Introduction to Linear Classifiers and Key Deep Learning Concepts

(12 September 2023)

Shweta Birla Dhakonia, Ph.D.

Senior Project Scientist

Translational Bioinformatics Group,

ICGEB, India

Data & Machine Learning

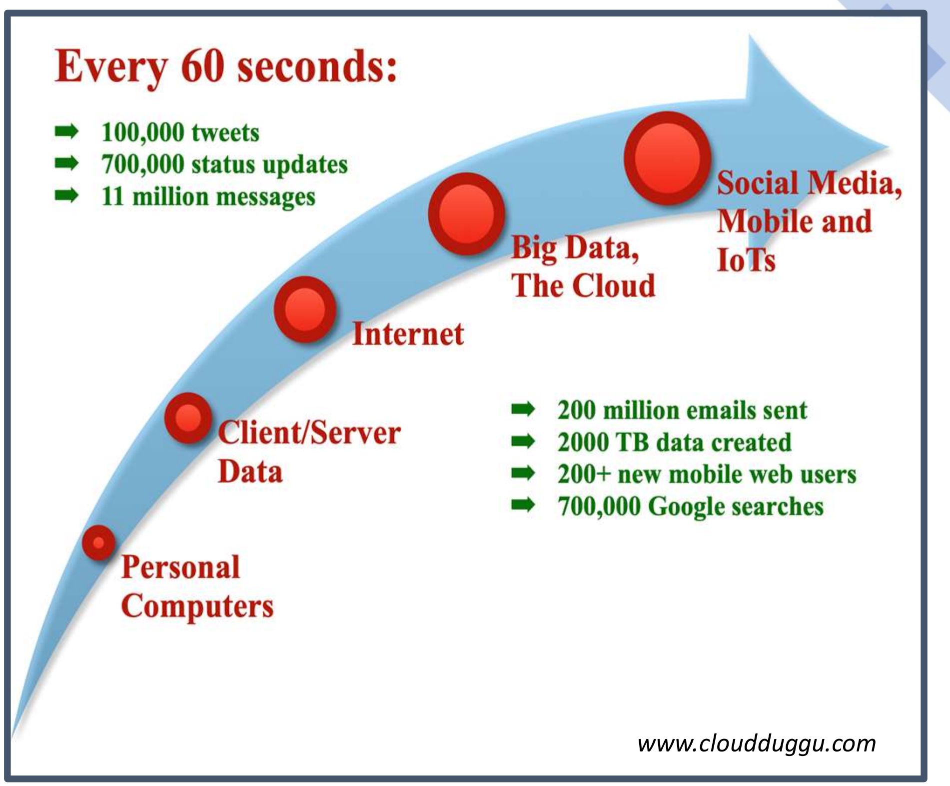
Data

- Generated at an unprecedented rate - technological advancements & and increasing digitalization
- Enormity and complexity- surpass the human ability to decipher hidden relevance/significance

↓

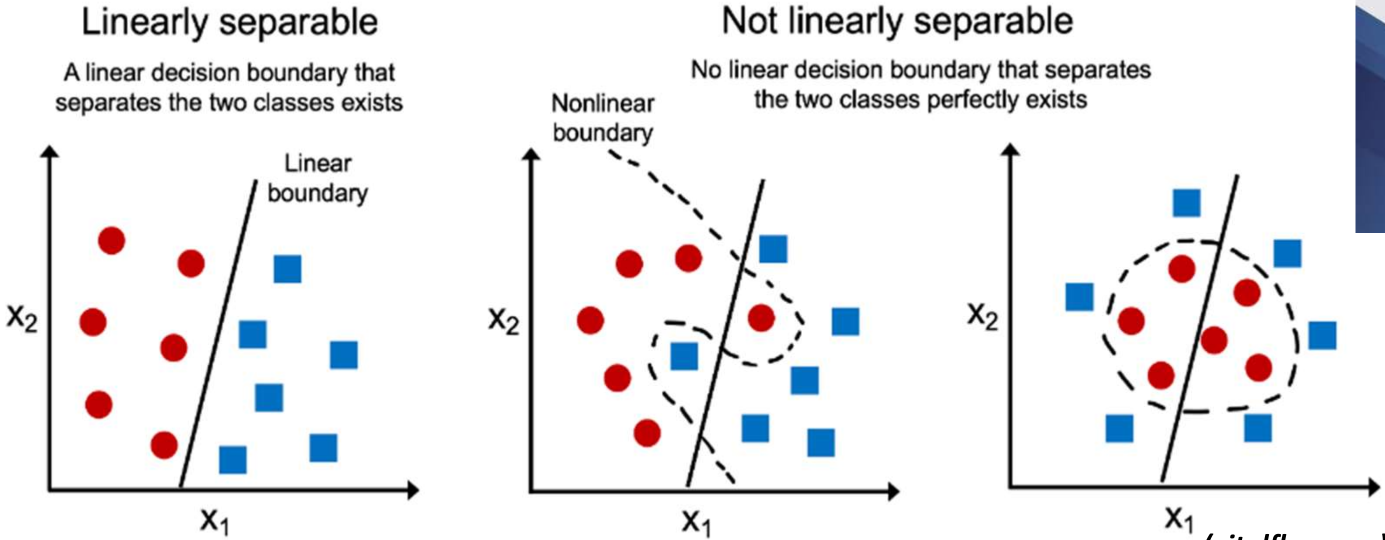
Machine learning (ML)

- To build models/machines that can learn from data and make predictions.
- Designed to analyze large and complex datasets, discover patterns, make predictions, and automate decision-making, - extremely challenging to do at scale.



