ROB313 Assignment 4 – Neural Networks

Objective

The objective of this assignment is to test various neural network configurations and observe the performance on the classification dataset *mnist_small.* The dataset contains 28x28 pixel images of digits that we are to classify appropriately as a digit between 0-9. The python package *autograd* was used to implement the neural network. We also explored metrics such as negative log-likelihood and accuracy to evaluate the model on all training, validation, and testing sets.

Code structure and strategy

The code was all written in one file, labelled *a4.py*. This file includes all functions provided in the starter code file *a4_mod.py*. The functions that were provided are:

- *forward_pass():* this function completes a forward pass iteration of the neural network. It was modified to include a logarithmic-softmax output function for the 10-neuron output layer.
- *negative-log_likelihood():* this function computes the neg. log-likelihood of any datapoints x compared to actual results provided in y. This function was modified to assume a categorical likelihood given conditional log probabilities.
- *nll_gradients():* computes the gradients of one training iteration of the neural network.

The main function written for this assignment is *neural_network().* This function has arguments that allow it to handle all problems in this assignment that require training a neural network. Provided a mini-batch size, a number of iterations, and a list of possible learning rates, it optimizes the weights of a 4-layer neural network and is capable of outputting training neg. log-likelihood per iteration, validation accuracy and neg. log-likelihood, and test neg. log-likelihood. There also exist parameters *visual,* which creates visualizations of the first layer weights, and *digit,* which creates figures for the digits that are most difficult to classify.

Other modularized helper functions including *Xavier_init()* and *generate_accuracy()* were essential to producing the results for the assignment. More information about the code can be found in the README.md file.

Q3. Neural network validation and training performance

While implementing a 4-layer neural network with 100 neurons in the hidden layers, we measured the training neg. log-likelihood and the validation neg. log-likelihood for each iteration. These results were plotted for each learning rate (0.0001 and 0.0005) over 3000 iterations.

In figures 1 and 2, we can see that the validation error makes a much smoother curve than the training error plot. This is because the training error is computed based on the randomly sampled mini-batch of size 250, whereas the validation error is computed using the full-batch method. We observe the expected trajectory of the validation neg. log-likelihood that is expected: the training error continuously decreases but the validation error reaches a minimum and then slightly increases as the model begins overfitting. The algorithm in the function *neural_network()* also finds the minimum value of the validation error and returns the iteration number which it occurs at. The values can be seen in table 1, and the trends can be observed in figures 1 and 2.

Table 1, NN results for 100 hidden neurons

Learning rate	Total num. iterations	Optimal validation neg. log-likelihood	Num optimal iteration
0.0001	4000	177.90	3067
0.0005	4000	179.67	340

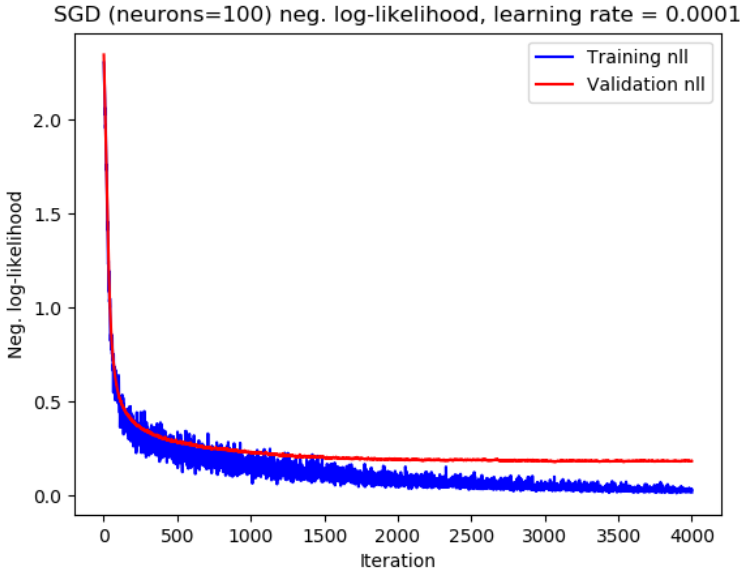


Figure 1, Training and validation neg. log-likelihood (normalized) for learning rate 0.0001, 100 hidden neurons

