CARNEGIE MELLON UNIVERSITY

10601A-F16: Homework #7 - “Neural Networks”

TA-in-charge:
Logan Brooks

Office Hours:

Thu. 10/27	10am–11am	GHC 6219
Fri. 10/28	10am–11am	GHC 6219
Sat. 10/29	2pm–4pm	GHC5, Citadel Teaching Commons, Table 6
Sun. 10/30	2pm–4pm	GHC5, Citadel Teaching Commons, Table 6
Mon. 10/31	2pm–4pm	GHC 6219
Tue. 11/01	2pm–4pm	GHC 6219

Assigned: Wednesday, 26 October 2016.
Due: 12:59:59pm on Tuesday, 1 November 2016.
Late Penalty: 20% per day.

Policy on Collaboration among Students, version of Fall 2016

Some of the homework assignments used in this class may have been used in prior versions of this class, or in classes at other institutions. Avoiding the use of heavily tested assignments will detract from the main purpose of these assignments, which is to reinforce the material and stimulate thinking. Because some of these assignments may have been used before, solutions to them may be (or may have been) available online, or from other people. **It is explicitly forbidden to use any such sources, or to consult people who have solved these problems before. It is explicitly forbidden to search for these problems or their solutions on the internet.** You must solve the homework assignments completely on your own. I will be actively monitoring your compliance, and any violation will be dealt with harshly. Collaboration with other students who are currently taking the class is allowed, but only under the conditions stated below.

The purpose of student collaboration is to facilitate learning, not to circumvent it. Studying the material in groups is strongly encouraged. It is also allowed to seek help from other students in understanding the material needed to solve a particular homework problem, provided no written notes are shared, or are taken at that time, and provided learning is facilitated, not circumvented. **The actual solution must be done by each student alone**, and the student should be ready to reproduce their solution upon request. In the case of programming assignments, **all code must be written by each student alone**. We will strictly enforce this policy. **The presence or absence of any form of help or collaboration, whether given or received, must be explicitly stated and disclosed in full by all involved.** Specifically, **each assignment must contain a file named collaboration.txt where you will answer the following questions**:

- Did you receive any help whatsoever from anyone in solving this assignment? Yes / No. If you answered 'yes', give full details? (e.g."Jane explained to me what is asked in Question 3.4").
- Did you give any help whatsoever to anyone in solving this assignment? Yes / No. If you answered 'yes', give full details? (e.g. "I pointed Joe to section 2.3 to help him with Question 2").

If you gave help after turning in your own assignment and/or after answering the questions above, you must update your answers before the assignment's deadline, if necessary by emailing the TA in charge of the assignment.

Collaboration without full disclosure will be handled severely, in compliance with CMU's Policy on Cheating and Plagiarism.

Students are responsible for pro-actively protecting their work from copying and misuse by other students. If a student's work is copied by another student, the original author is also considered to be at fault and in gross violation of the course policies. It does not matter whether the author allowed the work to be copied or was merely negligent in preventing it from being copied. When overlapping work is submitted by different students, **both students will be punished**.

All violations (even first one) of course policies will always be reported to the university authorities, will carry severe penalties, usually failure in the course, and can even lead to dismissal from the university. This is not an idle threat — it is my standard practice. You have been warned!

0 General Instructions

The goal of this assignment is to familiarize you with several implementation decisions involved in training neural networks, by ***implementing a neural network with one hidden layer, entirely from scratch***. You will be using datasets from the two domains similar to the ones used in the "Decision Tree" assignment (assignment 5). A major change in this assignment is that some input attributes are multi-valued or continuous-valued instead of the binary-valued attributes of Assignment 5. Notice that neural networks are well suited for continuous valued inputs. They are also different from linear regression models as they accommodate non-linearity, which allows us to consider a larger space of functions than simple linear functions. However, training (or "fitting") the model by minimizing the training set error poses several challenges, as there might be multiple local minima in the parameter (weight) space.

Your goal is to achieve the lowest error rates on the test sets for both domains. **(The test sets are hidden from you, which means you are not able to, and not allowed to, access them.)** You can (and should) experiment with different configurations for the two domains using the development data.

The first task is to predict whether a song was a 'hit', meaning that it made it onto the Billboard Top 50 — each instance has a label "Hit?" equal to "yes" (1) or "no" (0). The attributes are: year of release(multi-valued discrete, range $[1900, 2000]$), length of recording (continuous, range $[0, 7]$), jazz (binary discrete, "yes"/"no"), rock and roll (binary discrete, "yes"/"no").

The second task is to predict the final student scores in a high school course. The attributes are student grades on 2 multiple choice assignments M1 and M2, 2 programming assignments P1 and P2, and the final exam F. The scores of all the components are integers in the range $[0, 100]$. All the attributes are multivalued discrete. Again, check the csv files to see the attribute values. The final output is also integer with a range $[0, 100]$. Notice that, for this problem, we are performing regression, rather than classification, which was done in the decision tree assignment.

