a3

Homework Assignment # 3

Due: Friday, April 16, 2021, 11:59 p.m.
Total marks: 100

Question 1. [50 MARKS]

In this question, you will implement multivariate linear regression, and polynomial regression, and learn their parameters using Stochastic Gradient Descent (SGD). An script in python has been given to you, called `script_regress_q1.py`, to run the regression algorithms on a dataset. You will be running the algorithms on the GraduateAdmissions_v1.0 data set, which has $n = 500$ samples and $d = 7$ features. The features are composed of GPA, TOEFL grades and a few other criteria. You are asked to train some models to predict the admission probability based on these features. The features are augmented to have a column of ones (to create the bias term), in `dataloader.py` (not in the data file itself). Baseline algorithms, including mean and random predictions, are used to serve as sanity checks. We should be able to outperform random predictions, and the mean value of the target in the training set.

(a) [20 MARKS] Implement a mini-batch stochastic gradient descent approach to obtain the linear regression solution (see Algorithm 3 in Chapter 7 of the notes). The update has the form $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{g}_t$, where $\mathbf{g}_t$ is the mini-batch gradient on iteration t. We will use a slightly better heuristic stepsize choice than the last assignment, where we the inverse of an accumulating sum of gradient norms: $\eta_t = (1 + \bar{g}_t)^{-1}$ where

$$\bar{g}_t = \bar{g}_{t-1} + \frac{1}{d+1} \sum_{j=0}^{d} |g_{t,j}|$$

with $g_{t,j}$ the j-th entry in the vector (the array) $\mathbf{g}_t$ and $\bar{g}_0 = 1$ where t starts at 1. Initialize the weights to zero and set the default number of epochs to 100 (as in the barebones). Add this code to `algorithms.py`. Report the error obtained by your linear regression model, as outputted by `script_regress_q1.py`.

(b) [20 MARKS] Implement polynomial regression with a $p = 2$ degree polynomial, using the same approach as in Part a. This means using a mini-batch stochastic gradient descent approach with the given stepsize approach. Add this code to `algorithms.py`. As a hint, consider calling the `LinearRegression` algorithm you wrote in the first question, within `PolynomialRegression`, to avoid code duplication and to re-use an already debugged algorithm. Report the error obtained by your polynomial regression model, as outputted by `script_regress_q1.py`.

(c) [10 MARKS] Rather than the scalar step size, we can have a vector of step sizes: a different stepsize for each dimension in the weight vector. Moving to a vector of stepsizes makes it even more important to have adaptive stepsize approaches, as it is impractical to choose different step sizes for each dimension manually. The AdaGrad algorithm is similar to the stepsize approach above, but uses a different stepsize for each dimension. We now maintain a vector $\bar{\mathbf{g}}_t \in \mathbb{R}^{d+1}$ of accumulating gradient values for each dimension

$$\bar{\mathbf{g}}_t = \bar{\mathbf{g}}_{t-1} + \mathbf{g}_t^2$$

where $\bar{g}_{t=0} = \vec{0}$ where t starts at 1, and the squaring is done element-wise in $\mathbf{g}$, i.e., for every $j = 0, 1, \ldots, d$

$$\bar{g}_{t,j} = \bar{g}_{t-1,j} + g_{t,j}^2$$

Q1:

(a)

Error for Linear Regression Heuristic: 0.9763321274966993

(b)

Error for Polynomial Regression Heuristic: 0.7041705634726718

(c)

Error for Linear Regression AdaGrad: 0.9482430286425186

Error for Polynomial Regression AdaGrad: 0.6571093869241363

The vector stepsize $\boldsymbol{\eta}_t$ is composed of entries $\eta_{t,j} = 1/\sqrt{g_{t,j}}$. The update multiplies each gradient dimension with its own stepsize, which is an element-wise product between the vector $\boldsymbol{\eta}_t$ and $\mathbf{g}_t$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \boldsymbol{\eta}_t \cdot \mathbf{g}_t$$

where $\cdot$ means element-wise product. Add AdaGrad as a stepsize option within `LinearRegression`. Report the error obtained by your linear regression model trained with AdaGrad, as outputted by `script_regress_q1.py`.

