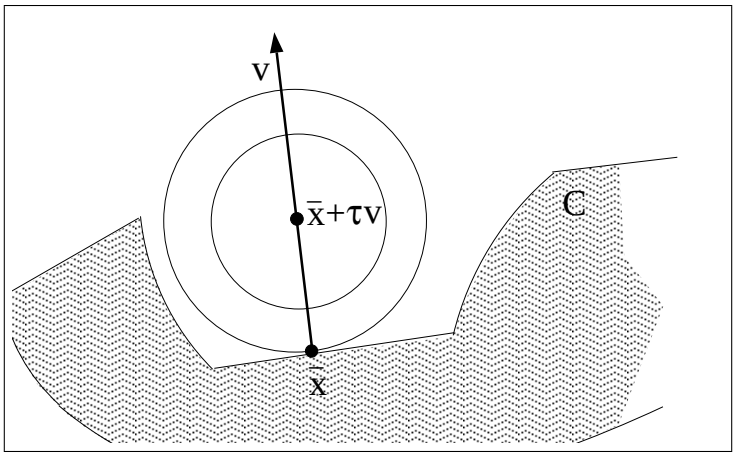

Figure 1: The p-normal v of C at $\bar{x}$ points from $\bar{x}$ to the center of a closed ball that touches C at only $\bar{x}$. Sorry for the inconsistent notation.

(b) [8 points] Prove that g is a p-subgradient of the indicator function I_C at x only if g is p-normal to C at x. Don't assume C is convex. Drop some terms, write in terms of $\epsilon = 1/\rho$, apply Cauchy-Schwarz, and incorporate $\tau = \min(\epsilon, \delta/2||g||)$.

2.2 Decomposable norms

Recall the 'nuclear' (or 'trace') norm is

$$||A||_* = \sum_{i=1}^{r} \sigma_i$$

where $\{\sigma_i\}$ are the singular values of the rank-r matrix $A \in \mathbb{R}^{m \times n}$ with $m \leq n$. We will now prove its subdifferential is

$$\partial||A||_* = \{UV^T + W : U^T W = 0, WV = 0, ||W||_2 \leq 1, W \in \mathbb{R}^{m \times n}\} \tag{6}$$

where $A = U\Sigma V^T$ is singular-value decomposed (into $U \in \mathbb{R}^{m \times r}$, $\Sigma \in \mathbb{R}^{r \times r}$, and $V \in \mathbb{R}^{n \times r}$) and $||\cdot||_2$ is the spectral norm. There is a direct way to grind this out with matrix derivatives. Instead, we'll take the scenic route, featuring projections, orthogonal decompositions, and the usual matrix inner product $\langle A, B \rangle = \sum_{ij} A_{ij} B_{ij} = \text{tr}(A^T B)$. The key insight is that, at each A, the subdifferential of the nuclear norm admits a subspace $\mathcal{T}$ upon which it can be 'decomposed.'

$$\mathcal{T} = \{UY^T + XV^T : X \in \mathbb{R}^{m \times r}, Y \in \mathbb{R}^{n \times r}\} \cap \{\text{matrices with orthonormal rows}\}$$

As we'll see in recitation, a similar property is shared by the ℓ_1 norm. Let $\Pi_{\mathcal{T}}$ and $\Pi_{\mathcal{T}^\perp}$ denote projections upon $\mathcal{T}$ and its orthogonal complement, respectively.

(a) [0 points] It turns out that $\Pi_{\mathcal{T}}(A) = UV^T$.

(b) [4 points] Prove that W in (6) is orthogonal to $\mathcal{T}$. Just simplify the definition.

Using the above results, we can set $Z = UV^T + W$, and rewrite (6) as:

$$\partial||A||_* = \{Z : \Pi_{\mathcal{T}}(Z) = UV^T, ||\Pi_{\mathcal{T}^\perp}(Z)||_2 \leq 1\} \tag{7}$$

We're calculating the subdifferential of a norm, not some arbitrary convex function, so let's exploit this additional structure. In particular, the dual norm

$$||Z||_*^* = \max_{||A||_* \leq 1} \langle Z, A \rangle$$

can be used to rewrite the subdifferential as

$$\partial||A||_* = \{Z : \langle Z, A \rangle = ||A||_*, ||Z||_*^* \leq 1\}$$

That follows from applying the subdifferential calculus (in particular, the max rule) to the standard subgradient definition. The trace norm's dual is the spectral norm; we won't prove that. We can complete the proof by showing that

$$\partial||A||_* = \{Z : \langle Z, A \rangle = ||A||_*, ||Z||_2 \leq 1\}$$

(c) [4 points] Via just equalities, prove any Z satisfying (7) satisfies $\langle Z, A \rangle = ||A||_*$. Use previously mentioned techniques.

