GITAM School of Technology

CSEN3011: Artificial Neural Networks

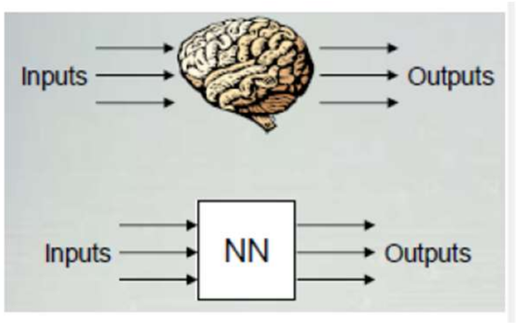
Unit 1 Syllabus

Introduction to Neural Networks

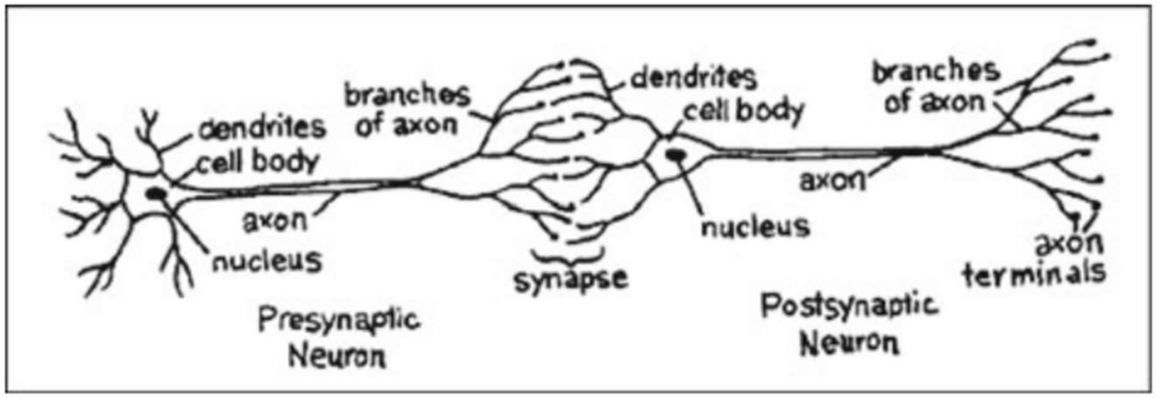
- Introduction
- The Basic Architecture of Neural Networks
- Training a Neural Network with Backpropagation
- Practical Issues in Neural Network Training
- Common Neural Architectures

Introduction to Neural Networks

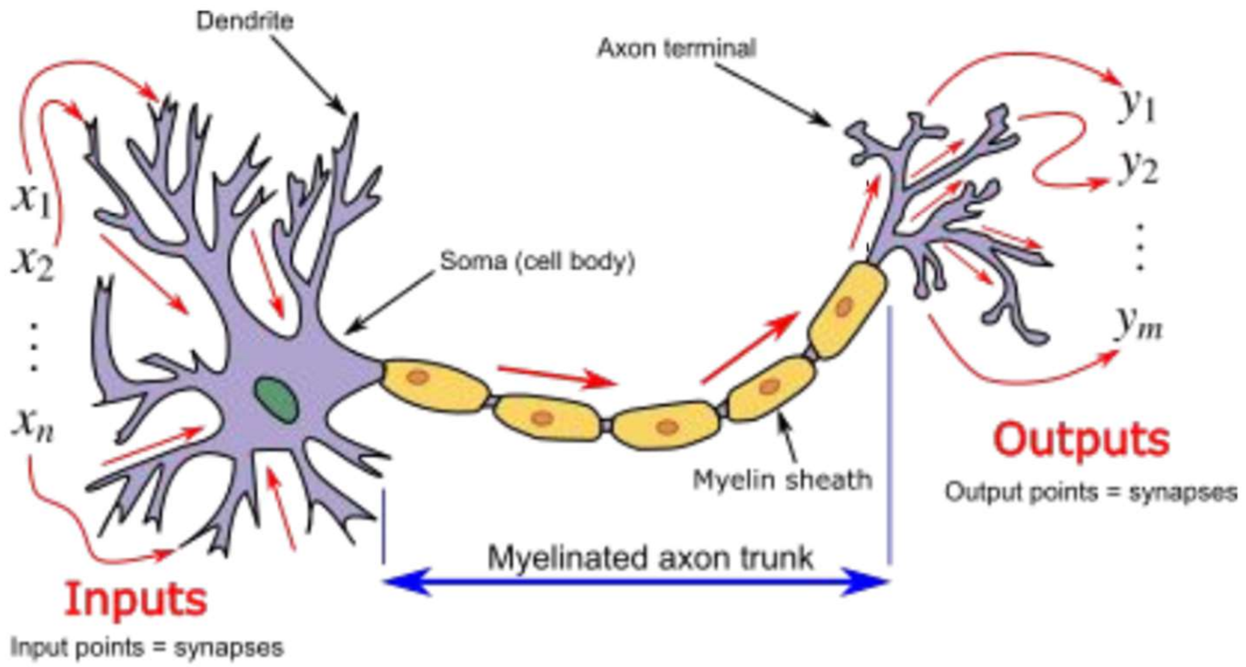
- **Artificial neural network (**ANN) is a network of **artificial neurons** which are conceptually derived from biological neurons.
  - Each artificial neuron has inputs and produces a single output which can be sent to multiple other neurons.
- **ANNs** usually simply called **neural networks** (**NNs**) or, more simply yet, **neural nets**, are computing systems inspired by the biological neural networks that constitute human brain



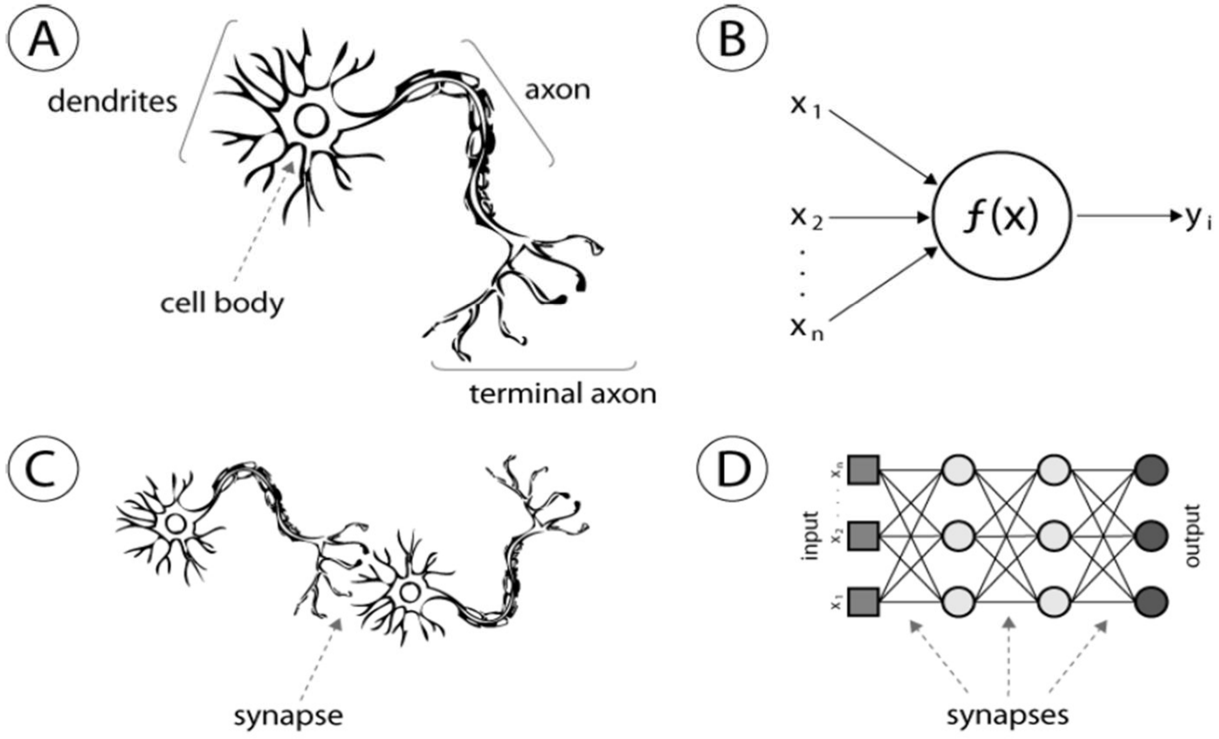
Neuron (in Brain)