Question 2. [30 MARKS]

In this question, you will use the paired t-test to compare the performance of two models. You will compare the above LinearRegression model and PolynomialRegression model, both using AdaGrad. You will run this comparison using `script_regress_q2.py`. You hypothesize that PolynomialRegression is better than LinearRegression, and so want to run a one-tailed test to see if that is true.

(a) [5 MARKS] Define the null hypothesis and the alternative hypothesis. Use μ_1 to be the true expected squared error for LinearRegression and μ_2 the true expected squared error for PolynomialRegression.

(b) [10 MARKS] Before running the paired t-test, you should check if the assumptions are not violated. One way to satisfy the assumption for the paired t-test is to check if the errors are (approximately) normally distributed with (approximately) equal variances. To do this, you need to implement the `checkforPrerequisites` method in `script_regress_q2.py`. For each model, you can plot a histogram of its errors on the test set. You can do so using the two vectors of errors and the function `plotTwoHistograms` function to visualize the error distributions simultaneously. Discuss why it is ok or not ok to use the paired t-test to get statistically sound conclusions about these two models.

(c) [15 MARKS] Regardless of the outcome of Part b, let's run the paired t-test. (Note, I am not advocating that you check for violated assumptions and then ignore the outcome of that step. The goal of this question is simply to give you experience actually running a statistical significance test. Presumably, in practice, you would pick an appropriate one after verifying assumptions). To run this test, you need to compute the p-value. To do this implement the `getPValue` method, which returns the p-value for the one-tailed paired t-test. Report the p-value. Would you be able to reject the null hypothesis with a significance threshold of 0.05? How about of 0.01?

Question 3. [20 MARKS]

In this question, you will implement multivariate logistic regression, and learn its solution using Stochastic Gradient Descent (SGD). We will be examining some of the practical aspects of implementing binary classification, including for a large number of features and samples. An initial script in python has been given to you, called `script_classify_q3.py`, and associated python files. The implementation of logistic regression class shall be provided in `algorithms.py`. You will be running on a physics data set, with 8 features and 100,000 samples (called susysubset). The features are augmented to have a column of ones (to create the bias term), in dataloader.py (not in the data file itself). We should be able to outperform random predictions, provided by a random classifier.

(a) [15 MARKS] Implement a mini-batch stochastic gradient descent approach to logistic regression,

Q2:

(a)

Null hypothesis: $\mu_1 - \mu_2 = 0$
Alternative hypothesis: $\mu_1 - \mu_2 > 0$

(b)

Form two diagrams, we know they are normally distributed,

and their mean and standard deviation are similar. As a result, we can use paired t-test to concluded those two mode.

(c)

P-value is 0.0 in the report. Therefore, I am be able to reject the null hypothesis under the significance threshold of 0.05 and 0.01.

using AdaGrad. Report the error, using the number of epochs given in the script.

(b) [5 MARKS] Implement a mini-batch stochastic gradient descent approach to logistic polynomial regression. As a hint, consider calling the `LogisticRegression` algorithm you wrote in (a), within `PolynomialLogisticRegression`, to avoid code duplication and to re-use an already debugged algorithm. Report the error, using the same parameters and step size approach as (a).

Q3
(a)
Error for Logistic Regression: 45.720000000000006
(b)
Error for Logistic Regression: 48.519999999999996

Homework policies:

Your assignment should be submitted as a single pdf document and a zip file with code, on eClass. The questions must be typed (e.g., Latex) or must be written legibly and scanned. All code (if applicable) should be turned in when you submit your assignment.

Because assignments are more for learning, and less for evaluation, grading will be based on coarse bins. **The grading is atypical**. For grades between (1) 80-100, we round-up to 100; (2) 60-80, we round-up to 80; (3) 40-60, we round-up to 60; and (4) **0-40, we round down to 0**. The last bin is to discourage quickly throwing together some answers to get some marks. The goal for the assignments is to help you learn the material, and completing less than 50% of the assignment is ineffective for learning.

Policy for late submission of assignments: Grades for late assignments will be penalized by 1 full bin per day late (see above) and any assignment submitted more than 2 days late will receive a grade of 0.

All assignments are individual, except when collaboration is explicitly allowed. All the sources used for the problem solution must be acknowledged, e.g. web sites, books, research papers, personal communication with people, etc. Academic honesty is taken seriously; for detailed information see the University of Alberta Code of Student Behaviour.

Good luck!