Lab 3: Neural Networks

Data Set

We have covered Neural Networks and Gradient Descent in class. Now we will create a network to classify handwritten digits from the MNIST database (http://yann.lecun.com/exdb/mnist/). There are 10 classes in the data that are handwritten numbers 0 through 9. This is a historically significant dataset.

Problem

You will need to submit your report. The report should include results, using different performance metrics to analyze the results. Also, you need to discuss your ideas and conclusions about the results. (e.g. You can say why a classifier is better or worse than another)

1. Use Keras (https://keras.io/) to load the mnist database into train and test sets.
2. Prepare data for use in a neural network deep learning model. Think about what shape the data needs to be in for a neural network. The default shape for a MNIST image is 28 x 28 pixels. Normalize the pixel data to be between 0 and 1.
3. Create a sequential model neural network that has 2 hidden layers that are 32 neurons wide. The final shape should be: Input, 32, 32, 10. This will be your baseline model. This network should use the Stochastic Gradient Descent optimizer. It should use the ReLU activation on each layer except the last one which needs to be a softmax. The loss function should be categorical crossentropy and the metric to use should be accuracy. Batch size should be 64 and epochs should be 20. You will use the testing dataset as your validation data for the model in the fit command. The accuracy you will be asked about for the rest of the lab is the validation accuracy. You will also compare every future model against this one.
4. Now create another network that has the shape: Input, 64, 64, 10. How does increasing the number of neurons per hidden state affect accuracy and training speed? Does it make sense to increase the number of neurons in each layer? If so, why and by how much?
5. Now create a network with the shape: Input, 32, 32, 10 again. This time adjust the SGD optimizer and set the learning rate to 0.01, the decay rate to 0.000001, and momentum rate to 0.9. How does this model compare to the two before it in terms of accuracy and computation speed? Is using momentum in our models a good idea?
6. Now create a network with the shape: Input, 32, 32, 10. This time we will adjust the batch size. Run one model with a batch size of 128 and again with a batch size of 32. How does batch size effect computation speed and accuracy? Why do you think that is?
7. Now we are going to add another hidden layer. Create a network with the shape: Input, 32, 32, 32, 10. How did adding another layer effect computation time and accuracy? Why do you think that is?
8. Now create a network with the shape: Input, 128, 128, 10. This time we are going to introduce a dropout (https://keras.io/layers/core/). After the input and both hidden layers add a dropout layer. Set the dropout value to 0.5. How does this effect accuracy and computation time? Think

about what dropout is doing to our network. Does it make sense to use dropout and how many neurons are being used in each layer during training? Does this effect training speed?

9. Think about all of the networks you have made thus far. Combine the positive aspects of each network before and create the ‘best’ network you can. Give your accuracy and parameters of your network. Why did you chose the network that you did?
10. Now evaluate your models. List out the parameters that made the most amount of change from the baseline model in order from most to least change and give your reasoning for why that is and how we can apply them to our networks.