Homework 2
Canonical Forms, Applications
Descent Algorithms & Line Search, and Gradient Descent

CMU 10-725/36-725: Convex Optimization (Fall 2017)

OUT: Sep 15
DUE: **Sep 29, 5:00 PM**

START HERE: Instructions

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 3.4"). Second, write your solution independently: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only.
- **Submitting your work:** Assignments should be submitted as PDFs using Gradescope unless explicitly stated otherwise. Each derivation/proof should be completed on a separate page. Submissions can be handwritten, but should be labeled and clearly legible. Else, submissions can be written in LaTeX. Upon submission, label each question using the template provided by Gradescope. Please refer to Piazza for detailed instruction for joining Gradescope and submitting your homework.
- **Programming**: All programming portions of the assignments should be submitted to Gradescope as well. We will not be using this for autograding, meaning you may use any language which you like to submit.

1 Schur Complement and Row Selection (25 pts) [Yichong]

1. Given a symmetric matrix $X \in \mathbb{S}^n$ partitioned as

$$X = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}$$

with $A \in \mathbb{S}^k$, and A is nonsingular. The matrix

$$S = C - B^T A^{-1} B$$

is called the Schur complement of A in X. Here $\mathbb{S}^n$ represents the set of all $n \times n$ symmetric matrices.

   (a) [**5pts**] Suppose $A \succ 0$, i.e., A is positive definite. Consider the problem

$$\min_{u \in \mathbb{R}^k} u^T A u + 2u^T B v + v^T C v$$

   where $u \in \mathbb{R}^k$ is to be optimized and $v \in \mathbb{R}^{n-k}$ is fixed. Compute the optimal value of this problem as a function of v and S.

   (b) [**5pts**] Prove that if $A \succ 0$, then $X \succeq 0$ if and only if $S \succeq 0$, i.e., X is positive semi-definite if and only if S is positive semi-definite.
   Hint: Use the fact that M is positive semi-definite if and only if $w^T M w \geq 0$ for all w; and use the result of part (a).

2. [**15pts**] Suppose $X \in \mathbb{R}^{n \times p}$. The following program appears in row subset selection problems:

$$\begin{aligned} \min_{\pi \in \mathbb{R}^n} \ & \operatorname{tr}\left(\left(X^T \operatorname{diag}(\pi) X \right)^{-1} \right) \\ \text{s.t. } & \pi \geq 0, 1^T \pi = 1. \end{aligned}$$

   Suppose the objective value is $+\infty$ when $X^T \operatorname{diag}(\pi) X$ is singular. Show that the above program is convex. Here $\pi \in \mathbb{R}^n$ and $\operatorname{diag}(\pi)$ is a $n \times n$ matrix with π being on the diagonal.
   Hint: Treat the trace as a sum of p variables, where each of the p variables are set to some expression involving X and π. Transform the equation defining each of these elements into an SDP constraint, by using the the identity: $t \geq u^T B^{-1} u \Leftrightarrow \begin{bmatrix} B & u \\ u^T & t \end{bmatrix} \succeq 0$..

2 Applications (25 pts) [Hao]

1. **(Extremal Volume Rectangle)** [**10pts**] Formulate the following problem as a convex optimization problem:

   Find the rectangle

$$R = \{ x \in \mathbb{R}^n \mid l \preceq x \preceq u \}$$

   of maximum volume, inscribed in a polyhedron $P = \{x | Ax \preceq b\}$, where for vectors x and y of the same dimension, $x \preceq y$ means $x_i \leq y_i$ for each i (x_i is the i-th element of x). The variables are $l, u \in \mathbb{R}^n$. Your answer should not involve an exponential number of constraints.
   Hint: $sup_{l \leq x \leq u} a^T x = \sum_{j=1}^n sup_{l_j \leq x_j \leq u_j} a_j x_j$.

2. **(Statistical estimation)** [**15pts**] Suppose $x_i, i = 1, ..., n$, are independent random variables with Poisson distributions

$$\mathbf{prob}(x_i = k) = \frac{e^{-\mu_i} \mu_i^k}{k!}$$

with unknown means μ_i. The variables x_i represent the number of times that one of n possible independent events occurs during a certain period.

