CS589: Machine Learning - Spring 2017

Homework 3

Getting Started: Download the assignment archive from Moodle and unzip the file. This will create the directory structure shown below. The data files for the data set are under the 'Data' directory. You will write your code under the Submission/Code directory. Make sure to put the deliverables (explained below) into their respective directories.

```
HW03
--- Data
    |--Movielens
--- Submission
    |--Code
    |--Figures
```

Data Sets: The data set we will use in this assignment is derived from the 20 million rating Movielens data set. The data set consists of ratings for movies entered by users of the Movielens web service. The full data set you will use contains 5,000,000 ratings for 20,402 different movies entered by 34,395 different users. The data set consists of one row per rating. The rating value is contained in the last column of each row and takes values between 1 and 5. It is preceded by a length-20 real-valued user descriptor vector and a length-20 binary movie descriptor vector. Your task is to learn a regression model to predict ratings based on using the movie and user descriptor vectors as features. The full data set is hosted on Amazon Web Services and will be used for the last question in this assignment only. We will use a smaller data set for development purposes, consisting of 14,166 ratings for 5,217 different movies entered by 101 different users. Both data sets have been split into training and test tests. All data sets are stored in CSV format.

Provided Code: For this assignment, several starter code files will be provided. For questions 1 and 2, we will use a plain Python Spark simulator library that implements a core subset of the full pySpark API. This library is contained in mySpark.py. See mySpark.py for the set of Spark transformations and actions that are implemented. You will add your code for question 1 to warmup.py, which contains stubs for all functions you need to implement for question 1. You will add your code for question 2 to ols.py, which contains stubs for all functions you need to implement for question 2. *local_run_me.py* is a driver script that will run all of your code for questions 1 and 2. Scripts for question 3 will run your code on Amazon Web Services to assess parallel scalability. These scripts will be provided after use of AWS is discussed in class.

Deliverables: This assignment has two types of deliverables: a report and code files.

- **Report:** The solution report will give your answers to the homework questions (listed below). The maximum length of the report is 5 pages in 11 point font, including all figures and tables. You can use any software to create your report, but your report must be submitted in PDF format.
- **Code:** The second deliverable is the code that you wrote to answer the questions. Your code must be written in Python 2.7 and PySpark (mySpark) 2.1. You may use NumPy from within Spark, but you may not use SciKitLearn for this assignment.

Submitting Solutions: When you complete the assignment, you will upload your report and your code using Gradescope.com. Place your final code in Submission/Code. Create a zip file of your submission directory, Submission.zip. Upload this single zip file to Gradescope as your solution to the HW03-Programming assignment. Gradescope will run code correctness checks for Question 1 and Question 2. Finally, upload your report as your solution to the HW03-Report assignment. The submission time for your assignment is considered to be the later of the submission timestamps for your code and report.

Academic Honesty Statement: Copying solutions from external sources (books, web pages, etc.) or other students is considered cheating. Sharing your solutions with other students is considered cheating. Posting your code to public repositories like GitHub is also considered cheating. Collaboration indistinguishable from cheating will be treated as cheating. Any detected cheating will result in a grade of 0 on the assignment for all students involved, and potentially a grade of F in the course.

Introduction: In this assignment, you will work with Python, Amazon AWS and Apache Spark to learn a linear regression model for predicting movie ratings based on user and/or movie features. You will work with a small local copy of the data set to develop and test the model using a Python-only implementation of the full pySpark API called mySpark. You will then move your code to PySpark/AWS to assess its parallel scalability when run in a Spark cluster. Starter code is provided in the Code directory for local experimentation. Code and instructions for running experiments on AWS with Spark will be released following Spring break.

1. (*30 points*) **mySpark Warm-Up Exercises** Complete the code stubs provided in *warmup.py* to answer the following warm-up questions. Your answers to these questions must manipulate RDDs based on the functions available in the mySpark RDD implementation, which are a subset of functions provided in the full pySpark API. Your code must not use for loops or other types of iterators to iterate over the data directly. As your answer to each question, include your code listing for the corresponding code stub in your report. You can use functions you specify for the earlier questions to help you complete the later questions. Your answer will be autograded for correctness, so do not change the function signatures and make sure to use the specified return types. Assume that the rows of the input RDDs contain Numpy arrays of shape (D,) for some D. You can run $local\_run\_me.py$ to see the output of your code. Also see $local\_run\_me.py$ for how the RDDs used for this question are generated. This can help you to debug your solutions.

1.1. (*5 pts*) Write the function `count(rdd)` that determines the number of rows in an RDD. The result should be returned as a float.

1.2. (*5 pts*) Write the function `mean(rdd)` that computes the sample mean of each column contained in an RDD. Return your answer as a Numpy array of shape (D,). See `https://en.wikipedia.org/wiki/Sample_mean_and_covariance#Sample_mean` for formulas.

1.3. (*5 pts*) Write the function `std(rdd)` that computes the sample standard deviation of each column contained in an RDD. Return your answer as a Numpy array of shape (D,). See `https://en.wikipedia.org/wiki/Unbiased_estimation_of_standard_deviation` for formulas (use devision by N, not N-1).

1.4. (*5 pts*) Under the assumption that $D = 2$ (the RDD has two columns), write the function `dot(rdd)` that computes the inner product (dot product) between the two columns. See `https://en.wikipedia.`

org/wiki/Dot_product#Algebraic_definition for formulas.

