Day 2 - Backpropagation Algorithm

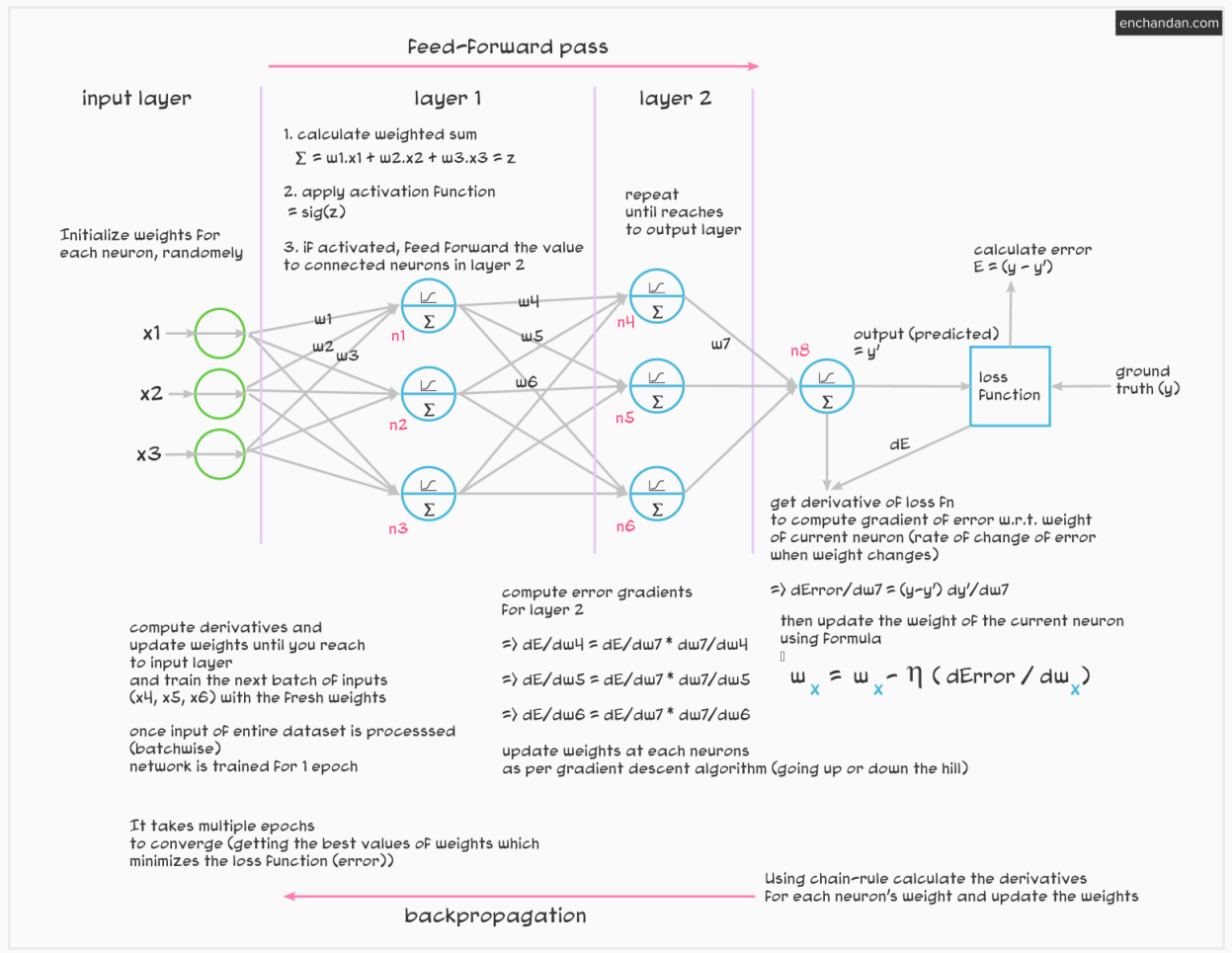
Tags: Artificial Neuron, BackPropagation, Bias, Epoch, FeedForward, NeuralNetwork, ReversePass, Training, Weights

Backpropagation Algorithm

Author: **Chandan Kumar**

enchandan.com

Process of Training NN in One Picture



Context

Suppose our training dataset has 10,000 inputs

We choose the batch_size of 32

Iterations required to process entire dataset once → $10000/32 \approx 312$

Steps

- First iteration, take **32 inputs**, so **32 neurons** required at the Input layer (say $x1, x2, x3 ... x32$)

- **Initialize the weights randomly** for each input neuron and pass the values (input value and weights) to the next layer
- At each neuron in the next layer
  1. Calculate the weighted sum
     $z = \sum wx + b$
     $z = w1.x1 + w2.x2 + w3.x3 + ... + w32.x32 + bias$

     Note: I have not mentioned bias neuron in the picture above (as it's optional in the neural network)
  2. Apply activation function
     $sig(z)$
  3. If activated, feed forward (pass) the value to the connected neurons in the next layer - layer2
- Each layer keeps the computation done on each of it's neurons (weighted sum, parameters etc). As it will be used during reverse pass.
- Repeat until you reach to the **Output layer**
- The activated weighted sum at the Output layer is the prediction (for regression problem at least, for classification problem, we might need to apply function like `Softmax()` to get the prediction)
- Find the gap between the predicted output with the ground truth and compute the error using a loss function e.g. **MSE** (Mean squared error)
- Get derivative of the loss function to w.r.t. the weights of the Output layers to compute the error gradients (at output neurons)

  $Error\ Gradient = \frac{\delta Error}{\delta w}$
- Update weights at the current neuron as per Gradient Descent Step
- Move to the previous layer in the network and using chain-rule calculate error gradients for that layer and update the weights of all the neurons in that layer
- Repeat the process until you reach to the input layer
- You'll have fresh weights on the input neurons. Now take another batch of 32 inputs and train the network using the fresh weights

- Once the entire training dataset is processed. you have completed training an epoch.
- Usually it takes multiple epochs to converge (getting the best values of weights which minimizes the chosen loss function of the network)