Class 19

October 24, 2022

Contents

Recap: Gradient Descent 1
Neural Networks 1
GD for NN 2
J vs epoch graph 3
Three ways to GD 4

Recap: Gradient Descent

- $y = w^T x$
- $J(w) = 1/2(\hat{y} - y)^2$ and

$$\frac{\partial J}{\partial w_i} = -(\hat{y} - y)x_i$$

- GD

$$w_i^{t+1} = w_i^t + \eta(\hat{y} - y)x_i$$

Now I want to apply this on a neural network. But it's more complex since there are many neurons.

Neural Networks

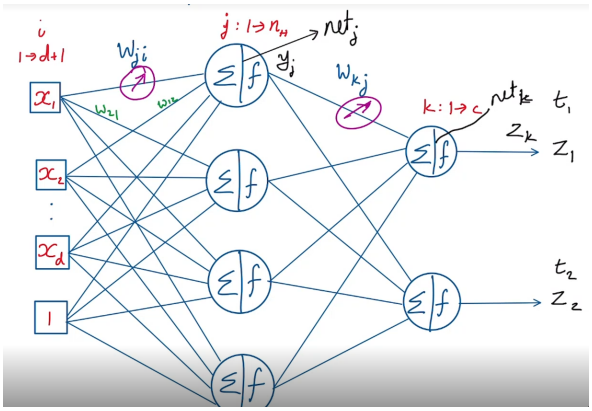

Figure 1: Neural Network Structure

First layer:

- i
- d dimensions
- $x_1 - x_d$ and 1 (for the constant in the linear sum).
- $w_{ji} :=$ Weight from i^{th} neuron to the j^{th} neuron in the hidden layer

Let's take one hidden layer:

- j
- j from 1 to n_H where n_H is the number of neurons in the hidden layer
- Output of the hidden layer after $\sum := net_j$.
- Output of the hidden layer after $\sum$ and $f := y_j$.
- Similarly we have w_{kj} here.

Final Layer:

- k
- k from 1 to c (where $|c|$ is the number of classes being classified by this neural network)
- Output of the hidden layer after $\sum := net_k$.
- Output of the hidden layer after $\sum$ and $f := z_k$.

Ground truth t.

GD for NN

$$J(w) = 1/2 \sum_{i=1}^{c} (t_k - z_k)^2 = 1/2 \|t_k - z_k\|^2$$

$$\Delta w = -\eta \frac{\partial J}{\partial w} \implies \Delta w_{pq} = -\eta \frac{\partial J}{\partial w_{pq}}$$

Finding this for output layer is easy. But for input layer it is difficult (**CREDIT ASSIGNMENT PROBLEM**).

Then same update rule $w^{t+1} = w^t + \Delta w$.

Updating w_{kj}

$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial net_k} \frac{\partial net_k}{\partial w_{kj}} = -\delta_k y_j$$

$$\delta_k = -\frac{\partial J}{\partial net_k} = -\frac{\partial J}{\partial z_k} \frac{\partial z_k}{\partial net_k} = (t_k - z_k) f'(net_k)$$

$$\therefore \Delta w_{kj} = \eta \, \delta_k \, y_j$$

Updating w_{ji}

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j}\frac{\partial y_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}} = \frac{\partial J}{\partial y_j} f'(net_j)\, x_i$$

$$\begin{aligned}
\frac{\partial J}{\partial y_j} &= \frac{\partial}{\partial y_j}\left[1/2\sum_{k=1}^{c}(t_k - z_k)^2\right] \\
&= -\sum_{k=1}^{c}(t_k - z_k)\frac{\partial z_k}{\partial y_j} \\
&= -\sum_{k=1}^{c}(t_k - z_k)\frac{\partial z_k}{\partial net_k}\frac{\partial net_k}{\partial y_j} \\
&= -\sum_{k=1}^{c}(t_k - z_k) f'(net_k) w_{kj} \\
&= -\sum_{k=1}^{c}\delta_k w_{kj}
\end{aligned}$$

Define $\delta_j := f'(net_j)\sum_{k=1}^{c}\delta_k w_{kj}$. Then,

$$\frac{\partial J}{\partial w_{ji}} = -\delta_j x_i$$

and

$$\Delta w_{ji} = \eta\, \delta_j\, x_i$$

J vs epoch graph

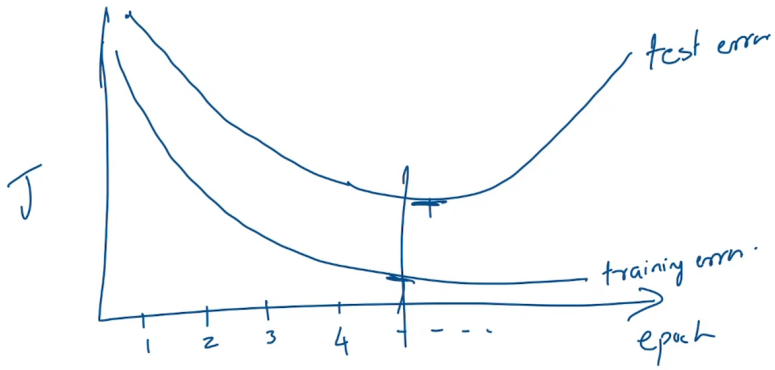

Figure 2: Error vs Epoch Graph

- epoch: one instance where we backpropagate and update each weight

Goes down (sort of exponential decay but might have some jitters here and there).

As the training error keeps coming down.

For the test set however, the testing error will go down but after some time the test error will start going up (overfitting).

Three ways to GD

- Stochastic GD
  - Pick training samples at random. Then forward and backward pass for that and so on.
  - So we randomly shuffle the training samples and go for all the points.
  - Just order of the training points is changed.
  - **Update is done after each sample**
- Batch GD
  - Update after the whole dataset
  - In this case epoch => update after going over all samples
- Mini Batch GD
  - Update after each mini batch
  - Shuffle after each epoch