1.5. (*5 pts*) Under the assumption that $D = 2$ (the RDD has two columns), write the function `corr(rdd)` that computes the sample Pearson's correlation coefficient between the two columns. See `https://en.wikipedia.org/wiki/Pearson_correlation_coefficient#For_a_sample` for formulas.

1.6. (*5 pts*) Under the assumption that $D = 1$ (the RDD has single column), and that the column contains only whole numbers between 0 and K for some K, write the function `distribution(rdd)` that computes the empirical probability distribution of values in the RDD. If N is the number of rows in the RDD and $y_1, ..., y_N$ are the values, then the empirical probability of the value k is $p[k] = \frac{1}{N}\sum_{n=1}^{N}[y_n = k]$ where $[y_n = k]$ is 1 if $y_n == k$ and is 0 otherwise. Return p as a Numpy array of size $(K + 1,)$.

2. (*50 points*) **mySpark Linear Regression** In this question, you will implement ordinary least squares linear regression using the mySpark interface to interact with RDDs. Your code must not use for loops or other types of iterators to iterate over the data directly. Once you have reduced the data to a size that is independent of the number of rows in the original data set, you may apply Numpy operations to the result. You can also use Numpy functions inside Map, Reduce and other mySpark RDD functions. Add the code for each question to the stubbs provided in *ols.py*. You can use functions you specify for the earlier questions to help you complete the later questions. As your answer to each question, include the code for the corresponding stub in your report. Your answers will be autograded for correctness, so do not change the function signatures and make sure to use the specified return types. You can run $local\_run\_me.py$ to see the output of your code.

The formula for the optimal OLS weight vector that you will implement is given below where i indexes the data cases, N is the number of data cases, $\mathbf{x}_i$ is a row vector containing the features for data case i and y_i is a scalar containing the rating for data case i.

$$\mathbf{w} = \big(\sum_{i=1}^{N}\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_i\big)^{-1}\big(\sum_{i=1}^{N}\mathbf{x}_i^{\mathrm{T}}y_i\big)$$

Note that the equations above do not include an explicit bias term b. Instead, each feature vector $\mathbf{x}_i$ must be augmented with a 1 (ie: [2, 17,...,0] -> [2, 17,...,0,1]) so that the bias can be learned implicitly along with the other weights. This will result in weight and feature vectors of length 41 (40 features plus the bias term).

2.1. (*10 pts*) Write code for the computation $\sum_{i=1}^{N}\mathbf{x}_i y_i$ using Map and Reduce. Your code should return a Numpy vector with shape (41,) as a result.

2.2. (*10 pts*) Write code that computes $\sum_{i=1}^{N}\mathbf{x}_i^T\mathbf{x}_i$ using Map and Reduce. Note that this operation is an outer product and your code should return a Numpy array with shape (41,41) as a result.

2.3. (*10 pts*) Write code to obtain the linear regression weight vector $\mathbf{w}$ according to the formula shown above. You should make calls to the previous two functions and combine the results using Numpy. Your code should return a Numpy vector with shape (41,) as a result.

2.4. (*10 pts*) Given a weight vector $\mathbf{w}$ you can make predictions using the equation $\hat{y}_i = \mathbf{w} \cdot \mathbf{x}_i$. Write code to implement this computation assuming you are given the weight vector $\mathbf{w}$ as a Numpy vector with shape

(41,). Your code should return an RDD with one row per data case containing the predicted value $\hat{y}_i$ as a float.

2.5. (*10 pts*) Write Spark code to compute the MAE of the predictions obtained using a weight vector $\mathbf{w}$ represented as a Numpy vector with shape (41,). Recall that the MAE is defined as:

$$\frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Your code should return the MAE as a float. Note that since there is no easy way to merge two RDDs, the easiest way to solve this problem is to modify the solution to the previous question so that the result is a pair $(y_i, \hat{y}_i)$ that you can operate on further to compute the MAE.

3. (*20 points*) **pySpark Linear Regression on AWS** In this question, you will test your code at scale using Amazon AWS. Separate instructions will be provided for setting up an Amazon AWS account, redeeming Amazon credits, starting a Spark cluster on AWS, and uploading and running your code.

3.1. (*10 pts*) Run your code on AWS and report the MAE your code obtains on the full test set as your answer to this question.

3.2. (*10 pts*) The procedure described above will run your code using between 1 and 8 Spark workers and will output the total time needed to train and test the model using your code for each number of workers. When the run completes, the code will output the amount of time used for each number of workers as a list of lists in the format [number of workers, run time in seconds]. Produce a line plot showing the run time of your code versus the number of workers and include it in your report as your answer to this question.

3.3. (*10 pts*) **Bonus:** Investigate different implementations of the linear regression learning code to see if you can find a solution with better parallel scalability than your initial solution. You may use the full pySpark API. If you find a significantly better implementation, describe what you changed and provide a line plot showing the run time of your alternate implementation versus the number of workers. Submit both your initial and updated code files.

3.4. (*10 pts*) **Bonus:** Investigate modifications to the basic OLS algorithm to see if you can find a solution with better accuracy on the full test data set. Things to try are feature normalization, feature selection, output normalization, regularization, etc. If you find a significant enhancement to the model, provide the resulting MAE, describe what you changed, and assess the impact on parallel scalability. Submit both your initial and updated code files.