Machine Learning Exercise 11

Shun-Lung Chang, Anna Ruby

XOR Classification Problem

The figure below depicts the architecture of the neural network we used in this problem. First, we added an all 1's vector as a bias term to the original training dataset. Then, in the single hidden layer we used, we have two neurons, no bias term, and a sigmoid function as the activation function. In the output layer we have two neurons with linear activation functions to process values from the hidden layer. Finally, a softmax function (with alpha equal to 1) is used to squash our output values into a hypothesis vector, which we used to make predictions. The weights between layers were randomly initialized using the following uniform distribution,

$$Unif(-\frac{1}{\sqrt{N}},-\frac{1}{\sqrt{N}}), \text{where N is the number of neurons in input layer.}$$

During the back-propagation procedure, we used 0.04 as the learning rate for both the output and hidden layer. The parameter settings are summarised in table 1.

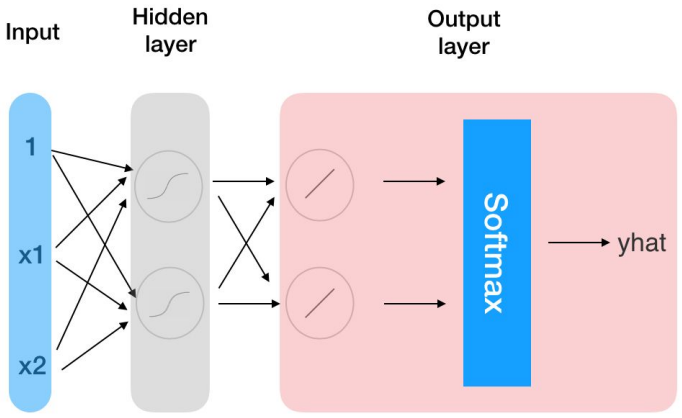

Figure 1: Architecture of the Neural Network for XOR problem.

Table 1: Parameter Setting of the Neural Network for XOR problem.

	Input	**Hidden Layer**	**Output Layer**
Number of Neurons	3	2	2
Activation Function	-	sigmoid	linear and softmax
Weights Initialization	-	$Unif(-\frac{1}{\sqrt{3}},-\frac{1}{\sqrt{3}})$	$Unif(-\frac{1}{\sqrt{2}},-\frac{1}{\sqrt{2}})$
Learning Rate	-	0.04	0.04

The learning curve of this network is shown below, where error is computed using the mean square error of the hypothesis vector and true labels. With our learning rate the error rate plumented to 0.07 after 3800th epoch, and then decreased with a infinitesimal change each epoch until 54844th epoch at which point the labels were all correctly predicted.

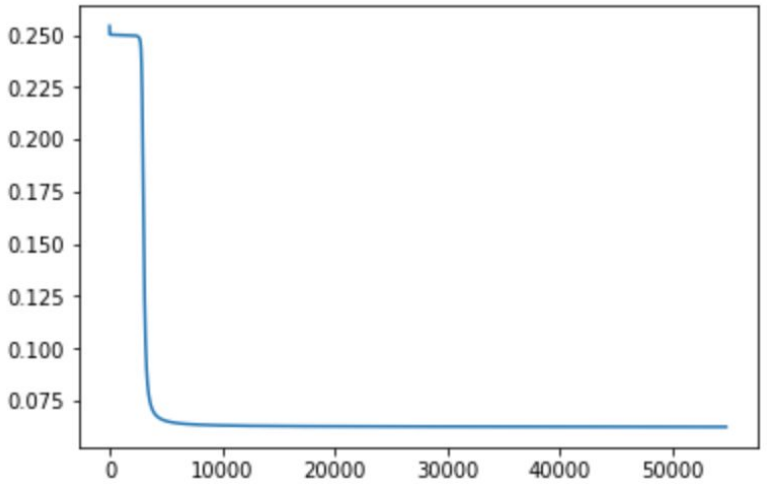

Figure 2: Mean Square Error against the number of epochs.

Digits Classification (Bonus Task)

In training a multi-layer perceptron classifier on the digits dataset, we have created a network with three hidden layers. For each hidden layer we have used hyperbolic tangent (i.e. tanh) as the activation function, with 256 neurons in the first two layers and 128 neurons in the third. The structure of the output layer is almost the same as we set in the XOR classification task, but here we have 10 neurons, instead of 2, as we wish to predict ten different classes in the digits dataset. The weights are again generated from a uniform distribution that we used in previous task. In order to prevent overfitting, random gaussian noise (with mean 0 and standard deviation 0.05) was added to the input values in each hidden layer. In the back propagation procedure all the learning rates were set to 0.01. Table 2 summarizes the parameter settings.

