10725/36725 Optimization
Homework 3

Due October 23, 2012 at beginning of class

Instructions: There are four questions in this assignment. Please submit your homework as 4 separate sets of pages with your name and userid on each set. For the last question which involves coding, please print out your code and graphs and attach them to the written part of your homework and email your code to submission10725f12@gmail.com. Refer to the course webpage for policies regarding collaboration, due dates, and extensions.

1 From Vector To Matrix Calculus [Aaditya]

1.1 PCA [10 points]

In this subsection we will warm up our vector calculus skills by showing two cool ways of interpreting the first principal component. One is the direction of maximum variance after projection and the second is the direction that minimizes reconstruction error. Note that the first principal component is the first eigenvector of the sample covariance matrix.

Consider n points $X_1, ..., X_n$ in p-dimensional space, and let X be the $n \times p$ matrix representing these points. Assume that the data points are centered, ie, $\vec{1}^\top X = \vec{0}$. Consider a unit vector $v \in \mathbb{R}^p$ and project all the points onto this vector (hence every point becomes a one-dimensional point on the direction of unit vector v).

1. Argue that the projection is given by Xv.
2. What is the sample mean of all the points after the projection?
3. What is the sample variance of all the points after the projection? (sample variance is the average squared distance of the points from their mean, but try to use vector notation)
4. Setup the problem of maximizing the sample variance of the projection onto v subject to a constraint on the L2-norm of v.

5. Solve using vector derivatives to show that the solution is the first PC.

So we have now proved that the direction of maximum covariance is the first PC. Now we show that the direction that minimizes reconstruction error is also the first PC.

1. Argue that the reconstruction of X_i using v is $(X_i^\top v)v$.
2. You projected X_i to $X_i^\top v$ and then reconstructed it using $(X_i^\top v)v$. What is the reconstruction error of X_i, when measured in L2-norm?
3. What is the total squared reconstruction error over all points?
4. Show that minimizing total squared reconstruction error is equivalent to minimizing $-\|Xv\|_2^2$.
5. Solve using vector derivatives to show that the solution is the first PC.

1.2 Matrix Factorization [15 points]

As its name suggests, matrix factorization involves factoring a matrix into a product of two other matrices. The SVD, LU and QR decomposition are examples of important types of factorizations. In some settings, we might postulate that a given $m \times n$ matrix Y of noisy values is actually low rank . One way to write this as an optimization problem (where $\|.\|_F$ is the frobenius norm, $\|.\|_*$ is the nuclear norm) is as follows:

$$\min_Z L(Z)$$

where

$$L(Z) = \|Y - Z\|_F^2 + \lambda\|Z\|_*$$

SVD Prove, using the subdifferential of the nuclear norm, that the solution to the above equation is achieved via SVD by subtracting λ from all singular values, and then setting negative ones to zero.

Since Z is postulated to be low rank, it can be decomposed into $Z = UV^\top$ where $U \in \mathbb{R}^{m\times k}$ and $V \in \mathbb{R}^{n\times k}$ for $k \geq rank(Z)$. Interestingly, here is a cool variational characterization of the nuclear norm (for a $k \geq rank(Z)$)

$$\|Z\|_* = \frac{1}{2} \min_{\{U\in\mathbb{R}^{m\times k}, V\in\mathbb{R}^{n\times k}: Z=UV^\top\}} \|U\|_F^2 + \|V\|_F^2$$

This allows us to re-pose the problem as:

$$\min_{\{U\in\mathbb{R}^{m\times k}, V\in\mathbb{R}^{n\times k}\}} L(U, V)$$

where

$$L(U,V) = \|Y - UV^\top\|_F^2 + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2)$$

Try to answer the following with compact notation (use vectors, matrices where possible)

1. Noting that $\|A\|_F^2 = \sum_{i,j} A_{ij}^2$, find $\partial L/\partial U_{ij}$
2. Noting that $\|A\|_F^2 = \sum_i A_i^\top A_i$, find $\partial L/\partial U_i$
3. Noting that $\|A\|_F^2 = Tr(A^\top A)$, find $\partial L/\partial U$ and $\partial L/\partial V$

Note that L is convex in $Z = UV^\top$, but non-convex in U, V. However, we could conceive of an alternate EM-like way of getting a reasonable solution to $L(U,V)$. We could alternate between fixing V and doing a gradient descent on U, and then fixing U and doing a gradient descent on V. So, keep in mind that different ways of posing an optimization problem can open different avenues to solving it.

