CS577 HW2 Report

Subowen Yan

February 29, 2020

1 Problem statement

Self-implement and test a three-layer neural network for a three class-classification using python and without using a GPU framework. Using categorical cross-entropy for loss, sigmoid activation functions for hidden layers, and SoftMax for the output layer. The computation tree and its forward and backward traversal should be hardcoded in the program without having to support dynamic configurations. Using two or more data sets to evaluate. And splitting the data sets into training, validation and testing sets. Plotting training and validation loss and accuracy as a function of epochs. After the fine-tuning model, retrain the final model and test performance on the test collection.

2 Proposed solution

The first layer that I will be using sigmoid. The hidden layer will be sigmoid as well. The output layer will be SoftMax. The loss function will be categorical cross-entropy. Using forward propagation and back-propagation to update weights and biases.

3 Implementation details

Forward pass will be done as follow:
Firstly, w1 * x + b1 passes the results to sigmoid. Secondly, w2 * last result + b2 passes the results to the sigmoid. Finally, w3 * last result + b3 passes the results to the SoftMax.
Backpropagation pass will be done as follow:
Firstly, calculate the derivative of cross-entropy and SoftMax using data from the results of forwarding pass. We will get dw3, db3, and a data matrix. Passing the data matrix * w3.T * derivative of sigmoid to the dense_back function, we will get dw2 and db2. Using last matrix * w2.T * derivative of sigmoid to the dense_back function, we will get dw2 and db2.
Using wi -= learning rate * dwi and bi -= learning rate * dbi to update w's and b's.

Data:
The first dataset is a game dataset. It is called the connect-4 game. This database contains all legal 8-ply positions in the game of connect-4 in which neither player has won yet, and in which the next
move is not forced. Number of Attributes: 42, each corresponding to one connect-4 square. Attribute Information: (x=player x has taken, o=player o has taken, b=blank). We need to predict win, loss or draw. I replace all x -> 2, o -> 1, b -> 0, win -> 0, loss -> 1, draw -> 2.
The second dataset consists of the multi-spectral values of pixels in 3x3 neighborhoods in a satellite image. The image has already been flattened. I just divide 255 for all of the input data to do normalization.

Number	Class
1	red soil
2	cotton crop
3	grey soil
4	damp grey soil
5	soil with vegetation stubble
6	mixture class (all types present)
7	very damp grey soil

Since we only need to do three-class classification. I only keep the number 1 red soil, 2 cotton crop, and 3 grey soil.

Instructions for running the problem:
putting the connect-4.data and sat.trn into the same folder of sample1.py and sample2.py folder. Run python sample1.py in the terminal. The same goes for sample2 scripts.

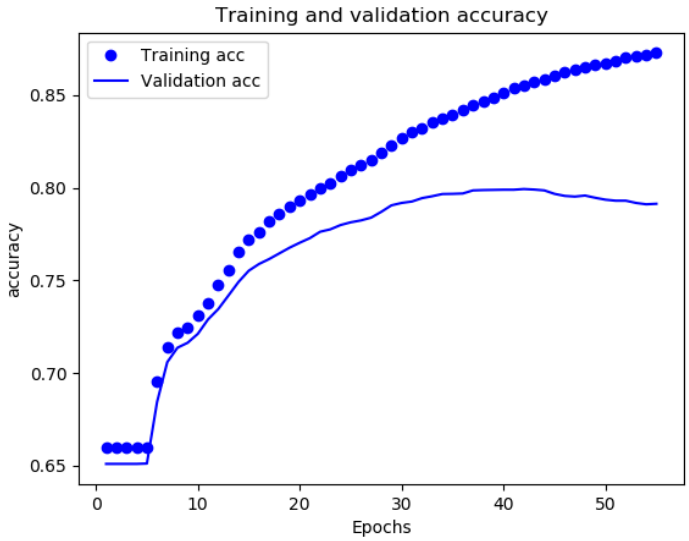
4 Results and discussion

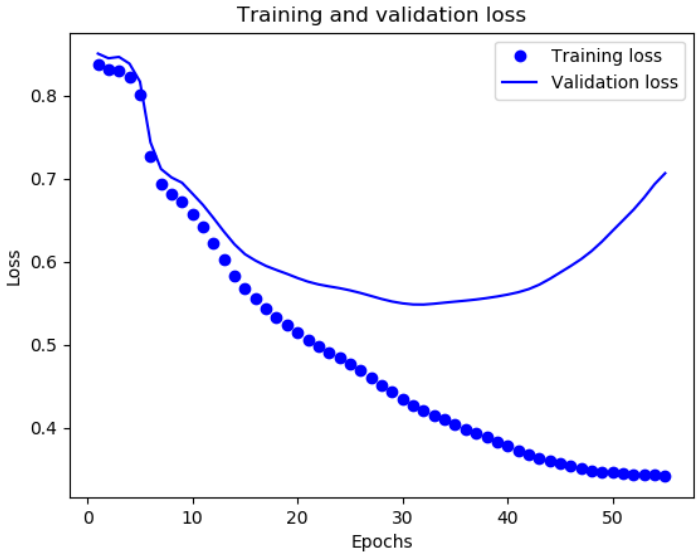
dataset 1:

The following graph shows the loss and accuracy sores for a number of epochs on training and validation. As we can see, around 32 epochs the graph shows overfitting. Therefore: the final model stops at 32 epochs. I assign random weights to the forward pass. I put training data into a forward pass. I have accuracy is 9.63%. After doing a neural network, the test accuracy is 80.56%.

The model that I use:

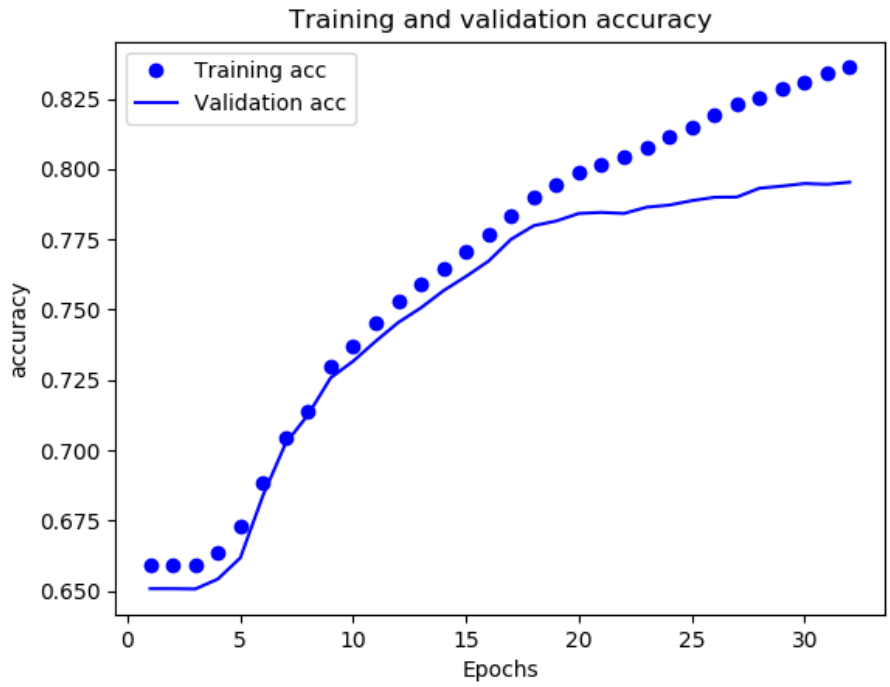
Layer name	Shape of output array	Activation function	Batch size
dense	512	sigmoid	32
dense	256	sigmoid	32
dense	3	SoftMax	32
output		categorical cross entropy	