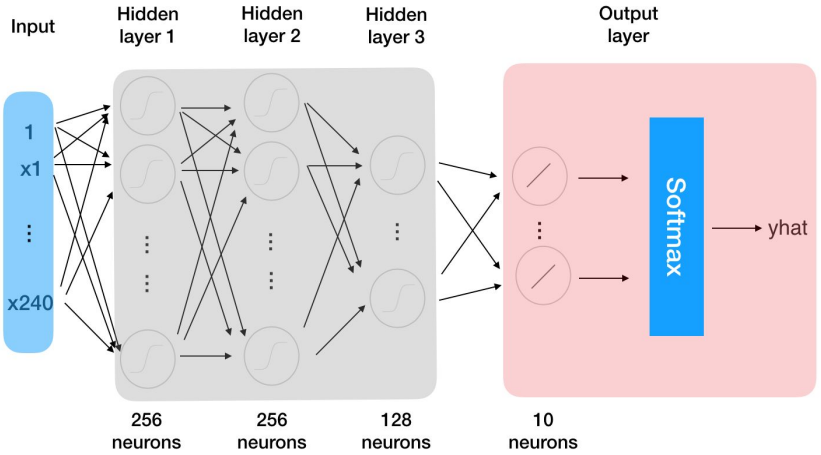

Figure 3: Architecture of the Neural Network for Digit Classification.

Table 2: Parameter Setting of the Neural Network for Digit Classification Problem

	Input	**Hidden Layer 1 and 2**	**Hidden Layer 3**	**Output Layer**
Number of Neurons	241	256	128	10
Activation Function	-	tanh	tanh	linear and softmax
Weights Initialization	-	Layer 1: $Unif(-\frac{1}{\sqrt{241}}, \frac{1}{\sqrt{241}})$ Layer 2: $Unif(-\frac{1}{\sqrt{256}}, \frac{1}{\sqrt{256}})$	$Unif(-\frac{1}{\sqrt{256}}, \frac{1}{\sqrt{256}})$	$Unif(-\frac{1}{\sqrt{128}}, \frac{1}{\sqrt{128}})$
Learning Rate	-	0.01	0.01	0.01
Random Noises in Input Values	-	gaussian(0, 0.05)	gaussian(0, 0.05)	-

Before being used in the learning task, the features in the dataset were normalized, and, in order to find a better number of epochs, we applied cross-validation. The training set (comprising the first 100 images from each class in the digits dataset) was divided into two sets of equal size. The first 50 images in each class became the training set and the latter 50 images the validation set. As can been seen in the figure below, the misclassification rates for validation set did not change notably after approximately 700 epochs, and the training error reached 0 at 941st epoch. In light of the results, we decided to pick a number between 700 and 900 when predicting test dataset. By trial and error as well, we finally set the number of epochs to 800. As such, using the aforementioned parameter settings, we produced a misclassification rate for the test dataset of 0.055.

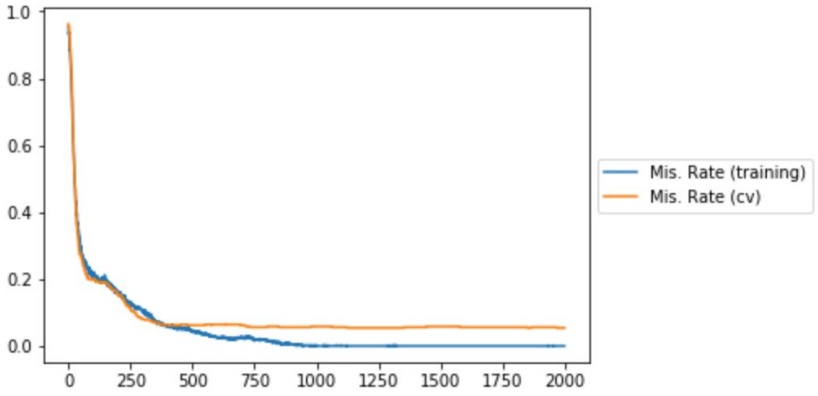

Figure 4: Misclassification Rates (Training and cross-validation) against the number of epochs.

In this task, we chose the hyperbolic tangent function as our activation function as it usually converges faster than the standard sigmoid function, as Lecun et al. (1988) have pointed out. However, the hyperbolic tangent function converges too fast such that the training error could easily reach 0, running the risk of overfitting. To prevent this situation, we added gaussian noise during training. This trick indeed slowed the speed of convergence during the training procedure and also incrementally improved performance in cross-validation.

Reference

[1] LeCun Y., Bottou L., Orr G.B., Müller K.R. (1998) Efficient BackProp. In: Orr G.B., Müller KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 1524. Springer, Berlin, Heidelberg