Conceptual Session 2
CHT and DL

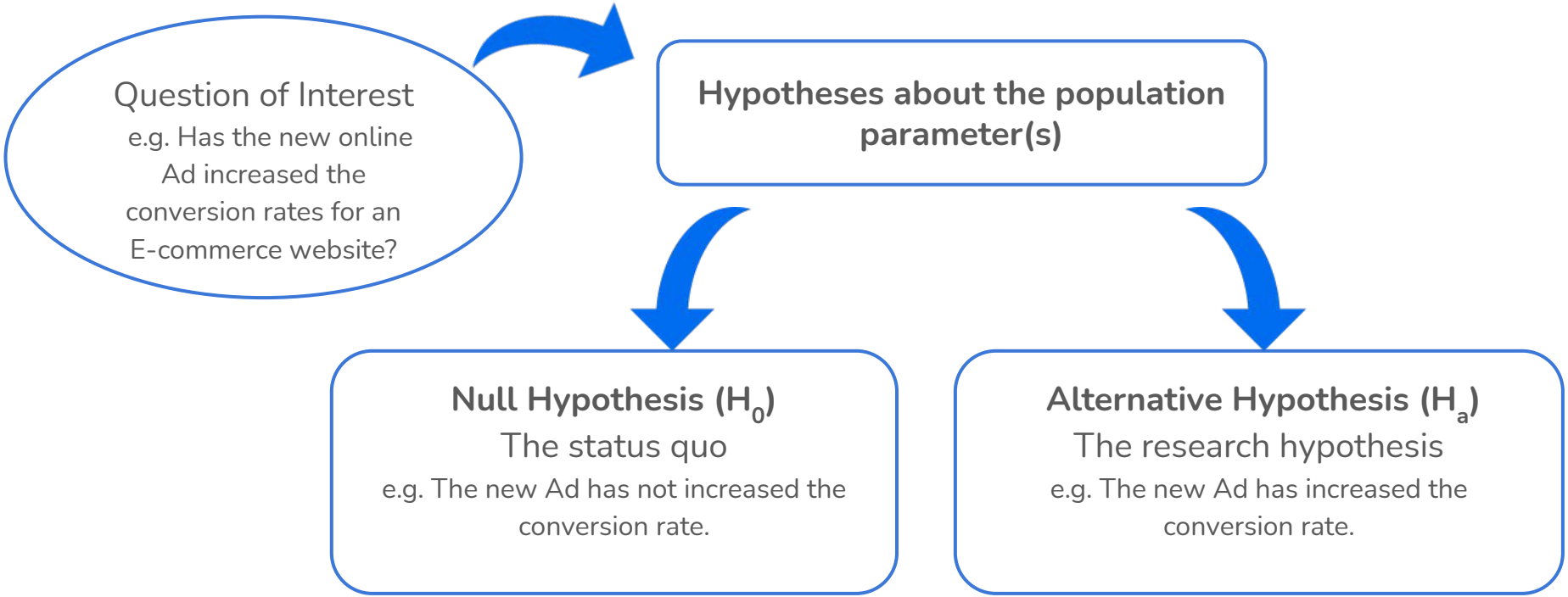
Learning Objective of the Session

1. Hypothesis Testing
    a. Hypothesis Formulation
    b. One Tailed Test vs Two Tailed Test
    c. Type I and Type II Errors
2. Classification
    a. Logistic Regression
    b. SVM and kernel Trick
    c. Perceptron
    d. Model Evaluation
3. Deep Learning
    a. Neural networks
    b. Convolutional Neural Networks

Discussion Question - Hypothesis Testing

1. What is hypothesis testing and what are different types of hypotheses?
2. What are some of the key terms involved in hypothesis testing?
3. What is the difference between one-tailed and two-tailed tests?
4. What are the steps to perform a hypothesis test?

