Neural Networks

Topics covered so far

- Neural Networks
  - General Introduction
  - Reminder of nonlinear features
  - Single unit and activation functions
  - Multiple layers
  - Architecture
  - Cross Entropy Loss
  - Gradient descent
  - Basic Training Algorithms, SGD, Minibatch

Discussion questions

1. What is a neural network and what do different layers in a neural network represent?
2. What is an activation function and what are the different types of activation functions?
3. What do forward and back propagation mean in neural networks?
4. What is the gradient descent algorithm and how does it work?
5. How do we reduce overfitting in neural networks?

Neural Networks

Artificial Neural Networks (ANNs) are inspired by biological neural networks and employ a collection of interconnected artificial neurons to extract the patterns from the given data.

It consists of three types of layers:

- **Input layer**
  - Represents dimensions of the input vector (one node for each dimension)
- **Hidden layer(s)**
  - Represents the intermediary nodes that divide the input space into regions with (soft) boundaries
  - Given enough hidden nodes, we can model any arbitrary input-output relation
  - It takes in a set of weighted input and produces output through an *activation function*
- **Output layer**
  - Represents the output of the neural network
  - Mostly, it doesn't have an activation function

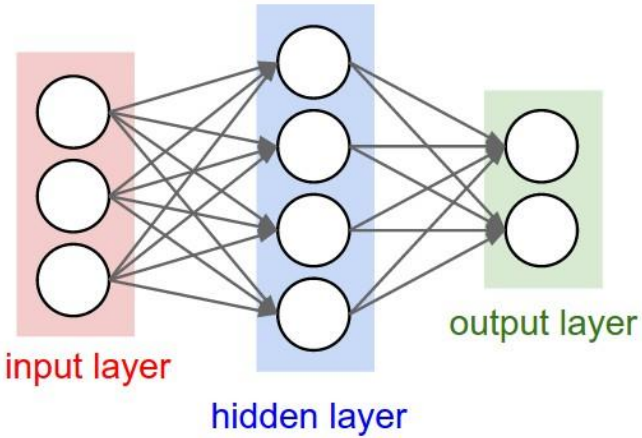
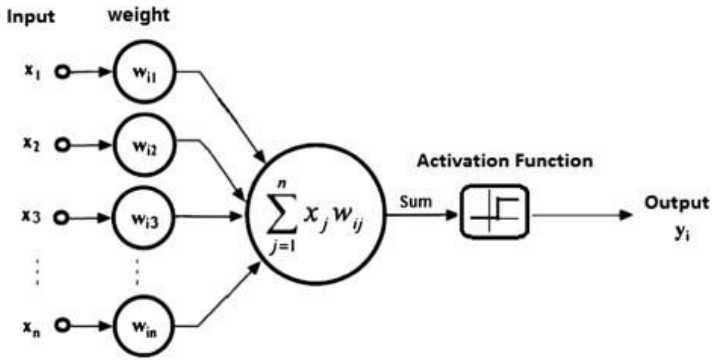


Image Source

Activation functions

- An artificial neural network works in three steps:
  - First, it multiplies the input signals with corresponding weights
  - Second, it adds the weighted signals together
  - Third, it converts the result into another value using a mathematical transformation (activation function)
- For the third step, there are multiple mathematical functions available that can be used for the activation function.



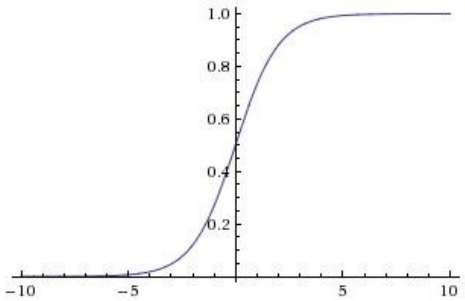
- The purpose of the activation function is to act like a switch for the neuron.
- The activation function is critical to the overall functioning of the neural network. Without it, the whole neural network will mathematically become equivalent to one single neuron!
- The activation function is one of the critical components that give neural networks the ability to deal with complex problems, by tackling the nonlinearity of the patterns in the data.

