**Machine Learning and Artificial Intelligence for Bioinformatics**

**Homework 6 – Due October 13th at 10am.**

**Each of the 2 questions below is worth 50 points.**

This homework needs to be completed on the Google Collaboratory, and the results submitted as screenshots in a .doc or .pdf. Please include the completed run of the corresponding code the question refers too along with your written answer (you can include additional code if you want). You will need to

Also you are welcome to run Tensorflow code outside of the Collaboratory, if you have such a setup, please note though that the submission need to follow the same format, meaning code cells –> output as shown on the Collaboratory (for example do not submit Python interactive command terminal code)

**Note:** You need to write this code yourself, no collaboration or code copying. Computer code is like English writing, I can identify plagiarism as I read the responses. For students who copy code from others, the question will be graded with zero points.

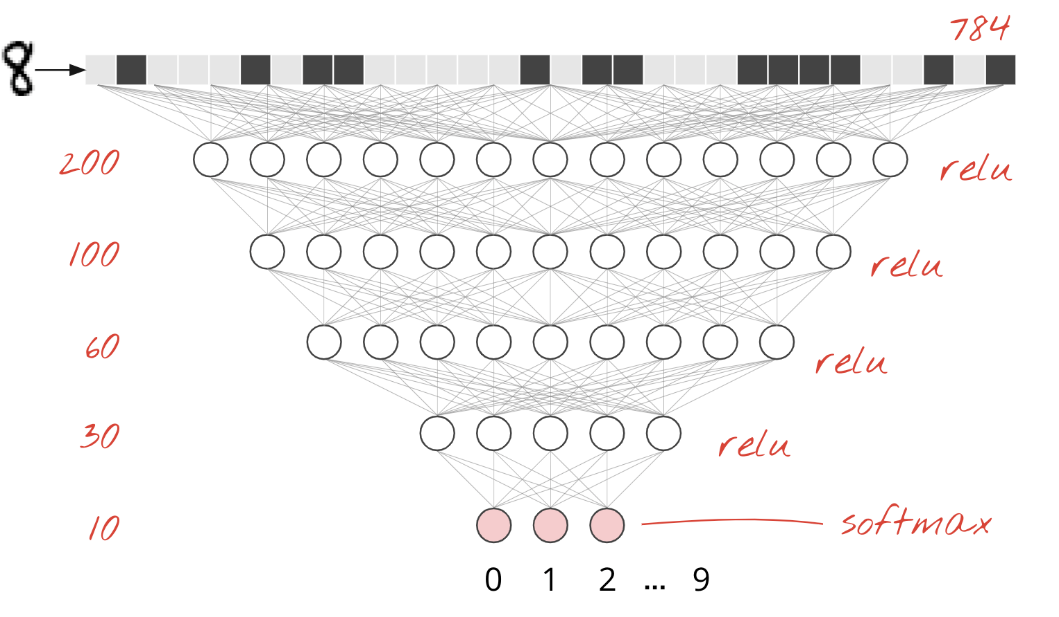
**Question 1.**

This exercise is simple, if you completely understood the simple feed forward neural network for the MNIST we saw in the previous homework and covered in the last lecture, and which you can reference towards writing this code:

https://colab.research.google.com/github/AviatorMoser/keras-mnist-tutorial/blob/master/MNIST in Keras.ipynb

For this question, implement the neural network shown on the picture below, and feed it with the MNIST data for training. Show your implemented code and outputs produced during the training, including the testing for accuracy of the predictions. As always it is best to work on the Google Collaboratory, and this has to be implemented with Tensorflow.

How do you observe this larger and more complex network performing, compared to the smaller one we discussed in the class and is available on the .ipynb above ?