We consider an experiment designed to determine the means μ_i. The experiment involves m detectors. If event i occurs, it is detected by detector j with probability p_{ji}. We assume the probabilities p_{ji} are given (with $p_{ji} \geq 0$, $\sum_{j=1}^{m} p_{ji} \leq 1$). y_{ji} is the variable that indicates the number of event i recorded by detector j, and the total number of events recorded by detector j is denoted y_j,

$$y_j = \sum_{i=1}^{n} y_{ji}, \ j = 1, ..., m.$$

Formulate the maximum likelihood estimation problem of estimating the means μ_i, based on observed values of $y_j, j = 1, ..., m$, as a convex optimization problem.

Hint: The variables y_{ji} subject to the Poisson distribution with means $p_{ji}\mu_i$, i.e.,

$$\mathbf{prob}(y_{ji} = k) = \frac{e^{-p_{ji}\mu_i}(p_{ji}\mu_i)^k}{k!}.$$

3 Descent Algorithms and Line Search (25 pts) [Yifeng]

For a differentiable convex function $f(x)$ with Lipschitz gradient, we have proved in class the gradient descent with backtracking has a convergence rate of $O(1/k)$, where $k \geq 1$ is the number of iterations that the algorithm is run for.

When $f(x)$ is not differentiable, but can be represented as the sum of a non-differentiable convex function $h(x)$ and differentiable convex function $g(x)$, proximal gradient descent (which will be introduced in class later) can be used. Never mind if you don't know proximal gradient descent, because this question is self-contained.

In this problem, you will show that the convergence rate for proximal gradient descent with backtracking is also $O(1/k)$. We will setup for proximal gradient descent and lead you to go through the proof. We assume that the objective $f(x)$ can be written as $f(x) = g(x) + h(x)$ (more details on these functions are given below); then, we compute the iterates

$$x^{(i)} = \text{prox}_{t_i h}\left(x^{(i-1)} - t_i \nabla g(x^{(i-1)})\right), \tag{1}$$

where $i \geq 1$ is an iteration counter, $x^{(0)}$ is the initial point, and the $t_i > 0$ are step sizes (chosen appropriately, during iteration i). The proximal mapping $\text{prox}_{th}(x)$ is defined as:

$$\underset{z}{\text{argmin}} \frac{1}{2t}||z - x||_2^2 + h(z).$$

To be clear, we are assuming the following conditions here.

(A1) g is convex, differentiable, and $\text{dom}(g) = \mathbb{R}^n$.

(A2) ∇g is Lipschitz, with constant $L > 0$.

(A3) h is convex, not necessarily differentiable, and we take $\text{dom}(h) = \mathbb{R}^n$ for simplicity.

(A4) The step sizes t_i are either taken to be constant, i.e., $t_i = t = 1/L$, or chosen by backtracking line search; either way, the following inequality holds:

$$g(x^{(i)}) \leq g(x^{(i-1)}) - t\nabla g(x^{(i-1)})^T G_t(x^{(i-1)}) + (t/2)\|G_t(x^{(i-1)})\|_2^2, \tag{2}$$

where t is the step size at any iteration of the algorithm, and we define

$$G_t(x^{(i-1)}) = (1/t)\left(x^{(i-1)} - x^{(i)}\right).$$

(In case you are wondering, this inequality follows from assumption (A2), but you can just take it to be true for this problem.)

Now, finally, for the problem. Assume, for all parts of this problem except the last one, that the step size is fixed, i.e., $t_i = t = 1/L$.

(a, 5pts) Derive the following (helpful) inequality:

$$f(x^{(i)}) \leq f(z) + G_t(x^{(i-1)})^T(x^{(i-1)} - z) - (t/2)\|G_t(x^{(i-1)})\|_2^2, \quad z \in \mathbb{R}^n.$$

Note: You can directly use the fact (2) and the following fact throughout question (a) to (e) without proof:

$$h(z) \geq h(x^{(i)}) + (G_t(x^{(i-1)}) - \nabla g(x^{(i-1)}))^T(z - x^{(i)}), \quad z \in \mathbb{R}^n.$$

(b, 5pts) Show that the sequence of objective function evaluations $\{f(x^{(i)})\}$, $i = 0, \ldots, k$, is nonincreasing (don't worry about the case when $x^{(i)}$ is a minimizer of f). (By the way, this result basically says that proximal gradient descent is a "descent method".)

(c, 5pts) Derive the following (helpful) inequality:

$$f(x^{(i)}) - f(x^*) \leq \frac{1}{2t}\left(\|x^{(i-1)} - x^*\|_2^2 - \|x^{(i)} - x^*\|_2^2\right),$$

where x^* is a minimizer of f (we assume $f(x^*)$ is finite). (By the way, this result, taken together with what you showed in part (b), implies that we move closer to the optimal point(s) on each iteration of proximal gradient descent.)

