Homework 6

Due: 10am Thursday 11/30/17

1. *Proximal Gradient Method*

The *proximal operator* of a function $r : \mathbf{R}^d \to \mathbf{R}$ is defined as

$$\text{prox}_r(z) = \underset{w}{\text{argmin}} \left(r(w) + \frac{1}{2}\|w - z\|_2^2 \right).$$

In class, we saw how to use the ℓ_1 regularizer to encourage sparsity. In this problem, we will see a different regularizer that *enforces* sparsity.

(a) Define the *k-sparse indicator* $\mathbf{1}_k : \mathbf{R}^d \to \mathbf{R} \cup \{\infty\}$,

$$\mathbf{1}_k(w) = \begin{cases} 0 & \text{nnz}(w) \leq k \\ \infty & \text{otherwise} \end{cases}$$

where $\text{nnz}(w) =$ the number of non-zero entries of w.
Compute the proximal operator of the k-sparse indicator $\mathbf{1}_k$.

(b) A function $f : \mathbf{R}^d \to \mathbf{R}$ is *convex* if the line between any two points on the function lies (weakly) above the graph of the function. Formally, $\forall x, y \in \mathbf{R}^d$ and $\forall t \in [0, 1]$,

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y).$$

Find a counterexample in two dimensions that shows the 1-sparse indicator is not convex. (That is, $d = 2$ and $k = 1$.)

(c) Our goal now is to solve the Sparse Least Squares (SLS) problem,

$$\min \|Xw - y\|^2 + \mathbf{1}_k(w).$$

Write pseudocode describing how the proximal gradient method could be used to solve this problem. (That is, write the prox and gradient steps explicitly for this loss function and regularizer.)

(d) Code the proximal gradient method for the SLS problem and run it on the instance in

`http://github.com/orie4741/homework/blob/master/ProxGradHomework.ipynb`.

You may find the julia function `sortperm` useful. Recall that the Lipschitz constant of the gradient of the least squares objective is $L = 2\|X\|^2$, where $\|X\|$ is the largest singular value of X. Make sure to use an appropriate step size to ensure convergence.
Plot the objective value as a function of the number of iterations. (This is called a convergence plot; it helps us understand how quickly the method finds a solution, and the quality of that solution.) You may want to plot the y axis of the plot on a log scale.

(e) Run the algorithm starting at multiple locations and create a histogram of the objective value. Use 100 iterations for each run. What do you observe?

(f) Solve the LASSO problem (ℓ_1 regularized least squares regression) using the proximal gradient method on this problem. You may use the code from the demo in class, found at `https://github.com/ORIE4741/demos/blob/master/ProximalGradient.ipynb`.

(g) Does LASSO converge to the same place starting from different initial vectors w^0?

(h) Compare the SLS solution with the LASSO solution. Which is more sparse? Which achieves a better objective value? Which method is more reliable?

2. *Stochastic proximal gradient method.*

(a) Write pseudocode for the stochastic proximal gradient method applied to the Sparse Least Squares problem above. Use one data example at each iteration to compute the stochastic approximation to the gradient.

(b) Code the stochastic proximal gradient method for the Sparse Least Squares problem.

(c) Using the same data used in `ProxGradHomework.ipynb`, plot the objective value as a function of the number of iterations.

(d) How long does the stochastic proximal gradient method take compared to the standard proximal gradient method? Compare both the number of iterations and the time required for convergence. You may find Julia's `@time` macro useful: place it in front of line of code to evaluate the running time of that line.

(e) Run the stochastic algorithm starting at multiple locations and create a histogram of the final objective values. What do you observe?

3. *Grading by Matrix Completion.* ORIE 4741 uses peer grading to determine scores on final projects. Each project has an underlying quality; some are good, some less good. Some students are fair graders, and report the project quality as their grade. Some are easy graders, and report a higher grade. Some are harsh graders, and report a lower grade. As a result, some students fear that peer grading is unfair: why should their grade suffer simply through the random chance of a harsh reviewer?

   An ideal solution might be to have every student grade every project. However, this solution is rarely popular with reviewers. Instead, in this homework problem, we will explore whether we can predict the ratings that all other reviewers *would have* given had they reviewed all projects. We will try this technique both on a synthetic data and on the real peer-review rating scores for ORIE4741 projects in 2017.

   Formally, let's define our problem. There are d students enrolled in ORIE 4741 who have formed n project groups. Each student is responsible for grading p projects. (In our class, p=2.) We'll collect the grades into a grade matrix $A \in \mathbb{R}^{n\times d}$: A_{ij} will represent the grade that student j would assign to project i.

   Of course, we cannot assign each student to grade every project. Instead, we make peer review assignments $\Omega = \{(i_1, j_1), \dots, \}$. Here, $(i, j) \in \Omega$ if student j is assigned to grade project i.

   Unfortunately, this means that some projects are assigned harder graders than other projects. Our goal is to find a fair way to compute a project's final grade. We consider two methods:

   - *Averaging.* The grade g_i for project i is the average of the grades given by peer reviewers:

   $$g_i^{\text{avg}} = \frac{1}{p} \sum_{j:(i,j)\in\Omega} A_{ij}$$

   - *Matrix completion.* We fit a low rank model to the grade matrix and use it to compute an estimate $\hat{A}$ of the grade matrix by solving

   $$\text{minimize} \quad \sum_{(i,j)\in\Omega} \ell(A_{ij}, (X^TY)_{ij}) + r(X) + r(Y),$$

   where $X \in \mathbb{R}^{k\times n}$ and $Y \in \mathbb{R}^{k\times d}$. We will try a few different losses ℓ and regularizers r to see which work best.

   We will then compute our estimate $\hat{A}$ as

   $$\hat{A} = X^TY.$$

   In other words, $\hat{A}$ is the rank-k matrix that matches the observations best in the sense of Huber error.

We compute the grade g_i for project i as the average of these estimated grades:

$$g_i^{\text{mc}} = \frac{1}{n} \sum_{j=1}^{n} \hat{A}_{ij}$$

In this problem, we will consider which of these two grading schemes, averaging or matrix completion, is better.

(a) *Analytical problem.* Consider $m = 2$ project groups and $n = 4$ peer graders. Suppose group 1 did well on their project and deserves a grade of 6; whereas group 2 deserves a grade of 3. Graders 1 and 2 are easy graders, and graders 3 and 4 are harsh.

Each project is graded by three graders. The grades given are

$$X = \begin{bmatrix} \times & 8 & 4 & 4 \\ 4 & 4 & 2 & \times \end{bmatrix}.$$

Here, an $\times$ in the (i, j)th entry means the jth student was not responsible for grading the ith project.

Use both methods, averaging and matrix completion, to compute grades for the two groups. Here, you should be able to compute the results of both methods by hand (on paper). Explain how you computed $\hat{A}$.

Compare your results. Which grading method would you say is more fair?

(b) *A more realistic example.* Work on the Jupyter notebook at `github.com/ORIE4741/homework/blob/master/MatrixCompletionforPeerGrading.ipynb`.