CMSC25400 — Homework 6

University of Chicago, Winter 2017 — due Thursday, March 9 at 11:59pm

In this assignment you will code up a simple feed-forward neural network (NN) and train it by backpropagation to solve the full 10-digit MNIST handwritten digit classification task.

1. We recommend that (at least for your initial implementation) you use a neural net with three layers of neurons:
   (a) An input layer ($\ell = 0$) consisting of $785 = 28 \times 28 + 1$ neurons, of which the ouput of neuron $0 \leq i \leq 784$ is just the value of pixel i in the image, while the last neuron is a bias neuron whose output is always 1.
   (b) A hidden layer ($\ell = 1$) consisting of N_{hidden} neurons plus a single bias neuron;
   (c) An output layer ($\ell = 2$) consisting of 10 neurons corresponding to the 10 classes.
2. Use a fully connected NN with the sigmoid transfer function, i.e. the output of neuron number i in layers $\ell = 1, 2$ should be $[\mathbf{x}_\ell]_i = \phi(\mathbf{x}_{\ell-1} \cdot \mathbf{w}_{\ell,i})$ where $\mathbf{x}_\ell$ is the vector of all the outputs in layer ℓ and $\phi(u) = (1 + e^{-u})^{-1}$. Note that this notation already suggests that you might be able to implement your network with vector and matrix operations.
3. For the outputs use squared error loss, i.e., if the correct label of a given example is $\widehat{y}$, and the output of the output layer is $\mathbf{x}_2$, then the loss should be $\mathcal{E} = \| \mathbf{x}_2 - \boldsymbol{e}_{\widehat{y}} \|^2$, where $\boldsymbol{e}_{\widehat{y}}$ is the $(\widehat{y} + 1)$'th standard unit basis vector, e.g., for $\boldsymbol{e}_3 = (0, 0, 0, 1, 0, 0, 0, 0, 0, 0)^\top$.
4. Train your neural network example-by-example with stochastic gradient descent, as discussed in class, with a fixed learning rate η. You will need to cycle through the training data multiple times (multiple epochs) and stop when error on the holdout set starts increasing substantially. The network should be started with random weights.
5. Training data is provided in the csv files `TrainDigitX.csv.gz` and `TrainDigitY.csv.gz` (if you use Python, the `loadtxt` function can automatically decompress these files). The test files are similarly named.
6. Experiment with varying the learning rate η and the number of hidden units, say $N_{\text{hidden}} \in \{32, 64, 128, 256\}$. For extra credit, also experiment with increasing the number of hidden layers, and changing the transfer function in the output layer to soft-max. If you are really ambitious, you can also try coding up a convolutional neural net (for a comparison for how well different algorithms work for this data see `http://yann.lecun.com/exdb/mnist/`).
7. For this assignment you may use matrix libraries, but please do not use a neural network library or somebody else's implementation: the goal of the assigment is to give you the experience of coding up a neural network "from scratch".

Your writeup for this assignment should include the following:

1. A short description of your code and what choices you made during the implementation.
2. A study of how performance varies as a function of η, N_{hidden} and the number of epochs. Try and optimize these parameters for the best performance on a hold-out set. Include plots for the error rate vs. each of these three parameters (with the other two set to reasonable values) on the test set.

In addition to your writeup please hand in the following:

1. Your full code in a form that the TAs can easily run it on the data if they want to verify it.
2. Your predictions on the two test sets TestDigitX and TestDigitX2 (for the second one we do not publish reference labels).