Assignment #2: Logistic Regression
CSCI 374 Fall 2022 Oberlin College
Due: Monday October 3 at 11:59 PM

Background

Our second assignment this semester has four main goals:

1. Implement logistic regression as a valuable method to supervised machine learning (and an eventual building block for neural networks),
2. Practice with data pre-processing to prepare datasets for supervised learning,
3. Investigate the learning process during training.
4. Practice working with a partner on code development and scientific experimentation.

Gitting Started

To begin this assignment, please follow this link:

https://classroom.github.com/a/PqMZTAjr

Data Sets

For this assignment, we will learn from four pre-defined data sets from publicly available sources, each representing a binary classification task:

1. **monks1.csv**: A data set describing two classes of robots using all nominal attributes and a binary label. This data set has a simple rule set for determining the label: `if head_shape = body_shape ∨ jacket_color = red, then yes (1), else no (0)`. Each of the attributes in the monks1 data set are nominal. Monks1 was one of the first machine learning challenge problems (http://www.mli.gmu.edu/papers/91-95/91-28.pdf). This data set comes from the UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/datasets/MONK%27s+Problems
2. **banknotes.csv**: A data set describing observed measurements about banknotes (i.e., cash) under an industrial print inspection camera. The task in this data set is to predict whether a given bank note is authentic or a forgery. The four attributes are each continuous measurements. The label is 0 if the note is authentic, and 1 if it is a forgery. This data set comes the UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/banknote+authentication
3. **occupancy.csv**: A data set of measurements describing a room in a building for a Smart Home application. The task in this data set is to predict whether or not the room is occupied by people. Each of the five attributes are continuous measurements. The label is 0 if the room is unoccupied, and a 1 if it is occupied by a person. This data set comes the UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+

4. **seismic.csv**: A data set of measurements describing seismic activity in the earth, measured from a wall in a Polish coal mine. The task in this data set is to predict whether there will be a high energy seismic event within the next 8 hours. The 18 attributes have a mix of types of values: 4 are nominal attributes, and the other 14 are continuous. The label is a 0 if there was no high energy seismic event in the next 8 hours, and a 1 if there was such an event. This data set comes the UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/seismic-bumps

The file format for each of these data sets is the same as in Homework #1:

- The first row contains a comma-separated list of the names of the label and attributes
- Each successive row represents a single instance
- The first entry (before the first comma) of each instance is the label to be learned, and all other entries (following the commas) are attribute values.
- Some attributes are strings (representing nominal values), others are real numbers. Each label is a 0 or 1.

Program

Your assignment is to write a program called **logistic** that behaves as follows:

1) It should take as input six parameters:
    a. The path to a file containing a data set (e.g., monks1.csv)
    b. The learning rate η to use during stochastic gradient descent
    c. The percentage of instances to use for a training set
    d. The percentage of instances to use for a validation set
    e. A random seed as an integer

    For example, if I wrote my program in Python 3, I might run

    ```
    python3 logistic.py monks1.csv 0.01 0.6 0.2 12345
    ```

    which will create a logistic regression model, trained using a learning rate of $\eta = 0.01$ on monks1.csv with a random seed of 12345, where 60% of the data will be used for training, 20% is used for a validation set, and the remaining 20% (100 – 60 – 20) will be used for testing.

2) Next, the program should read in the data set as a set of instances, which should be split into training, validation, and test sets (using the random seed input to the program). Unlike Homework 1, we will need to do some pre-processing of the data to prepare it for learning.

    a. For the continuous attributes in the banknotes.csv, occupancy.csv, and seismic.csv data sets, we will need to scale them to appropriate ranges for learning. This is because some of the values have very large magnitudes in occupancy.csv and seismic.csv — if we did not do this, some of our net values that go into the sigmoid function inside the logistic regression machine would either be too large or too negative, crashing the program.

To scale our continuous attributes, we will want to find the maximum and minimum value of that attribute across all instances in the data set. Then, we will replace each value with the following:

$$newValue = \frac{oldValue - \min}{\max - min}$$

b. For the nominal attributes in monks1.csv and seismic.csv, you will need to convert each of the attributes into *m-1* indicator variables using **one hot coding** (where the attribute originally took *m* values). For example, the body_shape attribute in monks1.csv takes *m* = 3 values (round, square, octagon), so we can create *m*-1 = 2 indicator variables:

$body\_shape_{Round} = 1$ if *body_shape* = round, else 0
$body\_shape_{Square} = 1$ if *body_shape* = square, else 0.

3) You should create a logistic regression model by learning a set of weights for the model using stochastic gradient descent on the training set. You should stop training when either: (1) your accuracy on the validation set reaches 99%, or (2) you have trained for 500 epochs (i.e., you looped through the entire training set 500 times).
4) Next, predictions should be made for each instance in the test set using the logistic regression learned in Step 3. Calculate the resulting confusion matrix using those predictions and the true labels of the instances in the test set. The confusion matrix should be output as a file with its name following the pattern:
results_<DataSet>_<LearningRate>r_<Seed>.csv
(e.g., results_monks1_0.01r_12345.csv).