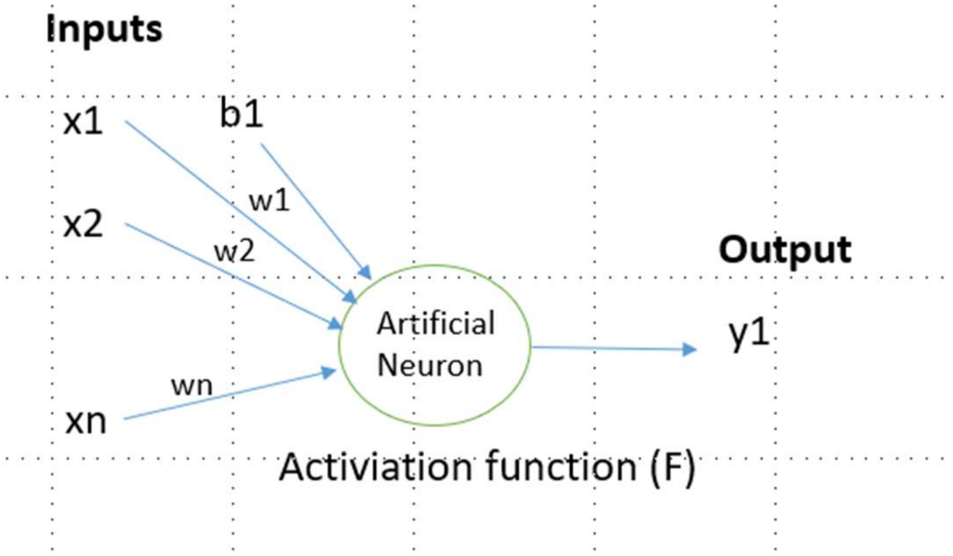
Biological neural network



Neurons and the brain



Artificial Neuron (or Perceptron)



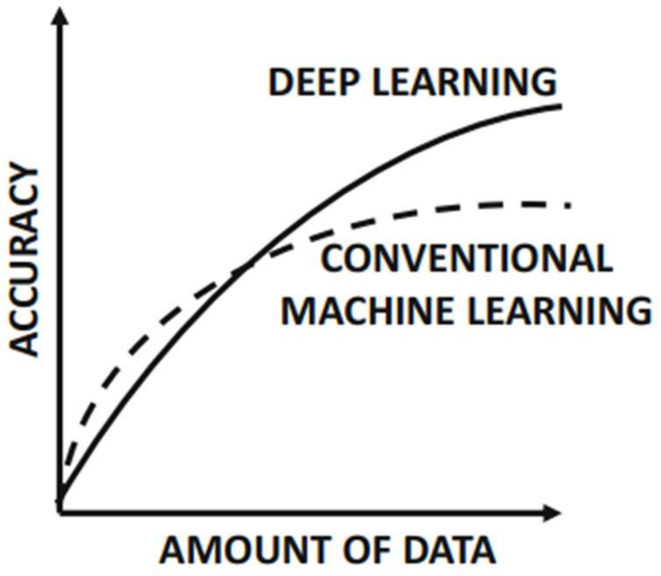
- Weights can be thought of as the strength of the connection.
- Weight affects the amount of influence a change in the input will have upon the output.
- A low weight value will have no change on the input, and alternatively a larger weight value will more significantly change the output.

- Activation function **decides, whether a neuron should be activated or not by calculating weighted sum of inputs and further adding bias with it**.
- The purpose of the activation function is to introduce non-linearity into the output of a neuron.
- Bias can be defined as the constant which is added to the product of features and weights. It is used to offset the result. It helps the models to shift the activation function towards the positive or negative side.

- y1 = F**(**∑(wi*xi) + b1**)**

- An artificial neural network **computes a function** of the inputs by propagating the computed values
    - from the input neurons to the output neuron(s) and using the weights as intermediate parameters.
- Learning occurs by **changing the weights** connecting the neurons.
    - Just as external stimuli are needed for learning in biological organisms, the external stimulus in artificial neural networks is provided by the training data containing examples of **input-output pairs** of the function to be learned.
- For example, the training data might contain pixel representations of images (input) and their annotated labels (e.g., carrot, banana) as the output.
    - These training data pairs are fed into the neural network by using the input representations to make predictions about the output labels.
    - The training data provides feedback to the correctness of the weights in the neural network depending on how well the predicted output (e.g., probability of carrot) for a particular input matches the annotated output label in the training data.

- The **weights between neurons are adjusted in a neural network in response to prediction errors.**
  - The goal of changing the weights is to modify the computed function to make the predictions more correct in future iterations.
  - Therefore, the weights are changed carefully in a mathematically justified way so as to reduce the error in computation on that example.
  - By successively adjusting the weights between neurons over many input-output pairs, the function computed by the neural network is refined over time so that it provides more accurate predictions.
- Therefore, if the neural network is trained with many different images of bananas, it will eventually be able to properly recognize a banana in an image it has not seen before.
- This ability to accurately compute functions of unseen inputs by training over a finite set of input-output pairs is referred to as **model generalization**.



An illustrative comparison of the accuracy of a typical machine learning algorithm with that of a large neural network. Deep learners become more attractive than conventional methods primarily when sufficient data/computational power is available. Recent years have seen an increase in data availability and computational power, which has led to a “Cambrian explosion” in deep learning technology.

Humans Versus Computers: Stretching the Limits of Artificial Intelligence

- **Computing the cube root** of a large number is very easy for a computer, but it is extremely difficult for humans.
- **Recognizing the objects in an image** is a simple matter for a human, but has traditionally been very difficult for an automated learning algorithm.
- Many **deep learning architectures that have shown such extraordinary performance** are not created by indiscriminately connecting computational units.
- **Biological networks** are connected in ways **we do not fully understand.**

Advantage of neural network

- A key advantage of neural networks over traditional machine learning is that the former provides a higher-level abstraction of expressing **semantic insights about data domains** by architectural design choices in the computational graph.
- The second advantage is that neural networks provide a **simple way to adjust the complexity of a model by adding or removing neurons** from the architecture according to the availability of training data or computational power.

Recent advancement

- The **“Big data” era** has been enabled by the advances in data collection technology; virtually everything we do today, including purchasing an item, using the phone, or clicking on a site, is collected and stored somewhere.
- Furthermore, the development of powerful **Graphics Processor Units** (GPUs) has enabled increasingly efficient processing on such large data sets.
- Furthermore, these recent adjustments to the algorithms have been enabled by **increased speed of computation, because reduced run-times enable efficient testing** (and subsequent algorithmic adjustment).
- The rapid advances associated with the three pillars of improved data, computation, and experimentation have resulted in an increasingly optimistic outlook about the future of deep learning.

The Basic Architecture of Neural Networks

Single Layer Network:

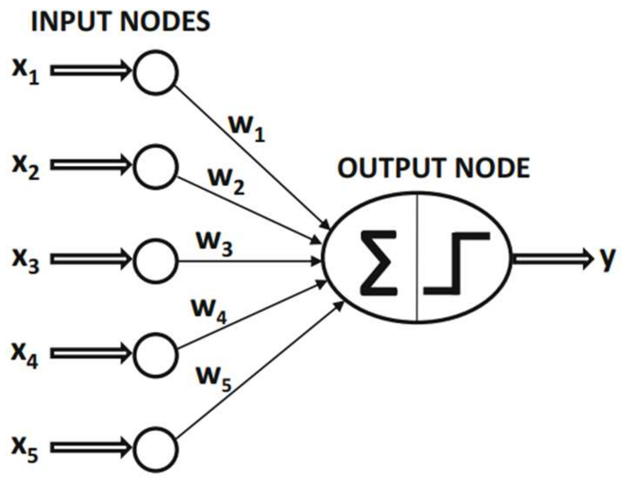
In the single layer network, a set of inputs is directly mapped to an output by using a generalized variation of a linear function. This simple instantiation of a neural network is also referred to as the **perceptron**.

Multi Layer Network:

In multi-layer neural networks, the neurons are arranged in layered fashion, in which the input and output layers are separated by a group of hidden layers. This layer-wise architecture of the neural network is also referred to as a **feed-forward network.**

Single Computational Layer: The Perceptron

- The simplest neural network is referred to as the perceptron. This neural network contains a single input layer and an output node.



(a) Perceptron without bias

The basic architecture of the perceptron

- Consider a situation where each training instance is of the form $(\overline{X},y)$, where each $\overline{X} = [x_1,\ldots x_d]$ contains d feature variables, and $y \in \{-1, +1\}$ contains the observed value of the binary class variable.
- By “observed value” we refer to the fact that it is given to us as a part of the training data, and our goal is to predict the class variable for cases in which it is not observed.

- For example, in a credit-card fraud detection application,
  - the features might represent various properties of a set of credit card transactions (e.g., amount and frequency of transactions), and the class variable might represent whether or not this set of transactions is fraudulent.
  - Clearly, in this type of application, one would have historical cases in which the class variable is observed, and other (current) cases in which the class variable has not yet been observed but needs to be predicted.

