Nikhil Chakravarthy

Problem 1 – Warm Start Transfer Learning

Evaluate the effects of “Warm Start” when changing the data trained on.

Problem description:

I have taken images from the MNIST database representing handwritten digits 0 through 9, labeled as the digit the writer defined it as. I first train a multilayer perceptron neural network to classify these images. Then, using the parameter weights of this network as a starting parameter set, I train a new network on a separate set of images (warm start). These images are “shifted”, meaning that I shifted the flattened image array (originally 28x28, now 1x784) by 100 values to “muddy” the data. Then lastly, I train a network from scratch on this shifted data. I then test all of these networks on 2 test data sets. One is untouched, and one is shifted like the training sets for the last 2 networks. I will refer to the original network trained on unchanged data as “base”, “warm” will refer to the network trained on the shifted data but with initial parameters set to the “base” trained parameters, and “treat” will refer to the network trained on the shifted data from scratch.

All networks have 1 hidden layer of 8 nodes, as I found that additional hidden layers reduced performance and increased convergence time, and layers with more than 8 nodes reduced performance.

The purpose of this is to test the transfer learning method of “warm start”, which should theoretically improve training time and performance over both domains of data.

Experiment:

2400 images are taken as the training set and 400 images are taken as the test set. Then, each are split into equal sets of 1200 and 200 respectively. The first sets of 1200 and 200 are the training and test sets of the data that will not be changed. The second sets of 1200 and 200 are the training and test sets of the data that will be shifted by 100 values. The shift is then applied to the second sets. The last step of preprocessing this data is to create a “scaler” based on the unchanged training set, then fit all the sets to that “scaler”. I found this helps overall network performance. Then, the first network “base” is created and trained on the untouched training set, and evaluated against the regular and shifted test sets. The second network “warm” is created by training “base” on the shifted data. Then “warm” is evaluated on the regular and shifted test sets. Lastly, “treat” is created and trained on the shifted data. All performance metrics are stored, including number of iterations to converge to a final network (to objectively test speed of convergence). This is repeated 5 times, and the performance metrics are averaged

Results (found in prob1/results/mnist_2400_.csv):

These 3 values describe the accuracies of each network on the test set of unchanged data. Oddly enough, the network trained over the unchanged data performed the worst. However, it is notable that the warm start network performed the best overall by a significant margin.

base_acc_reg: 0.795
warm_acc_reg: 0.8550000000000001
treat_acc_reg: 0.8149999999999998

These 3 values describe the accuracies of each network on the test set of shifted data. The “base” network somehow performed better on the shifted set than the unshifted set. However, the warm start network again performed the best overall.

base_acc_treat: 0.805
warm_acc_treat: 0.86
treat_acc_treat: 0.835

These 3 values describe the iteration counts needed for each network to converge from its initial weight parameters to parameters than minimize loss efficiently. Theoretically, the warm start network should have the lowest iterations since its initial parameters should already somewhat match the layout of the data. However, in my experiment, the base and treat networks required around the same number of iterations while the warm start network needed almost 6 times more iterations to converge.

base_iter: 2150.0
warm_iter: 11731.0
treat_iter: 2309.0

Overall, using warm start did result in better performance on both the unshifted and shifted domains of MNIST images. However, the warm start network did take far longer to converge to its final parameters, meaning warm start actually hindered converge time. This could be due to the shift of 100 values being so large as to actually require vastly different parameters to correctly map the network’s outputs to the labels. From this, I can conclude that warm start is useful in creating a network that has a larger domain of things it can accurately map to labels (both shifted and unshifted images were classified at high accuracies, higher than both the base and treat networks).

Used “sklearn” to create the Multilayer Perceptron, so special thanks to:

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.