ISYE 6740 Computational Data Analysis: Homework 3

Due: Sunday March 26, 2017, 11:59pm

100 Points Total Version 1.0

Instruction: Please write a report to answer the questions and include the plots in the report. You can write the code in R, Python or Matlab and submit your code and report via email. You can not use any existing package to directly solve the problems (e.g, ”lm” in R). In other words, you need show the iterative procedures in the code.

1) Gradient Descent for Multiple Linear Regression (35 points) Multiple linear regression attempts to model the relationship between two or more explanatory variables and a response variable by fitting a linear equation to observed data. Given a data set $\{x_{i1}, \dots, x_{ip}, y_i\}_{i=1}^n$ of n statistical units. Assume that

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^{\mathrm{T}} \\ \mathbf{x}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{x}_n^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix}, \quad \mathbf{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}.$$

Then we have regression matrix $X \in \mathbb{R}^{n\times p}$, the response vector $Y \in \mathbb{R}^{n\times 1}$ and assume that $Y = X\beta + \epsilon$, where $\beta \in \mathbb{R}^{p\times 1}$ is the unknown parameters to be estimated and $\epsilon \in \mathbb{R}^{n\times 1}$ is a random vector with standard n-dimensional normal distribution.

It is known that the unbiased estimator of β is $\hat{\beta} = (X^\top X)^{-1}X^\top Y$. However, when n and p is large, the computation of the inverse of $X^\top X$ is very time consuming. Therefore, it is very often to use gradient descent method to obtain a good estimation of β. Specifically, we aim to solve the following convex optimization problem:

$$\beta^* = \arg\max_{\beta\in\mathbb{R}^{p\times 1}} f(\beta), \quad f(\beta) \triangleq \frac{1}{2}\|Y - X\beta\|_2^2.$$

(a) One important term in gradient descent algorithms is the specific expression of gradient. Please write down the gradient $\nabla f(\beta)$. (5 points)

(b) Another important term in gradient descent is the step size in each iteration. In this problem, we choose a **fixed** step size ahead of time and use it in every iteration. It is known that if the step size is less than or equal to $1/L$, where L is the Lipschitz constant such that the following equation holds:

$$\|\nabla f(x) - \nabla f(y)\|_2 \le L\|x - y\|_2, \quad \forall x, y \in \mathbb{R}^{p\times 1},$$

then the gradient descent algorithm converges to the global optimal point (because f is convex). Please write down the step size you use in every iteration and explain why you use it. For example, you can derive L and then choose the step size to be $1/L$. (10 points)

(c) Please explain the rule of stopping your gradient descent algorithm. (5 points)

(d) The file "`MLR.csv`" is a 1000-by-31 matrix of which ith row is ith statistical unit $\{x_{i1}, \ldots, x_{ip}, y_i\}_{i=1}^n$. More specifically, the first 30 columns correspond to X and the 31st column (last column) corresponds to Y. Here we use the all-zero vector in the initialization. Please draw a plot of $f(\beta)$ versus number of iterations to demonstrate the convergence of your algorithm. (10 points)

(e) The file "`True_Beta.csv`" contains the true regression coefficients $\beta \in \mathbb{R}^{30\times 1}$. Please compare the result β^* returned by your algorithm with the true β by computing the mean squared error $\|\beta^* - \beta\|_2^2/30$. (5 points)

2) Stochastic Gradient Descent for Multiple Linear Regression (35 points) When the number n of statistical units is very large, the computation and storage complexity of computing the gradient of objective function are also very large. Therefore, in practice, most gradient descent works in a stochastic way.

Stochastic gradient descent is powerful in minimizing the objective function that has a form of sum. We again consider the multiple linear regression problem and we keep using the notation in Problem 1. Equivalently, we can see a good estimation of β as the solution to the following convex optimization problem:

$$\beta^* = \arg\max_{\beta \in \mathbb{R}^{p\times 1}} g(\beta), \quad g(\beta) \triangleq \frac{1}{n}\sum_{i=1}^n g_i(\beta), \quad g_i(\beta) \triangleq \frac{1}{2}(y_i - \mathbf{x}_i^\top \beta)^2.$$

