M.Tech Program

Advanced Industry Integrated Programs

Jointly offered by University and LTIMindTree

Applied Machine Learning

Knowledge partner

LTIMindtree

Implementation partner

L&T EduTech

Modules to cover….

1. Supervised Learning
2. **Advanced Learning Algos**
3. Unsupervised Learning and Recommender Systems

Advanced Learning Algos

Advanced Learning Algos

Overview of Neural Networks

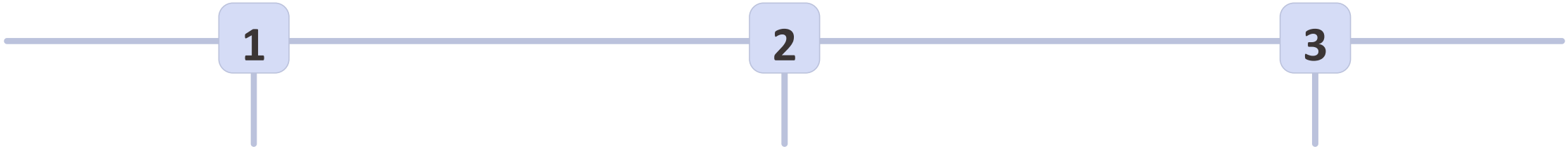
Neural networks were first invented many decades ago with the goal of mimicking how the human brain learns and thinks.

Even though today's neural networks are different from the brain, biological motivations still influence how we think about artificial neural networks.

Advanced Learning Algos

Overview of Neural Networks

Development of Neural Networks



1950s

Neural networks research began, laying the foundations for this field.

1980s-1990s

Neural networks gained popularity, with early applications in handwritten digit recognition.

2005 and beyond

Deep learning reignited interest in neural networks, leading to breakthroughs in speech recognition and computer vision.

Advanced Learning Algos

Overview of Neural Networks

Working of Neural Networks

The Human Brain

The human brain consists of neurons that send electrical impulses to each other, forming connections.

Artificial Neural Networks

Artificial neural networks use a simplified mathematical model of biological neurons, enabling complex computations and learning.