Decision boundaries

- Ability to create imaginary lines called “Decision Boundaries”
- Primary Goal: to differentiate or classify data into distinct categories
- Classifiers (algorithms): linear classifiers non-linear classifiers



(vitalflux.com)

Classifiers

Characteristic	Linear Classifier	Nonlinear Classifier
Decision Boundary	Linear (e.g., straight line, hyperplane)	Nonlinear (e.g., curves, circles, complex shapes)
Common Models	Logistic Regression, Linear SVM, Perceptron	Decision Trees, Random Forest, Kernel SVM, Neural Networks
Use Cases	Suitable when data has approximately linear relationships	Suitable when data exhibits complex or nonlinear patterns
Examples of Use	Basic image classification, spam email detection, sentiment analysis	Image recognition, speech recognition, natural language processing, complex pattern recognition
Interpretability	Often more interpretable due to simplicity	May be less interpretable due to complexity
Computational Demands	Lower computational demands	Higher computational demands, especially for deep neural networks

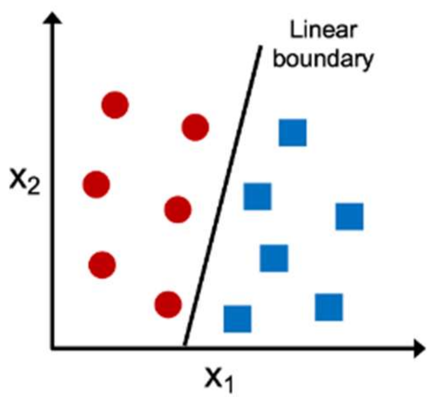
<u>Linear Classifiers (LC)</u>

"Foundation of many ML models essential for building intelligent systems"

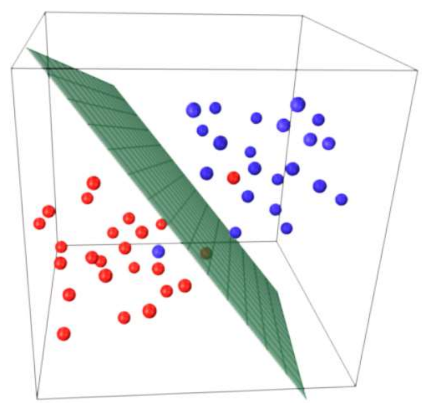
- Family of algorithms that learn to separate data into distinct classes
- Predictions - a linear combination of **input features**
- Define a decision boundary can be visualized by plotting the data in N dimensions
- Some popular linear classifiers **include SVM, logistic regression, and perceptron**

" Key idea is to find a linear decision boundary that separates different classes"

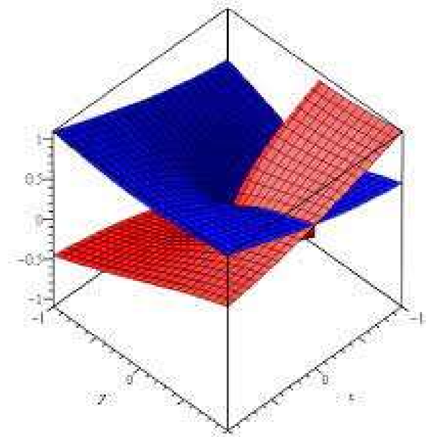
<u>A line in 2D</u>



<u>A plane in 3D</u>



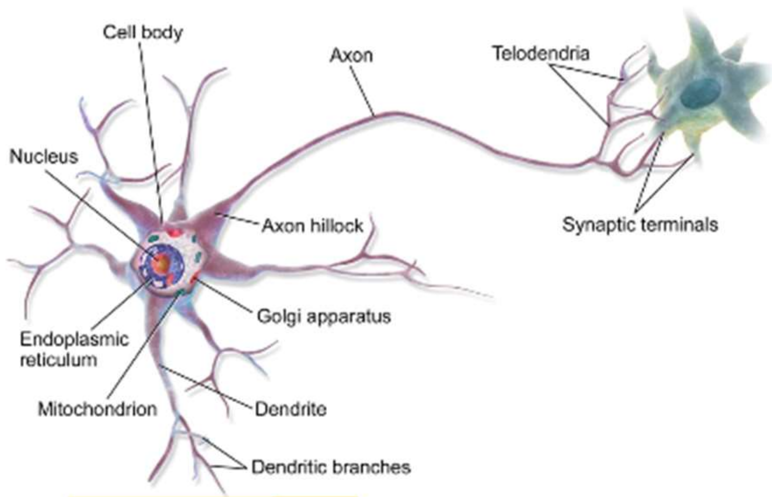
<u>A hyperplane in higher dimensions</u>



