Project 1 - Image Segmentation

Ankit Arya

September 25, 2012

ECE-8493 - Introduction to Neural Networks.

Taught by Dr. Jenny Du.

1 Technical Description

Note- This description is put together using "`http://ufldl.stanford.edu/wiki/index.php/Neural_Networks`" as a major resource. The equations present in the resource also guide my implementation of neural networks.

A neural network is constructed by connecting neurons to one another, such that output of a neuron can be the input of another. For example, here is a small neural network:

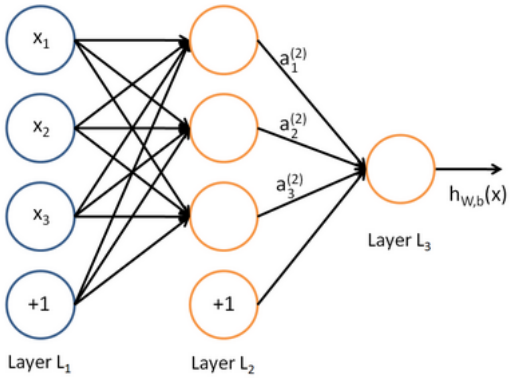

Figure 1: Neural Network with 3 input units, 3 hidden units, and one output.

1.1 Forward Propogation

x_1, x_2, x_3..... are the inputs. Weights are defined as$W$, where W_{ij}^{l} denotes the parameter (or weight) associated with the connection between unit j in layer l, and unit i in layer l + 1. We will write $a_i^{(l)}$ to denote the activation (meaning output value) of unit i in layer l. For l = 1, we also use $a_i^{(l)} = x_i$ to denote the $i-th$ input. Given a fixed setting of the parameters W,b, neural network defines a hypothesis $h_{W,b}(x)$ that outputs a real number. Also lets $z_i^{(l)}$ denote the total weighted sum of inputs to unit i in layer l, including the bias term. $f(.)$ is the sigmoid activation function. Equations of $forward\ propogation$ can be represented compactly as:

$$z^{(l+1)} = W^{(l)}x + b^{(1)} \tag{1}$$

$$a^{(l+1)} = f(z^{(l+1)}) \tag{2}$$

1.2 Backpropopogation Algorithm

Given a training set of m examples, the overall cost function can be defined as:

$$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)})\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2$$
$$= \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{W,b}(x^{(i)}) - y^{(i)}\right\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2$$

The goal is to minimize the above stated cost function using batch gradient descent or other advanced optimization algorithm.

The weight decay parameter λ controls the relative importance of the two terms. Note also the slightly overloaded notation: $J(W,b;x,y)$ is the squared error cost with respect to a single example; $J(W,b)$ is the overall cost function, which includes the weight decay term.

Step by step process algorithm is shown below:

1. Perform a feedforward pass, computing the activations for layers , , up to the output layer , using the equations defining the forward propagation steps.

2. For the output layer (layer n_l), set
   $\delta^{n_l} = -(y - a^{(n_l)}) . * f'(z^{(l)})$

3. For $l = n_l - 1.n_l - 2, ...$
   Set
   $\delta^l = ((W^l)^T\delta^{(l+1)}) . * f'(z^{(l)})$

4. Compute the desired partial derivates:
   $\nabla_{W^l} J(W,b;x,y) = \delta^{(l+1)}(a^l)^T$,
   $\nabla_{b^l} J(W,b;x,y) = \delta^{(l+1)}$

2 Neural net Implementation

Project is executed by running the file *main.m*

Steps to train the network:

1. Specify parameters used:

   input_layer_size =3 (total of 900 training samples)

   hiddlen_layer_size =3

   num_labels= 3 (3 assigned classes: flowers, leaves, background)

2. Randomly intialize Weights to train the network - $Theata1 and Theta2$

3. Set value for max_iteration and the regularization parameter *lambda*. Value used:

   $max\_iter = 300$

   $lambda = 1$

   neural net stops when number of iterations are completed.

4. Training the network includes call to the nnCostFunction using sigmoid activation, which includes implementation of the feed forward and the backpropogation algorithm . The functions returns the optimized value of the weights which are used for prediciton.

5. Once Theta1 and Theta2 are returned, function in predict.m is called.

   $[p, soft\_class] = PREDICT(Theta1, Theta2, testData)$ outputs the predicted label of testData given the trained weights of a neural network $(Theta1, Theta2)$

   p is the hard classification result, where as soft_class contains the soft classification map.

NOTE - Learning rate is not chosen manually, as instead of implementing Gradient Descent I have used an advanced optimizer $fmincg$. These advanced optimizers are able to train our cost/loss functions efficiently as long as we provide them with the gradient computations.

3 Results

Image Segmentations results are shown as follows:

Figure 2: Orignal Image

(a) Flowers (b) Leaves (c) Background

Figure 3: Image Segmentation

For hard classification and and soft classification maps please see the section below.

4 Matlab Code

Matlab code is attached as a tarball with email. Main.m is the file that is used to run the program. Resulting segmentations are stored as flower,

learves, and background (all PNG files). Hard and soft classification maps are stored as hard_map.mat and soft_map.mat.

Bibliography

1. Ng, Andrew, and Jiquan Ngiam. "UFLDL Tutorial - Ufldl." Unsupervised Feature Learning and Deep Learning. N.p., n.d. Web. 25 Sept. 2012. "http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial"