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

EE381V: Adv. Convex Optimization — Spring 2019

Problem Set Three

Constantine Caramanis Due: Tuesday, February 12, 2019.

Reading Assignments

Not directly related to this problem set, but nevertheless of interest: Read "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`.

1. (?) Read Chapter 3 in S. Bubeck's notes, especially the portion about Conditional Gradient / Frank-Wolfe, and Accelerated Gradient Descent. Also read Chapter 3 in Nocedal and Wright, about backtracking line search. Note that an e-version of this book is available for free from UT Austin Libraries.

Computational Problems

Save your completed code in a file named `hw3.py`, or in an analogously named Jupyter 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. Revisiting Lasso: ISTA[1]

   We will revisit the Lasso problem from the previous problem set. Recall that Lasso is least squares regression with ℓ^1 regularization:

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

   You will use the same diabetes regression dataset as in Problem Set 2, this time 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. Compare the performance of proximal gradient to the subgradient and Frank-Wolfe methods you implemented in the last assignment. For subgradient and Frank-Wolfe, use the fixed step size schedules given last time. For proximal gradient, the step size is only involved in the update for the smooth term, so you can use a constant step size. For all three methods, 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 three lines, one for each optimization method.

[1] ISTA is the term given for running proximal gradient descent on Lasso.

2. More Lasso, ISTA and FISTA[2]

   In the previous problem, the function was not strongly convex. Repeat the above idea, but now in the presence of strong convexity. Thus, consider the objective function

   $$f(\boldsymbol{x}) = \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \|\boldsymbol{x}\|_1,$$

   where $A \in \mathbb{R}^{3{,}000\times 1{,}500}$ is randomly generated (i.e., you should generate it randomly). Plot the solution obtained using the subgradient method, ISTA and FISTA. Note that for the exact coefficients for FISTA, you should refer to Bubeck (or elsewhere). In class, we only specified Nesterov's accelerated algorithm for smooth functions.

3. In logistic regression (we will see more of this in the next problem), we seek to minimize a convex but not strongly convex function of the form:

   $$\min: \ \log \sum_{i=1}^{m} \exp(\boldsymbol{a}_i^\top \boldsymbol{x} + b_i),$$

   where each $\boldsymbol{a}_i \in \mathbb{R}^n$. Let $m = 3,000$ and $n = 1,500$, randomly generate data, and solve the above problem 3 times using Gradient Descent and Accelerated Gradient Descent, and plot the results. Fix your random seed so that your experiments are easily repeatable.

4. More Logistic Regression

   Some more details about Logistic Regression and the previous problem: 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:

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

[2] FISTA is the term given for running accelerated proximal gradient descent on Lasso ("Fast ISTA = FISTA").

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:

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

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).

5. (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).

6. (Optional) Play around with variants of Nesterov's acceleration, to explore the issues brought up in the Su, Boyd, Candes paper cited above.

Written Problems

1. Given a strongly convex function $\Phi(\boldsymbol{x})$, the *Bregman-Divergence* associated to $\Phi(\cdot)$ is defined via
$$D_\Phi(\boldsymbol{x}\|\boldsymbol{y}) = \Phi(\boldsymbol{x}) - \Phi(\boldsymbol{y}) - \langle \nabla\Phi(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y}\rangle.$$
Given a convex function $\Phi(\cdot)$, and hence the associated Bregman divergence, we can define a projection operation onto a convex set $\mathcal{X}$, with respect to this Bregman divergence:
$$\Pi_{\mathcal{X}}^{\Phi}(\boldsymbol{y}) = \arg\min : D_\Phi(\boldsymbol{x}\|\boldsymbol{y}), \quad \text{s.t.} \quad \boldsymbol{x} \in \mathcal{X}.$$
For $\Phi(\boldsymbol{x})$ given by
$$\Phi(\boldsymbol{x}) = \sum x_i \log x_i.$$
show that the projection onto the simplex
$$\Delta_n = \{\boldsymbol{x} \in \mathbb{R}^n \, : \, \sum x_i = 1, \, x_i \geq 0\}.$$
is given by $L1$ renormalization:
$$\boldsymbol{y} \mapsto \frac{\boldsymbol{y}}{\|\boldsymbol{y}\|_1}.$$

2. For Φ strongly convex and twice differentiable, and for the Bregman divergence defined as above, show that:
$$D_\Phi(\boldsymbol{x}\|\boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_{\nabla^2\Phi(\boldsymbol{z})},$$
for some $\boldsymbol{z} \in [\boldsymbol{x}, \boldsymbol{y}]$, i.e., for some $\boldsymbol{z}$ in the convex combination of $\boldsymbol{x}$ and $\boldsymbol{y}$.

Recall that for a positive definite matrix M, the Euclidean norm with respect to M is given by
$$\|\boldsymbol{x}\|_M^2 = \boldsymbol{x}^\top M \boldsymbol{x}.$$