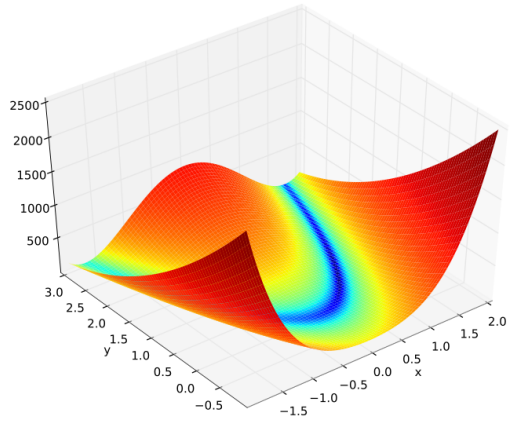
CMPSCI 690: Optimization for Computer Science Midterm Project

Due: Submit on Moodle by March 8 2016

Abstract

This is an individually carried out mid-term project based on the material covered in the course. Solutions should be *computer formatted*. This is a fairly **long** assignment. It cannot be done over one weekend. Begin early. Late submissions will be penalized at 10% of the grade per day. If you cannot finish the assignment, turn in what ever you have managed to complete. A partial grade is better than no grade! You may discuss the questions with other students taking this class, but your final answers must be your own. You can also discuss these questions on the class forum on `piazza.com`. Any plagiarism will be dealt with following the College of Information and Computer Sciences policy on cheating.

Question 1: Rosenbrock's function (20 points)



The Rosenbrock function, illustrated above, is specified as $f(x, y) = (1 - x)^2 + 100(y - x^2)^2$. Your task is to implement the following methods for find the minimum of this function and compare their performance:

- Steepest descent: Show the behavior of steepest descent for a variety of learning rates (step sizes) and initial starting points.
- Newton's method: Show the convergence rate of Newton's method.

Question 2 Subgradients (20 points)

- **part a** (10 points) If $f(x) = \max_{i=1}^{n}\{f_1(x), \ldots, f_n(x)\}$, where each f_i is a perhaps non-smooth convex function, show that f is convex, and derive a rule for computing its subgradient $g(x) \in \partial f(x)$ at some point x. Use your rule to compute the subgradient of the function $f(x) = \|x\|_1$.

- **part b** (10 points) Consider the projected subgradient method, which is defined as follows:

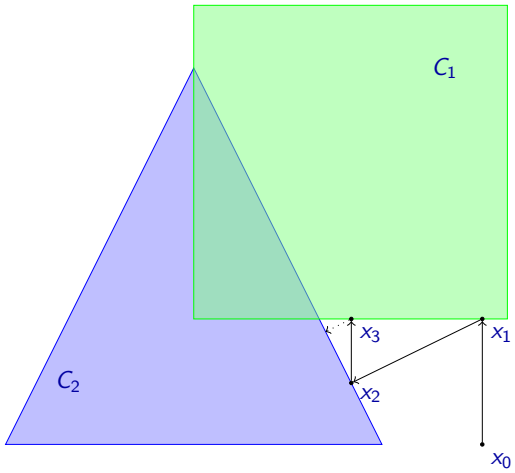
$$x_{k+1} \leftarrow P(x_k - \alpha_k \partial f(x_k))$$

Implement the method on the following optimization problem:

$$\begin{aligned} \min \quad & {}_{x \in X} f(x) = x_1 + 2x_2 - 1 \\ \text{where} \quad & X = \{x \in \mathbb{R}^2 | x_1 + x_2 = 1, x_1, x_2 \geq 0\} \end{aligned}$$

Apply the projected subgradient method starting at the point $x^0 = (0, 1)$. Graph the convergence of the method. Explain the behavior of the algorithm.

Question 3 (25 points)



Recall that the Kacmarz algorithm for solving an overdetermined system of linear equations is based on the method of successive projections (illustrated in the above diagram). Implement the Kacmarz algorithm in any language of your choice, and study its behavior for different choices of the next set (equation) to project on. In particular, compare the following three choices:

- *Cyclic:* Here, the successive projections are on successive hyperplanes, numbered sequentially.
- *Random:* Here, a random hyperplane is chosen to project on next.
- *Distal:* Here, the furthest hyperplane is chosen to project on.

Can you construct example problems where one method outperforms the other?

