The University of Texas at Austin
Department of Electrical and Computer Engineering

EE381V: Large Scale Optimization — Spring 2020

Problem Set Two

Constantine Caramanis — Due: Friday, March 13, 2020.

Computational Problems

Save your completed code in a file named hw2.ipynb, and please upload this, as well as a pdf printout of your notebook. Don't use stock optimization code, you should develop the core part of this assignment yourself. All plots should have titles, axis labels, and lines with different colors, markers, and legend labels.

1. Lasso: Sub-Gradient Descent, ISTA, FISTA, and Frank Wolfe.

   Lasso is the least squares regression problem with ℓ^1 regularization:

   $$\min_{\boldsymbol{x}} \left[f(\boldsymbol{x}) = \frac{1}{2} ||A\boldsymbol{x} - \boldsymbol{b}||_2^2 + \lambda ||\boldsymbol{x}||_1 \right].$$

   You will use a diabetes regression dataset split into training and test sets (`A_train.npy`, `b_train.npy`) and (`A_test.npy`, `b_test.npy`). As the objective above can be partitioned as $f(x) = g(x) + h(x)$ where $g(x)$ is smooth and $h(x)$ has a "simple" prox operator, proximal gradient can be used to solve this problem. Implement four algorithms: subgradient method, ISTA, FISTA, and Frank Wolfe, and compare their performance. For all methods, try to optimize the step size and regularization parameters for performance on the test set after at least 10^4 iterations, while only performing optimization over $\boldsymbol{x}$ on the training set. Plot the performance in terms of the (unsquared) error $||A\boldsymbol{x}_t - \boldsymbol{b}||$ for all three methods on the test and training set separately. There should be two plots, one for training error and one for test error, each with four lines, one for each optimization method.

2. Logistic Regression

   Logistic regression is a simple statistical classification method which models the conditional distribution of the class variable y being equal to class c given an input $\boldsymbol{x} \in \mathbb{R}^n$. We will examine two classification tasks, one classifying newsgroup posts, and the other classifying digits. In these tasks the input $\boldsymbol{x}$ is some description of the sample (e.g., word counts in the news case) and y is the category the sample belongs to (e.g., sports, politics). The Logistic Regression model assumes the class distribution conditioned on $\boldsymbol{x}$ is log-linear:

   $$p(y = c|\boldsymbol{x}, b_{1:C}) = \frac{e^{-b_c^\top \boldsymbol{x}}}{\sum_{j=1}^{C} e^{-b_j^\top \boldsymbol{x}}},$$

   where C is the total number of classes, and the denominator sums over all classes to ensure that $p(y|\boldsymbol{x})$ is a proper probability distribution. Each class $c \in 1, 2, \ldots, C$ has a parameter b_c, and $\mathbf{b} \in \mathbb{R}^{nC}$ is the vector of concatenated parameters $\mathbf{b} = [b_1^\top, b_2^\top, \ldots, b_C^\top]^\top$. Let $X \in \mathbb{R}^{N \times n}$ be the data matrix where each sample $\boldsymbol{x}_i^\top$ is a row and N is the number of samples. The

maximum likelihood approach seeks to find the parameter $\mathbf{b}$ which maximizes the likelihood of the classes given the input data and the model:

$$\max_{b_{1:C}} \; p(y_{1:N}|x_{1:N}, b_{1:C}) = \prod_{i=1}^{N} p(y_i|\boldsymbol{x}_i, b_{1:C}) = \prod_{i=1}^{N} \frac{e^{-b_{y_i}^\top \boldsymbol{x}_i}}{\sum_{j=1}^{C} e^{-b_j^\top \boldsymbol{x}_i}}.$$

For the purposes of optimization, we can equivalently minimize the negative log likelihood:

$$\min_{\beta} \ell(\beta) = -\log p(\mathbf{y}|X, \beta) = \sum_{i=1}^{N} \left(\beta_{y_i}^\top x_i + \log \sum_{j=1}^{C} e^{-\beta_j^\top x_i} \right).$$

After optimization, the model can be used to classify a new input by choosing the class that the model predicts as having the highest likelihood; note that we don't have to compute the normalizing quantity $\sum_{j=1}^{C} e^{-b_j^\top \boldsymbol{x}}$ as it is constant across all classes:

$$y = \arg\max_j p(y = j|\boldsymbol{x}, \beta) = \arg\min_j \beta_j^\top \boldsymbol{x}.$$

In this problem, you will optimize the logistic regression model for the two classification tasks mentioned above which vary in dimension and number of classes. The newsgroup dataset that we consider here has $C = 20$.