(d) [5 points] Via equalities followed by one last inequality, prove any Z satisfying (7) satisfies $||Z||_2 \leq 1$. Use $||A||_2 = \max_{x \neq 0} ||Ax||/||x||$, where $||\cdot||$ is the ℓ_2 norm, and that $\Pi_{\mathcal{T}}(Z)$ has orthonormal rows.

(e) [1 points] Prove (in one sentence) why any Z satisfying $\langle Z, A \rangle = ||A||_*$ and $||Z||_2 \leq 1$ satisfies (7).

3 Logistic Regression (Kevin)

In this question, we will derive a fast and numerically stable training procedure for binary logistic regression based on the iteratively-reweighted least squares algorithm. First, we define the logistic function as

$$\sigma(t) = \frac{1}{1 + e^{-t}}. \tag{8}$$

(a) [3 points] Compute the derivative $\sigma'(t)$ and write it in terms of $\sigma(t)$.

We will learn a probabilistic binary classifier from a set of training data. Let $\mathcal{X}^-, X^+ \subseteq \mathbb{R}^n$ denote finite sets of negative and positive examples respectively.

Given an instance, $x \in \mathbb{R}^n$, we let the conditional probabilities be

$$P(Y = -|X = x, \theta) = \sigma(x^T\theta), \text{ and} \tag{9}$$
$$P(Y = +|X = x, \theta) = 1 - \sigma(x^T\theta). \tag{10}$$

Here, $\theta \in \mathbb{R}^n$ is a linear function of the features that we will learn by maximizing the likelihood of our training set. We define the likelihood function as

$$L(\theta|\mathcal{X}^-, \mathcal{X}^+) = \prod_{x\in\mathcal{X}^-} P(Y = -|X = x, \theta) \prod_{x\in\mathcal{X}^+} P(Y = +|X = x, \theta). \tag{11}$$

As the likelihood function is non-negative and log is strictly increasing, it is equivalent to maximize the log-likelihood.

(b) [3 points] Write down and simplify the log-likelihood function, $f(\theta) = \log L(\theta|\mathcal{X}^-, \mathcal{X}^+)$. Your final answer should not be in terms of σ.

(c) [4 points] For a fixed x, write $\nabla \log P(Y = -|X = x, \theta)$ with respect to θ in terms of σ and x using matrix differentials.

(d) [4 points] For a fixed x, write $\nabla^2 \log P(Y = -|X = x, \theta)$ with respect to θ in terms of σ and x using matrix differentials. What statistical quantity does this expression remind you of?

(e) [3 points] Write down the gradient, $\nabla f(\theta)$.

(f) [3 points] Write down the Hessian, $\nabla^2 f(\theta)$.

When applying Newton's method to this problem, we compute the search direction, $\Delta\theta^k$, on the kth iteration by solving the system $\nabla^2 f(\theta^k)\Delta\theta^k = \nabla f(\theta^k)$. Normally we would use a line search to guarantee convergence with Newton's method, but let's hope for the best and let $\theta^{k+1} = \theta^k + \Delta\theta^k$.

Now, we reformulate this Newton step as a weighted least-squares regression:

(g) [7 points] Write θ^{k+1} as the solution to an optimization problem of the form

$$\theta^{k+1} = \theta^k + \Delta\theta^k = \operatorname*{argmin}_{\gamma\in\mathbb{R}^n} \sum_{x\in X} w_x \left[y_x - h_x(\gamma)\right]^2. \tag{12}$$

That is, what are w_x, y_x and $h_x(\gamma)$ in terms of x, its label and $\sigma(x^T\theta^k)$.
Hint: w_x is in terms of x and $\sigma(x^T\theta^k)$, y_x is in terms of x, its label and $\sigma(x^T\theta^k)$, and $h_x(\gamma)$ is in terms of x. Also, recall that the solution of $\operatorname{argmin}_x x^TQx/2 - b$ is $x = Q^{-1}b$ for symmetric positive definite Q.

(h) [3 points] How does this iteratively-reweighted least squares approach that you just derived compare with the first-order methods we've discussed in class? Under what circumstances would the Newton method be better or worse? Your answer should be a few sentences and should describe a situation where you'd prefer using Newton's method and a situation where you'd prefer using a first-order method.