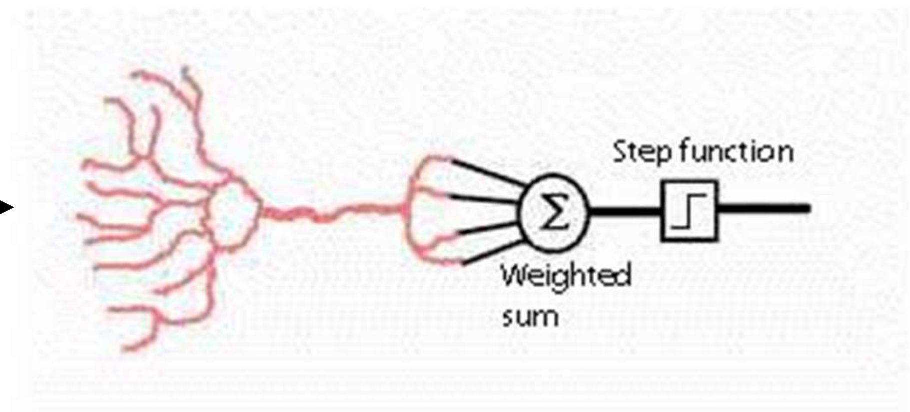
LC: Perceptron

- Perceptron mimics the way the human brain learns
- Developed in the 1950s to simulate the process of neural learning
- It was the first algorithm **capable of learning from data and making predictions**
- It provides a simple but powerful model for classification tasks

Neuron - a building block of the brain



Perceptron forms a basic unit of NN

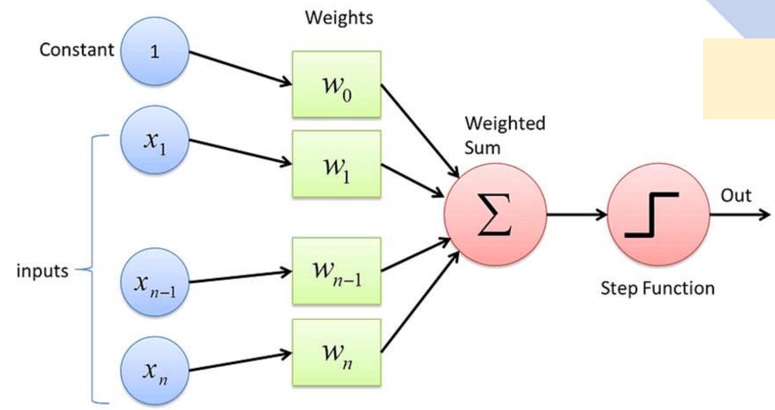


LC: Perceptron

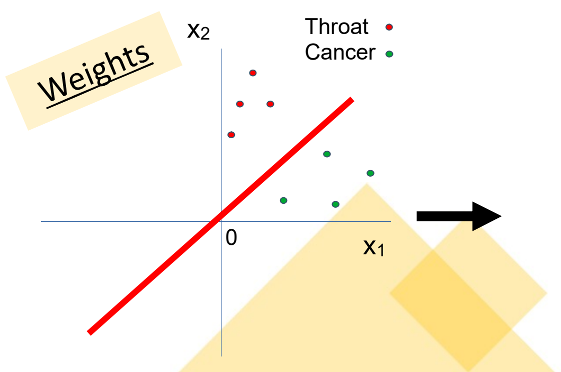
The perceptron consists of 4 parts:

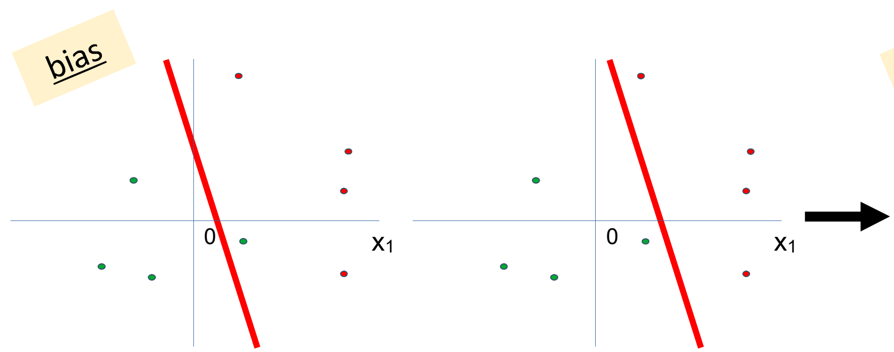
1) Input values/one input layer   2)Weights and Bias

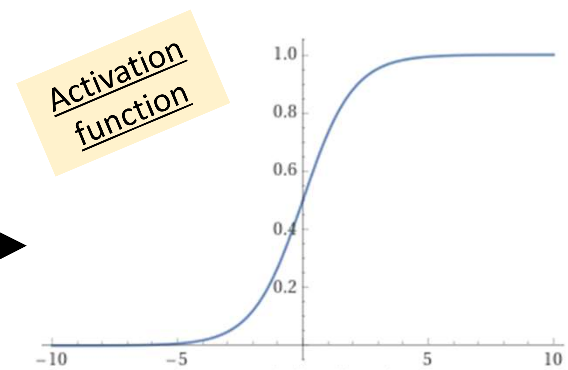
3)Net sum   4) Activation Function.



- Takes inputs, multiplies them by weights, adds a bias, sums them up, and then passes the result through an activation function =Output is binary (0 or 1).
- **Weights**: control the contribution of individual features to the decision boundary.
- **Bias:** shifts the decision boundary, allowing for more flexibility in classification. Both must be carefully tuned for best results.