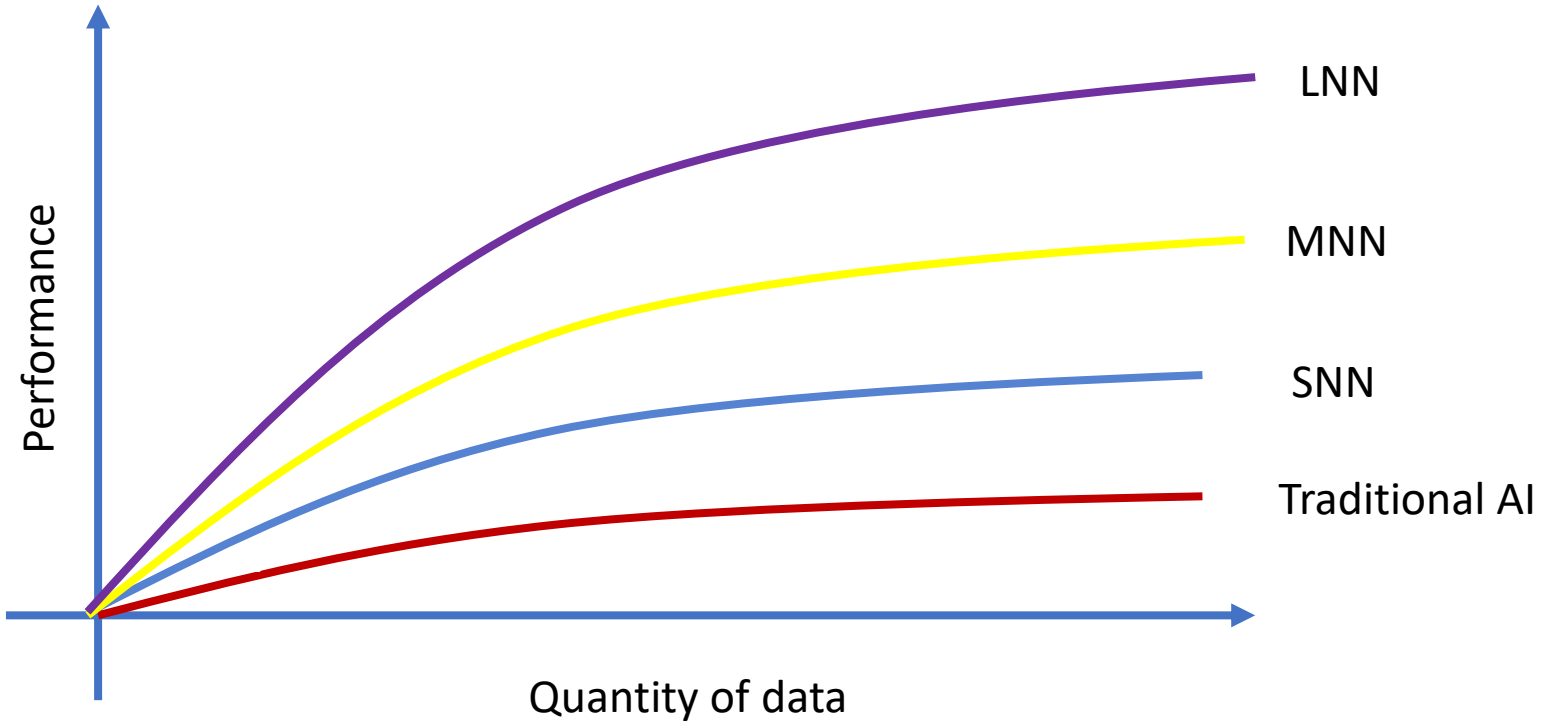
Neuron Models

Despite our limited understanding of the brain, simplified neuron models have led to powerful deep learning algorithms.

