Name:

AMATH 515 Homework Set 3

Due: Wednesday, March 3rd by midnight.

Let f be a closed proper convex function. The convex conjugate of f, called f^*, is defined by

$$f^*(z) = \sup_x \left\{ z^T x - f(x) \right\}.$$

(1) Compute the conjugates of the following functions.
   (a) $f(x) = \delta_{\mathbb{B}_\infty}(x)$.
   (b) $f(x) = \delta_{\mathbb{B}_2}(x)$.
   (c) $f(x) = \exp(x)$.
   (d) $f(x) = \log(1 + \exp(x))$
   (e) $f(x) = x \log(x)$

(2) Let g be any convex function; f is formed using g. Compute f^* in terms of g^*.
   (a) $f(x) = \lambda g(x)$.
   (b) $f(x) = g(x - a) + \langle x, b \rangle$.
   (c) $f(x) = \inf_z \{g(x, z)\}$.
   (d) $f(x) = \inf_z \left\{ \frac{1}{2}\|x - z\|^2 + g(z) \right\}$

(3) Moreau Identities.
   (a) Derive the Moreau Identity:

$$\text{prox}_f(z) + \text{prox}_{f^*}(z) = z.$$

   (b) Use the Moreau identity and 1a, 1b to check your formulas for

$$\text{prox}_{\|\cdot\|_1}, \quad \text{prox}_{\|\cdot\|_2}$$

   from last week's homework.

(4) Duals of regularized GLM. Consider the Generalized Linear Model family:

$$\min_x \sum_{i=1}^n g(\langle a_i, x\rangle) - b^T Ax + R(x),$$

Where g is convex and R is any regularizer.

(a) Write down the dual obtained by dualizing g.

(b) Specify your formula to Ridge-regularized logistic regression:

$$\min_x \sum_{i=1}^n \log(1 + \exp(\langle a_i, x\rangle)) - b^T Ax + \frac{\lambda}{2}\|x\|^2.$$

(c) Specify your formula to 1-norm regularized Poisson regression:

$$\min_x \sum_{i=1}^n \exp(\langle a_i, x\rangle) - b^T Ax + \lambda\|x\|_1.$$

Coding Assignment

Please download `515Hw3_Coding.ipynb` and `proxes.py` to complete problem (5).

(5) In this problem you will write a routine to project onto the capped simplex.

The Capped Simplex Δ_k is defined as follows:

$$\Delta_k := \left\{x : 1^T x = k, \quad 0 \le x_i \le 1 \quad \forall i.\right\}$$

This is the intersection of the k-simplex with the unit box.

The projection problem is given by

$$\text{proj}_{\Delta_k}(z) = \arg\min_{x\in\Delta_k} \frac{1}{2}\|x - z\|^2.$$

(a) Derive the (1-dimensional) dual problem by focusing on the $1^T x = k$ constraint.

(b) Implement a routine to solve this dual. It's a scalar root finding problem, so you can use the root-finding algorithm provided in the code.

(c) Using the dual solution, write down a closed form formula for the projection. Use this formula, along with your dual solver, to implement the projection. You can use the unit test provided to check if your code id working correctly.

(6) In this problem you will learn apply proximal operators to matrices to perform matrix completion. You'll find that you can recover most of the information in a low rank matrix despite only seeing a small percentage of the entries. At the end of this problem, you'll see something that is, in my opinion, **truly remarkable**.

Consider a matrix X, (for example, where it is a matrix of ratings, and $X_{i,j}$ is the rating that user i gave to item j). However, you only observed the entries $(i,j) \in \Omega$ where Ω is a subset of the entries $\{(i,j)\}_{i,j=1}^{n}$. In order to solve this, we assume that the matrix X is low rank, and we will use this to try prior knowledge to try and recover it. Similar to how the lasso or ℓ_1 penalty promotes sparsity in regression, the nuclear norm $\|\cdot\|_*$ promotes low rank matrices.[1] We will implement and experiment with a handful of approaches to the matrix completion problem.