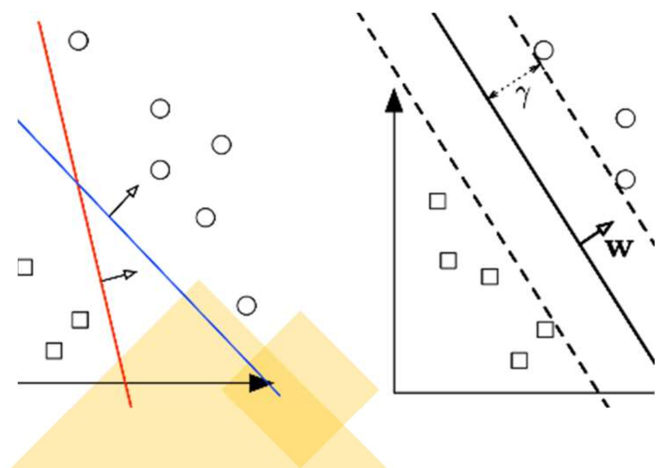


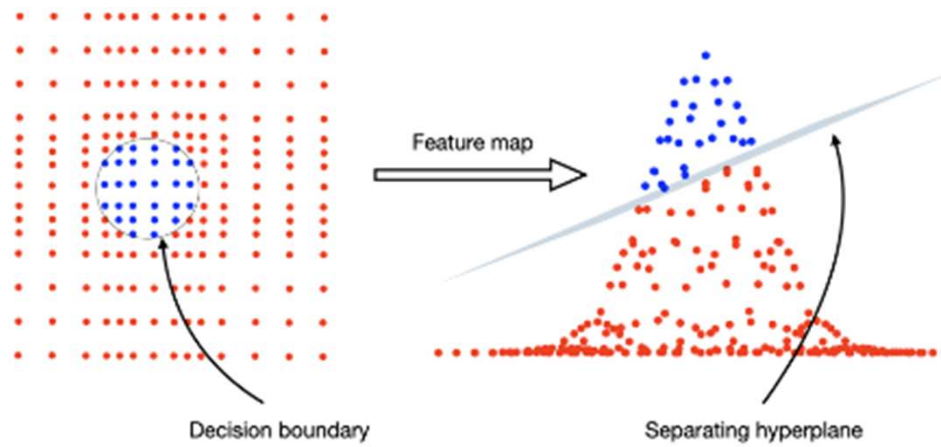


Output = sigmoid(sum)

LC: Support Vector Machines (SVM)

- SVM: Powerful linear classification algorithm
- Aims: To find a hyperplane that maximizes the margin between different classes
- Kernel function to project the data into a higher-dimensional space where the problem of finding a linear discriminant becomes easier
- Effective for binary and multiclass classification tasks

