CS 615 - Deep Learning

Assignment 2 - Artificial Neural Networks
Spring 2020
Xiangang Lai

1 Theory

1. Another common activation function is the hyperbolic tangent function, *tanh*, which is defined as:

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

What is $\frac{\partial tanh(x\theta)}{\partial \theta_j}$ (5pts)?

$$\begin{aligned}
\frac{\partial tanh(x\theta)}{\partial \theta_j} &= \frac{\partial tanh(z)}{\partial z}\frac{\partial z}{\partial \theta_j} \\
&= \frac{(e^z + e^{-z}) * (e^z + e^{-z}) - (e^z - e^{-z})^2}{(e^z + e^{-z})^2} x_j \\
&= \frac{(e^{2z} + e^{-2z}) - (e^{2z} + e^{-2z} - 2)}{(e^z + e^{-z})^2} x_j \\
&= \frac{2}{(e^z + e^{-z})^2} x_j \\
&= \frac{2}{(e^{x\theta} + e^{-x\theta})^2} x_j
\end{aligned}$$

2. Given a network with a single hidden layer, a softmax activation function at the output, a linear activation at the hidden layer, and a cross-entropy objective function what is:

   (a) $\frac{\partial J}{\partial \theta_{kj}}$ (5pts)
   Given

$$J = -\sum_{i=1}^{K} y_i \ln(\hat{y}_i) = \ln(\hat{y}_a)$$

$$\frac{\partial J}{\partial g(net_{O_k})} = -\frac{1}{\hat{y}_a}$$

$$\begin{aligned}
\frac{\partial g(net_{O_k})}{\partial net_{O_k}} &= \begin{cases} \frac{e^{h\theta_{:,a}}\left(\sum_{k=1}^{K} e^{h\theta_{:,i}}\right) - e^{2h\theta_{:,a}}}{\left(\sum_{k=1}^{K} e^{h\theta_{:,a}}\right)^2} = \hat{y}_a(1 - \hat{y}_a) & k = a \\ \frac{-e^{h\theta_{:,k}} e^{h\theta_{:,a}}}{\left(\sum_{k=1}^{K} e^{h\theta_{:,a}}\right)^2} = \hat{y}_a \hat{y}_k & k \neq a \end{cases} \\
&= \hat{y}_a(y_k - \hat{y}_k)
\end{aligned}$$

$$\frac{\partial net_{O_k}}{\partial \theta_{kj}} = h_j$$

Therefore

$$\frac{\partial J}{\partial \theta_{kj}} = \frac{\partial J}{\partial \hat{y}_a} \frac{\partial \hat{y}_a}{\partial net_{O_k}} \frac{\partial net_{O_k}}{\partial \theta_{kj}} = -(y_k - \hat{y}_k) h_j$$

(b) $\frac{\partial J}{\partial \beta_{ij}}$ (5pts)

$$\begin{aligned}
\frac{\partial J}{\partial \beta_{ij}} &= -\sum_{i=1}^{K} \left(\frac{\partial J}{\partial \hat{y}_a} \frac{\partial \hat{y}_a}{\partial net_{O_k}} \frac{\partial net_{O_k}}{\partial g\left(net_{h_j}\right)} \frac{\partial g\left(net_{h_j}\right)}{\partial net_{h_j}} \frac{\partial net_{h_j}}{\partial \beta_{ij}} \right) \\
&= -\sum_{i=1}^{K} -(y_k - \hat{y}_k)\theta_{jk} * 1 * x_i \\
&= +x_i \sum_{i=1}^{K} (y_k - \hat{y}_k)\theta_{jk}
\end{aligned}$$

2 Shallow Artificial Neural Networks

Hyperparameters You'll also have to experiment with different hyper-parameters. These include:

1. How to initialize your model (i.e the initial values of θ.
   Generate rand numbers with seed 61

2. Your learning rate.
   0.75

3. Your regularization amount (use L2 regularization).
   The coefficient is 0.008

Results

1. A graph of the average log likelihood for the training and testing sets as a function of the training iteration number.