(d, 5pts) Now, show that after k iterations, the accuracy that proximal gradient descent (with a fixed step size of $1/L$) obtains is $O(1/k)$, i.e.,

$$f(x^{(k)}) - f(x^*) \leq \frac{1}{2kt}\|x^{(0)} - x^*\|_2^2;$$

in other words, the convergence rate for proximal gradient descent is $O(1/k)$. (Put differently, if you desire ε-level accuracy, roughly speaking, then you must run proximal gradient descent for $O(1/\varepsilon)$ iterations.)

(e, 5pts) Establish the analogous convergence rate result when the step sizes are chosen according to backtracking line search, i.e.,

$$f(x^{(k)}) - f(x^*) \leq \frac{1}{2kt_{\min}}\|x^{(0)} - x^*\|_2^2,$$

where $t_{\min} = \min_{i=1,\ldots,k} t_i$.

4 Programming Problem (25pts) [Devendra]

This part consists of an implementation of a one-versus-all, L2 loss SVM also known as squared hinge loss, which is commonly used for multiclass classification. L2 SVM is differentiable and imposes a bigger penalty on points which violates the margin. In one-versus-all (OVA) approach, we train $|\mathbf{C}|$ binary classifiers, one for each class and during inference time, we select the class which classifies the test data with maximum margin.

Given a training set $\mathbf{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, where $\mathbf{x}_i \in \mathbb{R}^n$, $y_i \in 1, .., C$ where C is the number of classes, we would like to minimize the following objective function:

$$f(x) = \min_{\mathbf{w}^{(j)} \in \mathbb{R}^n} \frac{1}{2} \sum_{j=1}^{C} \|\mathbf{w}^{(j)}\|_2^2 + \frac{\lambda}{m} \sum_{(\mathbf{x}_i, y_i) \in S} \sum_{j=1}^{C} l(\mathbf{w}^{(j)}; (\mathbf{x}_i, y_i))^2$$

where

$$l(\mathbf{w}^{(j)}; (\mathbf{x}_i, y_i)) = \max\{0, 1 - (\mathbb{1}\{y_i = j\}(\langle \mathbf{w}^{(j)}, \mathbf{x}_i \rangle + b^{(j)}))\}$$

and

$$\mathbb{1}\{y_i = j\} = \begin{cases} 1 & \text{if } y_i = j \\ -1 & \text{if } y_i \neq j \end{cases}$$

$f(x)$ is also known as the cost function. In order to update the parameters $\mathbf{w}$ and b of the SVM objective function, gradient descent techniques are used.

Gradient computation Compute the gradient of the objective function $f(x)$ w.r.t the parameters $\mathbf{w}^{(j)}$. [**5pts**]

Implementation tasks You will implement batch gradient descent and mini-batch stochastic gradient descent algorithms in order to update the parameters $\mathbf{w}^{(j)}$ by minimizing the cost function $f(x)$. In both the cases, initialize the values of $\mathbf{w}^{(j)}$ with zero vector and update the parameters for a maximum of 200 epochs. There is no explicit need for bias parameter during implementation as an additional feature (all 1's) for the bias term has been included in the dataset.

- **Batch Gradient Descent**: In this, the parameters are updated by iterating through the full dataset in every epoch. Plot the value of the objective function, train accuracy, test accuracy after every epoch for $\lambda = 0.1, 0.5, 1, 10$. Use learning rate as 0.005. [**10pts**]
- **Mini-Batch Stochastic Gradient Descent**: In this, the parameters are updated by iterating through randomly sampled subsets of training data at every epoch. Plot the value of the objective function, train accuracy, test accuracy on every epoch for $\lambda = 0.1, 1, 30, 50$. Use learning rate as 0.001 and batch size as 5000. [**10pts**]

You can use any programming language to implement the above functionality.

Dataset description: The training and test data consists of features extracted from CIFAR-10 images which is available from here https://www.dropbox.com/s/4zgesaqlfxoe811/hw2_q4_dataset.zip. There are 4 csv files inside this compressed file whose descriptions are below:

- *train_features.csv*: training set features
- *train_labels.csv*: training set labels
- *test_features.csv*: test set features
- *test_labels.csv*: test set labels

The training set consists of $50,000$ examples and test set consists of $10,000$ examples. Each example is in a new line and has 401 features which are separated by comma delimiter. Overall there are 10 classes in this dataset.