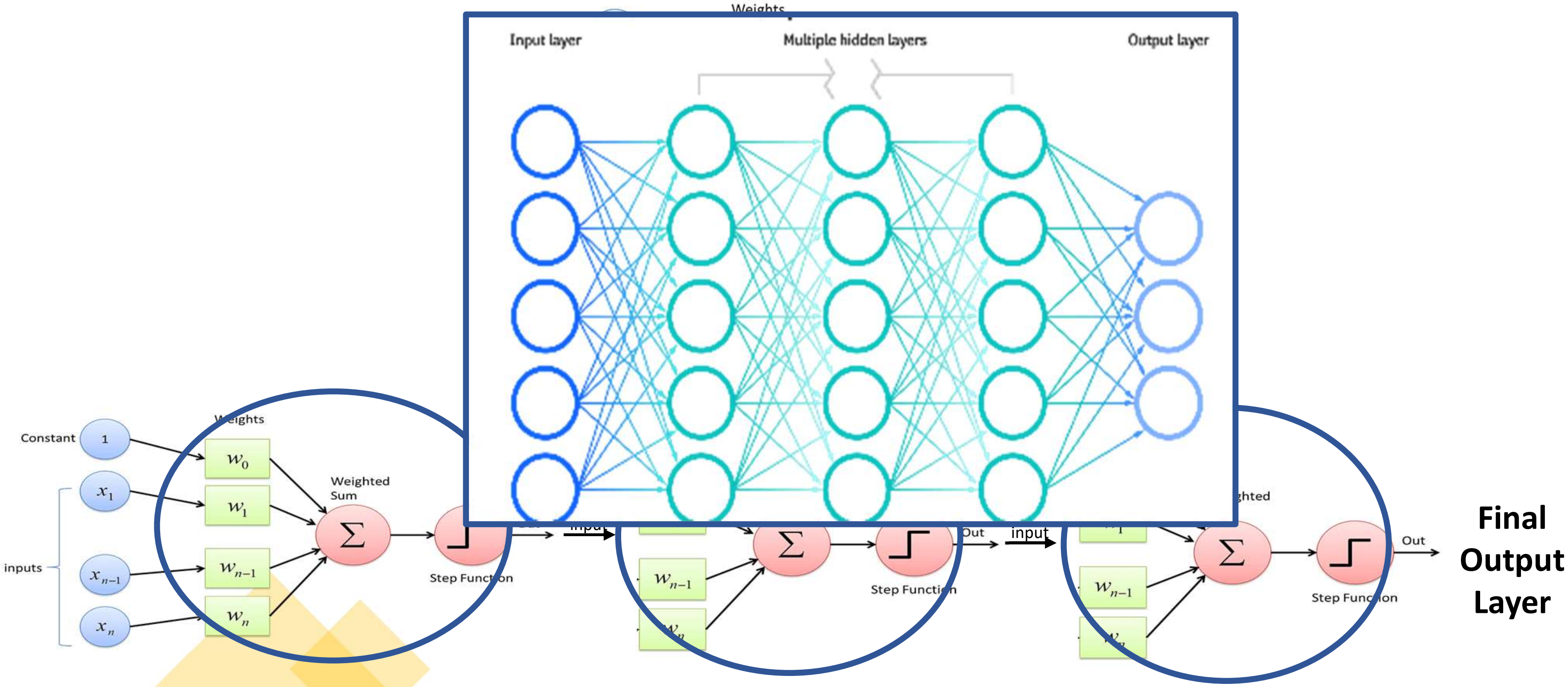




Next hands-on: Building a linear classifier using perceptron and SVM

Neural Network

- Many perceptrons combine together = NN.
- NN works the same way as the perceptron.