(a) In stochastic gradient descent algorithm, we make update based on the gradient at **one** statistical unit at each iteration. Keep using the same step size, stopping rule and the all-zero vector in Problem 1 to initialize the algorithm. Please draw a plot for the value of objective function $g(\beta)$ versus the number of iterations to demonstrate the convergence. (15 points)

(b) We consider a slightly variant of stochastic gradient descent, called mini-batch stochastic gradient descent. Instead of making update based on the gradient at one statistical unit, we randomly choose b statistical units from all the n units and make update based on the mean gradient of these b units, at each iteration. For example, if we choose $\{i_1, \ldots, i_b\}$ units from all the units, then the mean gradient is $(1/b) \cdot \sum_{j=1}^b \nabla g_{i_j}(\beta)$. This method combines the advantages of gradient descent and stochastic gradient descent. Keep using the same step size, stopping rule and the all-zero vector in Problem 1 to initialize the algorithm. Please draw three plots with $b = 10, 25, 100$ for the value of objective function $g(\beta)$ versus the number of iterations to demonstrate the convergence. (15 points)

(c) The file "`True_Beta.csv`" contains the true regression coefficients $\beta \in \mathbb{R}^{30\times 1}$. Please compare the result β^* returned by your algorithm with the true β by computing the mean squared error $\|\beta^* - \beta\|_2^2/30$. You may have four values, one for stochastic gradient descent and three for the mini-batch stochastic gradient descent with batch-size b is $10, 25$ and 100. (5 points)

3) Online Principal Component Analysis (30 points) Principal component analysis (PCA) is one of the most fundamental problems in machine learning, numerical linear algebra, and data analysis. However, when run on large data sets it may be the case that we cannot afford more than single pass over the data and that is why streaming PCA comes in.

	0	1	Recall	Miss rate
0	16993	11332	60%	40%
1	13629	19944	59%	41%
Precision	55%	64%		
False discovery rate	45%	36%	Accuracy	60%

Table 1: KNN with $k = 50$, 2-class classifier

We observe $A_1, \ldots, A_n \in \mathbb{R}^{d \times d}$ sampled independently from distributions that satisfy $\mathbb{E}[A_i] = \Sigma, i = 1, \ldots, n$ for symmetric positive semidefinite (PSD) matrix $\Sigma \in \mathbb{R}^{d \times d}$. The goal is to estimate the top eigenvector (eigenvector corresponds to the largest eigenvalue) of Σ.

The Oja's algorithm is one of the most popular algorithms to achieve this goal. The algorithm is summarized as follows:

- Initialization: $w_0 = (1/\sqrt{d}, \ldots, 1/\sqrt{d})^\top \in \mathbb{R}^{d \times 1}$.
- for $i = 1, \ldots, n$
    1. $w_i \leftarrow w_{i-1} + \eta_i A_i w_{i-1}$
    2. $w_i = w_i / \|w_i\|_2$.
- end for
- output w_n.

The metric used to measure the similarity of the output of algorithm w and the true top eigenvector v is $dist(w, v) \triangleq 1 - (w^\top v)^2$, which is the square of sine of the angle between w and v.

The algorithm also relies on the choice of step size η_i at ith iteration to converge. A fixed step size usually fails to make Oja's algorithm to converge.

In this problem we set $d = 20$. "`OPCA.csv`" contains a 20000-by-20 matrix of which the 1st to 20th row is matrix $A_1 \in \mathbb{R}^{20 \times 20}$, the 21st to 40th row is matrix A_2 and so on. Therefore, we have $A_1, \ldots, A_{1000}$ and they are all 20-by-20 matrices. "`True_eigvector.csv`" contains the true top eigenvector $v \in \mathbb{R}^{20 \times 1}$ of Σ.

(a) Setting a fixed step size $\eta = 0.01$ at each iteration. Please draw a plot for $dist(w_i, v)$ versus number of iterations, where w_i is the output of Oja's algorithm at ith iteration and v is the true top eigenvector of Σ. (15 points)

(b) Setting decreasing step sizes $\eta_i = 1/(100 + i)$ at ith iteration. Please draw a plot for $dist(w_i, v)$ versus number of iterations. (15 points)

For more details about the choice of step size, the reader is referred to the paper "Streaming PCA: Matching Matrix Bernstein and Near-Optimal Finite Sample Guarantees for Oja's Algorithm" written by Prateek Jain and other authors.