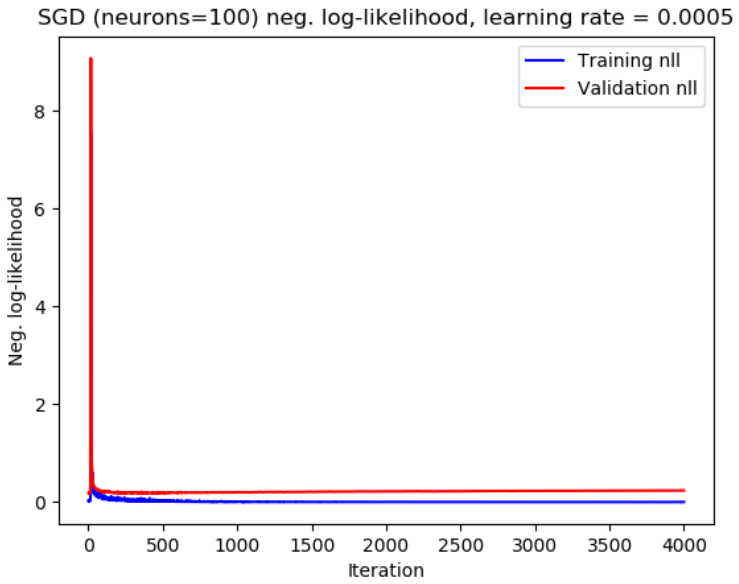


Figure 2, Training and validation neg. log-likelihood (normalized) for learning rate 0.0005, 100 hidden neurons

Q4. Neural network with various hidden layer sizes

For this experiment, we trained the neural network in the same manner as above, however this time we tested the accuracy as well as the negative log-likelihood for both validation and testing sets. We created models with various hidden layer sizes over multiple learning rates with a large number of iterations. For most models, we found that the large number of iterations was excessive, and that we could use early stopping to select an optimal iteration number.

Note that when we changed the number of neurons on the hidden layers, we use the negative log-likelihood as the primary metric for the optimal number of iterations (i.e. where we can early-stop). It is at this point that we calculate the validation and test accuracy as well as the negative log-likelihood on the test set.

Table 2, NN results for 10 hidden neurons, 4000 iterations

Learning rate	Optimal iteration num.	Validation NLL	Validation accuracy	Test NLL	Test accuracy
0.0001	1967	285.02	91.2 %	264.39	92.9 %
0.0005	734	289.31	91.1 %	268.21	92.3 %
0.001	1	488.29	90.1 %	411.46	91.9 %
Preferred learning rate:					**0.0001**

Table 3, NN results for 120 hidden neurons, 3000 iterations

Learning rate	Optimal iteration num.	Validation NLL	Validation accuracy	Test NLL	Test accuracy
0.0001	2891	175.67	94.6 %	151.12	94.9 %
0.0005	358	162.93	94.9 %	146.17	95.1 %
0.001	78	196.82	95.6 %	149.16	95.8 %
Preferred learning rate:					**0.0005**

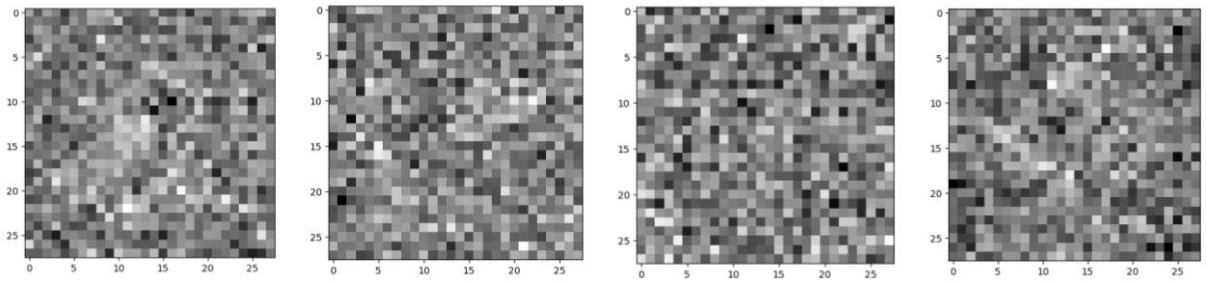
Table 4, NN results for 200 hidden neurons, 3000 iterations

Learning rate	Optimal iteration num.	Validation NLL	Validation accuracy	Test NLL	Test accuracy
0.0001	2907	172.32	94.4 %	150.40	95.8 %
0.0005	545	172.29	94.9 %	139.39	96.1 %
0.001	41	186.09	95.6 %	160.99	96.3 %
Preferred learning rate:					**0.0005**