Forward Propagation

Essential c

1.Forward

“ During tr

- **What it**

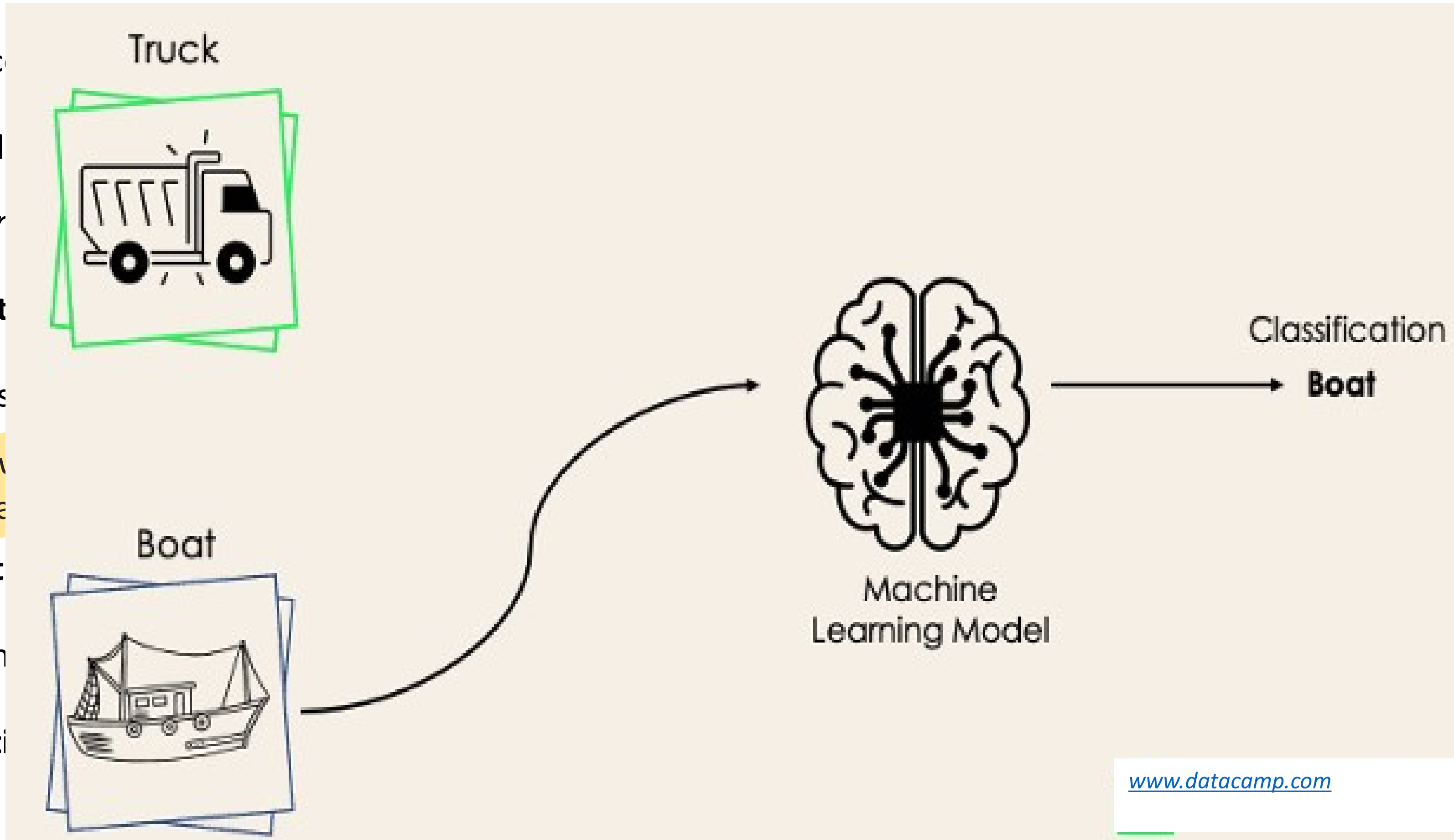
  weights

Adjusted du
to improve

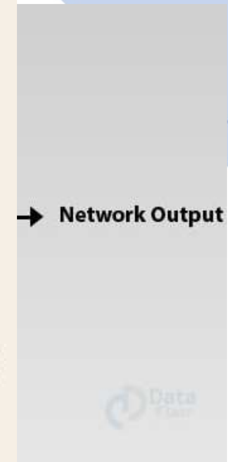
- **Weight**

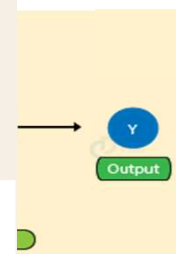
- **Bias:** sh

- **Activati**



www.datacamp.com

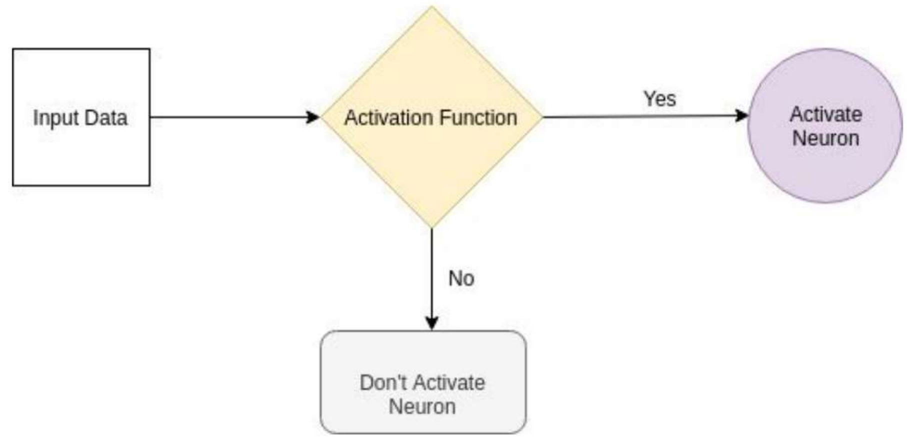


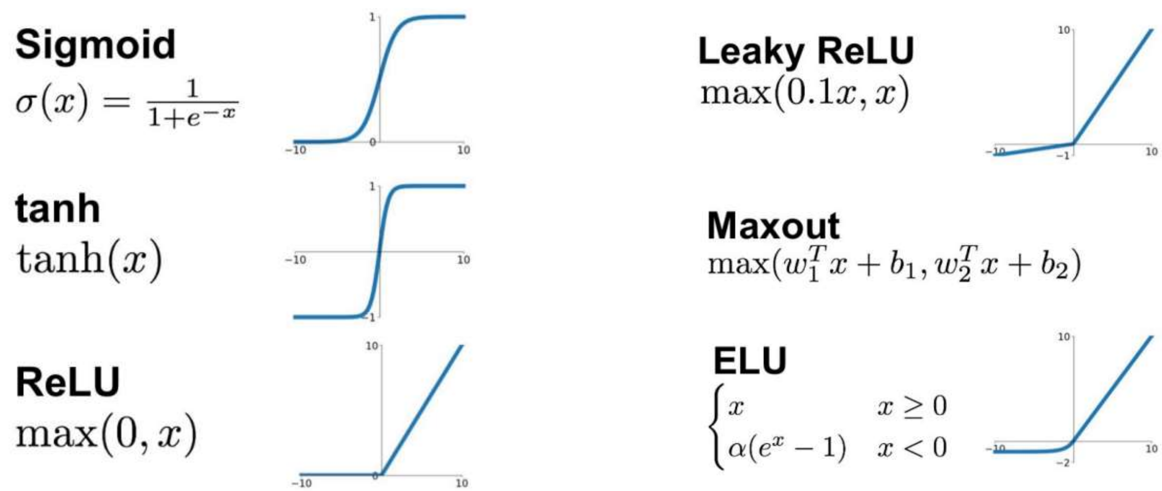


Activation Function

"Deciding authority"

- An activation function - binary switch for a perceptron - "on" or "off" based on its input.
- This function introduces non-linearity enabling NN to capture complex patterns - like image recognition and natural language processing.
- Types of activation function:

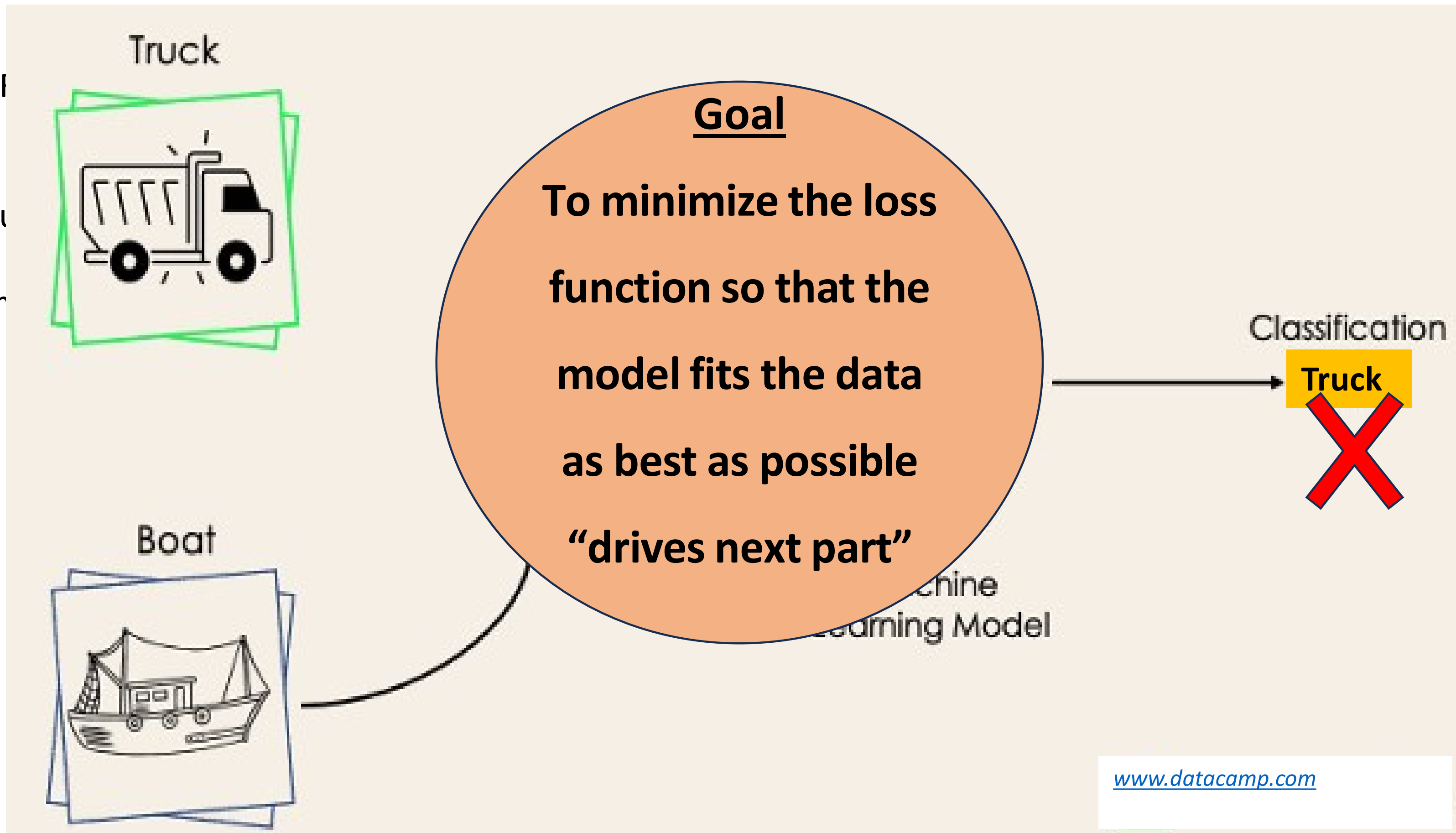




Understanding Loss Function

"Measures how bad our model is at its job"

- After
- Loss fu … good model
- Comm



Backward Propagation

2. Backward Propagation (Backpropagation):

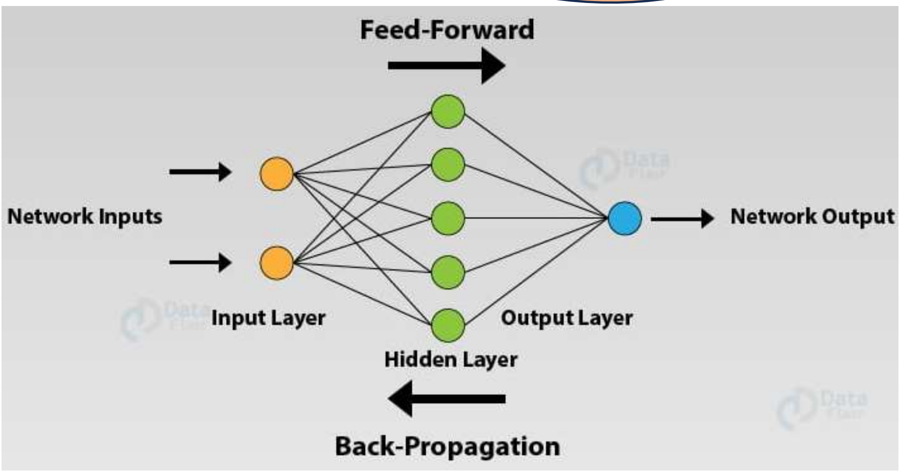
“Responsible for learning from the mistakes (errors) made during FP by adjusting the network's weights to minimize those errors”

What it is: Process involves calculating LF gradients (derivatives)/ how each weight in the network contributed to this error.

Steps:

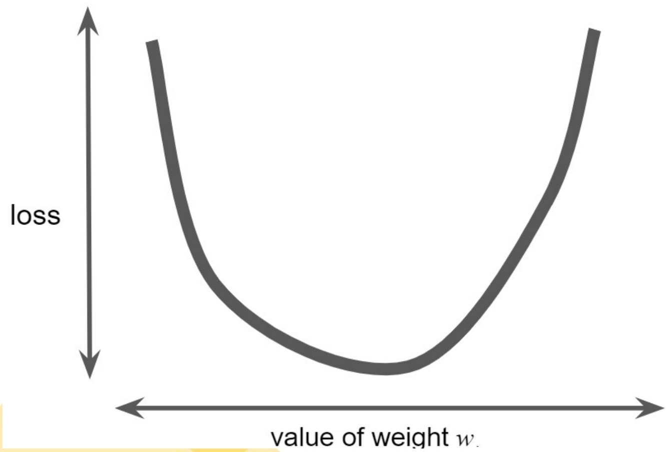
- **Gradient Calculation**: Computes gradients of the loss w.r.t. model parameters (weights and biases), indicating how to adjust them.
- **Backward Pass**: Passing gradients backward.
- **Weight Update**: Optimizes weights and biases with an algorithm like **gradient descent** to reduce the loss.
- **Iteration**: Steps repeated over the training data until the network minimizes its loss, improving its performance.