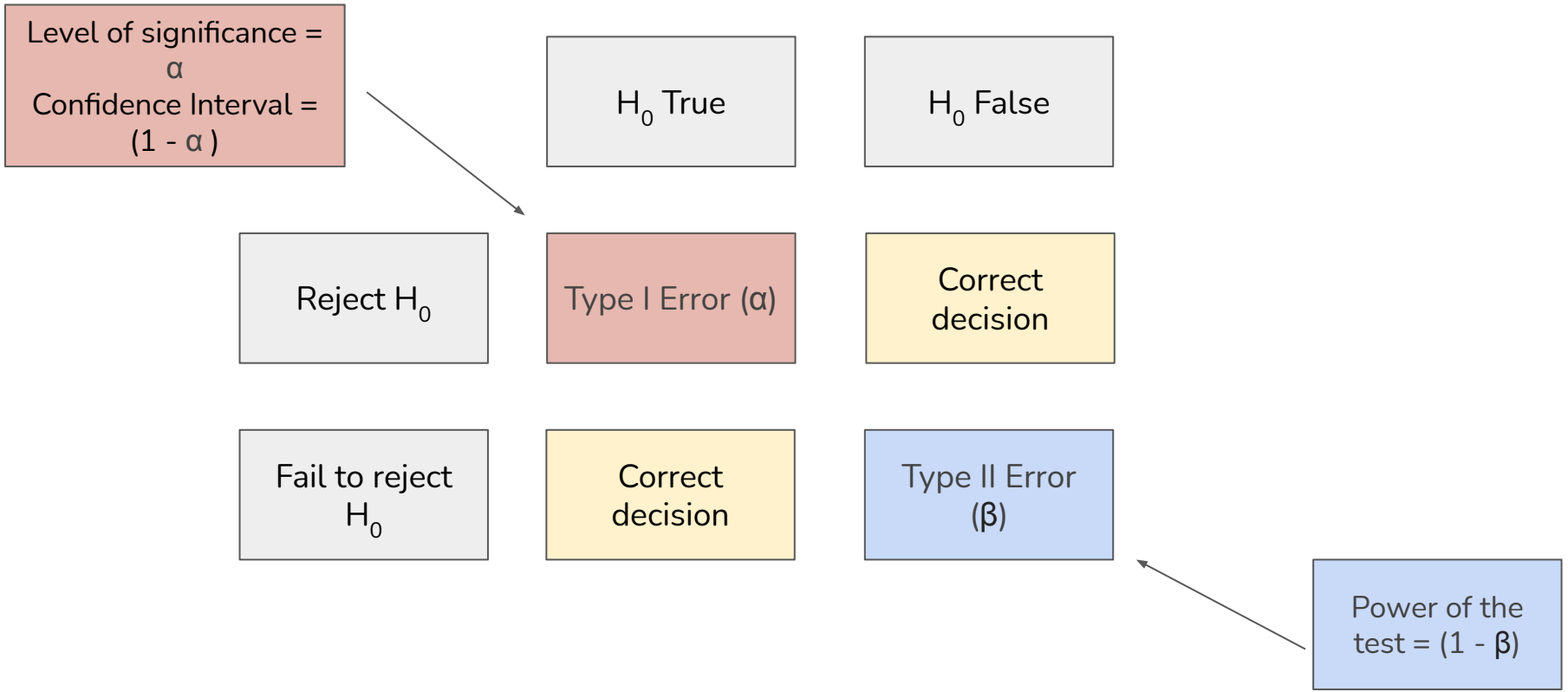
Introduction to Hypothesis Testing



Key terms in Hypothesis Testing

P-Value	• Probability of observing equal or more extreme results than the computed test statistic, under the null hypothesis. • The smaller the p-value, the stronger the evidence against the null hypothesis.
Level of Significance	• The significance level (denoted by α), is the probability of rejecting the null hypothesis when it is true. • It is a measure of the strength of the evidence that must be present in the sample data to reject the null hypothesis.
Acceptance or Rejection Region	• The total area under the distribution curve of the test statistic is partitioned into acceptance and rejection region • Reject the null hypothesis when the test statistic lies in the rejection region, else we fail to reject it
Types of Error	• There are two types of errors - Type I and Type II

Type I and Type II errors



Let's go through an example

Problem Statement: The store manager believes that the average waiting time for the customers at checkouts has become worse than 15 minutes. Formulate the hypothesis.

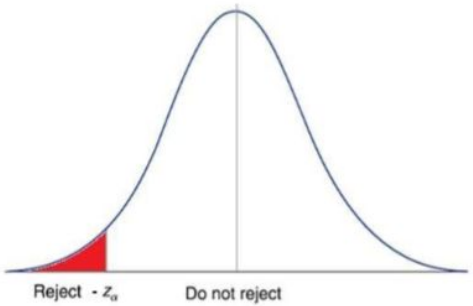
Null Hypothesis (H_0): The average waiting time at checkouts is less than equal to 15 minutes.

Alternate Hypothesis (H_a): The average waiting time at checkouts is more than 15 minutes.

Type I error (false positive): Reject Null hypothesis when it is indeed true. "The fact is that the average waiting time at checkout is less than equal to 15 minutes but the store manager has identified that it is more than 15 minutes".

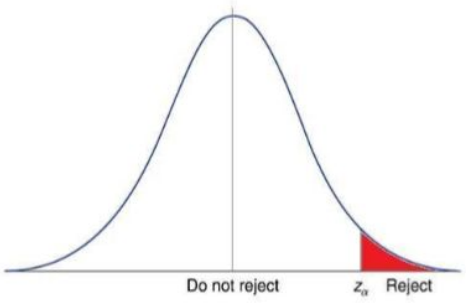
Type II error (false negative): Fail to reject Null hypothesis when it is indeed false. "The fact is that the average waiting time at checkout is more than 15 minutes but the store manager has identified that it is less than equal to 15 minutes".

One-tailed vs Two-tailed Test



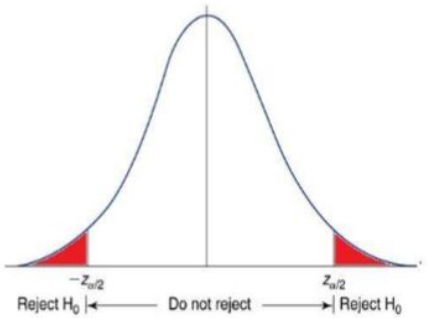
- Lower tail test.
- H_1: $\mu <$ .....

Reject H_0 if the value of test statistic is too small



- Upper tail test.
- H_1: $\mu >$ .....

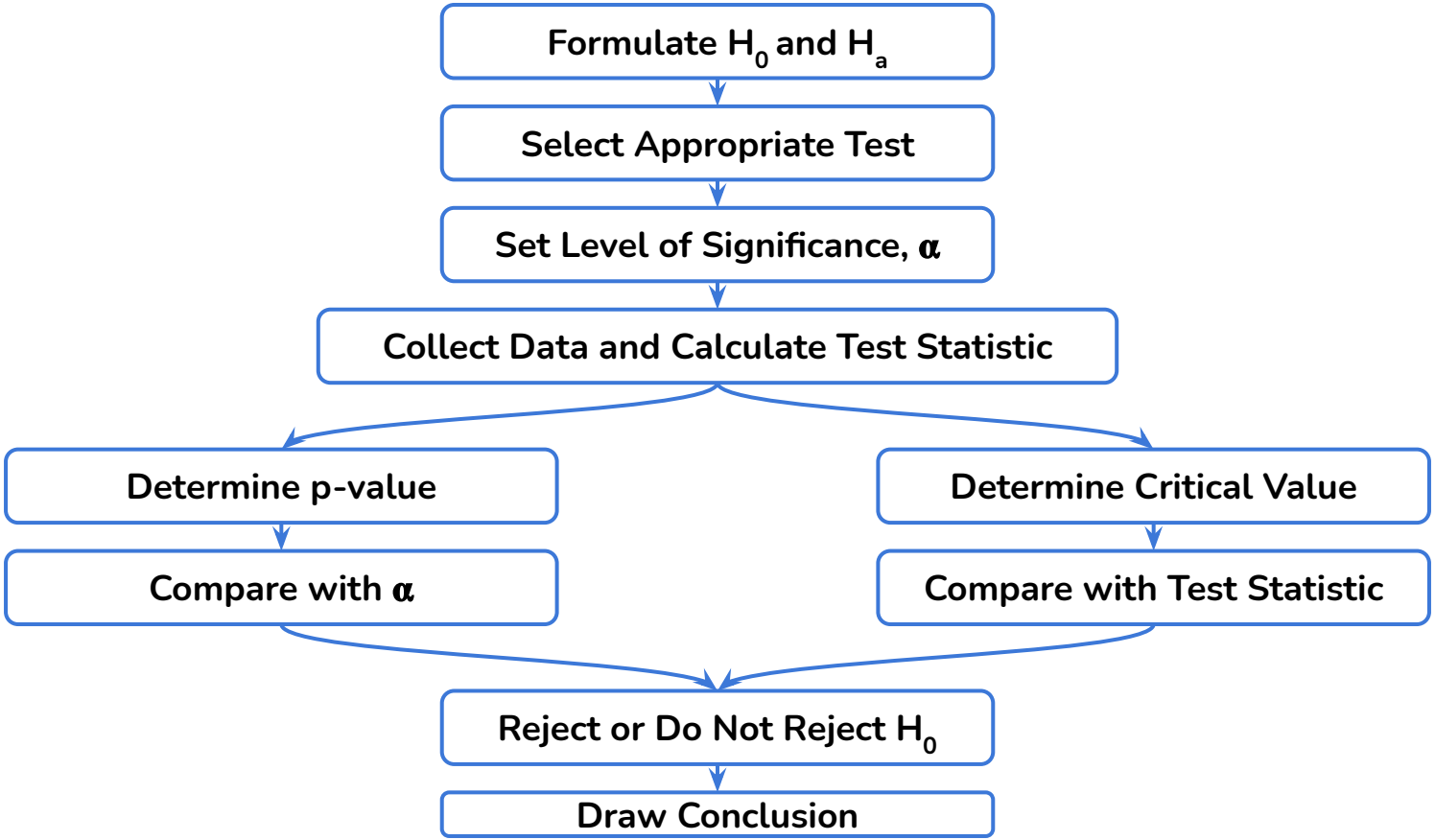
Reject H_0 if the value of test statistic is too large



- Two tail test.
- H_1: $\mu \neq$ .....

Reject H_0 if the value of test statistic is either too small or too large

Hypothesis Testing Steps



Classification models

Why do we use logistic regression?

