Assignment 5: Neural Networks

Commented [1]: Looks good to me, but it would be great that you review it one more time.

Commented [2R1]: Ok, going over it again now

For this assignment, you will conduct programming tasks in Jupyter Notebook and answer provided questions in a Google Doc or MS Word document. You will submit **2** documents to Blackboard:

1. A PDF file converted from your Jupyter Notebook.
2. A PDF file converted from your Assignment 5.1 Template Word Doc (document is linked in Blackboard assignment prompt).
    - This template has a copy of all short answer questions for this assignment.

Dataset: MNIST

The MNIST dataset is a set of images of handwritten digits from 0-9. They are a series of small images that have been aligned and scaled to be a similar size and orientation. I recommend loading the data using Keras if possible https://keras.io/api/datasets/mnist/.

Scikit-learn has another digits dataset built in, using `sklearn.datasets.load_digits`, but this is not MNIST, and this dataset is smaller.

Written Questions

(short answer, 2-3 sentences):

Why do the initial weights for a neural network need to be small and random?

Commented [3]: @akovacs@sandiego.edu @etarshizi@sandiego.edu - this is ready for review

Small and randomized weights allow for small, incremental gradient descent that doesn't overshoot the local minima and prevents us from overfitting early on by having the weights be too large.

Why do the initial weights for the perceptron NOT need to be small and random?
Gradient descent for a perceptron is convex and therefore guaranteed to find a global minimum if the data is linearly separable.

Why are neural networks able to learn more than a perceptron can?
The presence of a hidden layer and hidden units allows for complex decision boundaries, such as one that could solve XOR.

When you trained the neural network on the “two moons” dataset, you plotted the shape of the decision boundary as you increased the number of hidden units. What seems to be the best number of hidden units for this problem? How does the shape of the boundary change as you increase the number of hidden units?

As the number of hidden units increases (to a point), the decision boundary becomes more complex. The optimal number of units is 7+/-1, as it correctly classifies the largest number of points.

These next questions reference plots you will make in the MNIST section. Complete those plots, and then complete these written questions.

When you change the number of hidden units, does the network train faster? Is the output error affected by changing this parameter? What is the best number of hidden units for this problem?
Smaller amounts of hidden units require less computation complexity.
For sigmoid with lbfgs, 20 hidden units had the best performance.
Increasing the number of hidden units can, to a point, improve test set performance.

When you change the solver, does the network train better? What is the best solver for this problem?
Adam had the fastest training time, but the worst performance. In general, lbfgs was the most accurate solver.

When you change the activation function, does the network train better? What is the best activation function for this problem?
Logistic/Sigmoid had the best overall performance with an accuracy of >89%.