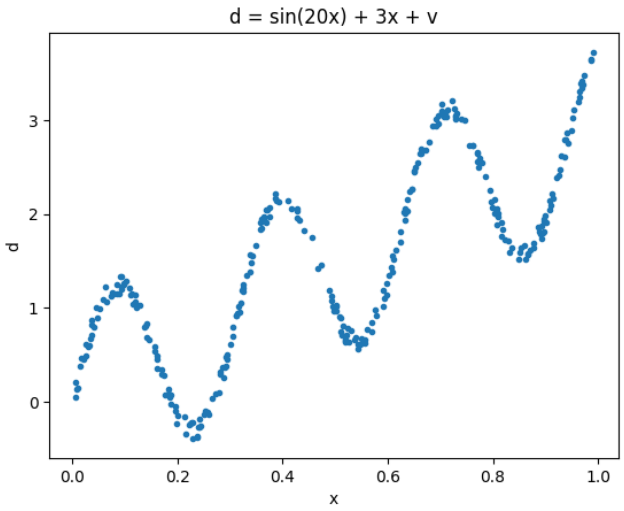
CS 559 Neural Networks – Homework 4

Zohair Hashmi | UIN: 668913771 | zhashm4@uic.edu

NOTE:

- Libraries used in python code: Numpy, Matplotlib.pyplot
- Random Seed set to 42.

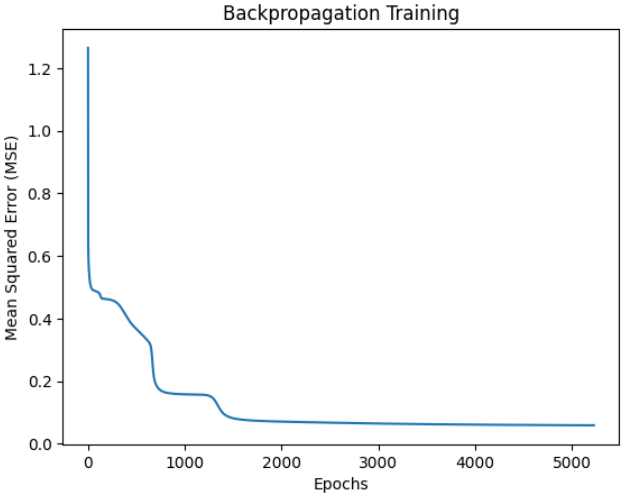
For $n = 300, x \in [0,1], v \in \left[-\frac{1}{10}, \frac{1}{10}\right]$ and $d = sin(20x_i) + 3x_i + v_i$, we get the following plot:



For our neural network, weights and biases initialized for $input \rightarrow hidden$ & $hidden \rightarrow output$ layers, in the following manner. Hence having 3N + 1 weights including biases.

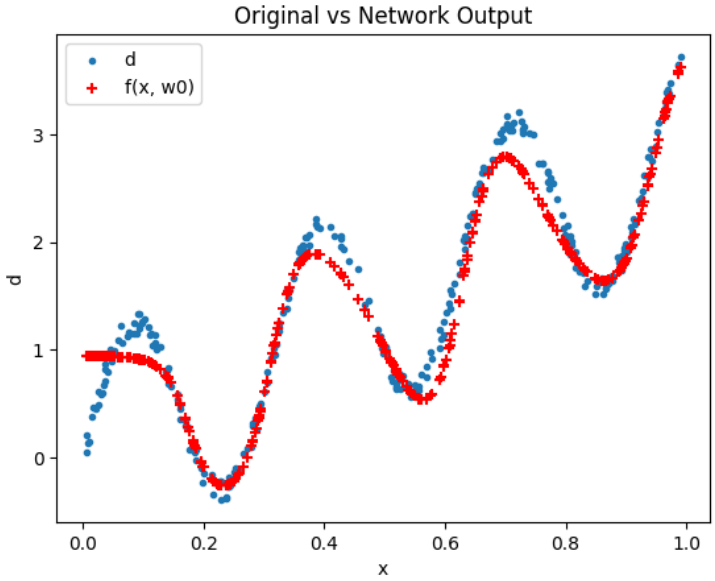
- weights_input_hidden → shape (1x24)
- weights_hidden_output → shape (24x1)
- bias_hidden → shape (1x24)
- bias_output → shape (1x1)

Using backpropagation, the neural network is trained and weights and biases are updated after each epoch, resulting in the following relationship of number of epochs and mean square error:



The final **Mean Square Error** achieved by our Neural Network is **0.059116**, where our network converges, considering a very minimal change in the error.

The final weights achieved are then used to estimate the value of $\boldsymbol{f(x, w_0)}$ and compared it with the original data points, in the following curve.



As we can observe, the weights obtained offer a good fit (red marks) to the given curve (blue marks), showing that neural network has trained efficiently using backpropagation algorithm.

Backpropagation Algorithm:

The following equations for gradient used in the code are derived using the formula:

$$\frac{\partial \mathrm{E}}{\partial \mathrm{w}} = -(\text{signal before w in Feed Forward Graph}) * (\text{signal before w in Feed Backward Graph})$$

$$\frac{\partial E}{\partial W0} = -\ x\ *\ error_{hidden}$$
$$\frac{\partial E}{\partial W1} = -\ output_{layer2}\ .error_{output}$$
$$\frac{\partial E}{\partial b0} = -\ error_{hidden}$$
$$\frac{\partial E}{\partial b1} = -\ error_{output}$$

Note: In the code the plus sign in the updating weights section caters for the negative sign in the above equation.

PSEUDOCODE

$n \leftarrow 300\ , x \leftarrow [x_1, \dots, x_n]\ , v \leftarrow [v_1, \dots, v_n]$
$d \leftarrow sin(20x) + 3x + v$
$weights_{inputHidden} \leftarrow random\_array(1, 24)$
$weights_{hiddenOutput} \leftarrow random\_array\ (24, 1)$
$bias_{hidden} \leftarrow zeros(1, 24)$
$bias_{output} \leftarrow zeros(1, 1)$

$Epochs \leftarrow 10000$
$\eta(Learning\ Rate) \leftarrow 0.05$

$mse\_history \leftarrow list[\,]$

$W0 \leftarrow copy(weights_{inputHidden})$
$W1 \leftarrow copy(weights_{hiddenOutput})$
$b0 \leftarrow copy(bias_{hidden})$
$b1 \leftarrow copy(bias_{output})$

For each Epoch:
$mse \leftarrow 0$
For each input in x:
Forward Pass
$output_{layer1} = x * weights_{inputHidden} + bias_{hidden}$
$output_{layer2} = tanh\,(output_{layer1})$
$network\_output = linear\,(layer2\_ouput * weights_{hiddenOutput} + bias_{output})$

Mean Square Error
$mse \leftarrow mse + \sum_{i=1}^{|S|}(d_i - f(x_i, w)^2$

Backward Pass
$error_{output} \leftarrow d - output_{network}$
$error_{hidden} \leftarrow error_{output} * W1 * (1 - tanh(Layer1output^2)$

Update Weights
$W1 = W1 + \eta * output_{layer2}\,.error_{output}$
$W0 = W0 + \eta * x * error_{hidden}$
$bias_{output} = bias_{output} + \eta * error_{output}$
$bias_{hidden} = bias_{hidden} + \eta * error_{hidden}$

$mse = mse/n$
$mse\_history.append(mse)$

 # Modify learning rate if MSE is increasing
If $epoch > 0$ and $mse\_history[-1] > mse\_history[-2]$:
$learning\_rate = learning\_rate * 0.9$

 # Stop if change in MSE is small
if $epoch > 0$ and $\left|\left|mse_{history[-1]} - mse_{history[-2]}\right|\right| < 1 * 10^{-6}$:
Print "Converged at epoch {epoch}, MSE: {mse}"
break