Assignment #3: Neural Networks
CSCI 374 Fall 2019 Oberlin College
Due: Sunday October 13 at 11:59 PM

Background

Our third assignment this semester has three main goals:

1. Implement neural networks as a powerful approach to supervised machine learning,
2. Investigate the impact of hyperparameters on neural network performance as evaluated on empirical data sets.
3. 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/-AUwZ2U1

Data Sets

For this assignment, we will learn from three 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 V jacket_color = red, then yes, else no.` 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. **mnist_5v8.csv**: A data set of optical character recognition of numeric digits from images. The task in this data set is to predict whether a handwritten number is a "5" or an "8". Each instance represents a different grayscale 28x28 pixel image of a handwritten numeric digit. The attributes are the intensity values of the 784 pixels. Each attribute is ordinal (treat them as continuous for the purpose of this assignment). The label is a 0 if the handwritten number is a "5", and a 1 if the handwritten number is an "8". This version of MNIST contains 100 instances of the handwritten numeric digits "5" and "8", randomly sampled from the original training data for MNIST. The overall MNIST data set is one of the main benchmarks in machine learning: http://yann.lecun.com/exdb/mnist/. It was converted to CSV file using the python code provided at: https://quickgrid.blogspot.com/2017/05/Converting-MNIST-Handwritten-Digits-Dataset-into-CSV-with-Sorting-and-Extracting-Labels-and-Features-into-Different-CSV-using-Python.html
3. **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 the other homework assignments.

Program

Your assignment is to write a program called **nn** 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 number of neurons to use in the hidden layer
   c. The learning rate η to use during backpropagation
   d. The percentage of instances to use for a training set
   e. A random seed as an integer
   f. The threshold to use for deciding if the predicted label is a 0 or 1

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

   ```
   python3 nn.py monks1.csv 20 0.01 0.6 12345 0.5
   ```

   which will create a neural network with 20 neurons in the hidden layer. It will then train the network 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. Predicted labels will be a 1 if the predicted probability $\hat{y}$ from the output layer is 0.5 or higher (and a 0 predict label otherwise)

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). You should use the same process for pre-processing the data used in Homework 2.

   Your validation and test sets should have the same size, using the remaining data that is not chosen for training. For example, if you input a training percentage of 60%, then 20% of the instances should be used for the validation set and the remaining 20% for test set. Alternatively, if you input a training percentage of 70%, then 15% should be used for the validation set and the remaining 15% for the test set.

3) You should create a neural network model that will be learned from the training data. It should have **one hidden layer** containing the number of neurons passed in as an input to your program (in Step 1 above). Then, it should have an output layer with **one neuron**. All of the neurons in the hidden layer and the output layer should use the **sigmoid function** as the activation/transformation function.

4) You should train your network using your inputted learning rate and the Backpropagation algorithm. As in Homework 2, 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).

5) After training the network, calculate its confusion matrix on the test set. Remember to use the threshold value input in Step 1 (the last command line input) when deciding on the predicted label (this is the *only* time we need to use the threshold – it does *not* affect Backpropagation):

$$\begin{aligned}&\text{if } \hat{y} \geq threshold\text{:}\\&\quad predicted = 1\\&\text{else:}\\&\quad predicted = 0\end{aligned}$$

Then the confusion matrix should be output as a file with its name following the pattern: results_<DataSet>_<Neurons>n_<LearningRate>r_<Threshold>t_<TrainingPercentage>p_<Seed>.csv (e.g., results_monks1_20n_0.01r_0.5t_0.6p_12345.csv).

Please note that you **are** allowed to reuse your code from Homework 2, which has already completed a large portion of this assignment for you.

Program Output

The file format for your output file should be the same as in Homework 1 (and 2). Please refer back to the instructions for Homework 1 for more details.

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.

As in Homework 2, **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 neural networks (e.g., using scikit-learn, PyTorch, or Tensorflow in Python), but instead implement your own from scratch.