Source: Quora

Types of activation functions

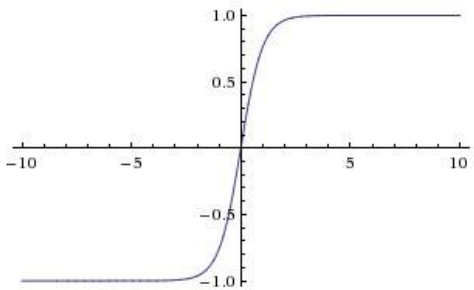
These are some activation functions that are generally used in neural networks:

1. The Sigmoid function
2. The Tanh function
3. The ReLU (Rectified Linear Unit) function



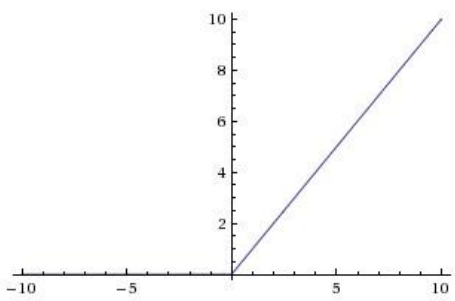
Sigmoid

- Range : 0 to 1
- Gives probabilities



Tanh

- Range : -1 to 1
- More steeper than sigmoid



ReLU

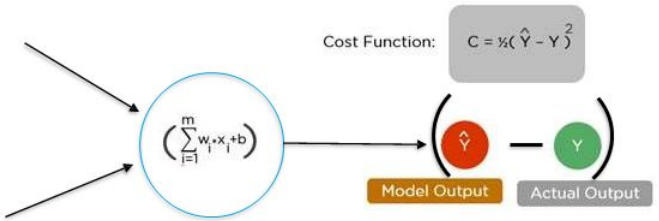
- Range : 0 to ∞
- Less computationally expensive

Image Source

Forward Propagation

- In the forward propagation, the input data is propagated forward from the input layer through the hidden layer until it reaches the final/output layer where predictions are made.
- At every layer, data gets transformed in three steps in every neuron:
  - Sum of weighted input at every neuron (by multiplying X by the hidden weight W) and the bias
  - Apply the activation function on the sum
  - Pass the result to all the neurons in the next layer
- The last layer is the output layer, which may have a sigmoid function (for binary classification) or a softmax function (if the network is a multi-class classifier). The output layer gives the predictions of the neural network.

After getting the predictions, we use an optimization algorithm that helps us to ***minimize*** *the* error (cost) function **E(x)** which is simply a mathematical function dependent on the model's **learnable parameters** which are used in computing the target values (**Y**) from the set of *predictors* (**X**) used in the model.



Back Propagation

- Back propagation is the process of learning that the neural network employs to re-calibrate the weights at every layer and every node to minimize the error in the output layer.
- During the first pass of forward propagation, the weights are random numbers.
- The output of the first iteration is not always accurate. The difference between the actual value / class and the predicted value / class is the error.
- All the nodes in all the preceding layers contribute to error and hence need to get their share of the error and correct their weights.
- This process of allocating a proportion of the error (error gradient) to all the nodes in the previous layer is called the back propagation.
- The goal of back propagation is to adjust weights in proportion to the error contribution and in iterative process identify the optimal combination of weights.
- At each layer, at each node, the gradient descent algorithm is applied to adjust the weights.

