Neural Network for MNIST Digit Recognition

April 26, 2016

Lei Kang

This document is derived from my Homework solution to UC Berkeley CS289A (Spring 2016). MNIST dataset also comes from that class.

Problem Statement

1. You will be using a hidden layer of size 200. Let $n_{in} = 784$, the number of features for the digits class. Let $n_{hid} = 200$, the size of the hidden layer. Finally, let $n_{out} = 10$, the number of classes. Then, you will have $n_{in} + 1$ units in the input layer, $n_{hid} + 1$ units in the hidden layer, and n_{out} units in the output layer.

   The input and hidden layers have one additional unit which always takes a value of 1 to represent bias. The output layer size is set to the number of classes. Each label will have to be transformed to a vector of length 10 which has a single 1 in the position of the true class and 0 everywhere else.

2. The parameters of this model are the following:

   - V, a n_{hid}-by-$(n_{in} + 1)$ matrix where the (i, j)-entry represents the weight connecting the j-th unit in the input layer to the i-th unit in the hidden layer. The i-th row of V represents the ensemble of weights feeding into the i-th hidden unit. Note: there is an additional row for weights connecting the bias term to each unit in the hidden layer.

   - W, a n_{out}-by-$(n_{hid} + 1)$ matrix where the (i, j)-entry represents the weight connecting the j-th unit in the hidden layer to the i-th unit in the output layer. The i-th row of W represents the ensemble of weights feeding into the i-th output unit. Note: again there is an additional row for weights connecting the bias term to each unit in the output layer.

3. You will be expected to train and run your neural network using both mean-squared error and cross-entropy error as your loss function (one network for each loss function). Suppose y is the ground truth label (using the same 1-of-n_{out} encoding as stated previously in point 1) and $z(x)$ is a vector containing each value of the units in the output layer given the feature vector x. Then, the mean-squared error is

$$J = \frac{1}{2}\sum_{k=1}^{n_{out}}(y_k - z_k(x))^2$$

   The cross-entropy error is given as

$$J = -\sum_{k=1}^{n_{out}}[y_k \ln z_k(x) + (1 - y_k)\ln(1 - z_k(x))]$$

4. All hidden units should use the tanh activation function as the choice of non-linear function and the output units should use the sigmoid function as its choice. Remember the sigmoid function is given as

$$g(z) = \frac{1}{1 + e^{-z}}$$

5. You will be using stochastic gradient descent to update your weights.

Derivation of stochastic gradient descent updates for V and W

Setting: for a single observation (x, y), let's derive the gradient updates in a matrix form:

V: 200×785

W: 10×201

x: 785×1

h = tanh[Vx]: 200×1 (hidden layer, tanh is tanh function)

h_1= [1;h]: 201×1 (hidden layer with bias term)

z = s(Wh_1): 10×1 (output layer, s is sigmoid function)

y: 10×1

For MSE loss function

$\nabla_W J = ((z-y)\times(1-z)\times z)h_1^T$: 10×201

$\nabla_V J = (W[:,1:]^T((z-y)\times(1-z)\times z)(1-h^2))x^T$: 200×785

where $W[:,1:]$is the weights connecting hidden layer and output layer without considering bias term, its dimension is 10×200

For cross entropy loss function

$\nabla_W J = (z-y)h_1^T$: 10×201

$\nabla_V J = (W[:,1:]^T(z-y)(1-h^2))x^T$: 200×785

Update:

$W = W - \eta \nabla_W L$

$V = V - \eta \nabla_V L$

To be more specific, let's look at output unit *i*, then for mean squared loss function

$$\nabla_{W_i} J = \frac{\partial J}{\partial z_i} \nabla_{W_i} z_i = \frac{\partial J}{\partial z_i} s(W_i h_1)(1 - s(W_i h_1)) h_1^T$$
$$= 2(s(W_i h_1) - y_i) s(W_i h_1)(1 - s(W_i h_1)) h_1^T$$

where h_1=[1; tanh(Vx)] which is 201 × 1

Then for hidden unit *i*:

$$\nabla_{V_i} J = \frac{\partial J}{\partial h_i} \nabla_{V_i} h_i = \frac{\partial J}{\partial h_i} (1 - \tanh^2(V_i x)) x$$
$$= [\sum_{j=1}^{k} W_{ji} s(W_j h)(1 - s(W_j h)) \times 2(s(W_j h) - y_j)](1 - \tanh^2(V_i x)) x$$

where h = tanh(Vx) which is 200 × 1

Now, considering cross entropy loss function, we only need to change $\frac{\partial J}{\partial z_i}$ and $\frac{\partial J}{\partial h_i}$ in the above derivation. Similar as MSE, gradients for cross-entropy loss can be written as:

$$\nabla_{W_i} J = \frac{\partial J}{\partial z_i} \nabla_{W_i} z_i = \frac{\partial J}{\partial z_i} s(W_i h_1)(1 - s(W_i h_1)) h_1^T$$
$$= (s(W_i h_1) - y_i) h_1^T$$

$$\nabla_{V_i} J = \frac{\partial J}{\partial h_i} \nabla_{V_i} h_i = \frac{\partial J}{\partial h_i} (1 - \tanh^2(V_i x)) x$$
$$= [\sum_{j=1}^{k} W_{ji} (s(W_j h) - y_j)](1 - \tanh^2(V_i x)) x$$