2 Duel with Dual Duo (Shiva)

In this question, we parlay local optimization into global optimization without convexity-like assumptions. We will attempt to maximize a global function with many arguments given that it consists of easy-to-maximize local functions with few arguments.

$\{x_v\}_{v\in V}$ are variables, each taking a value in a finite (typically constant-sized) set S. For each variable v, ϕ_v is a function of just that variable. $F \subset 2^V$ is a finite set of 'factors' f (overlapping groups of variables). $\{\psi_f\}_{f\in F}$ are functions of just the variables belonging to f. For example, since this is an election year, S could be a list of political parties and V could be a set of voters who belong to various factions (or cliques of friends) F. If $\phi_v(x_v)$ denotes how inherently inclined v is to vote for x_v, and ψ_f promotes some sort of groupthink, then the following guesses who everyone will vote for:

$$\max_x A(x) = \sum_{v\in V} \phi_v(x_v) + \sum_{f\in F} \psi_f(\{x_v\}_{v\in f})$$
$$\text{s.t. } \forall v \in V,\ x_v \in S$$

This is obviously equivalent to

$$\max_{x,y} A(x,y) = \sum_{v\in V} \phi_v(x_v) + \sum_{f\in F} \psi_f(\{y_v^f\}_{v\in f}) \tag{1}$$
$$\text{s.t. } \forall f \in F, v \in f,\ y_v^f = x_v$$
$$\forall v \in V,\ x_v \in S$$
$$\forall f \in F, v \in f,\ y_v^f \in S$$

in which we introduced factor-local variables y_v^f as copies of the global variables x_v.

2.1 Round 1

ϕ_v and ψ_f are arbitrary functions on a structureless set S. Fortunately, since S is small, they can be viewed as compact input-output tables. We can afford to rewrite our optimization program by associating an indicator decision variable with each table row, indicating if we choose the input value in that row:

$$\begin{aligned} \max_{a,b}\ & A(a,b) = \sum_{v\in V}\sum_{s\in S}\phi_v(s)a_{v,s} + \sum_{f\in F}\sum_{\{s_v\}_{v\in f}\subset S}\psi_f(\{s_v\}_{v\in f})\prod_{v\in f} b^f_{v,s_v} \\ \text{s.t. } & \forall f\in F, v\in f, s\in S,\ b^f_{v,s} = a_{v,s} \\ & \forall v\in V,\ a_v\in\mathcal{I} \\ & \forall f\in F, v\in f, b^f_v\in\mathcal{I} \end{aligned} \tag{2}$$

where $\mathcal{I}$ are vectors that make sense as indicator variables, i.e. vectors with a single 1 component and zeroes elsewhere. (We'll see why this is helpful in hindsight.)

(a) [2 points] Is (2) a convex program? Why or why not?

(b) [3 points] Take $A(a,b)$ and form the Lagrangian $L(\delta,a,b)$ by introducing Lagrange multipliers $\{\delta^f_{v,s}\}_{f\in F,v\in f,s\in S}$ for the first constraint line in (2).

This forms a max-min optimization problem:

$$\begin{aligned} \max_{a,b}\ \min_{\delta}\ & L(\delta,a,b) \\ \text{s.t. } & \forall v\in V,\ a_v\in\mathcal{I} \\ & \forall f\in F, v\in f, b^f_v\in\mathcal{I} \end{aligned} \tag{3}$$

(c) [1 points] What is the relationship between (2) and (3)?

We can also form a min-max optimization problem:

$$\begin{aligned} \min_{\delta}\ L(\delta) = \max_{a,b}\ & L(\delta,a,b) \\ \text{s.t.}\quad & \forall v\in V,\ a_v\in\mathcal{I} \\ & \forall f\in F, v\in f, b^f_v\in\mathcal{I} \end{aligned} \tag{4}$$

(d) [1 points] What is the relationship between (3) and (4)?

