Homework 4, due Monday October 31 COMS 4771 Fall 2016

Problem 1 (Linear regression; 20 points)**.** In this problem, we'll consider an optimization formulation for *linear regression*. This is a supervised learning problem where the output space is $\mathcal{Y} = \mathbb{R}$ (rather than a discrete space like $\{0, 1\}$). Instead of classification error, the goal here is to learn a function $f \colon \mathbb{R}^d \to \mathbb{R}$ so as to minimize the expected *squared error* with respect to a distribution P over $\mathbb{R}^d \times \mathbb{R}$: $\mathbb{E}[(f(\boldsymbol{X}) - Y)^2]$ (the expectation is taken with respect to the random couple $(\boldsymbol{X}, Y)$ which has distribution P). We consider the case where f is a *linear function* represented by a weight vector $\boldsymbol{w} \in \mathbb{R}^d$. Let S be an i.i.d. sample from P; consider the following optimization problem:

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \quad \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \frac{1}{|S|} \sum_{(\boldsymbol{x}, y) \in S} \big(\langle \boldsymbol{w}, \boldsymbol{x} \rangle - y\big)^2 .$$

(Above, $\lambda > 0$ is assumed to be a positive scalar.)

(a) Prove that the Hessian of the objective function, at any point $\boldsymbol{w} \in \mathbb{R}$, is positive definite. Explain why this establishes that the optimization problem is convex.

(b) Derive a gradient descent algorithm for solving the optimization problem (following the recipe from lecture). Give concise and unambiguous pseudocode for the algorithm, and be explicit about how to compute gradients. You may use vector addition, scaling, and inner products as primitive operations (in addition to usual arithmetic operations). Furthermore, assume the initial solution, step sizes, and number of iterations are provided as inputs.

(c) Suppose we add the following constraints to the optimization problem:

$$w_i^2 \;\leq\; 1 \quad \text{for all } i = 1, 2, \ldots, d\,.$$

Is the optimization problem still convex? Briefly explain why or why not.

(d) Same as (c), but with the following constraints instead (assuming d is even):

$$w_{2i-1} \;=\; 1 - w_{2i} \quad \text{for all } i = 1, 2, \ldots, d/2\,.$$

(Hint: can you write an equality constraint as a pair of inequality constraints?)

(e) Same as (c), but with the following constraints instead:

$$w_i^2 \;=\; 1 \quad \text{for all } i = 1, 2, \ldots, d\,.$$

Problem 2 (Logistic regression; 30 points)**.** Let S be a training data set from $\mathbb{R}^d \times \{0,1\}$ for a binary classification problem. Consider the following optimization problem for computing the MLE of the logistic regression parameters:

$$\min_{\beta_0 \in \mathbb{R}, \boldsymbol{\beta} \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} \left\{ \ln(1 + \exp(\beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x}_i \rangle)) - y_i(\beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x}_i \rangle) \right\}.$$

(a) Derive a gradient descent algorithm for solving the optimization problem. Give concise and unambiguous pseudocode for your algorithm, and be explicit about how to compute gradients. You may use vector addition, scaling, and inner products as primitive operations (in addition to usual arithmetic operations); and natural logarithm (ln) and exponential (exp) functions as subroutines. Furthermore, assume the initial solution, step sizes, and number of iterations are provided as inputs.

(b) Implement the gradient descent algorithm from part (a), except use $\beta_0 = 0$ and $\boldsymbol{\beta} = \mathbf{0}$ as the initial solution, choose the step sizes using a line search, and use as many iterations as are required to achieve a prescribed objective value. Run your gradient descent code on the data set `hw4data.mat` from Courseworks (which has training features vectors and labels stored as `data` and `labels`, respectively). Roughly how many iterations are needed to achieve an objective value that is at most 0.65064?[1]

(c) The feature vectors in the data set from `hw4data.mat` are three-dimensional, so they are relatively easy to inspect. Investigate the data by plotting it and/or computing some statistics about the features. Do you notice anything peculiar about the features? Use what you discover to design a simple invertible linear transformation of the feature vectors $\boldsymbol{x}_i \mapsto \boldsymbol{A}\boldsymbol{x}_i$ such that running your gradient descent code with this transformed data set $\{(\boldsymbol{A}\boldsymbol{x}_i, y_i)\}_{i=1}^n$ reaches an objective value of 0.65064 in (many) fewer iterations.

Describe the steps you took in this investigation, and your reasoning behind them, as well as your chosen linear transformation (represented as a 3×3 matrix). State roughly how many iterations were required to achieve this stated objective value.

(d) Modify your gradient descent code in the following way.

- Use only the first $\lfloor 0.8n \rfloor$ examples to define the objective function; keep the remaining $n - \lfloor 0.8n \rfloor$ examples as a hold-out set.[2]
- The stopping condition is as follows. After every power-of-two iterations of gradient descent, record the hold-out error rate for the linear classifier based on the current $(\beta_0, \boldsymbol{\beta})$. If this hold-out error rate is more than 0.99 times that of the best hold-out error rate previously computed, and the number of iterations executed is at least 32 (which is somewhat of an arbitrary number), then stop.

Run this modified gradient descent code on the original `hw4data.mat` data, as well as the linearly transformed data (from part (c)). In each case, report: (1) the number of iterations executed, (2) the final objective value, and (3) the final hold-out error rate.

Please also submit your gradient descent source code (both versions) in separate files.

[1] The actual minimum value is less than 0.65064.

[2] Normally you would not simply select the first $\lfloor 0.8n \rfloor$ examples, but rather pick a random subset of $\lfloor 0.8n \rfloor$ examples. But here, the order of the examples has already been randomized.