TCSS 455: Introduction to Machine Learning
Spring 2019, Homework #3
Due: Thursday, May 30

Please submit a hard copy of your answers to the homework problems below. Staple all of your pages together (and order them according to the order of the problems below) and have your name on each page, just in case the pages get separated. Write legibly (or type) and organize your answers in a way that is easy to read. Neatness counts!

For each problem, make sure you have acknowledged all persons with whom you worked. Even though you are encouraged to work together on problems, the work you turn in is expected to be your own. When in doubt, invoke the Gilligan's Island rule (see the syllabus) or ask the instructor.

All homeworks are due at the beginning of lecture on the due date. I will accept one homework up to one lecture late without penalty. You do not need to inform me – I will accept it automatically, no questions asked or documentation required.

---

1. **Backpropagation.** (3 pts) Solve problem 4.7 from the textbook by applying the BACKPROPAGATION algorithm from Table 4.2 (p.98). This entails that you should assume that the hidden unit c and the output unit d are sigmoid units. Use stochastic gradient descent. This means that in iteration 1, you should present the first training example and update the weights. In iteration 2, you should present the second training example and update the weights again. It is in iteration 2 that momentum starts playing a role.

   If working through this pen-and-paper example of backpropagation might seem slightly painful, please know that it is a great exercise to help you grasp this weight learning algorithm! This exercise also requires you to read about momentum, which is an important notion that we did not cover in class.

2. **Gradient descent weight update rule for a tanh unit.** (2 pts) Assume throughout this exercise that we are using gradient descent to minimize the error as defined in formula (4.2) on p.89 in the textbook:

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

   Recall that the corresponding weight update rule for a sigmoid unit like the one in Figure 4.6 on p.96 in the textbook is:

$$w_i \leftarrow w_i + \eta \cdot \sum_{d \in D} (t_d - o_d) \cdot o_d \cdot (1 - o_d) \cdot x_{i,d}$$

   Let us replace the sigmoid function σ in Figure 4.6 by the function "tanh". In other words, the output of the unit is now:

$$o = \tanh(net) = \tanh(w_0 x_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n)$$

   Derive the new weight update rule. Show your work, and indicate clearly in your answer what the weight update rule for the tanh unit looks like. Hint: $\tanh'(z) = 1 - (\tanh(z))^2$.

3. **Training a neural network with Keras.** (5 pts) For this problem you will implement and train your own neural network. To this end, you will use Keras, a high-level neural networks API. If not already done so, then install Keras on your computer with Tensorflow as the backend (Theano is NOT suggested). Instructions are available on `https://keras.io/`.

The file hw3.py in Files/homeworks/hw3 contains starter code for predicting the *extraversion score* of a Facebook user based on the 81 LIWC features. A corresponding dataset Facebook-User-Personality-LIWC-HW3.csv is provided. It is the same data that we used in the session about machine learning in Python, with the first two columns and the header row removed. The dataset contains information for 9500 users (rows). The first 81 columns are the LIWC feature values, extracted from the Facebook status updates. The remaining columns are respectively the personality trait scores for *openness*, *conscientiousness*, *extraversion*, *agreeableness*, and *neuroticism*. For this homework problem, only the 81 LIWC columns and the column for *extraversion* is relevant. You can ignore the other personality trait columns.

Reminder: Do not share the dataset with anyone who is not registered for TCSS555.

In the starter code, the dataset is split in 8000 instances for training, and 1500 instances for testing. When running the code, you will observe that linear regression works quite well, achieving a MSE of approximately 0.64. This corresponds to a RMSE of approximately 0.80, as we are used to seeing on the scoreboard for the project. The neural network on the other hand performs horribly bad. The problem is in this part of the code:

```
model.add(Dense(1,input_dim=81))
model.compile(optimizer='adam', loss='mse', metrics=['mse'])
model.fit(X_train,y_train)
```

This code is not wrong. It defines a rudimentary neural network with poor predictive performance. Your task is to change this code to *make the neural network perform better than the linear regression model* in terms of MSE.[1] You shouldn't have to make any changes to other parts of the starter code. In particular, don't change the code for fixing the random seed, as a fixed random seed allows us to reproduce the same train/test split of the data, so that we all get the same MSE for the linear regression model (among other things).

This homework problem is expected to help you get a better understanding of the design choices to make when defining and training a neural network. You will also experience that fine tuning a neural network requires more skills from a data scientist compared to the simple linear regression model, which performed reasonably well "right out of the box".

What to submit:

(a) (hard copy) A printout of the part of the code that you changed.

(b) (hard copy) Screenshots of your terminal window that contain the line:

```
MSE with linear regression: 0.642565125843
```

and the completed line:

```
MSE with neural network: ...
```

The displayed MSE obtained with the neural network should be *lower* than the MSE with linear regression, as evidence that you correctly completed the assignment.

(c) (hard copy) A brief description of interesting aspects about training neural networks that you learned while completing this assignment.

(d) (electronic submission on Canvas) Your file hw3.py. Your code should be compatible with Python 3.

[1]You might find this Keras "cheat sheet" helpful: `https://www.datacamp.com/community/blog/keras-cheat-sheet`