- The input layer contains d nodes that transmit the d features $\overline{X} = [x_1 \ldots x_d]$ with edges of weight $\overline{W} = [w_1 \ldots w_d]$ to an output node. The input layer does not perform any computation in its own right. The linear function $\overline{W} \cdot \overline{X} = \sum_{i=1}^{d} w_i x_i$ is computed at the output node. Subsequently, the sign of this real value is used in order to predict the dependent variable of X. Therefore, the prediction $\hat{y}$ is computed as follows:

$$\hat{y} = \text{sign}\{\overline{W} \cdot \overline{X}\} = \text{sign}\{\sum_{j=1}^{d} w_j x_j\}$$

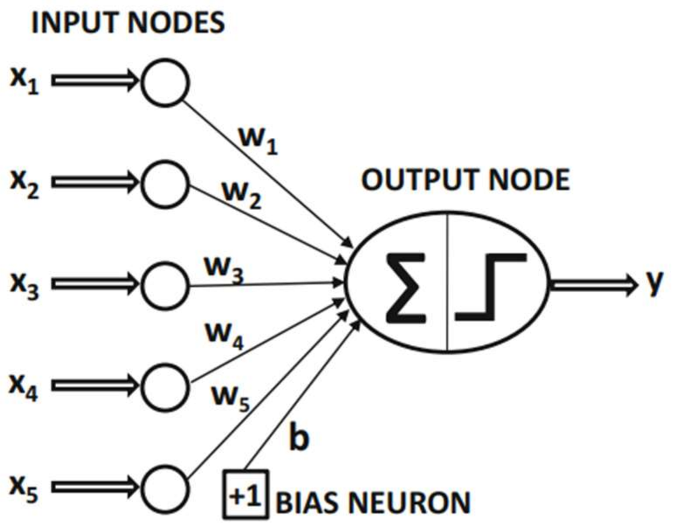
- The sign function maps a real value to either +1 or −1, which is appropriate for binary classification.
- The error of the prediction is $E(\overline{X}) = y - \hat{y}$

- In cases where the error value $E(\overline{X})$ is nonzero, the weights in the neural network need to be updated in the (negative) direction of the error gradient.

- The edges from the input to the output contain the weights $w_1 \ldots w_d$ with which the features are multiplied and added at the output node.
- Subsequently, the sign function is applied in order to convert the aggregated value into a class label. **The sign function serves the role of an activation function**. Different choices of activation functions can be used to simulate different types of models used in machine learning, like least-squares regression with numeric targets, the support vector machine, or a logistic regression classifier.
- We need to incorporate an additional bias variable b that captures the invariant part (when binary class distribution is highly imbalanced) of the prediction:

$$\hat{y} = \text{sign}\{\overline{W} \cdot \overline{X} + b\} = \text{sign}\{\sum_{j=1}^{d} w_j x_j + b\}$$

- The bias can be incorporated as the weight of an edge by using a bias neuron. This is achieved by adding a neuron that always transmits a value of 1 to the output node. The weight of the edge connecting the bias neuron to the output node provides the bias variable. An example of a bias neuron is shown in Figure 1.3(b).

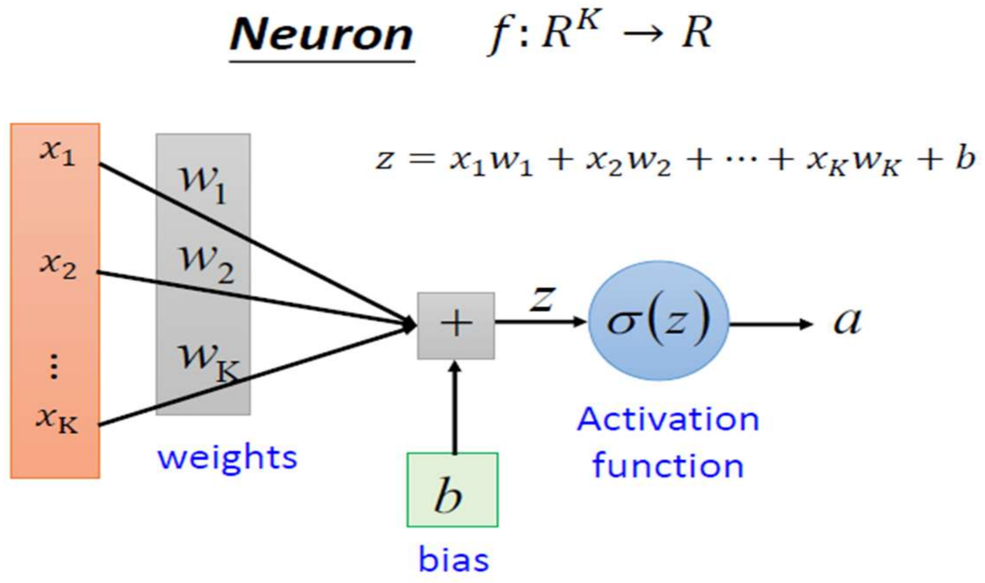


(b) Perceptron with bias

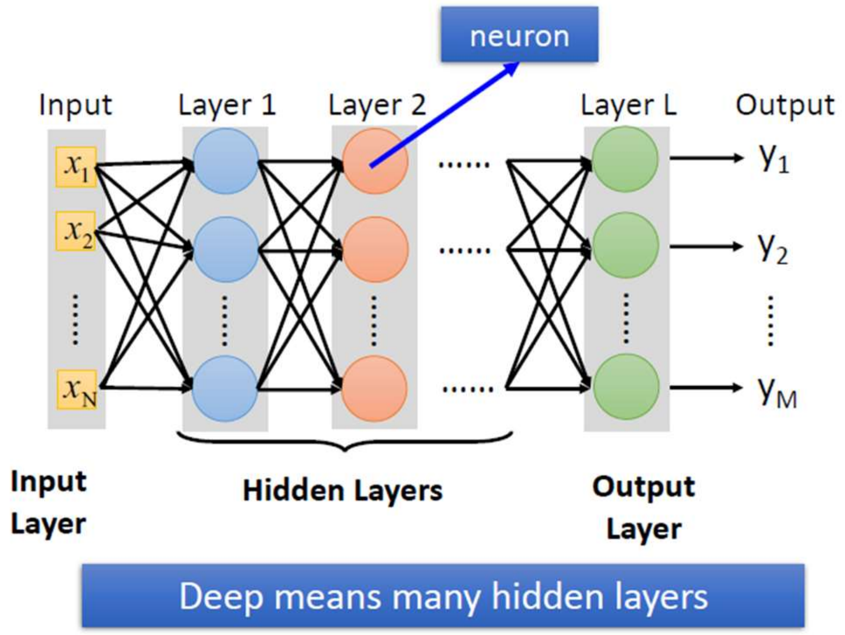
https://www.geeksforgeeks.org/effect-of-bias-in-neural-network/