Ans: The load data and pre-processing:

Graphical user interface, text, application

Description automatically generated

Create the model:

Table

Description automatically generated

Train and evaluate:

Graphical user interface, text, application, email

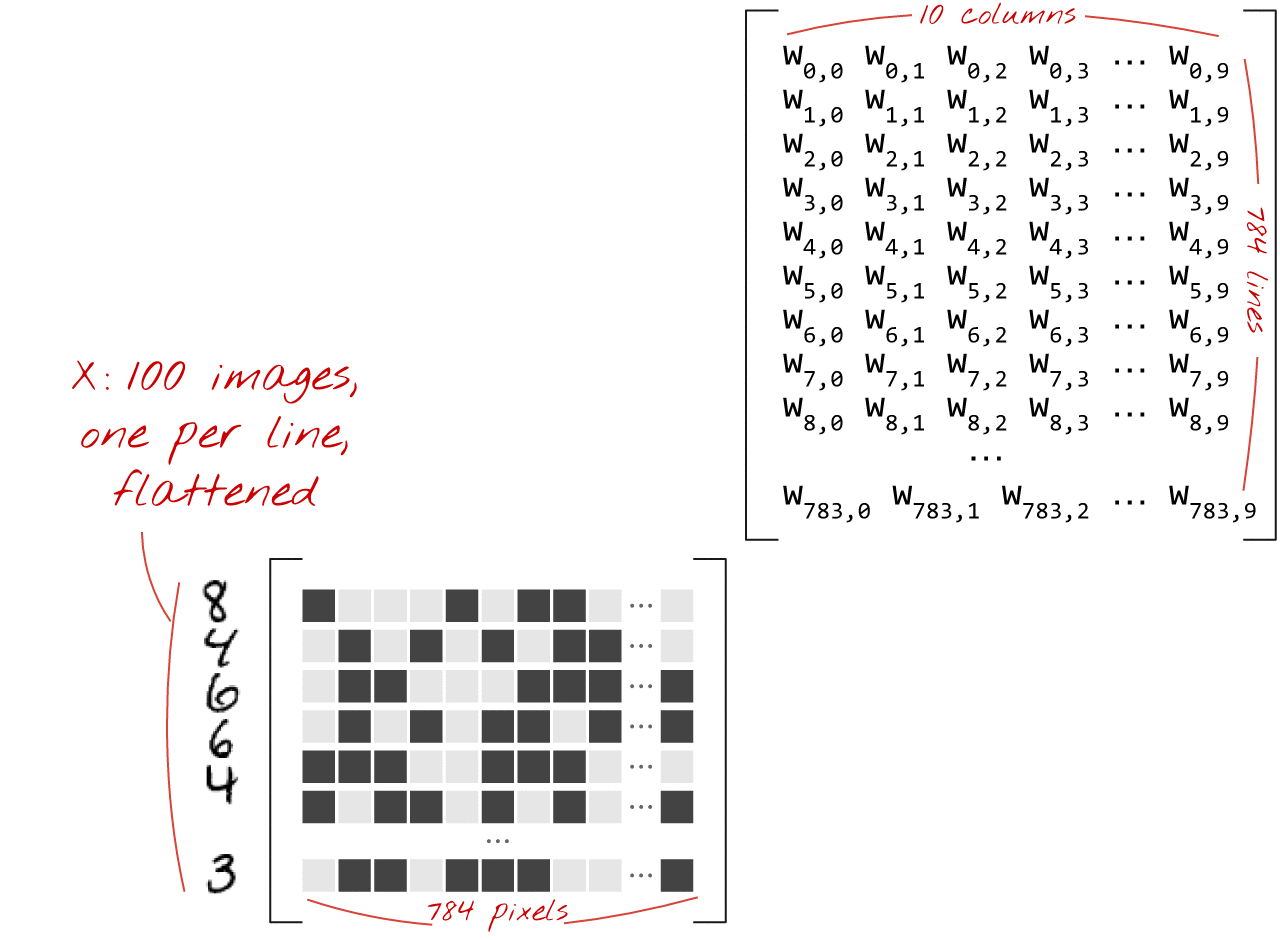
Description automatically generated

The larger model can get lower training loss and higher training accuracy; however, it doesn’t mean higher testing performance. In my experiment, I got higher testing loss and similar testing accuracy comparing with the smaller model in the sample note.