We will compare the performance of gradient descent and Nesterov's accelerated gradient method on the ℓ^2-regularized version of the logistic regression model:

$$\min_{\boldsymbol{\beta}} = \frac{1}{N} \sum_{i=1}^{N} \left(\beta_{y_i}^\top \boldsymbol{x}_i + \log \sum_{j=1}^{C} e^{-\beta_j^\top \boldsymbol{x}_i} \right) + \mu \, ||\boldsymbol{\beta}||^2 .$$

We discussed the idea behind "momentum" and the accelerated gradient descent method in class. More details of this can be found in Chapter 3 of Bubeck's notes.

Use the training and testing data contained in the four csv files packaged in `logistic_news.zip` on Canvas.

(a) Find the value of μ that gives you (approximately) the best generalization performance (error on test set). You obtain this by solving the the above optimization problem for different values of μ, and then checking the performance of the solution on the testing set, using the unregularized logistic regression loss. Note that this is not a question about an optimization method.

What value do you get for the test loss after convergence?

(b) Plot the loss against iterations for both the test and training data using the value of μ from part (a).

(c) How do the two algorithms differ in performance, and how does this change as you decrease μ?

(d) Explain the difference in convergence in terms of the condition number of the problem (note that the loss is μ-strongly convex).

(e) (Optional) Repeat using the data set `digits.zip`

3. (Optional) As we saw in class, Proximal Gradient is a descent algorithm. On the other hand, as you see from the above examples, accelerated (proximal) gradient is not. We can easily make it a descent method by adding a step where $\boldsymbol{x}$ is updated if the update has a lower objective value, but otherwise $\boldsymbol{x}_t = \boldsymbol{x}_{t-1}$. Note that since the accelerated method depends on the last two steps of the trajectory, the algorithm can still continue without getting stuck. Implement this small change in those examples above for which accelerated gradient did not give a descent method, and see how this version compares (i.e., plot the results on the same plot).

4. (Optional) Play around with variants of Nesterov's acceleration, to explore the issues brought up in the paper: "A Differential Equation for Modeling Nesterov's Accelerated Gradient Method: Theory and Insights" by W. Su, S. Boyd and E. Candes `https://arxiv.org/abs/1503.01243`.

5. Consider the problem of robust regression, where some small number of measurements have been potentially completely corrupted. One way to formulate an optimization problem to solve this robust regression is as follows:

$$\begin{aligned} \min_{\beta} : \quad & \|X\beta - \mathbf{y}\|_1 \\ \text{s.t.} : \quad & \beta \in \mathfrak{X}. \end{aligned}$$

The rationale for this formulation stems from the idea that because of the ℓ^1-error, huge errors are not disproportionately penalized, as they would be in the squared error formulation (this is the formulation we have worked with before, including in the previous problem), and therefore the optimal solution is less sensitive to outliers. You will solve this problem using Projected Subgradient Descent, and also Mirror Descent. Let the constraint set be the simplex:

$$\mathfrak{X} = \{\beta \,:\, \beta \geq 0, \sum \beta_i = 1\}.$$

(a) Write down the update for projected gradient descent.

The data for this problem is generated from the 20 newsgroups dataset [1]. First, the documents are represented as a vector of (normalized) word frequencies. Then, Nonnegative Matrix Factorization (NMF) is used to find "topics" which efficiently represent the collection of documents by assuming that each document is a distribution over topics [2]. You will find the data in the files `X.npy` and `y.npy`. X is an $n \times m$ matrix where n is the number of unique words considered and m is the number of topics. Given these topics, we represent a new document as a distribution over topics given by β. Given the (noisy) word frequencies y, your goal is to recover that distribution over topics β.

(b) (Optional) Explore the robustness properties of this formulation. Compare the performance of standard regression (least squares), with the above formulation using the corrupted observations `y_corr.npy`.

[1] `scikit-learn.org/stable/datasets/twenty_newsgroups.html`

[2] See `scikit-learn.org/stable/modules/generated/sklearn.decomposition.NMF.html` and the references therein.

6. **Mirror Descent**

Consider the same problem of robust regression that you had on the previous problem. There, you solved it using SGD. Repeat the problem, but now using Mirror Descent. Specifically:

(a) Write down the mirror descent update. For this, we will use the mirror map $\Phi(\beta) = \sum \beta_i \log \beta_i$ that we used in class. Compute the Bregman divergence, $D_\phi(\mathbf{x}, \mathbf{y})$ explicitly.

(b) Using the data in `X.npy` and `y.npy`, and using stepsizes of your choosing, compare the projected subgradient method (from the last problem) with mirror descent. What is β? Plot the objective above against iterations for both methods (in a single plot).

7. **Matrix Completion**.

In this problem we investigate **low-rank matrix completion**, the problem of finding a low-rank matrix given only a few (randomly sampled entries). While this is (clearly) not possible in general, somewhat remarkably, it is possible once some additional assumptions are made on the problem setup (for example, for a "random" low-rank matrix and random samples). The assumptions required for guarantees on matrix completion are beyond the scope of this class. Our goal is to develop a projected subgradient algorithm to solve such a problem.