By reintroducing our original notation and refactoring, (4) becomes:

$$\min_{\delta}\ L(\delta) = \sum_{v\in V}\max_{x_v\in S}\Phi_v(x_v) + \sum_{f\in F}\max_{\{y^f_v\}_{v\in f}\subset S}\Psi_f(\{y^f_v\}_{v\in f}) \tag{5}$$

where

$$\Phi_v(x_v) = \phi_v(x_v) + \sum_{f:v\in f} \delta^f_{v,x_v}$$
$$\Psi_f(\{y^f_v\}_{v\in f}) = \psi_f(\{y^f_v\}_{v\in f}) - \sum_{v\in f} \delta^f_{v,y^f_v}$$

This program can be solved in an alternating fashion. Suppose, for any fixed δ, every little part Φ_v and Ψ_f can be efficiently maximized subject to the constraints, yielding $\bar{x}$ and $\{\bar{y}^f\}_{f\in F}$. These local solutions may not be globally consistent with one another: we might have $\bar{y}^f_v \neq \bar{y}^{f'}_v$ for $f \neq f'$. Fixing those, we now want to minimize with respect to δ (which isn't constrained).

(e) [1 points] Is L convex in δ? Why or why not?

(f) [6 points] Give a procedure (not a big formula or matrix, please) which calculates a subgradient

$$g = \{g^f_{v,s}\}_{f\in F, v\in f, s\in S}$$

of L at δ. Describe, in English, what the subgradient qualitatively represents; which $\bar{x}$ and $\{\bar{y}^f\}_{f\in F}$ make the subgradient's norm small or large?

(g) [3 points] Suppose we had skipped the indicator variable transformation (2) and instead formed the dual from the Lagrangian of $A(x, y)$. What would the program corresponding to (5) be? Speculate on why (5) might be 'better.'

2.2 Round 2

Previously, we incorporated the $x \leftrightarrow y$ consistency constraints into the objective via the Lagrangian, formed a weak dual, then passed the integer constraints to factor-local optimization methods. Now we will relax the integer constraints and form a strong dual. The following replaces the integral decision variables of (1):

$$\max_{\substack{\{\mu_v\}_{v\in V} \\ \{\nu^f\}_{f\in F}}} \sum_{v\in V} \mathop{\mathrm{E}}_{x_v\sim\mu_v} (\phi_v(x_v)) + \sum_{f\in F} \mathop{\mathrm{E}}_{y^f\sim\nu^f} \left(\psi_f(\{y^f_v\}_{v\in f})\right) \tag{6}$$

with probability distributions $\{\mu_v\}_{v\in V}$ and $\{\nu^f\}_{f\in F}$ which are consistent with one another: the probability of s under μ_v must equal the marginal probability of s under ν^f.

(h) [3 points] Write this as a linear program. How many constraints are there?

(i) [5 points] Derive the dual of this linear program.

3 A bad day for the simplex algorithm (Kevin)

In this question, we'll explore the geometry of linear programming and a worst case for the simplex algorithm using a variant of the Klee-Minty cube. Let's define the cube in n dimensions as

$$0 \leq x_k \leq 2^k - 1 - 2\sum_{i=1}^{k-1} x_i. \qquad \forall k \in \{1, 2, \ldots, n\} \tag{7}$$

Note that for $k = 1$, the upper bound constraint degenerates to $x_1 \leq 1$.

(a) [2 pts] Sketch or plot the cube in $\mathbb{R}^2$.

Our worst-case example maximizes the sum of the x_i's over this cube.

(b) [1 pt] On your sketch, draw the objective function as a vector and circle the optimal vertex.

Let's run the simplex algorithm on the cube in $\mathbb{R}^2$ and see what it does.

(c) [2 pts] Convert the following linear program into standard form. Name your slack variables y_1 and y_2.

$$\max_x \ x_1 + x_2, \text{ subject to:} \tag{8}$$

$$0 \leq x_1 \leq 1 \tag{9}$$

$$0 \leq x_2 \leq 3 - 2x_1 \tag{10}$$

(d) [2 pts] Write out the simplex tableau for the point $x_1 = 0, x_2 = 0$ denoting z as the objective value. Which variables are in our starting basis?

