Implements the backpropagation algorithm for neural networks and apply it to the task of hand-written digit recognition.

1. Training Data

There are 5000 training examples in trainingdata.mat (or training\_x.txt & training\_y.txt), where each training example is a 20 pixel by 20 pixel grayscale image of the digit. Each pixel is represented by a floating point number indicating the grayscale intensity at that location. The 20 by 20 grid of pixels is “unrolled” into a 400-dimensional vector. Each of these training examples becomes a single row in our data matrix X. This gives us a 5000 by 400 matrix X where every row is a training example for a handwritten digit image. The second part of the training set is a 5000-dimensional vector y that contains labels for the training set. To make things more compatible with Octave/Matlab indexing, where there is no zero index, the digit zero is mapped to the value ten. Therefore, a “0” digit is labeled as “10”, while the digits “1” to “9” are labeled as “1” to “9” in their natural order.



Figure 1. Examples from the dataset

X(5000 x 400) has the form:

Y(5000 x 1) has the form:

2. Model representation

Our neural network is shown in Figure 2. It has 3 layers – an input layer, a hidden layer and an output layer. Recall that our inputs are pixel values of digit images. Since the images are of size 20 × 20, this gives us 400 input layer units (not counting the extra bias unit which always outputs +1). There are 25 units in the second layer and 10 output units (corresponding to the 10 digit classes)

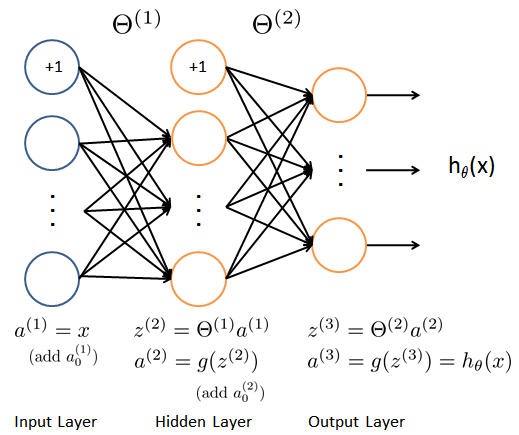


Figure 2: Neural network model

3. Regularized cost function

The cost function for neural network with regularization is given by

: computed as shown in the Figure 2

K: the total number of possible labels (K = 10)

The original labels were 1, 2, …, 10, for the purpose of training a neural network, we need to recode the labels as vectors containing only values 0 or 1, so that

For example, if x(i) is an image of the digit 5, then the corresponding y(i) (that you should use with the cost function) should be a 10-dimensional vector with y5 = 1, and the other elements equal to 0.

m: the total number of training examples

λ: 1

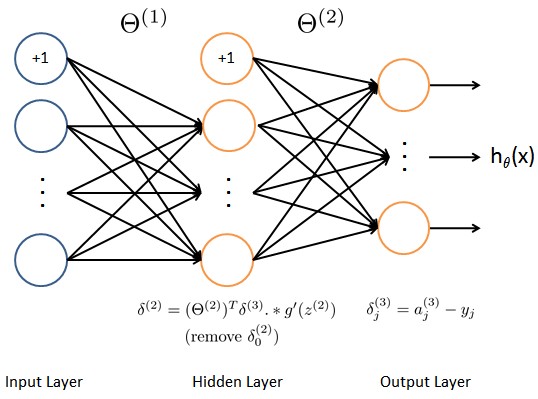
4. Random initialization

While training neural networks, it is important to randomly initialize the parameters for symmetry breaking. One effective strategy for random initialization is to randomly select values for ϴ(l) uniformly in the range {-ϵinit, ϵinit}. You should use ϵinit = 0.12. This range of values ensures that the parameters are kept small and makes the learning more efficient.

(, )

|  |
| --- |
|  |

5. Backpropagation algorithm



PseudoCode:

|  |
| --- |
| Load Training set {X y}: {(x(1), y(1)), …, (x(m), y(m))}  Set  For i = 1 to m  1. Perform forward propagation to compute for l = 2, 3, 4, …, L  Set          2. Using , compute      remove the bias row  3. Update Delta Value      end for loop |

6. Gradient descent

Now we have the cost function and partial derivative of the cost function with respect to each of our parameters, we can use gradient descent algorithm to find out parameters theta that minimize the cost function.

Recall gradient descent:

|  |
| --- |
| Repeat until convergence {  Run Backpropagation Algorithm    } |

7. Prediction & Calculate training accuracy

What the code does

1. Follows the instruction to implement the backpropagation algorithm for neural networks and apply it to the task of hand-written digit recognition.
2. Calculates the training set accuracy (the percentage of examples it got correct). The Training accuracy should be greater than 90%, the cost should be less than 1.0

Reference:

https://www.youtube.com/watch?v=18X68kLAfKY&index=51&list=PLZ9qNFMHZ-A4rycgrgOYma6zxF4BZGGPW