Question 4 (25 points)

In the lectures, we introduced the concept of *oblique* projections and contrasted them with orthogonal projections. Read the paper `Should one compute the Temporal Difference fix point or minimize the Bellman Residual ? The unified oblique projection view` attached with the assignment, which uses oblique projections to solve an optimization problem involved in optimal sequential decision making. Answer the following questions:

- Work through and describe the missing steps in Example 1. Is your result consistent with what the authors present?
- Explain in your own words, with a simple example, how the TD(0) method and the BR method involve oblique projections.
- **Optional extra credit 10 points**: Suppose we use the Kaczmarz algorithm to solve the same problem addressed in this paper? What new algorithm do we get?

Question 5 (35 points)

The goal of this problem is to implement a set of *matrix completion* algorithms on a real-world movie recommendation system called MovieLens (`http://movielens.umn.edu`). You are given a (highly sparse) matrix of user's ratings of movies, and are required to "complete" this matrix. Let $\mathcal{U}$ be a set of n users and $\mathcal{M}$ be a set of d movies. The unknown user ratings matrix R is of size $d \times n$. Matrix completion algorithms work by assuming that the rank k of R is low, so that $k \ll \min(n, d)$. You are given a training set matrix T of size $d \times n$ of known user recommendations, where each user recommends at least 20 movies, and each movie is recommended by at least one user. Your goal is to implement and test a variety of matrix completion methods, and test them on a sparse test matrix S. The goal is to find a method whose performance as measured by the root mean squared error (RMSE) is the lowest.

$$\text{RMSE}(\hat{R}, S) = \sqrt{\frac{1}{\|\Omega_S\|} \sum_{(i,j) \in \Omega_s} (\hat{R}_{ij} - S_{ij})^2}$$

where Ω_S is the set of non-zero indices of test matrix S, and $\hat{R}$ is the estimated recommendation matrix.

The paper attached to this assignment describes several matrix completion algorithms for the MovieLens dataset, and the web page cited above explains the format of the MovieLens datasets. For the 100k data set, you should use the u1.base, u1.test ... u5.base, u5.test files for five fold validation. The 1M data set does not come in this format, so we are giving you a shellscript file that will split the data into 5 training sets and 5 testing sets with 80-20 splits. Our recommendation is to begin with the smaller dataset (called 100k) of size 943 movies by 1682 users, and work your way up to the larger data set (called 1M), which has 6040 users and 3952 movies. Among the methods you should implement are the following:

- **Baseline methods:** A simple baseline method is a *global mean* algorithm that simply computes the mean (average) over all user ratings and uses this number to fill in all missing entries. Variants of this method include *user mean* (compute the mean of each user's ratings and use it to complete a column), and *movie mean*, where each row is completed by its mean value.

- **Mixture method:** A more sophisticated method is to complete each missing entry as a convex combination of a user rating and a movie rating, so that

$$\hat{R_{ij}} = \alpha_1 \mu_{\text{user}}(j) + \alpha_2 \mu_{\text{movie}}(i)$$

  The values $\alpha_1 + \alpha_2 = 1, \alpha_1, \alpha_2 \geq 0$ can be estimated by minimizing the RMSE function over the training data.

- **SVD method:** A commonly used (subgradient type method) method called *singular value thresholding* works as follows:

$$\begin{aligned} \hat{R}^q &= \text{shrink}(Y^{q-1}, \tau) \\ Y^q &= Y^{q-1} + \delta_q \mathcal{P}_{\Omega_T}(T - \hat{R}^q) \end{aligned}$$

  where $\tau > 0$ is some fixed constant, $Y^0 = 0$, and the shrink operator performs a soft threshold on the singular values of its operand. That is, if $X = U\Sigma V^T$ is the usual singular value decomposition of matrix X, then $\text{shrink}(X, \tau) = U\Sigma' V^T$ where $\Sigma' = \text{diag}\{(\sigma_i - \tau)_+\}$.

- **Matrix Factorization method:** Another common method is that of matrix factorization which expresses $\hat{R} = PQ^T$ where P is an n by k matrix, Q is a d by k matrix, and $k \ll \min(n, d)$. The squared error on the training set can then be expressed as:

$$\sum_{(i,j)\in\Omega_T} (P_i Q_j^T - T_{ij})^2$$

  Gradient Descent is applied to learn the matrices P and Q so that the updates given a rating in the training set are:

$$\begin{aligned} P_i^{'} &= P_i + \alpha(T_{ij} - P_i Q_j^T) Q_j \\ Q_j^{'} &= Q_j + \alpha(T_{ij} - P_i Q_j^T) P_i \end{aligned}$$

Feel free to experiment with other methods, as time permits. For evaluation of each method, you should average the RMSE score over the 5 test sets. Plot the improvement in performance of the SVD method over iterations to show its convergence. Show your results only for the 1M data set (the 100k is to help you for faster debugging).