CAP 5625: Programming Assignment 1

Due on Canvas by Friday, September 29, 2023 at 11:59pm

Preliminary instructions

You may consult with other students currently taking CAP 5625 in your section at FAU on this programming assignment. If you do consult with others, then you must indicate this by providing their names with your submitted assignment. However, all analyses must be performed independently, all source code must be written independently, and all students must turn in their own independent assignment. Note that for this assignment, you may *choose* to pair up with one other student in *your section* of CAP 5625 and submit a joint assignment. If you *choose* to do this, then both your names must be associated with the assignment and you will each receive the same grade.

> Though it should be unnecessary to state in a graduate class, I am reminding you that you may **not** turn in code (partial or complete) that is written or inspired by others, including code from other students, websites, past code that I release from prior assignments in this class or from past semesters in other classes I teach, or any other source that would constitute an academic integrity violation. All instances of academic integrity violations will receive a zero on the assignment and will be referred to the Department Chair and College Dean for further administrative action.

You may choose to use whatever programming language you want. However, you must provide clear instructions on how to compile and/or run your source code. I recommend using a modern language, such as Python, R, or Matlab as learning these languages can help you if you were to enter the data science, machine learning, or artificial intelligence field in the future.

All analyses performed and algorithms run must be written from scratch. That is, you may not use a library that can perform gradient descent, regression, least squares regression, optimization, etc. to successfully complete this programing assignment. The goal of this assignment is not to learn how to use particular libraries of a language, but to instead understand how key methods in statistical machine learning are implemented.

Note, credit for deliverables that request graphs, discussion of results, or specific values will not be given if the instructor must run your code to obtain these graphs, results, or specific values.

Brief overview of assignment

In this assignment you will be given advertising data from $N = 200$ training observations. The goal is to fit a model that can predict the amount of sales (in thousands of units) based on $p = 3$ features describing the amount of advertising budgets (in thousands of dollars) for TV, radio, and newspaper. Specifically, you will perform a least squares fit of a linear model using linear

regression, with the model parameters obtained by mini-batch gradient descent when applied to the training dataset.

Data

Data for these observations are given in `Advertising_N200_p3.csv`, with observations provided on each row (rows 2 through 201), and input features and response given on the columns (with the first row representing a header for each column). There are three quantitative features, given by columns labeled "TV", "radio", and "newspaper".

Detailed description of the task

Recall that the task of performing a least squares regression fit to training data $\{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ with mini-batch gradient descent is to minimize the cost function

$$J_{\mathcal{S}}(\beta) = \sum_{(x_i, y_i) \in \mathcal{S}} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2$$

where y_i is a response for observation i, x_{ij} is the value of feature j for observation i, and $\mathcal{S}$ is the set of size $n < N$ containing randomly chosen observations within a given mini batch, with the minimization across all mini batches each evaluated in turn. Moreover, recall that mini-batch gradient descent first computes the $(p+1)$-dimensional gradient vector $\frac{\partial J_{\mathcal{S}}(\beta)}{\partial \beta}$, and then simultaneously updates each parameter k, $k = 0,1, \dots, p$, as follows:

$$\beta_k \coloneqq \beta_k - \alpha \frac{\partial}{\partial \beta_k} J_{\mathcal{S}}(\beta)$$

where α is the learning rate and where the partial derivative of the cost function for a given mini-batch with respect to the kth parameter is

$$\frac{\partial}{\partial \beta_k} J_{\mathcal{S}}(\beta) = -2 \sum_{(x_i, y_i) \in \mathcal{S}} x_{ik} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)$$

To implement this algorithm, depending on whether your chosen language can quickly compute vectorized operations, you may implement mini-batch gradient descent using either Algorithm 1 or Algorithm 2 below (choose whichever you are more comfortable implementing). Note that in languages like R, Python, or Matlab, Algorithm 2 (which would be implemented by several nested loops) may be much slower than Algorithm 1. Note, if you are implementing Algorithm 1 using Python, then use numpy arrays instead of Pandas data frames for computational speed.

We will employ a batch size of $n = 10$, and you may need to explore different learning rate values to identify one that is not too large and not too small, such that it is likely for the algorithm to converge in a reasonable period of time. I would consider a learning rate of $\alpha = 2.5 \times 10^{-6}$, though I encourage you to explore how your model trains for smaller and larger

learning rates as well. For this assignment, assume that we will get close to the optimal parameter estimates within a fixed number of steps, with the number of iterations being $20{,}000$.