Advanced Learning Algos

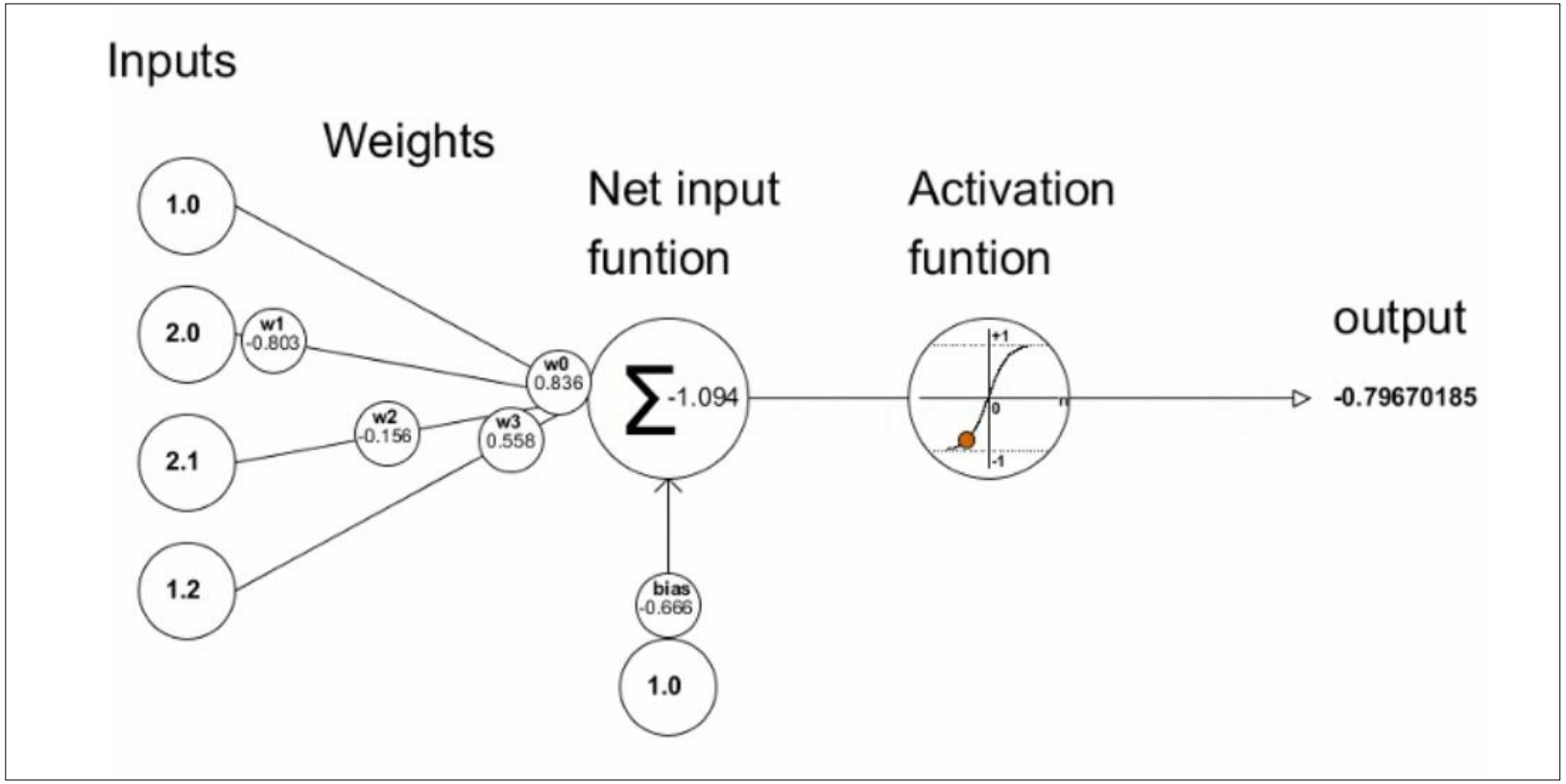
Overview of Neural Networks

Need of Neural Networks



Advanced Learning Algos

Overview of Neural Networks



[Introduction to Neural Networks and Their Key Elements… – Towards AI]

[Components of Neural Network | Neural Network Layers & Neurons (analyticsvidhya.com)]

[How Neural Networks Can Be Used For Data Mining? – GeeksforGeeks]

[Activation Functions In Neural Networks — Its Components, Uses & Types | by AnalytixLabs | Medium]

Advanced Learning Algos

Overview of Neural Networks

Forward Propagation

• **Input Layer:** Each feature in the input layer is represented by a node on the network, which receives input data.

• **Weights and Connections:** The weight of each neuronal connection indicates how strong the connection is. Throughout training, these weights are changed.

• **Hidden Layers:** Each hidden layer neuron processes inputs by multiplying them by weights, adding them up, and then passing them through an activation function. By doing this, non-linearity is introduced, enabling the network to recognize intricate patterns.

• **Output:** The final result is produced by repeating the process until the output layer is reached.

Advanced Learning Algos

Overview of Neural Networks

Backpropagation

• **Loss Calculation:** The network's output is evaluated against the real goal values, and a loss function is used to compute the difference. For a regression problem, the Mean Squared Error (MSE) is commonly used as the cost function.

• **Loss Function:** MSE

• **Gradient Descent:** Gradient descent is then used by the network to reduce the loss. To lower the inaccuracy, weights are changed based on the derivative of the loss with respect to each weight.

• **Adjusting weights:** The weights are adjusted at each connection by applying this iterative process, or backpropagation, backward across the network.

• **Training:** During training with different data samples, the entire process of forward propagation, loss calculation, and backpropagation is done iteratively, enabling the network to adapt and learn patterns from the data.

• **Activation Functions:** Model non-linearity is introduced by activation functions like the rectified linear unit (ReLU) or sigmoid. Their decision on whether to "fire" a neuron is based on the whole weighted input.

Advanced Learning Algos

Overview of Neural Networks

Demand Prediction

1 Input Feature

The input feature is the price of the T-shirt.

2 Logistic Regression

Logistic regression can fit a sigmoid function to predict the probability of being a top seller.