- Logistic Regression is a supervised learning algorithm which is used for the classification problems i.e. where the dependent variable is categorical
- In logistic regression, we use the sigmoid function to calculate the probability of the dependent variable
- The real life applications of logistic regression are churn prediction, spam detection etc.
- The below image shows how logistic regression is different from linear regression in fitting the model

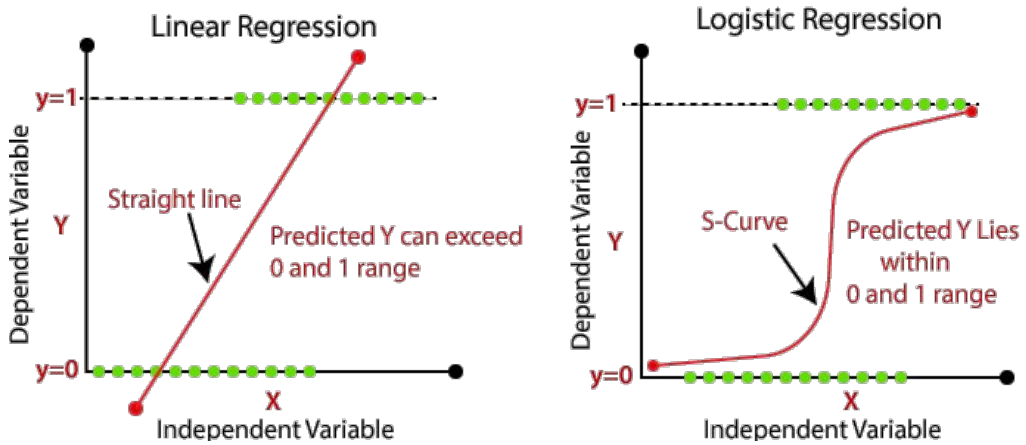
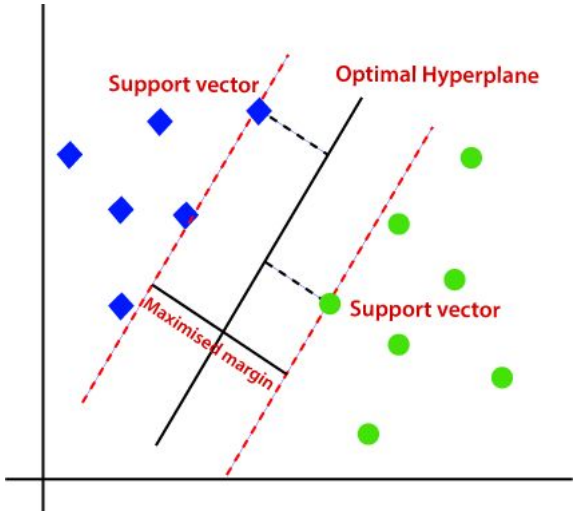


Image Source

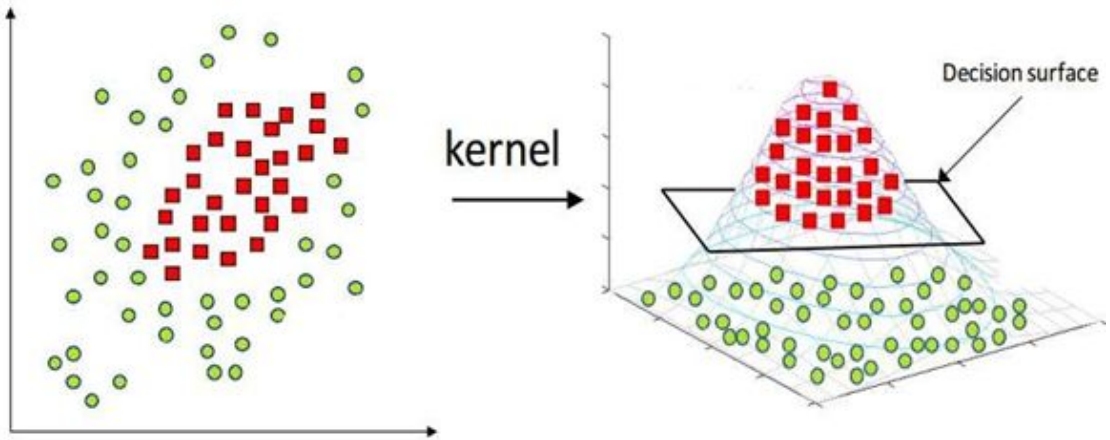
Support Vector Machines

- Support Vector Machines Can be used both for classification and regression tasks, but widely used for classification.
- It Aims at finding a hyperplane in an n-dimensional space that distinctly classifies all the observations in a data
- It is entirely possible to have number of hyperplanes that solve the purpose, we must find the one with maximum margin.
- Dimension of hyperplane is determined by # inputs
    - For 2 input features : line
    - For 3 input features : plane



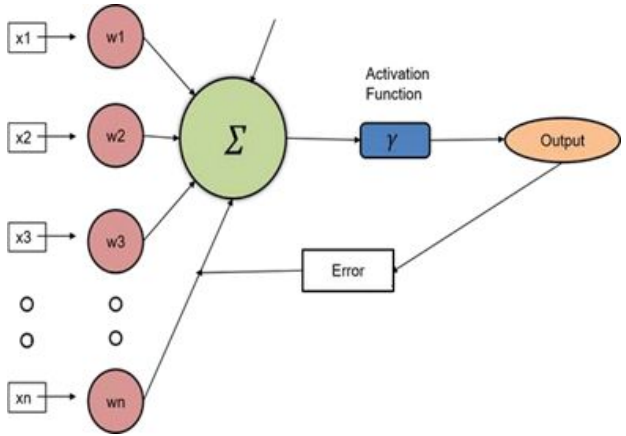
Kernel Trick

- Not all data is good enough to be linearly separable, which makes the job of a SVM difficult
- Not only does it become difficult, but it also leads to high computational costs.
- Kernel trick offers an efficient and less expensive way to transform data
- The trick is to represent the data in a low-dimension space instead of higher dimension.
- This is done by utilizing pairwise comparisons in the original data points.