Algorithm 1 (vectorized):

Step 1. Choose learning rate α and batch size n

Step 2. Generate N-dimensional response vector $\mathbf{y}$ and $N \times (p+1)$ design matrix $\mathbf{X}$, where the first column of $\mathbf{X}$ has all elements equal to one

Step 3. Randomly initialize the parameter vector $\beta = [\beta_0, \beta_1, \dots, \beta_p]$

Step 4. Randomly assign the N observations to a given batch, where the number of batches is $B = N/n$

Step 5. For each batch b, $b = 1,2, \dots, B$, update the parameter vector with the n observations in this batch as

$$\beta \coloneqq \beta + 2\alpha \mathbf{X}_b^T (\mathbf{y}_b - \mathbf{X}_b \beta)$$

where $\mathbf{X}_b$ is the $n \times (p+1)$ design matrix of the n observations in batch b and $\mathbf{y}_b$ is the n-dimensional response vector of the n observations in batch b

Step 6. Repeat Steps 4 and 5 for 20,000 iterations

Step 7. Set the last updated parameter vector as $\hat{\beta} = [\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p]$

Algorithm 2 (non-vectorized):

Step 1. Choose learning rate α and batch size n

Step 2. Generate N-dimensional response vector $\mathbf{y}$ and $N \times (p+1)$ design matrix $\mathbf{X}$, where the first column of $\mathbf{X}$ has all elements equal to one

Step 3. Randomly initialize the parameter vector $\beta = [\beta_0, \beta_1, \dots, \beta_p]$

Step 4. Create temporary parameter vector $\beta_{\text{temp}} = [\beta_0^{\text{temp}}, \beta_1^{\text{temp}}, \dots, \beta_p^{\text{temp}}]$

Step 5. Randomly assign the N observations to a given batch, where the number of batches is $B = N/n$ to create B batches (sets) of size n observations denoted $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_B$

Step 6. For each batch b, $b = 1,2, \dots, B$, update parameters with the n observations in this batch as

For each k, $k = 0,1, \dots, p$, find next value for parameter k as

$$\beta_k^{\text{temp}} \coloneqq \beta_k + 2\alpha \sum_{(x_i, y_i) \in \mathcal{S}_b} x_{ik} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)$$

Update the parameter vector as $\beta = \beta_{\text{temp}}$

Step 7. Repeat Steps 5 and 6 for 20,000 iterations

Step 8. Set the last updated parameter vector as $\hat{\beta}$

When randomly initializing the parameter vector, I would make sure that the parameters start at small values. A good strategy here may be to randomly initialize each of the β_j, $j = 0,1, \dots, p$, parameters from a uniform distribution between -1 and 1.

Deliverable 1: Illustrate the effect of iteration number of mini-batch gradient descent on the inferred regression coefficients by generating a plot (*e.g.*, using Excel, Matlab, R, etc.) of four lines (one for each of the $p = 3$ features and the intercept), with the y-axis as $\hat{\beta}_j$, $j = 0,1, \dots, p$, and the x-axis the corresponding iteration of mini-batch gradient descent that generated the particular $\hat{\beta}_j$. Label both axes in the plot.

Deliverable 2: Illustrate the effect of iteration number of mini-batch gradient descent on the cost by generating a plot (*e.g.*, using Excel, Matlab, R, Python, etc.) with the y-axis as the cost

$$J(\hat{\beta}) = (\mathbf{y} - \mathbf{X}\hat{\beta})^T(\mathbf{y} - \mathbf{X}\hat{\beta}) = \sum_{i=1}^{N}\left(y_i - \hat{\beta}_0 - \sum_{j=1}^{p} x_{ij}\hat{\beta}_j\right)^2$$

and the x-axis the corresponding iteration of mini-batch gradient descent that generated the particular cost. Label both axes in the plot.

Deliverable 3: Provide the estimates $\hat{\beta} = [\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p]$ of the best-fit model parameters.

Deliverable 4: Using the best-fit model parameters, compute the mean squared error (MSE) on the training set and report this value.

Deliverable 5: Provide all your source code that you wrote from scratch to perform all analyses (aside from plotting scripts, which you do not need to turn in) in this assignment, along with instructions on how to compile and run your code.

Deliverable 6 (extra credit): Upload your certificate for HackerRank Python(Basic) to the Canvas assignment titled "Submission of HackerRank Python(Basic) certification" before the submission deadline for Programming Assignment 1. This is worth up to 5% additional credit, which would allow you to get up to 105% out of 100 for this assignment.