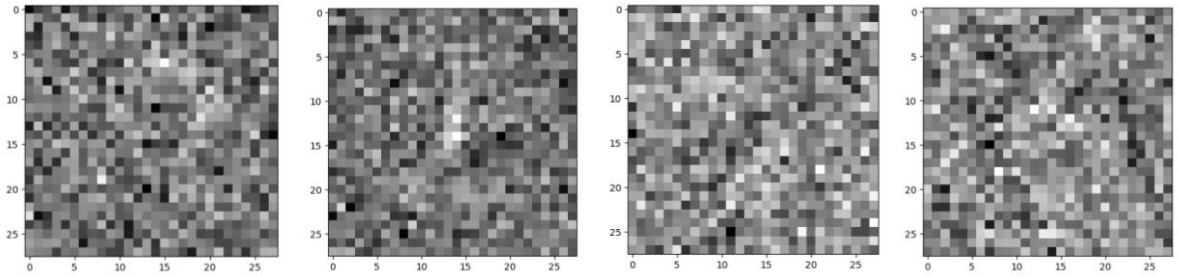
The preferred learning rates selected in tables 2, 3, and 4 were chosen based upon a trade-off between the number of iterations required to converge and the resulting test negative log-likelihood. The larger learning rate for 10 hidden neurons did not converge except randomly at the first iteration. Thus, the fewer neurons we have in our inner layers, the smaller our learning rates must be, because the model must be trained slower in attempt to reduce the amount of fluctuation in the training mini-batch results. Having a larger amount of hidden layer neurons also impacts the convergence properties – having more hidden neurons allows the neural network to converge very quickly at high learning rates. Additionally, as the learning rates increase, the generalization of the model (i.e. performance on data it has never seen before) tends to become better. This is the general case, but for some of the data presented above, this trend cannot be seen exactly because the model is trained with a random mini-batch subset of training points.

Q5. Visualization of the weights

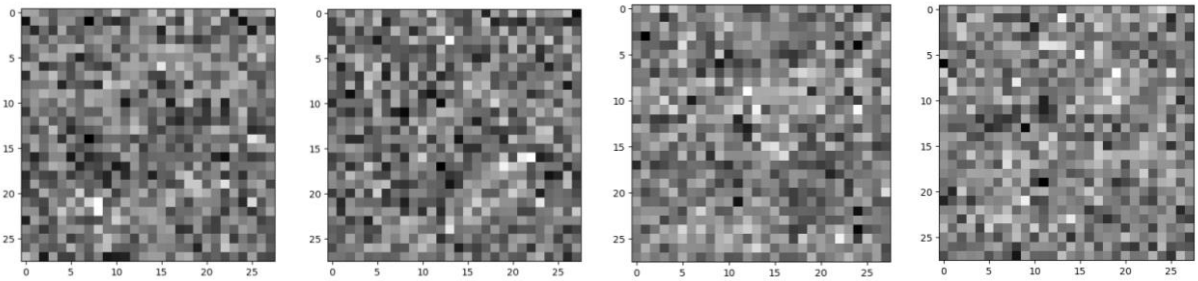
For this question, the neural network was trained with 100 hidden layer neurons, and thereafter we plotted 16 sets of weight vectors from the first hidden layer at random. Since the neural network is fully connected, the figures are the weight vectors transformed into 28x28 images on grayscale representing the weights (black is smaller weight, white is larger) from the input image into a specific neuron in the first hidden layer. Each neuron represents a feature of the image; thus, the weights represent the weighting of the feature in each of the flattened 784 pixels of the digit image. As we pass through 10,000 training photos, the weights must function for each classification (i.e. all digits 0-9), thus you can see some of the weights forming into what seems to resemble a hand-drawn digit. Each neuron contains specific weights such that it can help classify one or many of the digits.



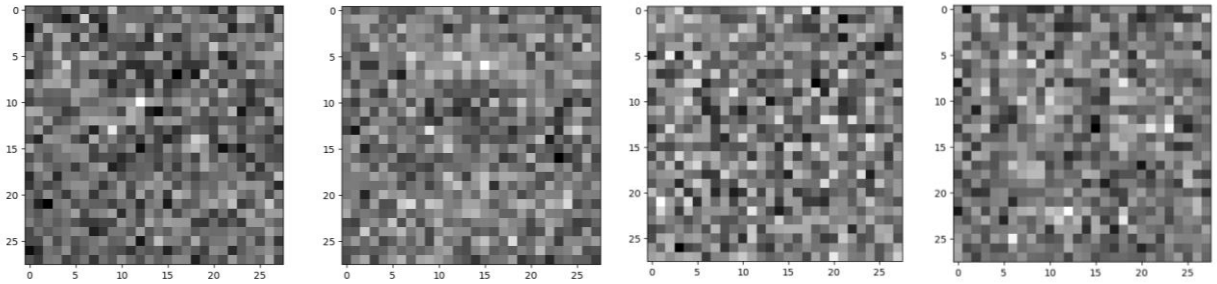
Figures 3-6, Weights of neurons 3, 7, 16, and 21