Figure 1.3: The basic architecture of the perceptron

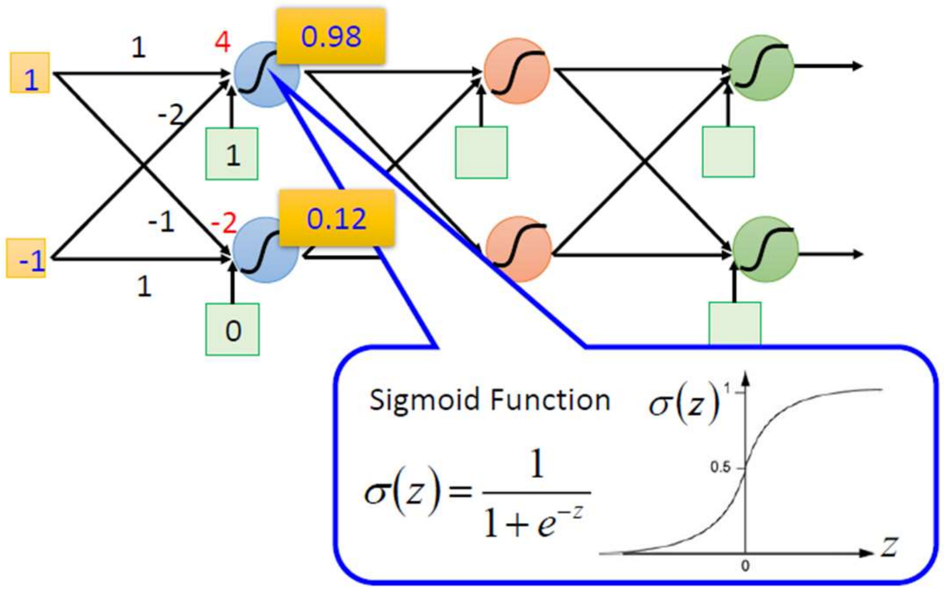
Elements of neural network



Neural Network

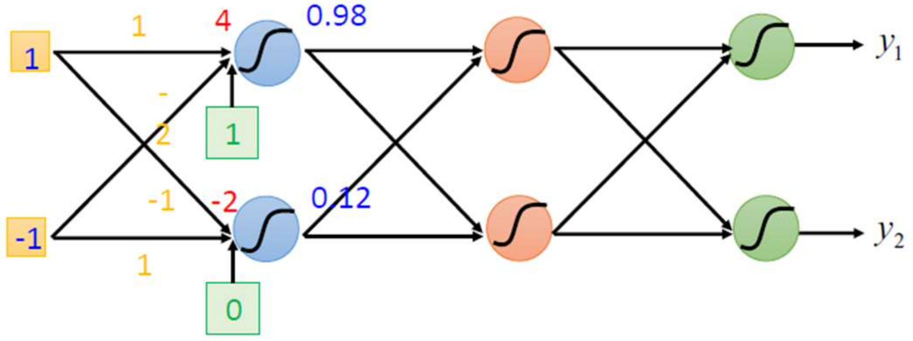


Example of Neural Network



https://keisan.casio.com/exec/system/15157249643325

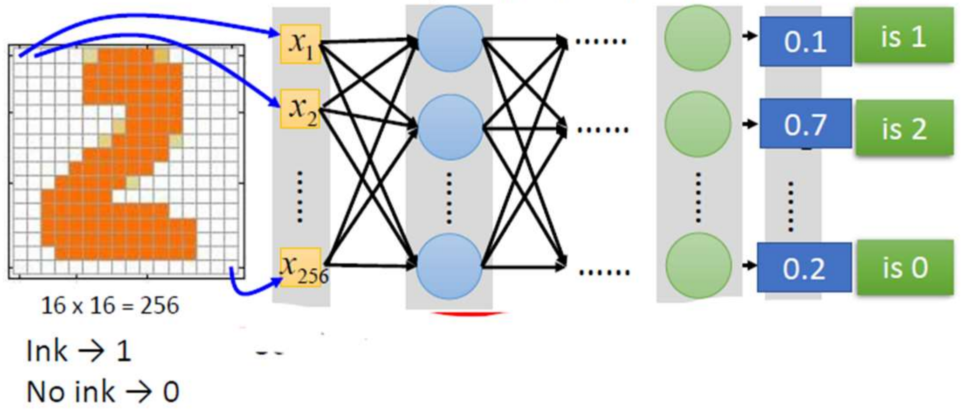
Matrix Operation



$$\sigma(\underbrace{\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\begin{bmatrix} 4 \\ -2 \end{bmatrix}}) = \begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$$

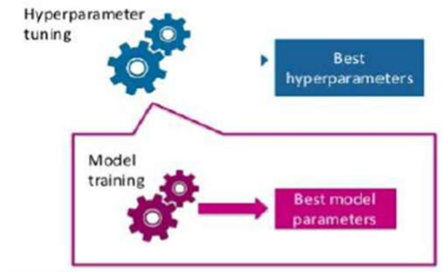
Neural Network Parameters

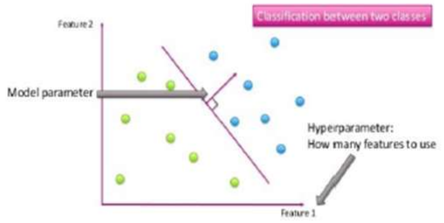
$\theta = \{W^1, b^1, W^2, b^2, \cdots W^L, b^L\}$



Parameters vs Hyperparameters

- A model parameter is a variable of the selected model which can be estimated by fitting the given data to the model.
- Hyperparameter is a parameter from a prior distribution; it captures the prior belief before data is observed.
  - These are the parameters that control the model parameters
  - In any machine learning algorithm, these parameters need to be initialized before training a model.





Neural Network: **Parameters vs Hyperparameters**

- Parameters:
  - $W^1, b^1, W^2, b^2, \cdots W^L, b^L$
- Hyperparameters:
  - Learning rate (α) in gradient descent
  - Number of iterations in gradient descent
  - Number of layers in a Neural Network
  - Number of neurons per layer in a Neural Network
  - Activations Functions
  - Mini-batch size
  - Regularizations parameters

Definition of learning

- We say, we are learning something when the performance is improving with our experience.
- Learning = Improving with experience at some task.
- Human’s Learn from experience

- A computer program is said to *learn* from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks T, as measured by P , improves with experience E.
- A computer program which learns from experience is called a machine learning program or simply a learning program

Examples

- Handwriting recognition learning problem
  - **Task T :** Recognizing and classifying handwritten words within images
  - **Performance P :** Percent of words correctly classified
  - **Training experience E :** A dataset of handwritten words with given classifications
- A robot driving learning problem
  - **Task T** : Driving on highways using vision sensors
  - **Performance P** : Average distance traveled before an error
  - **Training experience E :** A sequence of images and steering commands recorded while observing a human driver

Machine Learning

- Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.

- **The primary aim is to allow the computers learn automatically** without human intervention or assistance and adjust actions accordingly.

Terminologies of Machine Learning

- **Model:** A model is a **specific representation** learned from data by applying some machine learning algorithm(OR is the output of a machine learning algorithm run on data). A model is also called **hypothesis**.
- **Feature:** A feature is an individual measurable property of our data. A set of numeric features can be conveniently described by a **feature vector**. Feature vectors are fed as input to the model.

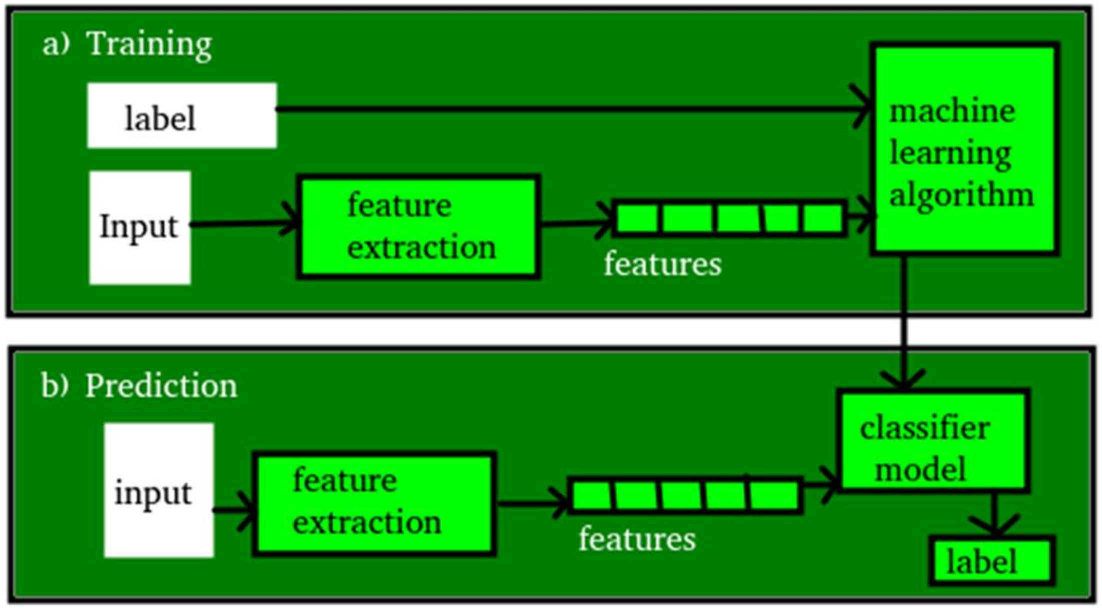
For example, in order to predict a fruit, there may be features like color, smell, shape, **etc. Note:** Choosing informative, discriminating and independent features is a crucial step for effective algorithms. We generally employ a **feature extractor** to extract the relevant features from the raw data.

- A **feature vector** is a vector containing multiple elements about an object.
- Putting feature vectors for objects together can make up a feature space **or feature space plot or scatter plot**.
- **Target (Label):** A target variable or label is the value to be predicted by our model. For the fruit example discussed in the features section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc

Terminologies of Machine Learning (cont.)

- **Training data set:** The set of data collected in order to train the machine.
- **Testing data set:** The set of data collected in order to test the machine for evaluating its performance.
- **Training:** The idea is to give a set of inputs(features) and it's expected outputs (labels), so after training we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- **Prediction:** Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label). But make sure if the machine performs well on unseen data, then only we can say the machine performs well.

Terminologies of Machine Learning (cont.)



The above figure clears the concepts/terminologies of ML

Machine Learning Models

- ***A machine learning model is defined as a mathematical representation of the output of the training process.***

- Machine learning is the study of different algorithms that can improve automatically through experience & old data and build the model.