Perceptrons

- Classification technique for binary classification.
- Also, c/a simplest type of neural network.
- Linear classification algorithm i.e., it separates the two classes using a line.
- Preferable for data where classes can we well separated using a line.
- Coefficients of the model are trained using gradient descent algorithm.



Confusion matrix

It is used to measure the performance of a classification algorithm. It calculates the following metrics:

1. **Accuracy:** Proportion of correctly predicted results among the total number of observations

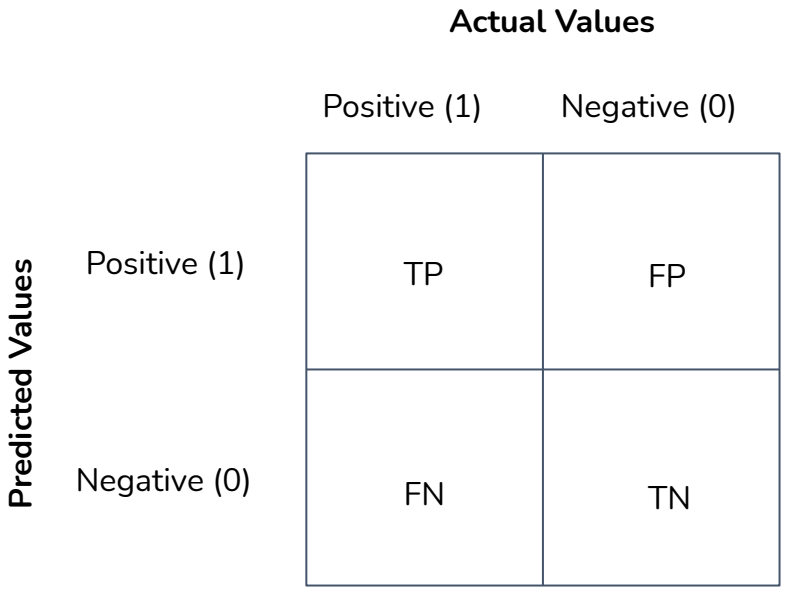
   Accuracy = (TP+TN)/(TP+FP+FN+TN)

1. **Precision:** Proportion of true positives to all the predicted positives i.e. how valid the predictions are

   Precision = (TP)/(TP+FP)

1. **Recall:** Proportion of true positives to all the actual positives i.e. how complete the predictions are

   Recall = (TP)/(TP+FN)



Neural Networks

Topics covered so far

- Neural Networks
  - General Intro
  - 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 an NN 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 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 an 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

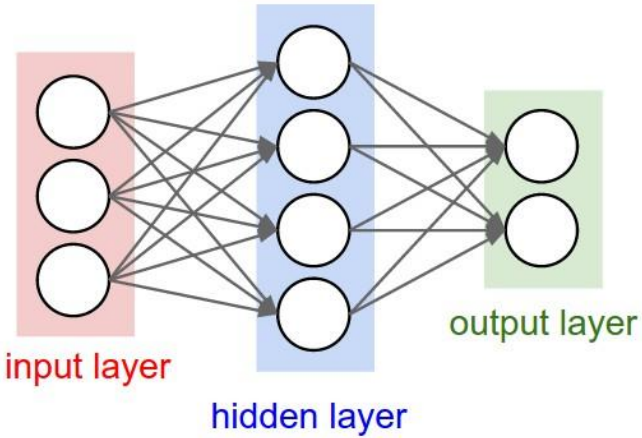
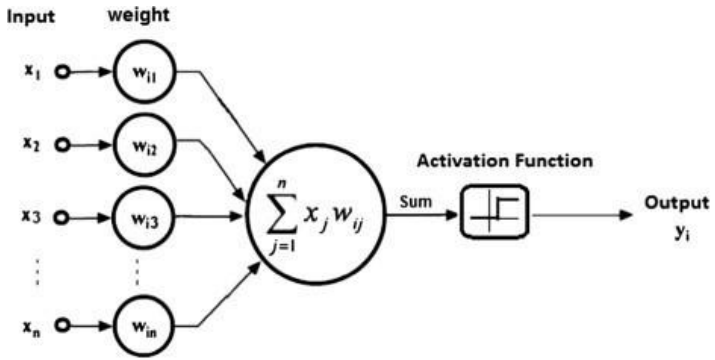


Image Source

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.