Suppose there is a true matrix $M \in \mathbb{R}^{m \times n}$ that we want to recover, but we are only given elements in the set $\Omega \subset [m] \times [n]$ (i.e. we know the value of m_{ij} if $(i, j) \in \Omega$). We want to solve the following constrained optimization problem

$$\begin{aligned} \min_X \quad & \|X\|_* \\ s.t. \quad & x_{ij} = m_{ij} \text{ for all } (i,j) \in \Omega \end{aligned}$$

where the variable of optimization $X \in \mathbb{R}^{m \times n}$ is a matrix. Here $\|\cdot\|_*$ is the "nuclear" norm, equal to the sum of singular values of the matrix. This norm is a convex but not smooth function of X; we will implement projected sub gradient descent for this problem.

The sub gradient of the $\|\cdot\|_*$ function is as follows: for any matrix X, if its SVD is $U\Sigma V'$, then a matrix $Z \in \partial\|X\|_*$ is in its sub gradient if and only if

$$Z = UV' + W$$

where W is such that *(a)* the column and row spaces of W are perpendicular to the corresponding ones of X, and *(b)* the spectral norm $\|W\|_2 \leq 1$. Recall that the spectral norm of a matrix is its maximum singular value. Also recall that if X is rank r, then the matrices U, V are of sizes $m \times r$ and $n \times r$ respectively, and have orthonormal columns.

(a) Given a matrix X, how will you generate an element $Z \in \partial\|X\|_*$ using a singular value decomposition function in Python (e.g., in np.linalg)?

(b) Given a matrix X, how will you project it onto the feasible set (i.e. the set of matrices that satisfy the constraints) ?

(c) Implement projected sub gradient descent with two choices for step sizes: $\eta_k = \frac{1}{k}$ and $\eta_k = \frac{1}{\sqrt{k}}$. You will need to use the files contained in `MatrixCompletion.zip`, which contains two 100×100 matrices: a low-rank matrix M, and the matrix O that represents the set Ω by having entries that are 0 or 1 (in particular, $O_{ij} = 1$ means $(i, j) \in \Omega$).

(d) Plot the relative error $\frac{1}{100^2}\|M - X_k\|_F^2$ between the true matrix and the k^{th} iterate, as a function of k, for both step size choices; do so on one plot.

(e) What is the rank of the intermediate iterates? Why is this the case?

Written Problems

(a) Prove that a matrix Z as described above in the matrix completion problem is indeed a sub gradient to the nuclear norm function at X. You can use the following fact about the nuclear norm: for any matrix $M \in \mathbb{R}^{m \times n}$, let $s = \min(m, n)$. Then for any matrices $A \in R^{m \times s}$ and $B \in R^{n \times s}$ that have orthonormal columns, we have that

$$\|M\|_* \ \geq \ \langle M, AB' \rangle$$

(b) (Optional) Re-do the computation we did in class, showing that if we use the mirror function

$$\Phi(\mathbf{x}) = \frac{1}{2}\|x\|_2^2,$$

then the Mirror Descent update for:

$$\begin{aligned} \min_{\mathbf{x}} : \quad & f(\mathbf{x}) \\ \text{s.t.} : \quad & \mathbf{x} \in \mathfrak{X}, \end{aligned}$$

is exactly projected subgradient descent.

(c) Here you will do some work that helps compute the Mirror Descent update you need for the computational problem above. As we discussed in class, and also as is explained in Section 4.2 of Bubeck's notes, the Mirror Descent update can also be obtained as:

$$x_{t+1} = \arg \min_{x \in \mathcal{X} \cap \mathcal{D}} : \ \eta \langle g_{x_t}, x \rangle + D_\Phi(x, x_t).$$

Therefore, Mirror Descent is only computationally useful if we can easily solve the problem:

$$\min_{u \in \mathcal{X}} : \ \langle z, u \rangle + \Phi(u).$$

In this problem, you will show that when $\mathcal{X}$ is the simplex, i.e., $\mathcal{X} = \Delta_n$, then this problem is indeed easy.

i. Consider the optimization problem:

$$\begin{aligned} \min : \quad & \langle z, u \rangle + \Phi(u) \\ \text{s.t.} : \quad & \sum_i u_i = 1. \end{aligned}$$

(Note that the constraints $\{u_i \geq 0\}$ are implicitly included as they are part of domΦ.) Write the Lagrangian for the problem. The variables will be u and a single variable λ for the single constraint. Write the optimality conditions for the problem.

ii. Using the optimality conditions, derive a closed form expression for u as a function of z.

iii. Now go back to the Mirror Descent update and write explicitly the Mirror Descent update using your work above.