Assignment #4: Classification and Regression with Neural Networks

CSCI 373 Spring 2024 Oberlin College

Due: Friday April 19 at 11:59 PM

Background

Our fourth assignment this semester has four main goals:

1. Practice using popular Python libraries for creating neural network machine learning models,
2. Implement programs that can solve both classification and regression tasks, depending on the user's input,
3. Conduct experiments to better understand the importance of correctly choosing model hyperparameters, and
4. Practice visualizing and interpreting results using charts.

Gitting Started

You can get started on the assignment by following this link:

https://classroom.github.com/a/u5RbYg_2

Data Sets

For this assignment, we will learn from four pre-defined data sets:

1. **mnist1000.csv**: A data set of optical character recognition of numeric digits from images. Each instance represents a different grayscale 28x28 pixel image of a handwritten numeric digit (from 0 through 9). The attributes are the intensity values of the 784 pixels. Each attribute is ordinal (treat them as continuous for the purpose of this assignment) and a nominal label. This version of MNIST contains 1,000 instances of each handwritten numeric digit, 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
2. **penguins.csv**: A data set describing observed measurements of different animals belonging to three species of penguins. The four attributes are each continuous measurements, and the label is the species of penguin. Special thanks and credit to Professor Allison Horst at the University of California Santa Barbara for making this data set public: see this Twitter post and thread with more information (https://twitter.com/allison_horst/status/1270046399418138625) and GitHub repository (https://github.com/allisonhorst/palmerpenguins).

3. **energy.csv**: A data set describing the energy consumption in 10-minute increments by appliances in a low-energy residence in Belgium. The task is to predict how much energy was consumed by appliances. Each of the 27 attributes are numeric and describe measurements from sensors in the residence or nearby weather stations, as well as energy usage by lights. This data set comes the UCI Machine Learning Repository: https://archive.ics.uci.edu/dataset/374/appliances+energy+prediction
4. **seoulbike.csv**: Another data set describing bike rentals in a metropolitan area (Seoul, South Korea). Again, the task is to predict how many bikes will be rented hourly throughout the day over a two-year period. The 11 attributes are a mix of 2 categorical and 9 numeric attributes, including information such as the season, whether it was a holiday, and current weather conditions. This data set comes the UCI Machine Learning Repository: https://archive.ics.uci.edu/dataset/560/seoul+bike+sharing+demand

The file format for each of these data sets is the same as Homeworks 1-3:

- 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 correct label we want to the computer to learn to predict, and all other entries (following the commas) are attribute values.
- Some attributes are strings and others are numbers. The labels for some data sets are strings (for classification tasks) and the labels for other data sets are numbers (for regression tasks)

Program

Your assignment is to write a program called **neuralnet.py** that behaves as follows:

1) It should take five parameters as input from the command line:

    a. The path to a file containing a data set
    b. The learning rate to use for training the neural network
    c. The number of neurons to put in a single hidden layer
    d. The percentage of instances to use for a training set
    e. An integer to use as a random seed

    For example, I might run

    ```
    python neuralnet.py penguins.csv 0.01 8 0.75 12345
    ```

    which will train a neural network with 8 hidden neurons using a learning rate of $\eta = 0.01$ to learn how to classify the `penguins.csv` data set using a random seed of `12345`, where `75%` of the data will be used for training (and the remaining `25%` will be used for testing)

2) Next, the program should read in the data set as a set of instances. Every categorical attribute should be converted to a **one-hot encoding** and every numeric attribute should be scaled using **max-min normalization** (as we did in Lab 4 and Homework 3). [Again, please do **not** rescale the label for this assignment]. For classification tasks (`mnist1000.csv` and `penguins.csv`), the **string label values should also be converted to integers** (e.g., Adelie = 0, Chinstrap = 1, Gentoo = 2), as we did in Lab 5.

3) After processing the data, it should be split into training and test sets (using the random seed input to the program), similar to the process in Homeworks 1-3.

4) A neural network model with the specified number of hidden neurons should be constructed. The **number of neurons in the output layer** should be the number of possible labels for classification tasks or one neuron for regression tasks. Hidden neurons should use `"sigmoid"` for the **activation function**, and output neurons should have nothing specified for the activation function. As in Lab 5, you should use `tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)` as the **loss function** for classification tasks; for regression tasks, you should use `tf.keras.losses.MeanSquaredError()`

5) The neural network model should be trained using the training set. As we did in Lab 5, 20% of the training data should be used as a **validation set** (when calling the `fit` method on the model), and a **`CSVLogger`** should save the loss on both the training set and validation set every epoch of training to a CSV file. The `CSVLogger` should also log the predictive performance of the model (either `"accuracy"` for classification tasks or `"mean_absolute_error"` for regression tasks) every epoch. Please use **250 as the number of epochs** during training (also specified when calling the `fit` method).

6) Next, predictions on the test set created in Step 2 should be made using the neural network trained in Step 5.
7) Rather than creating separate output files for every run of the program, instead create a single CSV file called `results.csv` that **adds one line per run of the program**, as described below.

Please note that you **are** allowed to reuse your code from Homeworks 1-3 and any labs to help you complete this assignment.

<u>Program Output</u>

The file format for your output file is different for this homework since we are performing both classification and regression.

The file format for your CSV (comma-separated values) output file should be as follows:

- The first row should be the names of the four columns: `Dataset`, `LearningRate`, `Neurons`, and `Performance` followed by a newline character (`\n`) to serve as the **header** of the table.
- Each run of the program then adds a line to the end of the file. The first three values of a line will come from the user's command line arguments, and the last will be the predictive performance of the model – the **accuracy** value on the test set for classification tasks and the **mean absolute error (MAE)** on the test set for regression tasks.

For example, the first few rows of your output file might look like:

```
Dataset,LearningRate,Neurons,Performance
penguins,0.0001,128,0.813953488372093
penguins,0.0001,256,1.0
mnist1000,0.0001,128,1.0
mnist1000,0.0001,256,1.0
energy,0.001,128,51.62748144201581
energy,0.001,256,51.168584989834564
seoulbike,0.0001,128,469.71249718321457
seoulbike,0.0001,256,379.3957882423023
```

<u>Programming Languages</u>

In this class, we are using the **Python** programming language for all of our labs and assignments. For this assignment, <u>you are allowed</u> to use any Python libraries (e.g., `pandas`, `numpy`, `tensorflow`) and not only those originally built into Python.

Research Questions and Experience

Please use your program to answer the four research questions in the provided `README.md` file. There are also three questions at the end of the `README.md` file that you should answer about your experience completing the assignment.

Please remember to commit your solution code, image files, and `README.md` file to your repository on GitHub. You do not need to wait until you are done with the assignment to commit your code. ***Make sure to document your code***, explaining how you implemented the different components of the assignment.

Honor Code

As with Homeworks 2 and 3, **each student is allowed to work with *one partner* to complete this assignment**. Groups are also encouraged to collaborate with one another to discuss the abstract design and processes of their implementations. However, sharing code or answers to the research questions (either electronically or looking at each other's files) between groups is not permitted.

Grading Rubric

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

Followed input and output directions: /5 points
Properly processed the data: /10 points
Created a neural network with the correct structure: /10 points
Correctly trained the neural network: /5 points
Correctly makes predictions and calculates test set performance: /10 points
Correctly answered the research questions: /50 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 at the top of each function explaining the purpose of the function (inline comments are also welcome).