- A machine learning model is similar to computer software designed to recognize patterns or behaviors based on previous experience or data

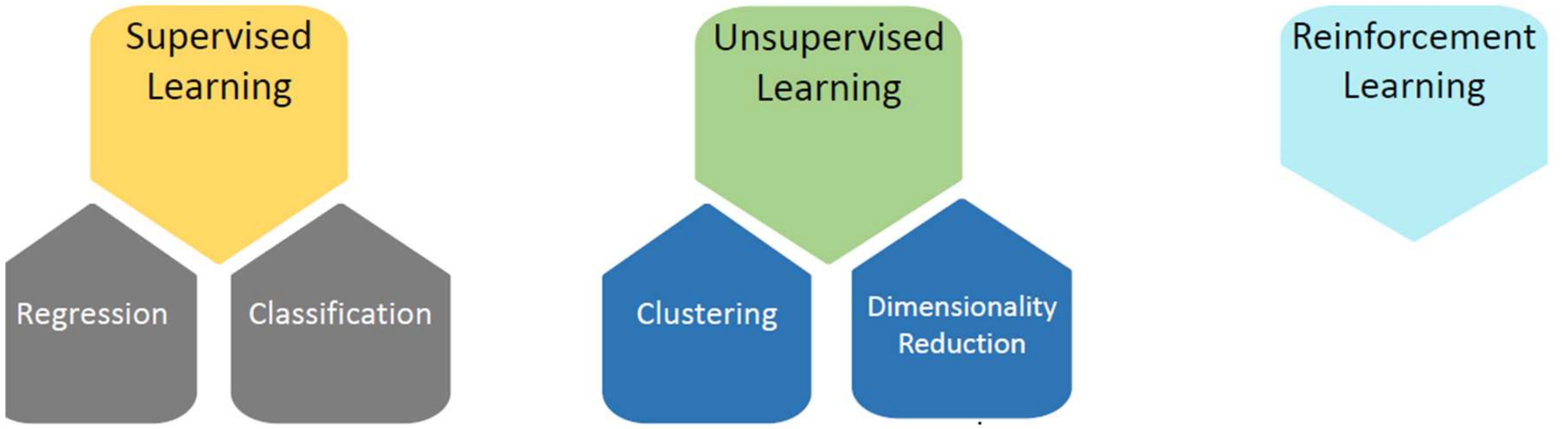
- The learning algorithm discovers patterns within the training data, and it outputs an ML model which captures these patterns and makes predictions on new data.

Contd..

- Machine Learning models can be understood as a program that has been trained to find patterns within new data and make predictions.
- These models are represented as a mathematical function that takes requests in the form of input data, makes predictions on input data, and then provides an output in response.

.

Types of ML algorithms



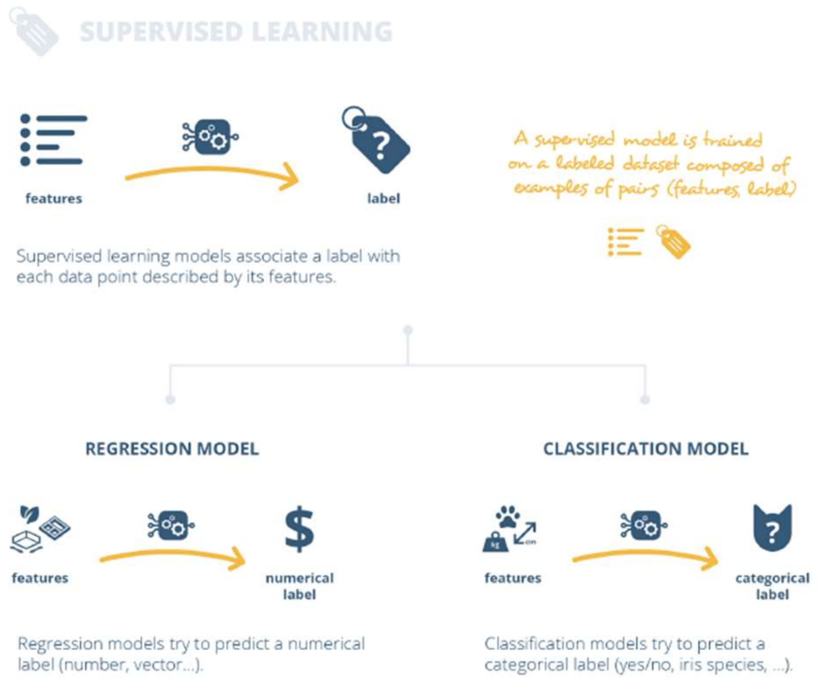
- Medical diagnosis
- Spam filtering
- Weather forecasting
- Image classification
- Fraud detection, etc.

- Targeted marketing
- Customer segmentation
- Image color compression
- Structure discovery
- Meaningful compression, etc.

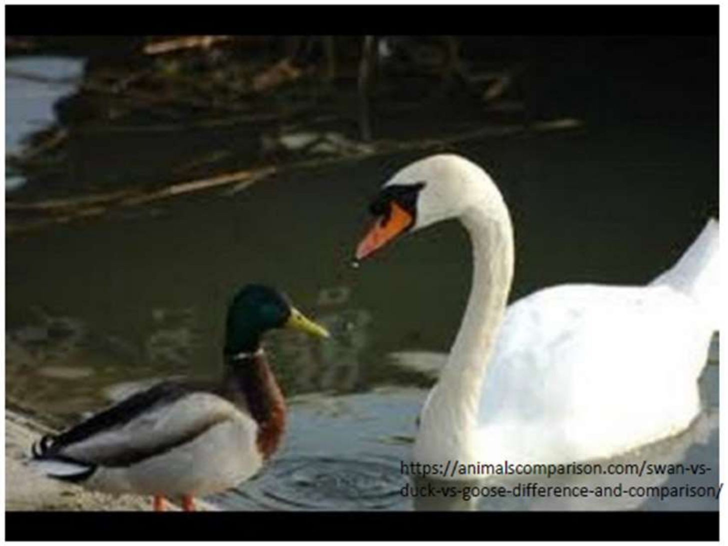
- Game AI
- Robot navigation
- Self driving cabs
- Real-time decisions
- Skill acquisition, etc.

Supervised Learning

- The data is labelled
- Label acts like a supervisor to guide the learning process.


SUPERVISED LEARNING
features
label
A supervised model is trained on a labeled dataset composed of examples of pairs (features, label)
Supervised learning models associate a label with each data point described by its features.
REGRESSION MODEL
features
numerical label
Regression models try to predict a numerical label (number, vector...).
CLASSIFICATION MODEL
features
categorical label
Classification models try to predict a categorical label (yes/no, iris species, ...).