Before you start coding your neural network, think about various decisions you need to make about your implementation. A few of them are:

- Initialization of weights
- Number of units in the hidden layer
- Network connectivity
- Number of iterations (or, when to stop the training)
- Pre-processing and post-processing of input and output

Generally, the number of hidden layers too is an important decision to be made, but in this assignment we only require you to implement a Neural Network with one hidden layer. There may be many more decisions to be made. All such decisions affect the speed and performance of your neural network, some more than others. You must use the **sigmoid** function as discussed in the class for introducing nonlinearity, even though other nonlinear functions exist.

Important: Autolab won't allow your code to be run for longer than 3 minutes. Hence, decide upon your stopping criteria so that it completes within 3 minutes. You may check the wall clock time to decide when to halt your program.

EVEN MORE IMPORTANT: Since the running times of your programs will be significantly longer than in past assignments, there will be queues on Autolab when multiple students submit solutions. It is your responsibility to submit early enough to account for possible waiting times on Autolab. Do not expect to submit your solution after 11pm on the due date and be able to see your results before the deadline.

You may use Python, Java, C, or C++ to complete this assignment. This assignment involves multiplication of vectors and matrices, which we encourage you to write the code for yourself. However, there are some packages available and installed on the VM that implement matrix math, which you may use instead:

For Python (python2): numpy, scipy

For Java: JAMA

For C: LAPACKE (`#include <lapacke/lapacke.h>`, `-llapacke -llapack -lblas`)

For C++ (C++11): (base only)

Do not use any other packages. We've provided you with attributes and labels split into training and development data in files `music*.csv` and `education*.csv`:

- `music_train.csv`, `education_train.csv`: attributes and labels for training data, with column names on the first line
- `music_dev.csv`, `education_dev.csv`: attributes for development data, without labels, with column names on the first line; shares the same format (but not necessarily the same number of records) as the test set that you will be evaluated on
- `music_dev_keys.csv`, `education_dev_keys.csv`: labels for development data, as a single column with no column names

The format is comma separated, one row per observation, one column per attribute. Your program will take the name of two file names as input:

- the first file contains labeled data, to be used to train the network (fit the weights).
- the second file contains unlabeled data, to be used to make predictions.

IMPORTANT: Do not use any standard neural network packages.

1 Neural Network Development

We expect two files: `NN_music.py` and `NN_education.py` (or `NN_music.java` and `NN_education.java`, or corresponding `*.c` or `*.cpp` files). You have training and development sets for both the domains. You will estimate the weights on your training set (the first file), printing the squared error after each iteration during training, and then use the learned weights to print the output decisions on the development/test set (the second file). (An iteration is defined as one pass through the entire data. If you use an incremental update method rather than a batch method, print out the error each time you iterate through the entire data set, not after updating for each point.) The output syntax expected by the grader is:

```
$ python NN_music.py <training_file> <test_file>
1022.6
```

```
1012.7
1005.8
...(do not print these dots; they just signify continuation)
TRAINING COMPLETED! NOW PREDICTING.
no
yes
yes
no
yes
...
```

For Java, the commands would be:

```
$ javac -cp Jama-1.0.3.jar *.java
$ java NN_music <training_file> <test_file>
$ java NN_education <training_file> <test_file>
```

For C, the commands would be:

```
$ gcc -O2 NN_music.c -I. -llapacke -llapack -lblas -o NN_music
$ gcc -O2 NN_education.c -I. -llapacke -llapack -lblas -o NN_education
$ NN_music <training_file> <test_file>
$ NN_education <training_file> <test_file>
```

For C++, the commands would be:

```
$ g++ -std=c++11 -O2 NN_music.cpp -I. -o NN_music
$ g++ -std=c++11 -O2 NN_education.cpp -I. -o NN_education
$ NN_music <training_file> <test_file>
$ NN_education <training_file> <test_file>
```

The output values shown in the examples do not represent actual output. In the above syntax, `<training_file>` is `music_train.csv` and `education_train.csv`, and `<test_file>` is `music_dev.csv` or `education_dev.csv` for the music and education domains, respectively. The autograder will run your program on the same training files and separate test datasets, and then evaluate your performance (error) on the test datasets. This error will be shown on the leaderboard.

Some useful questions to think about: Is your performance on the test set similar to your performance on the development set? Does improving your performance on development set always result in a better performance on your test set?

Grading criteria: A correct implementation for both the datasets will be sufficient for full credit. This is determined by **adequate performance on the test set**, and **strictly decreasing training error** on each iteration (Be sure not to overshoot a minimum and increase your training error!). Also, in this assignment, we will give extra credits to students with a good performance (low error rate) on test data (We will run with your latest submission). Note that we will set the threshold for extra credits as soon as the assignment is due, so students who are granted extensions won't cause you to lose bonus points by submitting high scoring submissions after the deadline.

Submit a `.tgz` file containing your source code(`NN_music.py`, `NN_education.py` or `NN_music.java`, NN_education.java, etc.), and collaboration.txt. You can create this archive by running the following command:

```
tar -cvf hw7.tgz *.py *.java *.c *.cpp *.txt
```

DO NOT put the above files in a folder and then tar gzip the folder. You must submit this file to the “homework7” link on Autolab.

Beware! You are allowed to run your code on the autograder only 10 times. Hence, work with the development set and test your performance on autolab only when you are very confident! Good Luck! May the global minimum be with you.