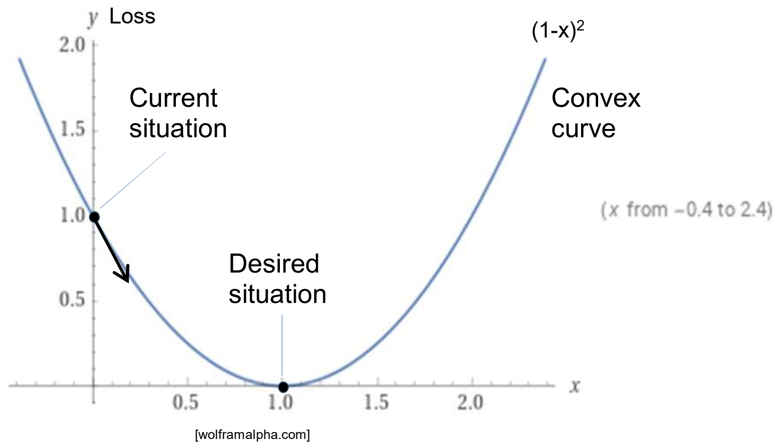
Crucial for adjusting the model parameters during training- an essential part of gradient descent



Reducing loss: Gradient Descent

- Step-by-step process by **optimization algorithms** (gradient descent) - teach the model - minimize the loss.
- Making small adjustments in weights and biases - reduce the mistakes our model is making.
- **Epoch:** An epoch represents one round of training for our model. In each epoch, the model learns from its mistakes and gets better.
- Training data is randomly shuffled and divided into **mini-batches** for computational speed-up.
- Mini-batches are used to compute each step, reaching **a local minimum** of the cost function.
- Types of gradient descent: stochastic gradient descent (SGD), Adam, RMSprop.





Reducing loss: Gradient Descent

Situation: blindfolded, standing in a hilly area → **I have a guide who will help me find the way.** **Clueless!**

Goal: Reaching the lowest point

1) **Slope =Loss function**
2) **Guide= Optimizer algorithm** – velocity + direction (Momentum)

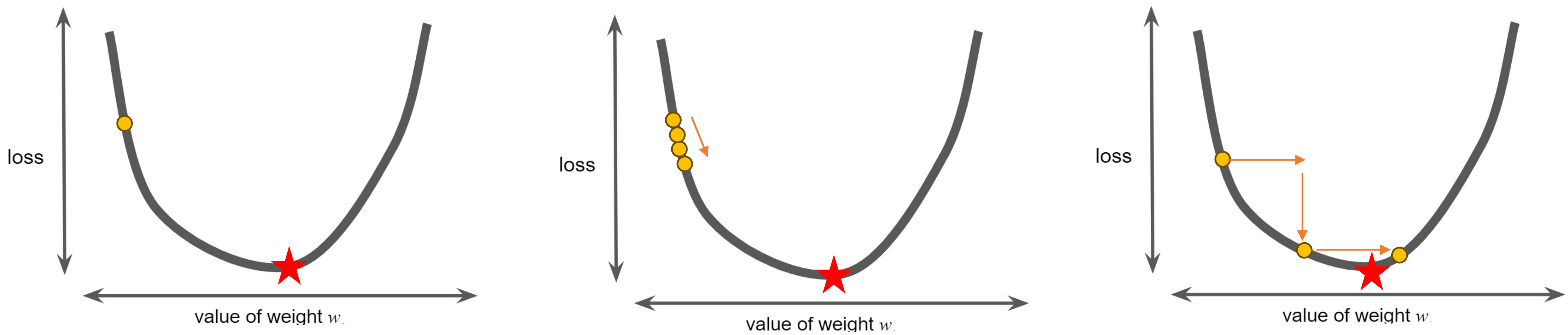
3) **Learning Rate**: How big my steps should be. “*controls how fast or slow I progress*”

- Small steps = move slowly and reach the lowest point accurately
- Larger steps= I might overshoot and miss the minimum.

4**) Epoch:** How many times do I want to follow my guide?
Each time = one epoch.
More epochs = higher chances to get closer to the lowest point.

5) **Minima**: Lowest point

Reducing loss: Gradient Descent



- **Optimizers**
- **Minima, gradient,**
- **learning rate –high or low**
- **Epochs**
- **Momentum**

- **Working of perceptron & NN**
- **Forward propagation**
- **Backward propagation,**
- **Activation function**
- **Loss function**

- **Linear classifiers**
- **Perceptron /SVM**

Adapted from https://developers.google.com/

Summarize

- Linear classifiers like perceptron and SVM are foundational algorithms in machine learning.
- Neural networks, comprised of multiple perceptrons, utilize forward and backward propagation to make predictions.
- Forward propagation involves input data, weights, bias, activation functions, and output prediction.
- Backward propagation adjusts weights based on error gradients to improve model performance.
- Gradient descent optimizes weights and biases to minimize the loss function and improve model accuracy

Thankyou