(e) [1 pts] Which variables are legal variables to enter our basis? Why?

Let's have the simplex algorithm choose the entering variable with the maximum objective coefficient in current tableau breaking ties by picking the variable with the smallest index.

(f) [2 pts] Which variable enters the basis, and which leaves? Write out the new tableau. What are the values of x_1 and x_2? From this point, how many more iterations will be required?

Let's take our intuition of what's happening from the 2-dimensional case and examine the performance of the simplex algorithm in $\mathbb{R}^n$. Here, we will consider the program

$$\max_x \ \sum_{i=1}^{n} x_i, \text{ subject to:} \tag{11}$$

$$0 \leq x_k \leq 2^k - 1 - 2\sum_{i=1}^{k-1} x_i. \qquad \forall k \in \{1, 2, \ldots, n\} \tag{12}$$

(g) [2 pts] How many vertices, edges and facets does the polytope being optimized over have as a function of n?

Now consider the n dimensional program in standard form, where y_k denotes the kth slack.

(h) [5 pts] Prove that any feasible basis includes either x_k, or y_k, but not both. Remember that a basis is a corner point and cannot be written as a linear combination of two other feasible points.

(i) [4 pts] Show the program has optimum $x^* = (0, 0, \ldots, 0, 2^n - 1)$ by writing out the corresponding tableau.

Given that x_i and y_i cannot be in a basis together, we know that when x_i enters, then y_i exits and vice versa. Thus, there are n transitions from any basis, which we will label from 1 to n. That is, when applying transition k to a basis it swaps the membership of x_k and y_k.

(j) [3 pts] If $\{a_1, a_2, \ldots, a_m\}$, where for all i, $a_i \in \{1, \ldots, n\}$, is the sequence of transitions that solves the n dimensional cube, show that the sequence $\{a_1, a_2, \ldots, a_m, n+1, a_1, a_2, \ldots, a_m\}$ solves the $n+1$ dimensional cube.
Hint: Remember your tableau from part (i).

(k) [2 pts] How many iterations does the simplex algorithm take when starting from $x = (0, 0, \ldots, 0)$ on the n-dimensional Klee-Minty cube? Use induction where your answer from (f) is the base case and your answer from (j) is the inductive step.

The case presented here is a contrived worst-case. As stated in class, in practice the simplex algorithm often performs quite well. Smoothed analysis, where one bounds the expected performance of an algorithm on a slightly perturbed worst-case input, is a tool that attempts to describe this phenomenon. See D. Spielman and S. Teng, "Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time", ACM Symposium on Theory of Computing, 2001 for more details.

4 Newton's method

In this problem, we will implement the Newton's method to solve logistic regression. Given a dataset with two types of binary labels (y_i) and and p dimensional features ($\mathbf{x}_i$) ($i = 1, \cdots, n$), we will estimate $\theta \in \mathcal{R}^p$ that minimizes the negative of log likelihood (equivalently maximizing the log likelihood). Our objective function would be

$$-\log(L(\theta|X, Y)) = \frac{1}{n}\sum_{i=1}^{n}(-y_i \log(\sigma_\theta(\mathbf{x}_i)) - (1 - y_i)\log(1 - \sigma_\theta(\mathbf{x}_i))) \tag{13}$$

where $\sigma_\theta(\mathbf{x}) = \dfrac{1}{1+\exp(-\mathbf{x}^T\theta)}$, $y_i \in \{0,1\}$ refers to i th element of Y, $\mathbf{x}_i \in \mathcal{R}^p$ corresponds to the ith column of X and n is the total number of samples.

Taking the derivative of the objective function with respect to the parameter (θ) results in the gradient,

$$\frac{d(-\log(L(\theta|X,Y)))}{d\theta} = \frac{1}{n}\sum_{i=1}^{n}(\sigma_\theta(\mathbf{x}_i) - y_i)\mathbf{x}_i) \tag{14}$$

Finally, the Hessian of our objective function results in

$$\frac{d^2(-\log(L(\theta|X,Y)))}{d\theta^2} = \frac{1}{n}\sum_{i=1}^{n}\left[\sigma_\theta(\mathbf{x}_i)(1-\sigma_\theta(\mathbf{x}_i))\mathbf{x}_i\mathbf{x}_i^T\right] \tag{15}$$

(a) [2 pts] Implement code for computing the objective function of the data (Eq.13). Please name it computeL.m or computeL.r. The input to the computeL function should be θ, X and Y. The output should be the objective function value.