Figures 7-10, Weights of neurons 36, 51, 54, and 57



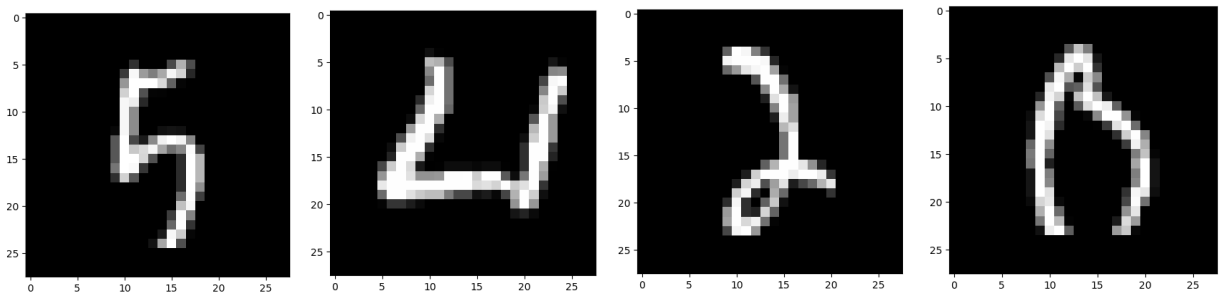
Figures 11-14, Weights of neurons 66, 79, 82, and 84



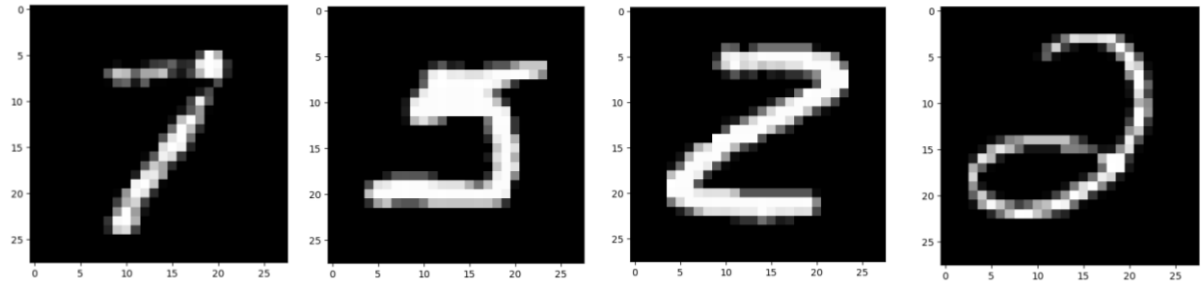
Figures 15-18, Weights of neurons 93, 95, 97, and 99

Q6. Difficulties classifying digits

Here we are plotting the 10 images of digits that are the most difficult for the neural network to classify (i.e. have the lowest class conditional probabilities). Some of the trends that we notice within these images is that they are slanted, or the connecting lines are unclear. There are multiple "2"'s included in this set of photos, which is expected because there are many different variations of drawing a number 2. Other than that, you can see a "4" with a vertical bar missing (figure 20) and a digit that a human could not even classify (figure 22).



Figures 19-22, Difficult images to classify (rank 1, 2, 3, and 4)



Figures 23-26, Difficult images to classify (rank 5, 6, 7, and 8)

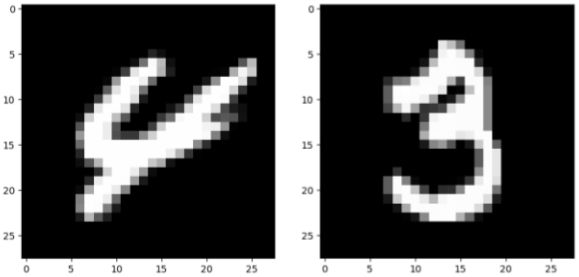


Figure 27-28, Difficult images to classify (rank 9, and 10)