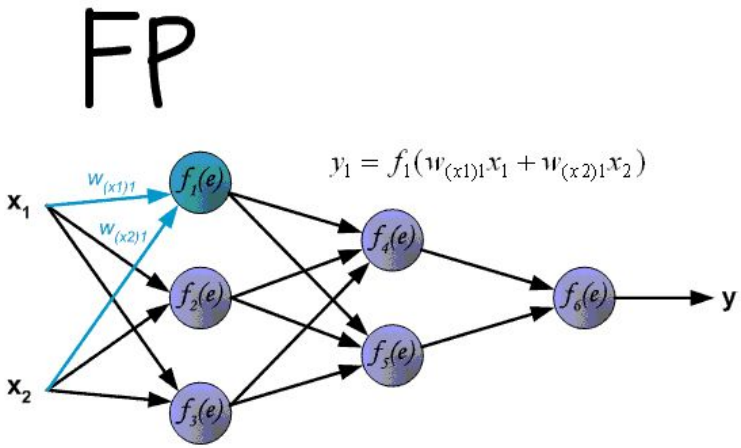


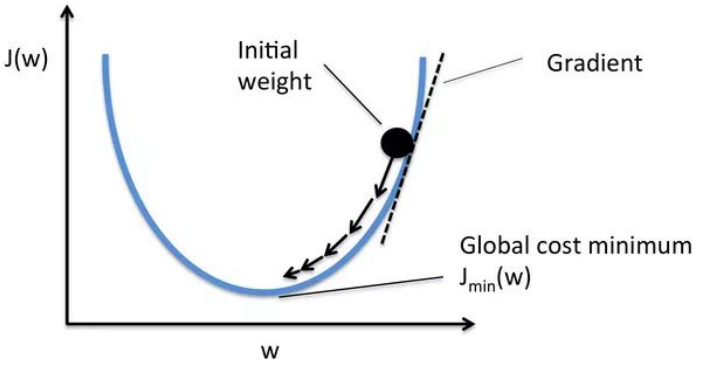
Image Source

Gradient Descent Algorithm

- The goal of optimization is to find a set of weights that minimizes the loss function.
- Optimization functions usually calculate the **gradient,** i.e., the partial derivative of the loss function with respect to weights, and the weights are modified in the opposite direction of the calculated gradient. This cycle is repeated until we reach the minima of loss function.
- The procedure of repeatedly evaluating the gradient and then performing a parameter update is known as **Gradient Descent Algorithm.**

Learning Rate:

- It is a hyperparameter which determines the step size (the amount by which the weights are updated).
- We can try out different values of learning rate to improve the results.



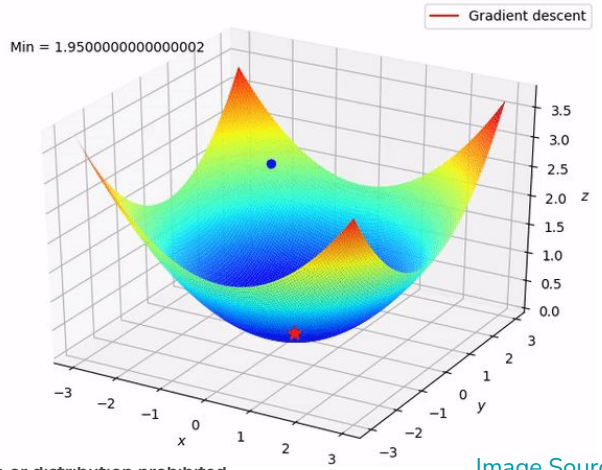


Image Source

Gradient Descent variations

	Batch Gradient Descent	Stochastic Gradient descent	Mini Batch Gradient Descent
Working	It updates the parameter by calculating gradients of the whole dataset	It updates the parameters by calculating gradients for each training example	It updates the parameters by calculating gradients for every mini batch of “n” training examples. It is a combination of batch and stochastic gradient descent
Advantages	It is computationally efficient and gives stable convergence	It is faster in learning than batch gradient descent and gives immediate performance insights	It is computationally efficient, fast to learn and gives stable convergence
Disadvantages	It learns very slowly and the chances of getting stuck in a local minima are high	It is computationally intensive and can give noisy gradients	It adds up one more hyperparameter to tune.