(b) [2 pts] Implement code for computing the gradient of the objective function with respect to θ (Eq.14). Please name it computeG.m or computeG.r. The input to the computeG function should be θ, X and Y. The output should be the gradient.

(c) [2 pts] Implement code for computing the Hessian of the objective function with respect to θ (Eq.15). Please name it computeH.m or computeH.r. The input to the computeH function should be θ and X. The output should be the value of the Hessian.

(d) [5 pts] Implement backtrack line search method (Boyd and Vandenberghe Ch. 9.2). Name the code backtrack.m or backtrack.r. The input to the function should be θ, the pointers to the functions computing the objective function (computeL) and the gradient (computeG), the direction along which you are moving (for the gradient method this will correspond to $-g$ and for the Newton's method this will correspond to $-H^{-1}g$), α, β and the input you need to compute the objective function and the gradient (X and Y). To prevent infinitely reducing the step size until the criterion is met, exit the program when step size is too small (say, when the stepsize is less than 10^{-6}). The output should include the new estimate of θ, the stepsize used, the new objective function value and the number of times computeL was called.

Please download wine-data.mat from `http://www.cs.cmu.edu/~ggordon/10725-F12/hws/hw3/`. This data includes the quality ratings of 1599 types of wine (1 corresponds to good, 0 to bad) and the volatile acidity (the first row of X) and the alcohol ratio (the second row of X) of each type.

(e) [2 pts] Run the Newton's method and the gradient descent method for 5 iterations with backtracking line search. Start from $\theta_0 = \mathbf{0}$ (e.g., a vector with all elements set to 0). Use $\alpha = 0.01$ and $\beta = 0.8$ for the backtracking line search. For both algorithms, plot the objective function values at each iteration including the initial value of the objective function.

(f) [2 pts] Plot your data (using distinct markers or color for the two classes) and the separating hyperplane estimated by the Newton's method and the gradient method. How would you conclude on the volatile acidity and alcohol level's relation to the quality of wine?

In addition to the printout, please submit your code by
1. zipping the three files for (a) - (d) into one zip file (please name it youAndrewID_hw3.zip)
2. send the zip file to submission10725f12@gmail.com before the due date. Please include in the subject line of your email, your name, AndrewID and the assignment number.

Now that we know your code works fine with low dimensional data, let's move onto more high dimensional data. Please download newton-data.mat from `http://www.cs.cmu.edu/~ggordon/10725-F12/hws/hw3/`. . The data consists of 4500 samples. The dimensionality of the features is 1000. Run the gradient descent method with backtracking line search for 30 iterations and the Newton's method with backtracking linear search for 10 iterations starting from $\mathbf{0}$.

(g) [2 pts] Plot the objective function values as a function of the number of iterations including the initial objective function values for both the gradient method and the Newton's method in one figure.

(h) [2 pts] Plot the objective function values as a function of the running time t for both the gradient method and the Newton's method in one figure. Please include the initial objective function value at $t = 0$. Compute the elapsed time per iteration. (Hint: you can use the tic and toc functions in MATLAB to record the elapsed times.)

(i) [2 pts] Plot the histogram of step sizes returned by the backtracking line search over all iterations for one run. Make separate plots for the gradient method and the Newton's method each. Does the stepsize distribution for the descent gradient method differ from that of the Newton's method? Explain why.

(j) [2 pts] Plot the histogram of number of times the function that computes the objective function was called by the backtracking line search over all iterations. Make separate plots for the gradient method and the Newton's method each. Does the histogram for the descent gradient method differ from that of the Newton's method? Explain why.

(k) [2 pts] Run your code again starting from $\theta_0 = \mathbf{1}$. Which algorithm do you think is more sensitive to the initial points? Explain why.