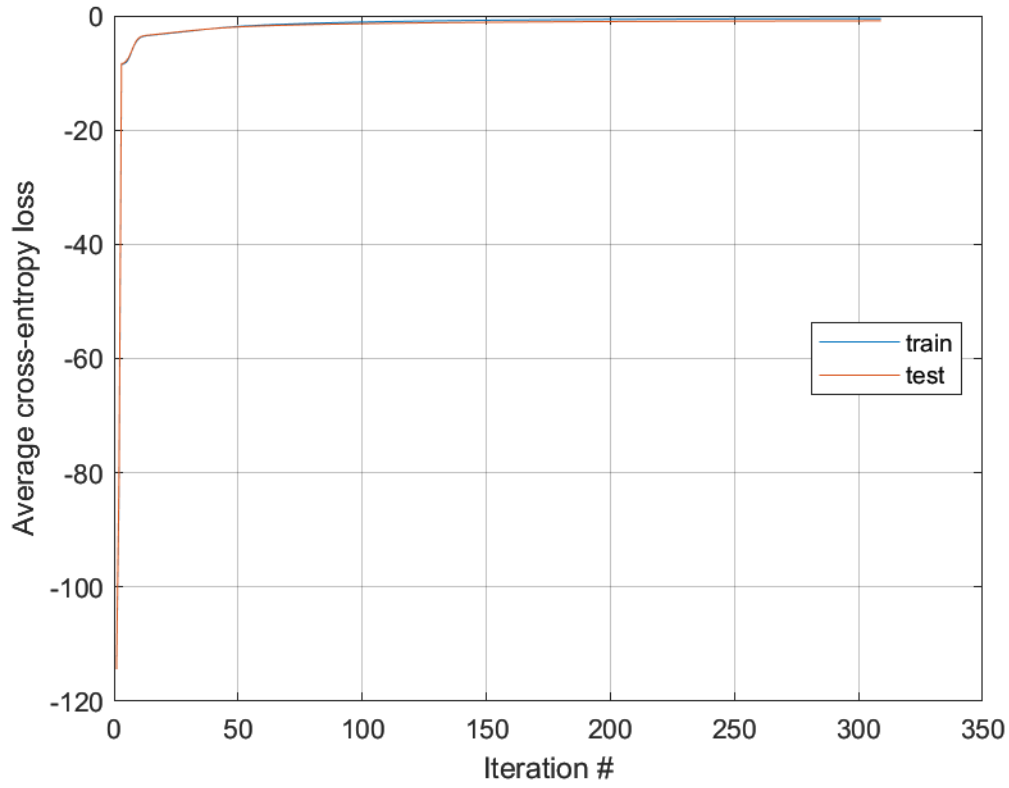

Figure 1: Gradient Descent Progress

2. The testing accuracy.
   0.95 for test dataset and 1 for train dataset

3. The confusion matrix for the testing data.

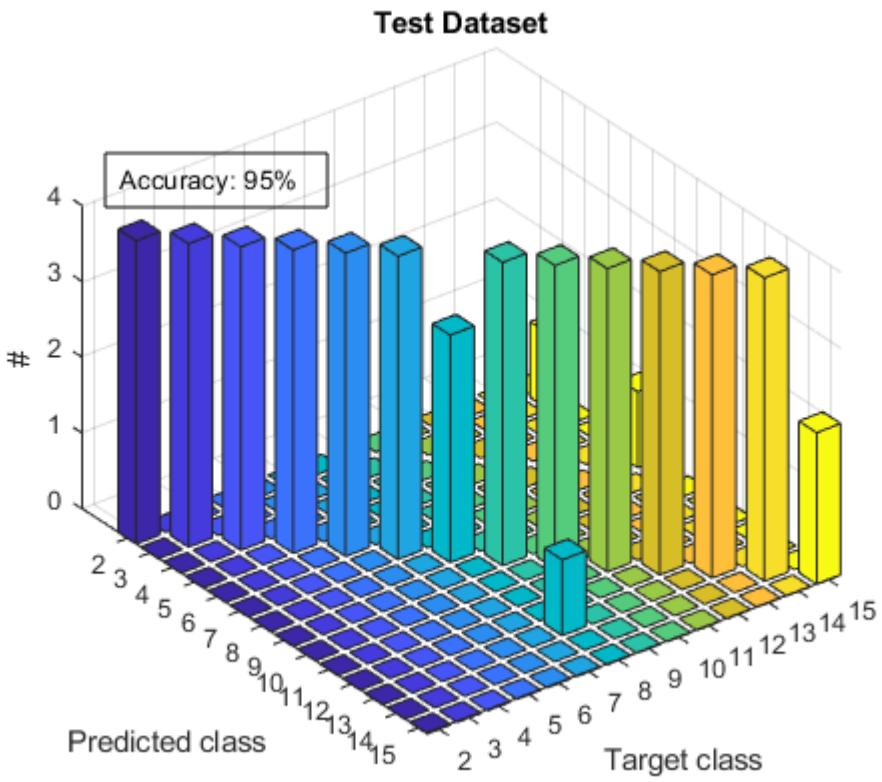

Figure 2: Confusion Matrix of test dataset

3 Multi-Layer ANN

1. The values of the hyperparameters that you chose.
   Learning rate: 0.75
   L2 Regulariation Coefficient: 0.0002
   Random seed: 120

2. A table of network configurations and their associated number of iterations till termination, training and testing accuracies.

Hidden Layer Vector	Number of Iterations	Train Accuracy	Test Accuracy
[500,20,15]	255	0.591836734693878	0.500000000000000
[800,50]	1055	1	0.892857142857143
[600,50]	1042	1	0.928571428571429
[50,10]	623	0.948979591836735	0.821428571428571
[50,30,10]	224	0.602040816326531	0.553571428571429
[500,20,15]	81	0.306122448979592	0.303571428571429

Two hidden layers typically generate good results while having three layers, the selection of the hyperparameters are more tricky