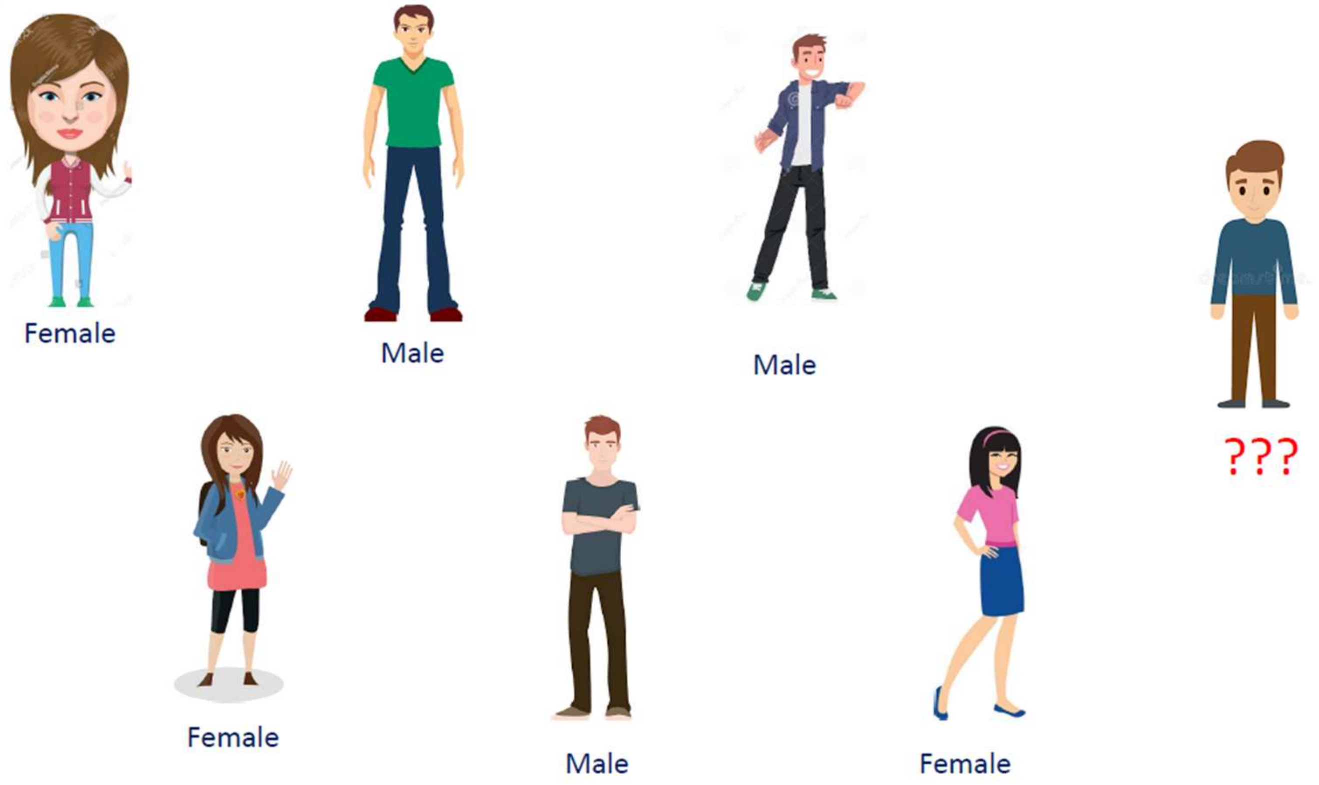
Example: Classification – Swan Vs. Duck


https://animalscomparison.com/swan-vs-duck-vs-goose-difference-and-comparison/

- If the neck length is large
  - ➔ Swan
- Else
  - ➔ Duck

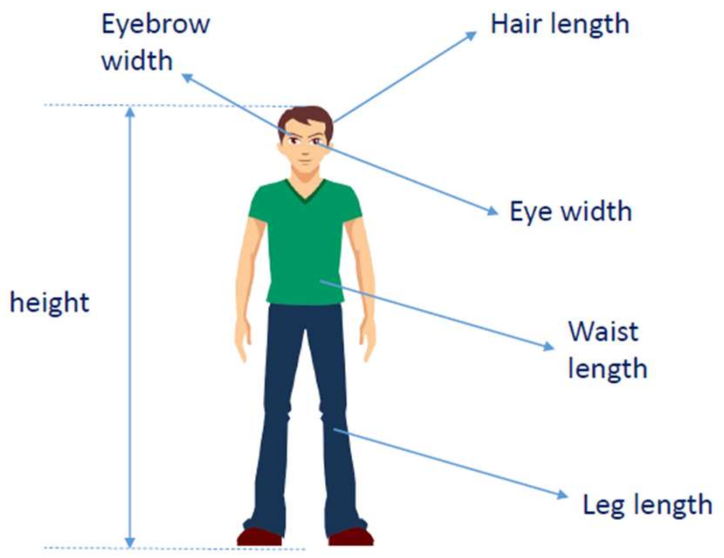
➔We need not explicitly define the rules, machine learning algorithm automatically learns from the given data

The data is labelled



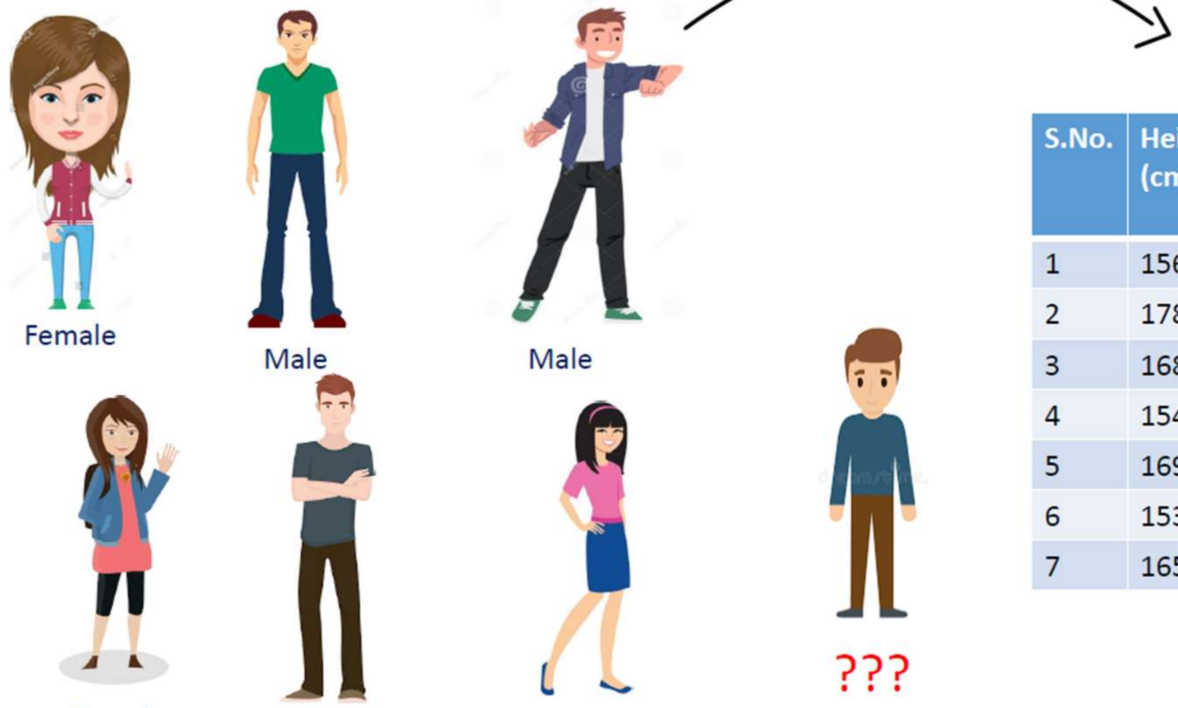
Supervised Learning

Feature Extraction



Supervised Learning

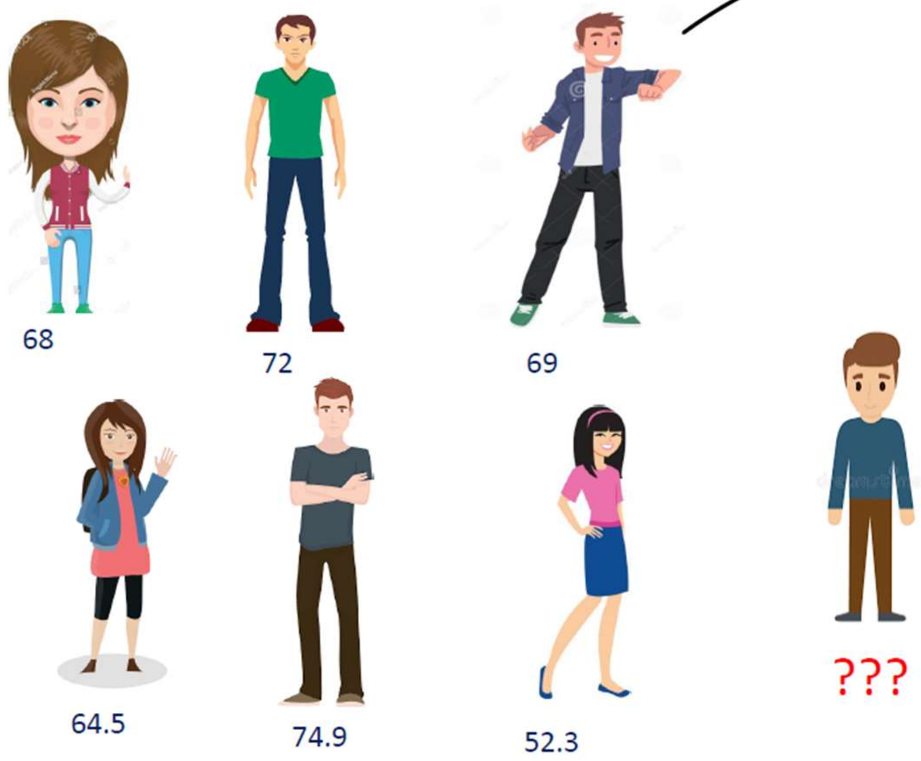
Classification



S.No.	Height (cm)	waist length (cm)	Gender
1	156	28	Female
2	178	32	Male
3	168	30	Male
4	154	29	Female
5	169	30	Male
6	153	24	Female
7	165	29	???

Supervised Learning

Regression



S.No.	Height (cm)	waist length (cm)	Weight (Kg)
1	156	28	68
2	178	32	72
3	168	30	69
4	154	29	64.5
5	169	30	74.9
6	153	24	52.3
7	165	29	???