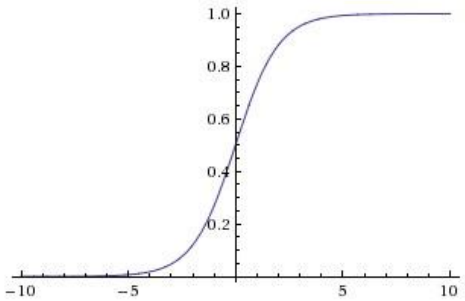
- The purpose of the activation function is to act like a switch for the neuron.
- The activation function is critical to the overall functioning of the neural network. Without it, the whole neural network will mathematically become equivalent to one single neuron!
- The activation function is one of 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

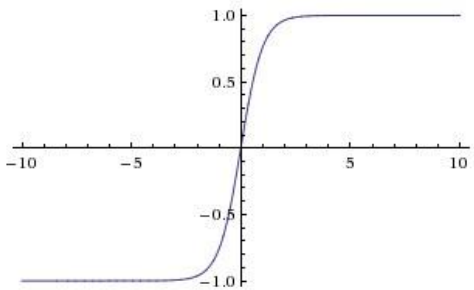
These are some activation functions that are generally used in neural networks:

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



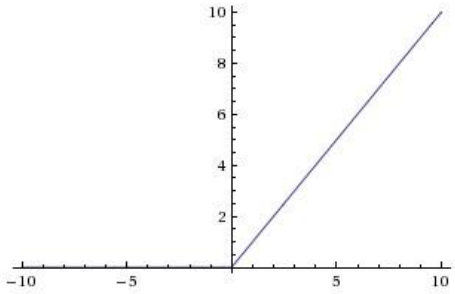
Sigmoid

- Range : 0 to 1
- Gives probabilities



Tanh

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



ReLU

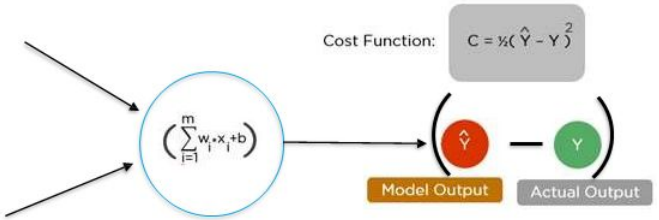
- Range : 0 to ∞
- Less computationally expensive

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 weighted input at every neuron by multiplying X by the hidden weight Wh 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*** 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 actual value / class and 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

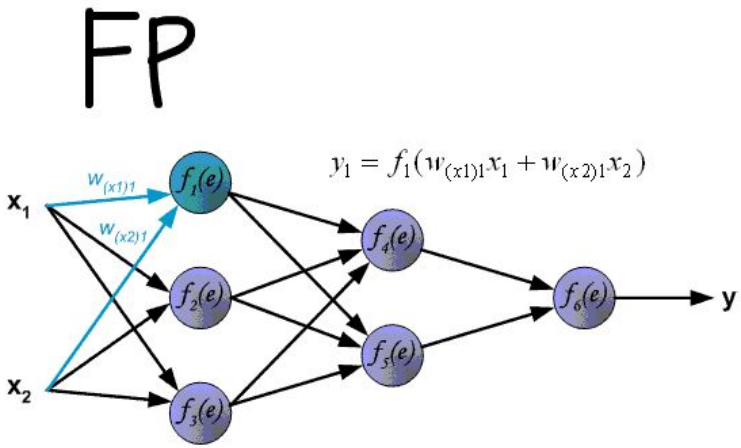
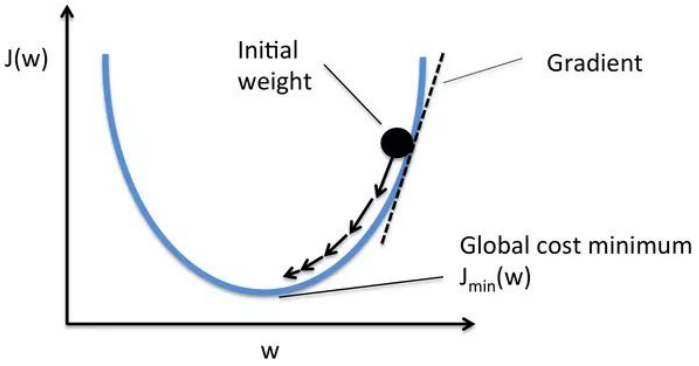


Image Source

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 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 called **Gradient Descent**