Please note that you **are** allowed to reuse your code from Homework 1 for generating random test/training sets, as well as for creating output files.

Program Output

The file format for your output file should be the same as in Homework 1. Please refer back to that assignment for more details.

As in Homework 1, the output for your program should be consistent with the random seed. That is, if the same seed is input twice, your program should output the exact same confusion matrix.

Programming Languages

I would recommend using either the **Java** or **Python** programming languages to complete this assignment. If you have a different preferred language, please talk to me to make sure that I will be able to run your submission in that language.

Different from Homework 1, **you are allowed to use external libraries** (e.g., Pandas and numpy in Python) for preprocessing the data set (i.e., reading it in, converting the nominal attributes to one-hot variables, and normalizing the continuous attributes). To receive ***full credit*** on the assignment, you should *not* use any pre-created implementations of logistic regression (e.g., using scikit-learn in Python), but instead implement your own from scratch.

However, if you have ***difficulties getting logistic regression to work*****,** you *can* use a pre-created implementation to *answer the research questions*. If you do so, please make sure to cite your source in the README file.

Research Questions

Please use your program to answer these questions and record your answers in a README file:

1) Pick a single random seed (your choice, document in your README file), 0.01 as the learning rate, 60% as the training percentage, and 20% as the validation percentage.
    a. Record the accuracies of your program on the test set for each of the four data sets.
    b. Create confidence intervals for each of your test set accuracies.
2) Pick 30 different random seeds (document them in your README file). Rerun your program on each data set using the same parameters as Question 1 (0.01 learning rate, 60% training percentage, 20% validation percentage).
    a. What was the average accuracy you observed across all 30 seeds for each data set?
    b. Did this average fall inside the confidence interval you calculated for Question 1?
    c. How many of the 30 seeds produced a test set accuracy that fell within your confidence interval calculated in Question 1? Does this match your expectation, given that you calculated 95% confidence intervals in Question 1?
3) Pick 10 different random seeds (document them in your README file). For the occupancy.csv data set using a learning rate of 0.01, track the accuracy of your model on the **validation set** after each epoch of stochastic gradient descent (i.e., after you feed the entire training set in).
    a. Plot a line chart of the validation set accuracies for each epoch (the epochs go on the x-axis, and the accuracy goes on the y-axis). You can use any tool of your choice to create the line charts: Excel, R, matplotlib in Python, etc. Include your line chart as an image in your GitHub repository.
    b. What trends do you see across the 10 random seeds? How does the accuracy of validation set change over time as the epoch increases? Were there differences between the 10 seeds, or did they all produce similar results?
    c. What do these results imply?

Bonus Question (up to 10 points)

4) Design and run your own experiments with logistic regression. Some ideas (not mandatory) could include:
    a. comparing test set performances with different learning rates to see how that impacts the learning process,
    b. comparing test set performances with different training and validation set sizes (i.e., different percentages as command line arguments),
    c. trying out different binary classification data sets that you download, or
    d. rerunning your knn solution from Homework 1 on each of the four data sets from this assignment and comparing the results.

This is completely up to you! The more elaborate the experiment, the more bonus points you can earn.

README

Within a README file, you should include:

1) Your answers to the questions above,
2) A short paragraph describing your experience during the assignment (what did you enjoy, what was difficult, etc.)
3) An estimation of how much time you spent on the assignment, and
4) An affirmation that you adhered to the honor code

Please remember to commit your solution code, results files, and README file to your repository on GitHub. You do not need to wait to commit your code until you are done with the assignment; it is good practice to do so not only after each coding session, but maybe after hitting important milestones or solving bugs during a coding session. ***Make sure to document your code***, explaining how you implemented the different components of the assignment.

Honor Code

Different from the first homework assignment, **each student is allowed to work with *one partner* to complete this assignment**. Groups are also allowed to collaborate with one another to discuss the abstract design and processes of their implementations. However, sharing code (either electronically or looking at each other's code) between groups is not permitted

Grading Rubric

Your solution and README will be graded based on the following rubric:

Followed input directions: /5 points
Properly read in and pre-processed the data: /10 points
Properly created training, validation, and test sets: /5 points
Correctly implemented Stochastic Gradient Descent: /20 points

Correctly implemented classification using logistic regression: /10 points
Followed output directions: /5 points
Correctly answered the research questions: /35 points
Provided requested README information: /5 points
Appropriate code documentation: /5 points

By appropriate code documentation, I mean including a header comment at the top of each file explaining what the file provides, as well as at least one comment per function explaining the purpose of the function.