Supervised Learning

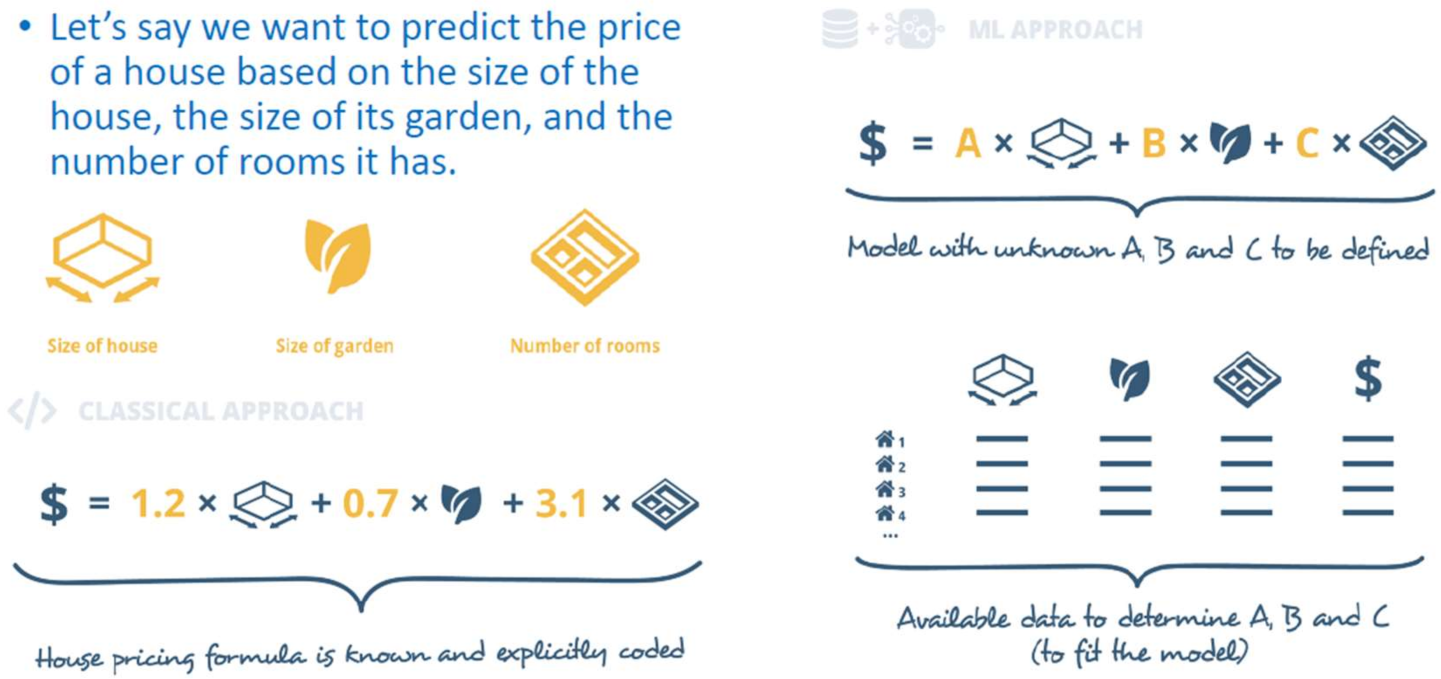
Regression or classification ?

Problem	Regression or Classfication?
E-mail spam and non-spam filtering	Classification
House Price Prediction	Regression
Detection of news article type	Classification
Marks Prediction	Regression
Grade prediction	Classification
Weather forecasting	Classification
Temperature forecasting	Regression

Linear Regression

Classic Approaches vs Machine Learning

- Let’s say we want to predict the price of a house based on the size of the house, the size of its garden, and the number of rooms it has.



Size of house
Size of garden
Number of rooms
CLASSICAL APPROACH
$ = 1.2 × + 0.7 × + 3.1 ×
House pricing formula is known and explicitly coded
ML APPROACH
$ = A × + B × + C ×
Model with unknown A, B and C to be defined
$
1
2
3
4
...
Available data to determine A, B and C (to fit the model)

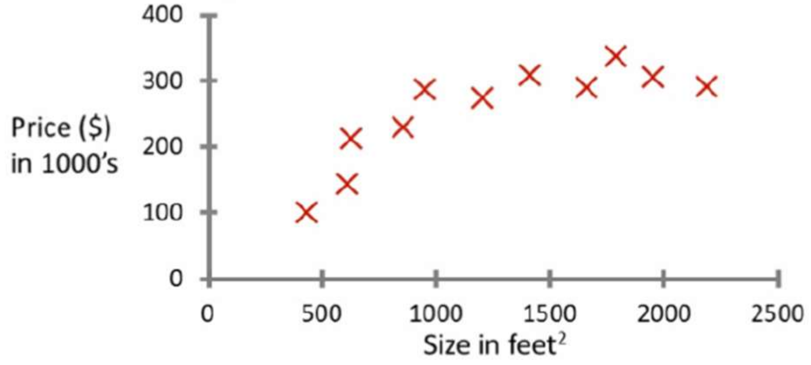
Problem Formulation

- Given a the training set of pairs $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$, where $x^{(i)} \in R^d$ and $y^{(i)}$ is a **continuous** target variable, the task is to predict for $x^{(m+j)}, j \geq 1$.

Let us start with an Example

- How do we predict housing prices
  - Collect data regarding housing prices and how they relate to size in feet.

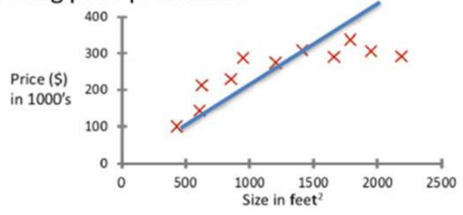
Housing price prediction.



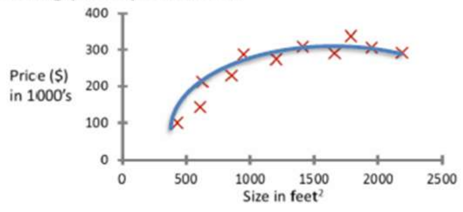
Example problem

- "Given this data, a friend has a house 750 square feet - how much can they be expected to get?"
- Straight line through data
  - Maybe $150 000
- Second order polynomial
  - Maybe $200 000

Housing price prediction.

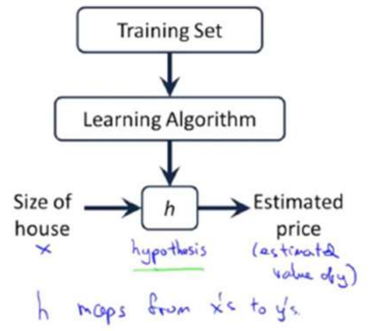


Housing price prediction.



Linear Regression

- With our training set defined - how do we use it?
  - Take training set
  - Pass into a learning algorithm
  - Algorithm outputs a function (denoted *h*)
  - This function takes an input (e.g. size of new house)
    - Tries to output the estimated value of Y
- How do we represent hypothesis *h* ?
  - Going to present *h* as $h_\theta(x) = \theta_0 + \theta_1 x$
  - Means Y is a linear function of x
  - θ_i are **parameters**

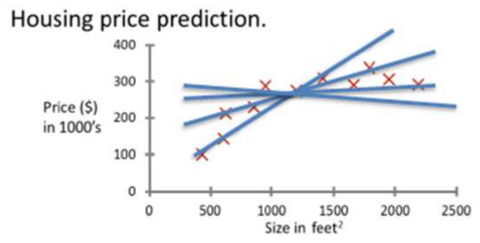


- A linear regression with one variable is also called **univariate linear regression**

- So in summary
  - A hypothesis takes in some variable
  - Uses parameters determined by a learning system
  - Outputs a prediction based on that input

Which line is best ?

- Many lines are possible !!
  Which is the best?
- A cost function lets us figure out how to fit the best straight line to our data.



- What makes a line different ?
  - Parameters θ_0, θ_1
- Which is the best line ?
  - The line that minimizes the difference between the actual and estimated prices.
- What is our objective ?
  - Choose these parameters θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples, i.e. minimize the difference between $h(x)$ and y for each/any/every example.

Objective

- Loss function: $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^i\right) - y^i\right)^2$

- $\underset{\theta_0, \theta_1}{Minimize} \; J(\theta)$

Multivariate Linear Regression

- Linear Regression with multiple input variables/features.

Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178

- More notations
  - n: number of features ($n = 4$)
  - m : number of examples (i.e. number of rows in a table)
  - $\mathbf{x}^i$: vector of the input for an example (so a vector of the four parameters for the ith input example)
    - x^3 is, for example, the 3rd house, and contains the four features associated with that house
  - $\mathbf{x}_j^i$ The value of feature j in the ith training example
    - x_2^3 is, for example, the number of bedrooms in the third house

LINEAR REGRESSION AND PERCEPTRON:

- To simulate linear regression using a single-layer neural network, you can create a neural network with just one linear neuron.
- The linear neuron has no activation function, and its output is a linear combination of the input features.

LOSS FUNCTION

- The perceptron algorithm was, heuristically designed to minimize the number of misclassifications, and convergence proofs were available that provided correctness guarantees of the learning algorithm in simplified settings. Therefore, we can still write **the (heuristically motivated) goal of the perceptron algorithm in least-squares form with respect to all training instances in a data set D**

$$\text{Minimize}_{\overline{W}}\, L = \sum_{(\overline{X},y)\in\mathcal{D}} (y-\hat{y})^2 = \sum_{(\overline{X},y)\in\mathcal{D}} \left(y - \text{sign}\{\overline{W}\cdot\overline{X}\}\right)^2$$

This type of minimization objective function is also referred to as a **loss function**.