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

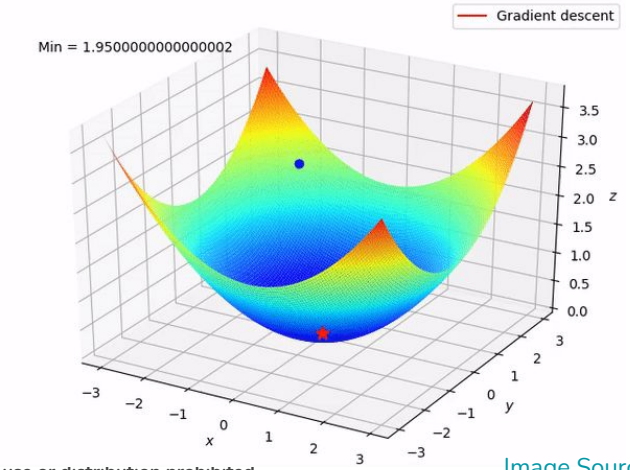


Image Source

Gradient Descent variations

	Batch Gradient Descent	Stochastic Gradient descent	Mini Batch Gradient Descent
Working	It updates the parameter by calculating gradients of 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 covered in previous modules) to reduce overfitting. There are also some other ways to reduce overfitting in a Deep Neural Network:

- **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.

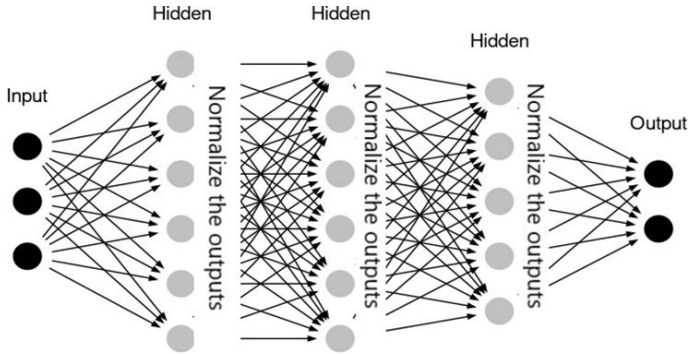
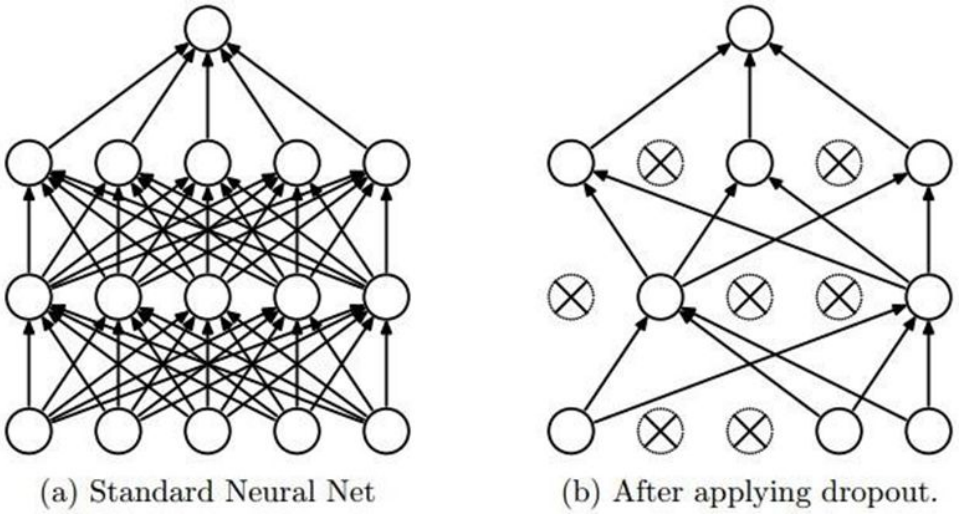


Image Source

Overfitting in NN

- **Dropout:** 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 over-fitting of the training data.

- In dropout, 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, 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)