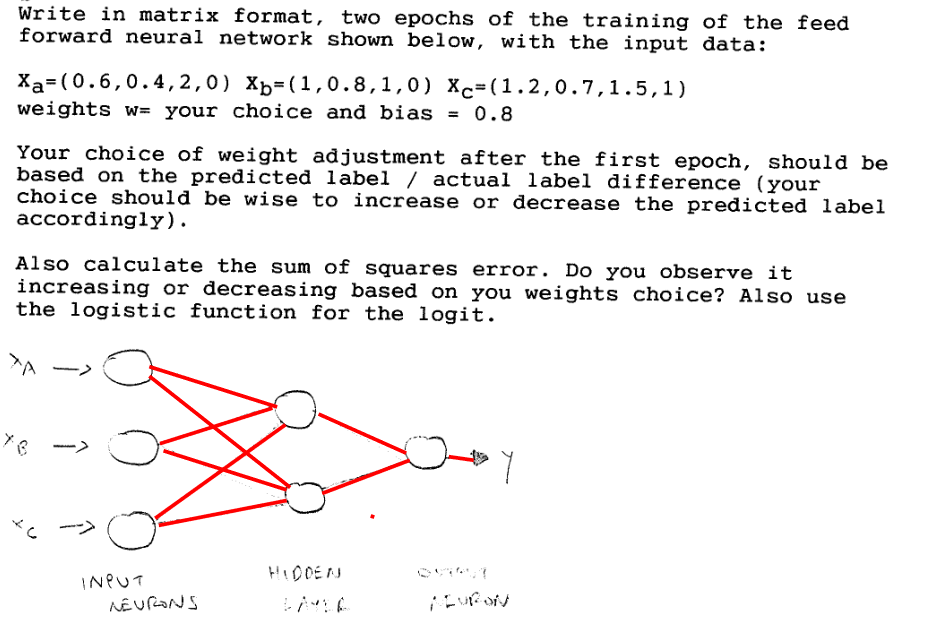
**Question 2.**

The goal of this question for you to implement manually two epochs (feel free to do more if you want), of the feed forward learning process of a neural network, using matrix multiplications (not Tensorflow) as we have seen in the image below and from the link of “neural networks without a PhD”:

https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist#3



We will have two differences from the above: first we will use a network of three inputs, we will additionally have a single hidden layer with two neurons, and a single output as shown in the picture below, from our earlier homework where we did the matrix multiplications by hand (but here you will need to implement them in Python).



Last item on each input vector above is the y. So the purpose of this exercise is to make you think about the dimensions of the matrices. The difference with the link of “neural networks without a PhD” (despite having a much smaller input), is that here we have an additional hidden layer, and a single output instead of 10. But if you remember what we mentioned that neurons do not “realize” if they are in the input layer, or hidden layer, they just receive inputs and do the matrix multiplications and activation function.

Essentially you need to do one additional matrix multiplication, giving the outputs from the first layer, as inputs to the second layer. The weights you select randomly, as we did in our previous homework.

Below you will find some guides on matrix multiplication with Numpy (and you have to use Numpy and the Collaboratory for running the code as usual).

<https://www.kdnuggets.com/2017/03/working-numpy-matrices.html>

<https://numpy.org/doc/stable/reference/generated/numpy.matrix.html>

<https://numpy.org/doc/stable/reference/generated/numpy.dot.html>

<https://www.geeksforgeeks.org/multiplication-two-matrices-single-line-using-numpy-python/>

<https://www.journaldev.com/32966/numpy-matrix-multiplication>

<https://likegeeks.com/numpy-matrix-multiplication/>

Also for the y, you will need to calculate the sum of squared errors in your code, as the last step in each epoch. Use this as example how to do the sum of squares error calculation (feel free to search around the web for additional examples):

<https://www.kite.com/python/answers/how-to-calculate-mean-squared-error-in-python>

Remember you need to do 2 epochs, and because we do not know the backpropagation yet (and you could not possibly do it manually), simply adjust the weights matrix manually at the end of the first epoch (as we did in our previous homework).

So your code has to show the matrix multiplications with numpy in each epoch, including the weights matrices. For those of you who find this very simplistic, feel free to implement two epochs with the MNIST dataset, and since in that case you would need very large weight matrices (which you could not hardcode), just initialize with random values (you would have to search how to create a random value matrix.