- The least-squares form of the objective function, the sign function is nondifferentiable, with step-like jumps at specific points. The perceptron algorithm (implicitly) uses a smooth approximation of the gradient of this objective function with respect to each example.

$$\nabla L_{\text{smooth}} = \sum_{(\overline{X},y)\in\mathcal{D}} (y-\hat{y})\overline{X}$$

- The training algorithm of neural networks works by feeding each input data instance $\overline{X}$ into the network one by one (or in small batches) to create the prediction $\hat{y}$. The weights are then updated, based on the error value E(X)=(y − $\hat{y}$). Specifically, when the data point X is fed into the network, the weight vector $\overline{W}$ is updated as follows

$$\overline{W} \Leftarrow \overline{W} + \alpha(y - \hat{y})\overline{X}$$

- The parameter α regulates the learning rate of the neural network. The perceptron algorithm repeatedly cycles through all the training examples in random order and iteratively adjusts the weights until convergence is reached.
- A single training data point may be cycled through many times. Each such cycle is referred to as an epoch. One can also write the gradient descent update in terms of the error $E(\overline{X}) = (y - \hat{y})$ as follows:

$$\overline{W} \Leftarrow \overline{W} + \alpha E(\overline{X})\overline{X}$$

- The basic perceptron algorithm can be considered a stochastic gradient-descent method, which implicitly minimizes the squared error of prediction by performing gradient-descent updates with respect to randomly chosen training points.
- The assumption is that the neural network cycles through the points in random order during training and changes the weights with the goal of reducing the prediction error on that point. Non-zero updates are made to the weights only when $y \neq \hat{y}$, which occurs only when errors are made in prediction. In mini-batch stochastic gradient descent, the aforementioned updates of Above Equation are implemented over a randomly chosen subset of training points S:

$$\overline{W} \Leftarrow \overline{W} + \alpha \sum_{\overline{X} \in S} E(\overline{X})\overline{X}$$

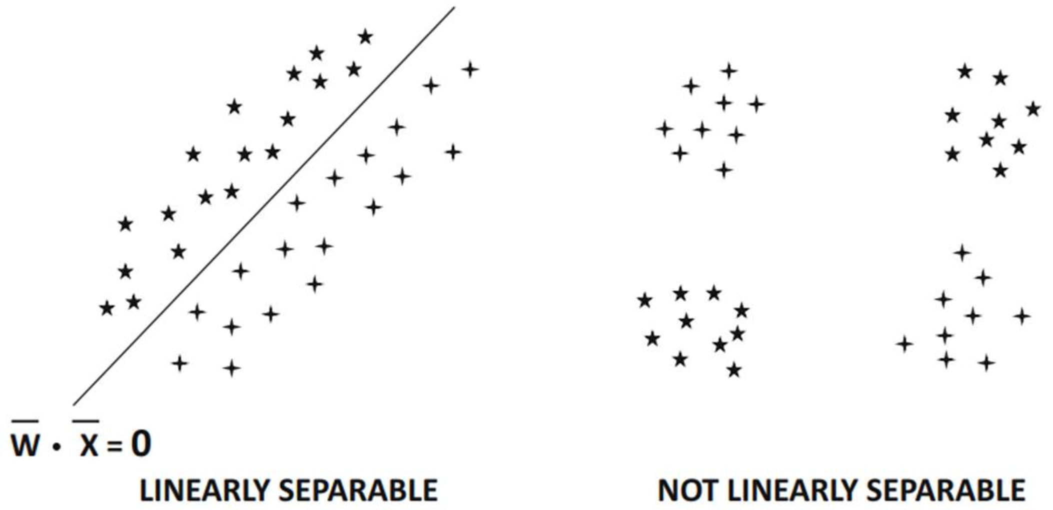
What is the Perceptron Doing?

- Tries to find a *linear separator* $\overline{W} \cdot \overline{X} = 0$ between the two classes.

- Ideally, all positive instances ($y = 1$) should be on the side of the separator satisfying $\overline{W} \cdot \overline{X} > 0$.

- All negative instances ($y = -1$) should be on the side of the separator satisfying $\overline{W} \cdot \overline{X} < 0$.

- Examples of linearly separable and inseparable data are shown below:



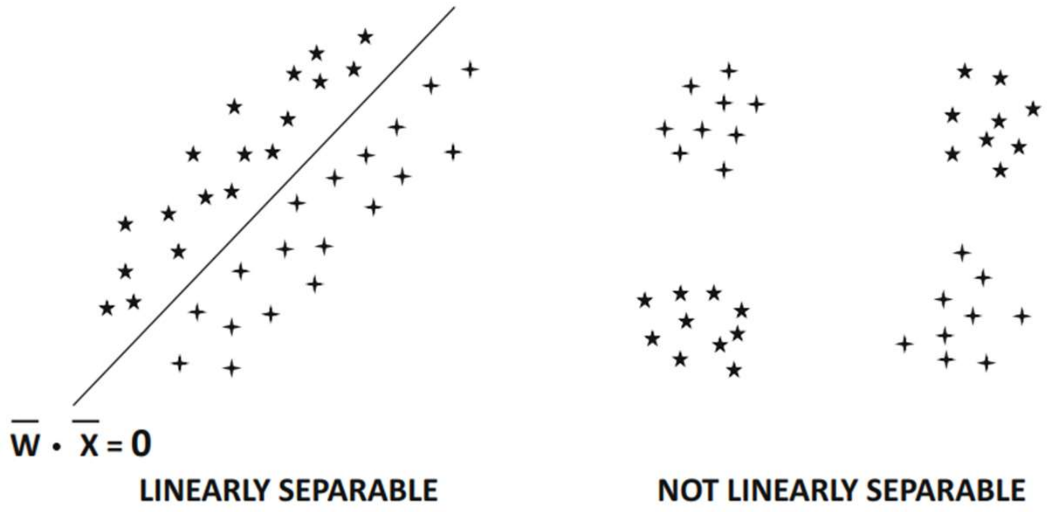
$\overline{W}.\overline{X} = 0$, linear separator between 2 classes

Positive instances (y=1) on side of separator $\overline{W}.\overline{X} > 0$

Negative instances (y=-1) on side of separator $\overline{W}.\overline{X} < 0$

Examples of linearly separable and inseparable data in two classes

- The type of model proposed in the perceptron is a linear model, in which the equation $\overline{W} \cdot \overline{X} = 0$ defines a linear hyperplane. Here, $\overline{W} = (w_1 \ldots w_d)$ is a d-dimensional vector that is normal to the hyperplane. Furthermore, the value of $W \cdot X$ is positive for values of $\overline{X}$ on one side of the hyperplane, and it is negative for values of $\overline{X}$ on the other side. This type of model performs particularly well when the data is linearly separable. Examples of linearly separable and inseparable data are shown below:



$\overline{W}.\overline{X} = 0$, linear separator between 2 classes

Positive instances (y=1) on side of separator $\overline{W}.\overline{X} > 0$

Negative instances (y=-1) on side of separator $\overline{W}.\overline{X} < 0$

Examples of linearly separable and inseparable data in two classes

What Objective Function Is the Perceptron Optimizing?

- Can we find a smooth loss function, whose gradient turns out to be the perceptron update? The number of classification errors in a binary classification problem can be written in the form of a 0/1 loss function for training data point $(\overline{X}_i, y_i)$ as follows:

$$L_i^{(0/1)} = \frac{1}{2}(y_i - \text{sign}\{\overline{W} \cdot \overline{X_i}\})^2 = 1 - y_i \cdot \text{sign}\{\overline{W} \cdot \overline{X_i}\}$$

The simplification to the right-hand side of the above objective function is obtained by setting both y_i^2 and $\text{sign}\{\overline{W} \cdot \overline{X_i}\}^2$ to 1, since they are obtained by squaring a value drawn from $\{-1, +1\}$. However, this objective function is not differentiable, because it has a staircase-like shape, especially when it is added over multiple points.

- This objective function is defined by dropping the sign function in the above 0/1 loss and setting negative values to 0 in order to treat all correct predictions in a uniform and lossless way:

$$L_i = \max\{-y_i(\overline{W} \cdot \overline{X_i}), 0\}$$

- Using calculus to verify gradient of this smooth objective function leads to the perceptron update, and the update of the perceptron is essentially $\overline{W} \Leftarrow \overline{W} - \alpha \nabla_W L_i$.

  The modified loss function to enable gradient computation of a non-differentiable function is also referred to as a **smoothed surrogate loss function**.

N.B.: The Loss function tells us how far we are from a desired solution. (Max of –ve value and 0 is 0, indicating loss is 0)