CSC 442 Project 4 — Machine Learning

Introduction

For this project, you are asked to explore applications of the gradient descent method to training logistic regression models and multilayer perceptrons (neural networks).

We will the Sonar dataset as discussed in class. More information about the dataset can be found here: https://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+%28Sonar%2C+Mines+vs.+Rocks%29

The dataset was produced by Gorman and Sejnowski (1988) and pertains to the classification of objects as either mines or rocks. The dataset consists of 20 samples with 60 real-valued features per sample and an integer label (zero or one, for binary classification). G&S included a number of experiments investigating the performance of neural networks on the data. They include a human performance benchmark as well. This publication is a great example of empirical work investigating fundamental principles of machine learning.

For the purposes of comparing our results amongst ourselves, I have split the data into training, development, and testing sections. The split sections consist of 144, 32, and 32 samples, respectively. Note that G&S used momentum during training (we're not doing that) and they carefully initialized their network by hand (see the paper; we're not doing that either), and they split using 192/16 train/test whereas we're using a tougher setup of 143/32. Your accuracy may not be as high as G&S — that is OK. The important thing to observe for this project is the relationships between the hyperparameters and their effect on overall performance.

Your work involves three sub-problems, intended to guide you through the process of developing a classifier for a new dataset.

Part One: Logistic Regression

I. Experiment with logistic regression to create a learning curve on this dataset. The x-axis should be a number of epochs and the y-axis should be accuracy of the logistic model. You should plot two lines — one each for accuracy on the training data and the development data. Note that in all of these cases you only train on the training data. For this problem you may either write your own logistic regression program, or you may use `logisticv2.py` as supplied with this project.

You will probably want to train for a few hundred epochs and use a starting learning rate of 1. You will know it is working if the training accuracy goes up from epoch to epoch. If this does not happen, try adjusting the learning rate. Allow your model to train until it seems to stop "improving", then train for at least a couple hundred more epochs, then stop.

Record the following:

Learning rate:	_______
Total number of epochs:	_______
Final development set accuracy:	_______
Final training set accuracy:	_______
Best development set accuracy:	_______
Best training set accuracy:	_______

Finally, using the data observed so far, create a learning curve as described at the beginning of this problem. Call this plot, "Accuracy vs Epochs (Logistic Regression).

II. Now that you have successfully trained a model, repeat the training process (but not the experimentation) 30 times. Your objective is to evaluate the variation in model performance with each random initialization. Create dictionaries from integer-valued epochs to lists of float-valued accuracies for development and training data. Then produce another learning curve; however, this time you should use the mean accuracy as determined over the 30 samples. Call this plot, "Mean Accuracy vs Epochs (Logistic Regression)".

III. Optional: For 5% bonus credit, also include a box-and-whiskers plot indicating the range of this data.

IV. Answer the following questions:

- Do you notice an asymptote for the training data? If so, what is its value and how many should you train to have a high probability of getting a good model?

- Do you notice an asymptote for the development data? If so, what is its value and how many should you train to have a high probability of getting a good model?

Part Two: Multilayer Perceptrons

Now it is time to explore a true multi-layer perceptron. You may either write your own program or use the supplied implementation using the Keras API.
To install Keras, see their website at https://keras.io.   If you have your own machine with python, it is probably easiest to install via a combined python/Keras distribution such as Anaconda (available at https://www.anaconda.com/distribution).

I. You should train an MLP with 12 hidden units using an initial learning rate of 1. (You are welcome to experiment with other learning rates, but this one worked for me.) Once you are reasonably confident that your MLP is learning (i.e., training set accuracy is reliably increasing) allow your MLP to learn "for a while" and see if you can detect another asymptote as you did when investigating logistic regression.

   Once you have finished training, produce another learning curve as before.   Call this one "Accuracy vs Epochs (MLP)".   At the time that you stop training, record the following:

Learning rate:	________
Total number of epochs:	________
Final development set accuracy:	________
Final training set accuracy:	________
Best development set accuracy:	________
Best training set accuracy:	________

II. Now that you have successfully trained a model, repeat the training process (but not the experimentation) 30 times.   Your objective is to evaluate the variation in model performance with each random initialization.   Create dictionaries from integer-valued epochs to lists of float-valued accuracies for development and training data. Then produce another learning curve; however, this time you should use the mean accuracy as determined over the 30 samples.   Call this plot, "Mean Accuracy vs Epochs (MLP)".

   Optional: For 5% bonus credit, also include a box-and-whiskers plot indicating the range of this data (this is in addition to the bonus marks for logistic regression!)

III. Answer the following questions:

- Do you notice an asymptote for the training data? If so, what is its value and how many should you train to have a high probability of getting a good model?

- Do you notice an asymptote for the development data?   If so, what is its value and how many should you train to have a high probability of getting a good model?

Part Three: Putting It All Together

Using everything you have learned answering problems one and two, identify your best estimate of your best setup (i.e., either logistic regression or neural networks, reasonable number of epochs, and learning rate), then train 30 more times using this setup and report mean, min, max, and standard deviation of TEST SET accuracy.

For 5% additional bonus credit (up to maximum of 15%) you may include a histogram of test set accuracy.

Additional Comments

If you encounter any major problems or computational bottlenecks please post public questions to piazza. I would like everyone to make as much progress as possible on this project given our short remaining timeframe.

Submission and Grading

Submit a single PDF writeup of your responses (no intense writing necessary — just include the labeled plots, include your name (and your group members names), and your answers to each of the questions in the same order that they were asked.

You do not have to submit any source code for this assignment.

Your submission will be graded according to the following rubric:

Plots

10	logistic regression learning curve — single sample
15	logistic regression learning curve — mean of 30 trials
10	neural network learning curve — single sample
15	neural network learning curve — mean of 30 trials

Questions

5	asymptote for logistic regression and estimated epochs for good performance
5	did you observe any overfitting for logistic regression? explain your answer.
5	did you observe any underfitting for logistic regression? explain your answer
5	asymptote for neural network and epochs for good performance (if observed)
5	did you observe any overfitting for logistic regression? explain your answer.
5	did you observe any underfitting for logistic regression? explain your answer
10	best overall dev set accuracy observed; describes the setup
optional	— 5% bonus for box-and-whiskers plot of logistic learning curve
optional	— 5% bonus for box-and-whiskers plot of neural network learning curve
optional	— 5% bonus for histogram of test accuracy using your chosen model