(a) One natural approach to setting up an optimization problem that models this situation is to assume that X is rank k, and solve the optimization problem

$$\begin{aligned} &\text{minimize } \|P \odot (Y - X)\|_F^2 \\ &\text{subject to } \operatorname{rank}(Y) \leq k \end{aligned} \tag{1}$$

where $\odot$ denotes the elementwise product, and P is a matrix with $P_{i,j} = 1$ when $(i,j) \in \Omega$ and 0 if $(i,j) \notin \Omega$ (essentially a matrix of which entries are measured). **Is this problem convex? Why or why not?**

(b) We relax the constraint above into a nuclear norm constraint

$$\begin{aligned} &\text{minimize } \|P \odot (Y - X)\|_F^2 \\ &\text{subject to } \|Y\|_* \leq K \end{aligned}$$

Is this problem convex? Why or why not? Rather than the problem above, we replace this with a penalized version,

$$\text{minimize } \|P \odot (Y - X)\|_F^2 + \lambda \|Y\|_* \tag{2}$$

because the projection onto the ℓ_1 ball requires a little bit of work to write (though it can be done exactly in $O(n \log(n))$, using a sort+O(n) operations to collect the result, and is faster if most of the entries are zero. This is even better for us since we're already computing an SVD so the singular values will come sorted.) This approach is known as soft-impute in the literature.

(c) Derive formulas for (or procedures for computing) the following proximal and projection operators

$$\operatorname{prox}_{t\|\cdot\|_*}(Y) = \arg\min_M \frac{1}{2t}\|Y - M\|_2^2 + \|M\|_*$$

[1] One interpretation of this assumption is that a low rank model is equivalent to a latent factor model with low dimensional latent factors. Assume that X is rank d. This is like saying that $X_{i,j} = \langle u_i, v_j \rangle$ for some set of vectors $\{u_i\}$ representing the rows (or users), and $\{v_j\}$, representing the columns (or items), with $u, v \in \mathbb{R}^d$

$$\text{proj}_{\text{rank}_k}(Y) = \arg\min_{\text{rank}(M)\leq k} \frac{1}{2}\|Y - M\|_2^2$$

$$\text{proj}_{*K}(Y) = \arg\min_{\|Y\|_*\leq K} \frac{1}{2}\|Y - M\|_2^2$$

For the third one, you may write the answer in terms of an ℓ_1 ball projection

$$\text{proj}_{\ell_1\leq R}(x) = \arg\min_{\|y\|_1\leq R} \frac{1}{2}\|x - y\|_2^2$$

(d) Suppose we want to solve the optimization problem in (b) for many values of λ to give ourselves the best chance of finding a good recovery (performing some kind of cross validation or scoring to pick the best one. In the coding section, we'll just compare the true reconstruction error $\|X - Y\|_F^2$ for simplicity in order to understand how changing the regularization parameter changes the performance of the model. This data wouldn't be available in practice since we only observe $P \odot X$ but there are ways to approximate that quantity. However, as you'll see, even for small examples, this problem can be somewhat time consuming. **What could we do to speed this up? Do you expect a larger value of λ to make the problem converge faster or slower?**

(e) Suppose that we rather than knowing the observed entries of X exactly, there were some large corrupted entries. **What could we change to make our problem robust to such corruptions?**

(f) Fill in Problem 6 in the jupyter notebook, the nuclear norm and rank projection in proxes.py

There are a couple of other tricks to make this scalable to real world big data sets which we're not making you do in this class. The biggest speed up would come from using a "partial" or "truncated" SVD at each step since we only care about the top components, along the lines of the algorithm in https://arxiv.org/pdf/1607.03463.pdf. Additionally, because they apply an iterative algorithm, you can both warm start the SVD itself with the previous SVD, and stop computing additional components once you know they will be "proxed" down to zero (either with the nonconvex projection or with nuclear norm prox).

For **extra credit** implement the LazySVD algorithm in arxiv.org/pdf/1607.03463.pdf with a warm start for the nuclear norm regularized problem, and adaptively quit computing components when you are sure that the rest will be sent to zero by the prox operator! You may have to write a custom accelerated proximal gradient descent function for this, as you'll need to save the decompositions between iterations!

Ultimately, for the best performance (assuming very low rank and very few observed entries), you would want to avoid multiplying the matrices together, in the SVD, making each matrix vector product $O(kn + |\Omega|)$, where the kn comes from the two factor matrices, and $|\Omega|$ comes from the sparsity of the gradient (the gradient of the smooth part is zero except for where you observed the entries of X).