Neural Network Assignment 5 Report

Sreeraj Rimmalapudi

Algorithm 1 Neural Network Function Approximation

function MAIN
    Initialize a uniformly chosen random weight vector **w** of size 3N + 1 where N is the number of hidden neurons.
    Let $x_1..x_n$ be the training samples
    Let $d_1..d_n$ be the desired output
    for $epoch = 1$ to 100000 **do**
        for each training sample x_i **do**
            Perform forward propagation and get output y_i
            Loss $= (d_i - y_i)^2$
            Perform back propagation to get gradients with respect to **w**
            Update the weights using gradient descent

function FORWARD PROPAGATION
    Let x be the input sample
    Let **w1** be $w_1..w_N$, **b1** be $w_{N+1}..w_{2N}$
    Let **w2** be $w_{2N+1}..w_{3N}$, b2 be w_{3N+1}
    hiddenout $= \tanh(\mathbf{w1} * x + \mathbf{b1})$; where **hiddenout** (output of the hidden neurons) is a vector of size N
    $y = \sum_{n=1}^{N}(\text{hiddenout}_i * w2_i) + \text{b2}$
    Store **w2**,x, **hiddenout**, y for backprop

function BACK PROPAGATION
    Get **w2**,x **hiddenout**, y from forward prop
    Let d be the desired sample output
    $cost\_derivative = 2 * (y - d)$
    $delta2 = cost\_derivative * 1$
    gradw2 = **hiddenout** * $delta2$
    $gradb2 = delta2$
    $\mathbf{delta1} = \mathbf{w2} * \text{delta2} \odot (1 - \mathbf{hiddenout}^2)$ (derivative of tanh function)
    gradw1 = x * **delta1**
    gradb1 = **delta1**
    Append **gradw1**, **gradb1**,**gradw2**, **gradb2** to form vector **gradw**

function GRADIENT DESCENT
    Let η be the learning rate
    for each w_i in **w** **do**
        $w_i = w_i - \eta * \text{grad}w_i$