Overfitting in NN

Deep neural networks are prone to overfitting since they try to capture complex patterns in the data but they may also capture the noise.

We can use the Ridge and Lasso regression techniques (that we have understood in previous courses) to reduce overfitting. There are also some other ways to reduce overfitting specific to Deep Neural Networks:

- **Batch Normalization:** Batch normalization is a technique for improving the performance and stability of neural networks. The idea is to normalize the inputs of each layer in such a way that they have a mean output activation of zero and standard deviation of one. This is analogous to how the inputs to networks are standardized.

  This approach leads to faster learning rates since normalization ensures there's no activation value that's too high or too low, as well as allowing each layer to learn independently of the others.

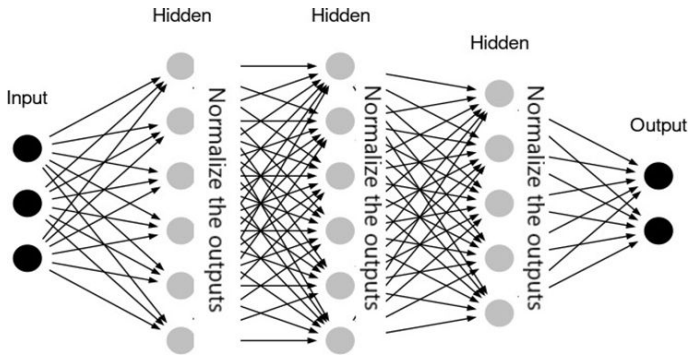
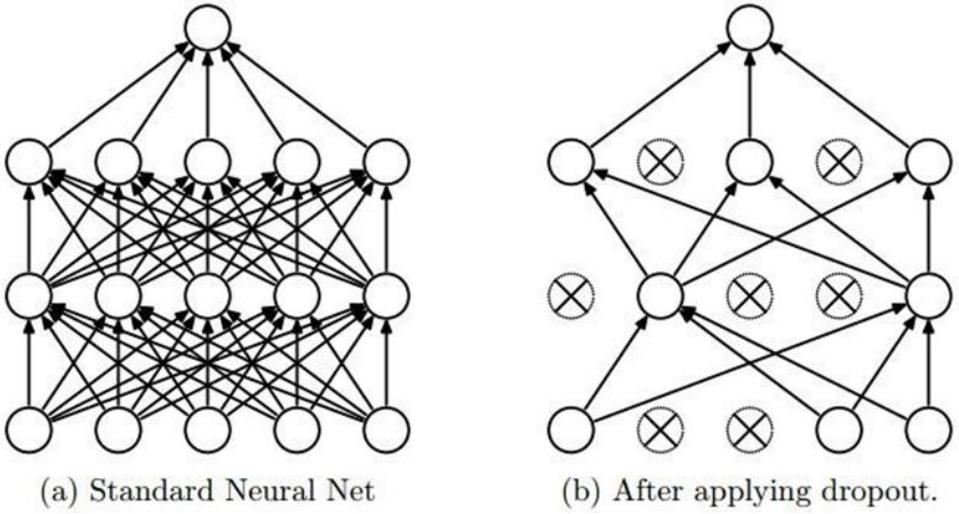


Image Source

Overfitting in NN

- A fully connected layer occupies most of the parameters, and hence, neurons develop co-dependency amongst each other during training which curbs the individual power of each neuron leading to overfitting of the training data. This is handled by using **Dropout layers**.

- In a dropout layer, we randomly shut down some fraction of a layer's neurons at each training step by zeroing out the neuron values.
- The fraction of neurons to be zeroed out is known as the dropout rate, say rd.
- The remaining neurons have their values multiplied by 1/(1-rd) so that the overall sum of the neuron values remains the same.



Source: (Srivastava et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. JMLR, 2014)

Case Study

Great Learning

Happy Learning !