There are two major advantages to using iteratively-reweighted least squares to solve for the Newton step when learning a logistic regression classifier. First, we can employ specialized solvers that have been optimized for weighted least squares. Second, this approach can be more numerically stable. In particular, instead of manipulating XWX^T, we can instead factorize $XW^{1/2}$ to compute the next iterate more accurately.

Additionally, by understanding that the logistic regression classifier is the solution to a particular weighted least squares problem, we gain insight into its statistical properties. For example, we can derive confidence intervals for and analyze the correlations between the classifier's parameters.

4 Node by node revisited (Wooyoung)

In Homework 1, given a small number of p-dimensional iid samples $T = \{X^{(1)}, \ldots, X^{(m)}\}$, we estimated the inverse of the covariance matrix Σ^{-1} that characterized the multivariate Gaussian distributions from which the samples were drawn. Although the sparsity of the inverse of the covariance was desired, we did not explicitly introduce the sparsity constraint to the objective function since we only knew how to deal with convex and differentiable functions. Now that we know how to deal with convex but non-differentiable objective functions, let's minimize the new objective function with the $L1$ norm term :

$$L(\Theta) = -\ell_T(\Theta) + \lambda||\Theta||_1 = -\log\,\det(\Theta) + \mathrm{tr}(S\Theta) + \lambda||P * \Theta||_1$$

where Θ is a symmetric positive semidefinite $p \times p$ matrix meant to estimate Σ^{-1}, λ is a non-negative scalar, the empirical covariance matrix $S = \frac{1}{m-1}\sum_{i=1}^{m}(X_i - \bar{X})(X_i - \bar{X})^T$ and the sample mean $\bar{X} = \frac{1}{m}\sum_{i=1}^{m} X_i$. $*$ refers to element-wise matrix multiplication and P is a square matrix of size $p \times p$ with all all off-diagonal elements set to one and diagonal set to zero. Note that in this way, we only penalize the off-diagonal elements of Θ. In this problem, we are going to implement the iterative soft-thresholding algorithm and the fast iterative soft-thresholding algorithm. Please download the data files from `http://www.cs.cmu.edu/~ggordon/10725-F12/hws/hw2/`. In addition to the printout, please submit your code by

1. zipping the three files for (a), (b) and (c) into one zip file (please name it youAndrewID_hw2.zip)
2. send the zip file to submission10725f12@gmail.com before the due date. Please include in

the subject line of your email, your name, AndrewID and the assignment number.

(a) [6 pts] Implement the soft-thresholding operator in Matlab or R. Please name the function softthresholder.m or softthresholder.r.

(b) [6 pts] Implement the iterative soft-thresholding algorithm in Matlab or R. Please name the function ista.m or ista.r.

(c) [6 pts] Implement the fast iterative soft-thresholding algorithm in Matlab or R. Please name the function fista.m or fista.r.

Run ISTA and FISTA for $K = 5000$ iterations with 30 values of logarithmically spaced λs between 0.001 and 0.3 (e.g., use logspace(log10(0.001),log10(0.30),30)), starting from the identity matrix. We will refer to the estimate of k th iteration with λ as $\Theta_\lambda^{(k)}$. Use a fixed step size $t = 0.001$. Also, run the steepest descent algorithm with L1-norm (you can either use your own implementation for Homework 1 or the solution) for $K = 5000$ iterations starting from the same initial point and the step size.

(d) [4 pts] Plot the log of objective function values minus constant, $\log(L(\Theta_\lambda^k) - c)$ for the train set each iteration $k = 1, \cdots, K$ for both ISTA and FISTA when $\lambda = 0.0106$. Set the constant $c = \min(L^{FISTA}(\Theta_\lambda^K), L^{ISTA}(\Theta_\lambda^K)) - 10^{-6}$ (where $L^{FISTA}(\Theta_\lambda^K)$ refers to the objective function value of Θ estimated with FISTA at K th iteration with λ.

(e) [4 pts] For ISTA and FISTA each, plot the coefficient profiles of the off-diagonal elements of $\Theta_\lambda^{(k=5000)}$ (as $\Theta^{k=5000}$ is a symmetric matrix, you can plot either the elements of the upper or the lower triangular matrix) as a function of $||P * \Theta_\lambda^{(k=5000)}||_1$ for all values of λ.

(f) [4 pts] For the steepest descent gradient method with L1-norm, plot the off-diagonal elements of $\Theta^{(k)}$ as in (e) as a function of $||P * \Theta^{(k)}||_1$ for $k = 1, \cdots, K$. Compare with the results in (e).