However, if you have ***difficulties getting the neural network 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) For the monks1.csv data set, pick a training set percentage and a random seed (document both in your README). Use 2 hidden neurons, a learning rate $\eta = 0.1$, and a threshold of 0.5.
    a. What is the test set accuracy you observed for your neural network?
    b. Repeat using the same training set percentage, random seed, and learning rate but with your logistic regression solution from Homework 2 (only use one student's implementation, if you are a group). What is the test set accuracy you observed for your logistic regression model on the monks1.csv data set?
    c. Create 95% confidence intervals for both accuracies. Which learner (neural network or logistic regression) had the highest accuracy? Was the difference statistically significant? What are the implications of your results?
2) For the mnist_5v8.csv data set, pick a random seed. Use a learning rate of $\eta = 0.001$, a training set percentage of 60%, and a threshold of 0.5. Create a neural network with each of the following **numbers of neurons**: 2, 5, 10, 20, and 50.
    a. What is the test set accuracy you observed for each number of neurons? Plot a line chart (using the tool of your choice: Excel, R, matplotlib in Python, etc.) of the test set accuracy as the number of neurons increased. Include your line chart as an image in your GitHub repository.
    b. How did the accuracy change as the number of hidden neurons change? Why do you think this result occurred?
3) For the mnist_5v8.csv data set, use the three learning rates $\eta = 0.001, 0.01, 0.1$. Use the number of neurons that gave the highest accuracy in Q2 (in case of ties, use the *smallest* number of neurons that tied for the highest accuracy), a training percentage of 60%, a threshold of 0.5, and the same random seed used in Q2. Track the accuracy on both the **training set** the **validation set** after each epoch of Backpropagation (i.e., after you feed the entire training set in).
    a. Plot the accuracy of the network on the *training set* for each epoch on a single line chart (again using your favorite tool) for all three learning rates (each learning rate should be a separate line on the same chart, where the x-axis is the epoch and the y-axis is the training set accuracy) . Include your line chart as an image in your GitHub repository.
    b. Plot the accuracy of the network on the *validation set* for each epoch on a single line chart (again using your favorite tool) for all three learning rates (each learning rate should be a separate line on the same chart, where the x-axis is the epoch and the y-axis is the validation set accuracy) . Include your line chart as an image in your GitHub repository.

c. Compare the training set accuracy across the three learning rates. What trends do you observe in your line charts? What do you think this implies about choosing a learning rate?

d. Compare the validation set accuracy across the three learning rates. What trends do you observe in your line charts? What do you think this implies about choosing a learning rate?

4) For the seismic.csv data set, use 10 hidden neurons, a learning rate $\eta = 0.01$, a training percentage of 60%, and your favorite random seed. Using **five different thresholds** (0.05, 0.1, 0.5, 0.9, 0.95) for converting the predicted probabilities into predicted labels, calculate the accuracy and recalls of your trained neural network on the test set.

a. What were the test set accuracies you observed for each threshold value? How did they change as the threshold changed?

b. What were the recalls on each label that you observed for each threshold value? How did they change as the threshold changed?

Remember that we calculate recall for a label as:

$$Recall_1 = \#\ of\ times\ we\ correctly\ predicted\ label\ 1 \Big/ \#\ of\ instances\ with\ a\ true\ label\ of\ 1$$

$$Recall_0 = \#\ of\ times\ we\ correctly\ predicted\ label\ 0 \Big/ \#\ of\ instances\ with\ a\ true\ label\ of\ 0$$

c. Remembering that a label of 1 is predicting a seismic event, and a label of 0 is predicting that there is no seismic event, which threshold do you think is ideal for this data set? Why do you think this threshold is the best?

Bonus Question (5 points)

Modify your program to be able to have multiple hidden layers, all with the same number of neurons. The number of hidden layers to create should be taken in as a seventh parameter on the command line (after the random seed).

Pick three different numbers of hidden layers. Repeat Question 1, except also vary the number of hidden layers based on the set of three that you picked (so that you have 15 different combinations of hidden layers and neurons per layer). How does changing the number of layers further impact the accuracy of the neural network as you also vary the number of hidden neurons per layer?

README

Within a README file, you should include:

1) Your name,
2) Your answers to the questions above,
3) A short paragraph describing your experience during the assignment (what did you enjoy, what was difficult, etc.)
4) An estimation of how much time you spent on the assignment, and
5) 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 two homeworks, **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 and output directions: /5 points
Properly read in and pre-processed the data into training, validation, and test sets: /5 points
Correctly implemented the neural network model: /20 points
Correctly implemented Backpropagation: /20 points
Correctly answered the research questions: /40 points
Provided requested README information: /5 points
Appropriate code documentation: /5 points