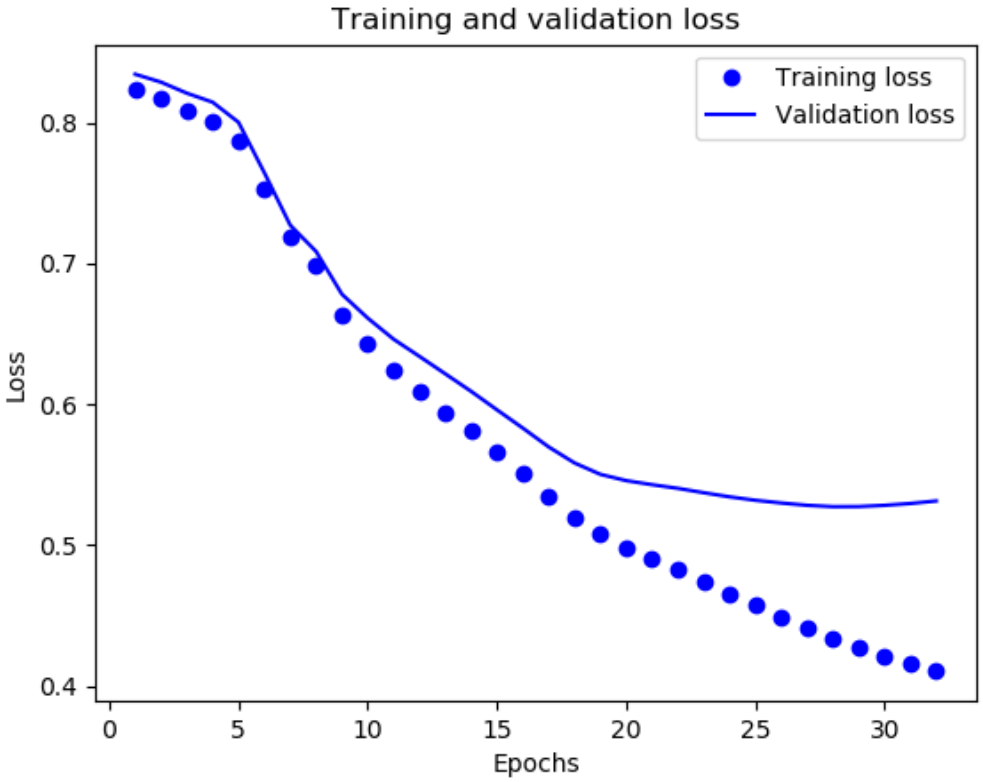




The following graph shows the loss and accuracy sores for number of epochs on training and validation after 32 epochs

.


Training and validation accuracy
Training acc
Validation acc
accuracy
0.825
0.800
0.775
0.750
0.725
0.700
0.675
0.650
0
5
10
15
20
25
30
Epochs

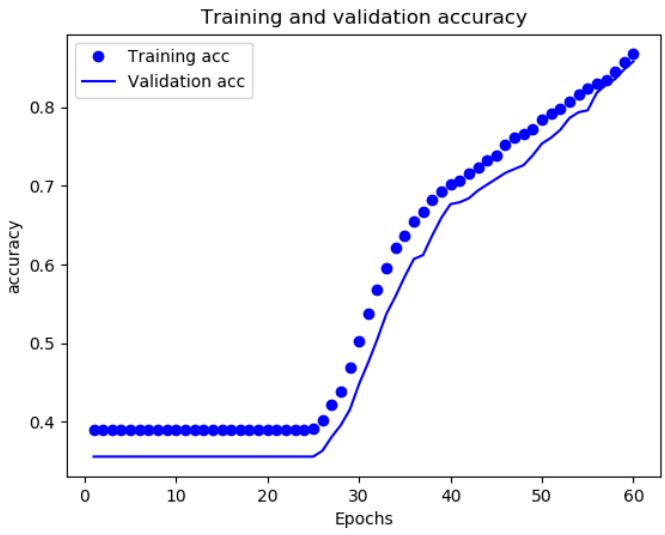

Training and validation loss
Training loss
Validation loss
Loss
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
30
Epochs

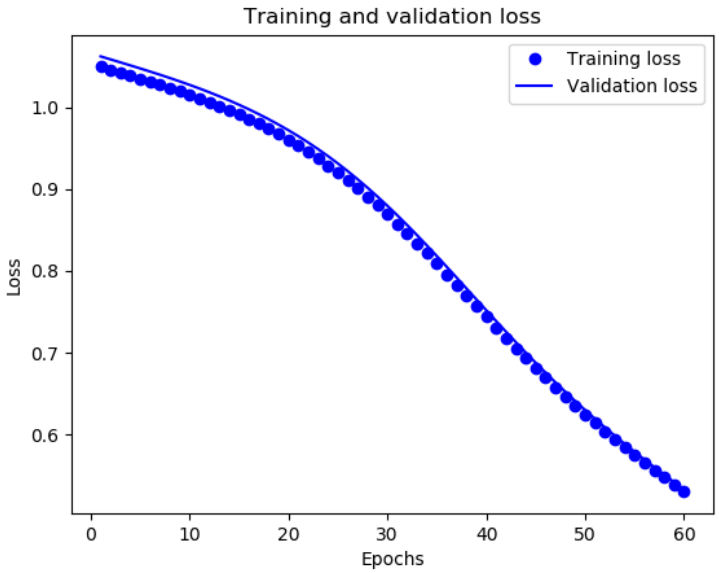
dataset 2:

The model that I use:

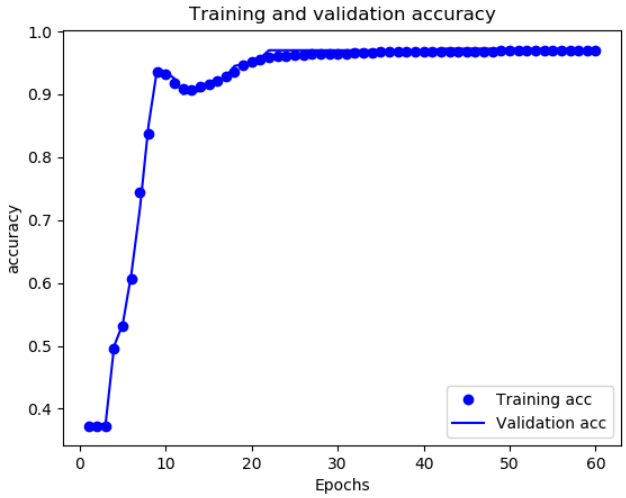
Layer name	Shape of output array	Activation function	Batch size
dense	512	sigmoid	16
dense	256	sigmoid	16
dense	3	SoftMax	16
output		categorical cross entropy	

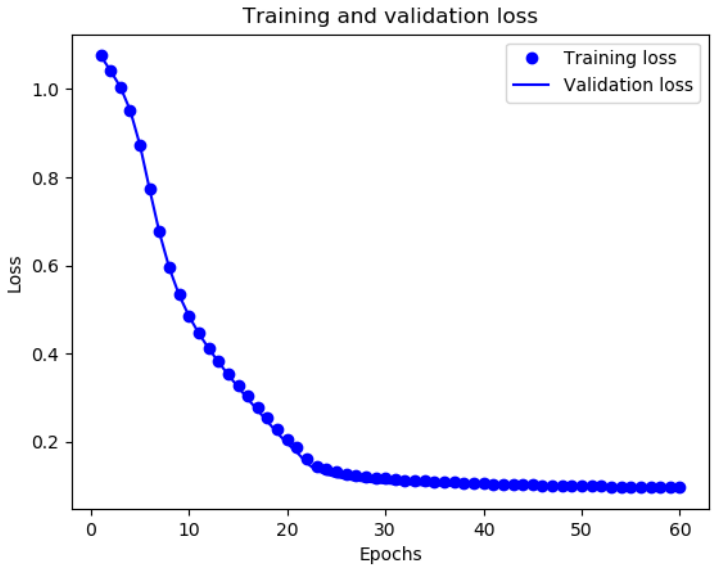
The following graph shows the loss and accuracy scores for the number of epochs on training and validation. From the graph, after 60 epochs, we know the learning rate is too small. The learning rate is 0.01.



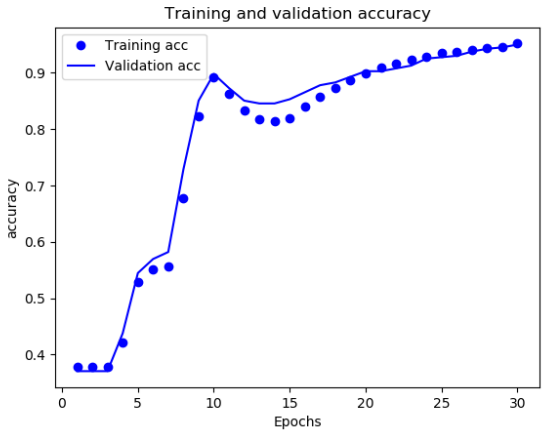


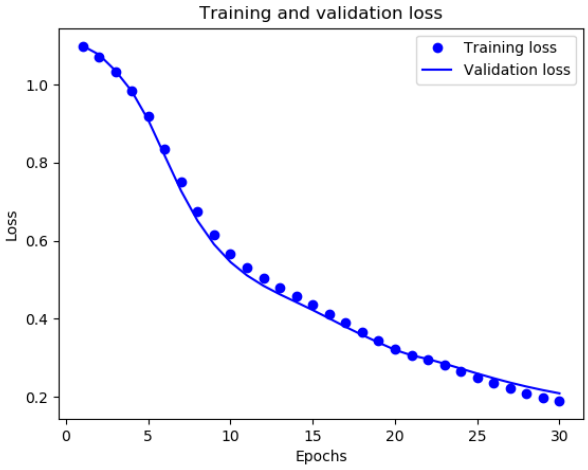
I change the learning rate to 0.09. we have the following graph.
As we can see, the learning rate works pretty well this time. We can stop training around 30 epochs.





The following graph shows the final results.





I assign random weights to the forward pass. I put training data into the forward pass. the accuracy is 18.89%. After doing a neural network, the test accuracy is 97.22%.

5 References

https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f
https://gombru.github.io/2018/05/23/cross_entropy_loss/
http://cs231n.github.io/optimization-2/
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.reset_index.html
http://cs231n.github.io/neural-networks-case-study/