CNN

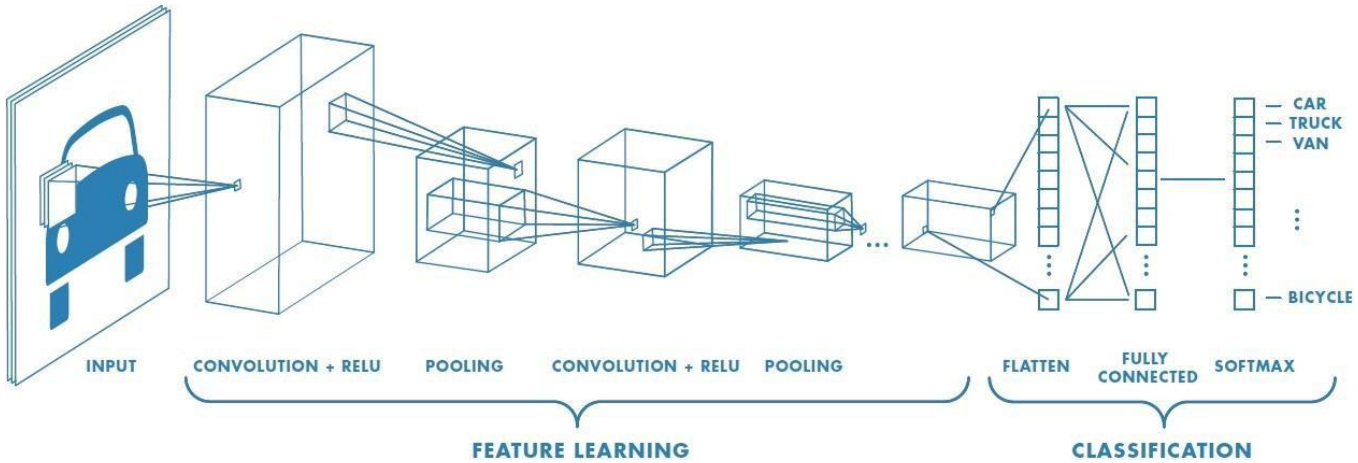
Discussion questions

1. What are convolutional neural networks and how are they different from ANNs?
2. How do filters/kernels work for feature detection in CNNs?
3. How do pooling, padding and stride operations work in CNNs?
4. What is transfer learning and how can we use this in CNNs?

Convolutional Neural Networks

CNNs are a special type of neural network designed to work with image data. CNNs use convolutional layers - hidden layers which perform convolution operations. They have some different characteristics to ANNs:

1. Unlike ANNs, CNNs capture the spatial structure of the image
2. CNNs follow the concept of parameter sharing i.e. one filter is applied over the whole image, because of which they are much more computationally efficient.



The first part in this architecture is the convolutional layer followed by the pooling layer and the second part is the fully connected layer. This whole architecture is called a convolutional neural network.

Image Source

The convolutional layer - Filter/Kernel

- A convolution operation uses a small array of numbers called a filter/kernel on the input image.
- Each filter is designed to identify a specific feature in the input space of the image, such as horizontal edges, vertical edges etc.
- A CNN is able to successfully capture the spatial and temporal dependencies in an image through the application of relevant filters.
- The role of the CNN is to reduce the images into a form which is easier to process, without losing features which are important for getting a good prediction.

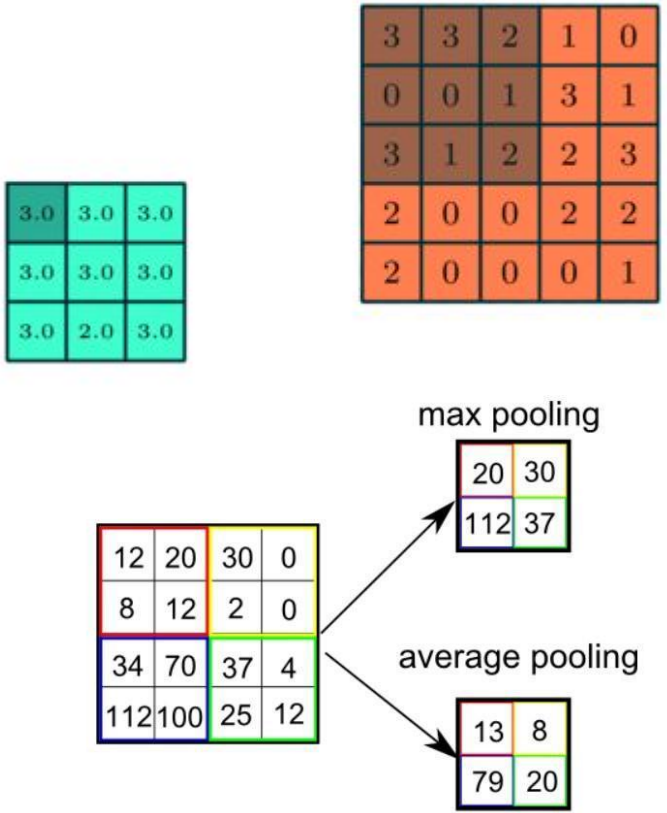
- This image shows how the convolution operation works in a CNN
- It uses a 3x3 filter on a 5x5 image
- The resulted feature is a 3x3 image which is the convolved feature

Pooling layer in CNNs

- After a convolution operation, we usually perform pooling to reduce the dimensions of the feature map
- It enables us to reduce the number of parameters, which both reduces the training time and the overfitting
- Pooling layers downsample each feature map independently, reducing the height and width, but keeping the depth same.

There are two types of pooling - Max and Average

- Max pooling just takes the maximum value whereas average pooling takes the average value in the pooling window
- Contrary to the convolution operation, pooling has no parameters.
- In these gifs, we can see how both max and average pooling work



Padding & Stride in CNNs

- **Stride** specifies how much we move the filter at each step. By default the value of the stride is 1 and is represented by the first figure
- We can also increase the value of stride if we want less overlap between the filters. It also makes the resulting feature map smaller since we are skipping over some locations.
- The second figure demonstrates the stride 2

We see that after using stride, the size of the feature map is smaller than the input. If we want to maintain the same dimensions, we can use **padding** to surround the input with zeros.

- The grey area around the input in the third figure is the padding.
- We either pad with zeros or the values on the edge, to match the dimensions of the feature map with the input.
- Padding is commonly used in CNNs to preserve the size of the feature maps, otherwise they would shrink at each layer, which is not desirable.

Great Learning

Happy Learning !