3 Neuron Model

Logistic regression units can be thought of as simplified models of neurons in the brain.

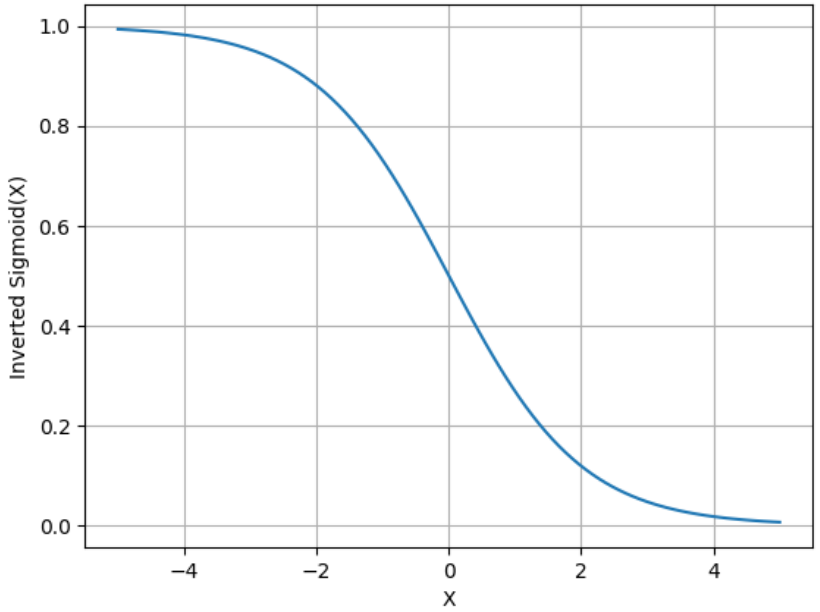
4 Neuron Outputs

Neurons input one or more numbers, compute a formula, and output a probability.

Advanced Learning Algos

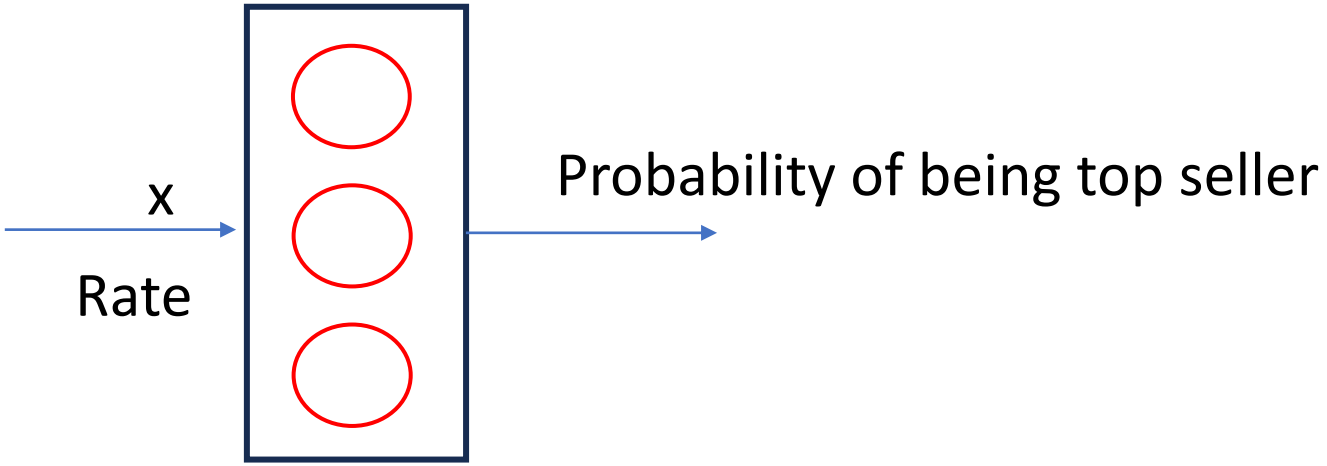
Overview of Neural Networks

Demand Prediction



$$x = price$$

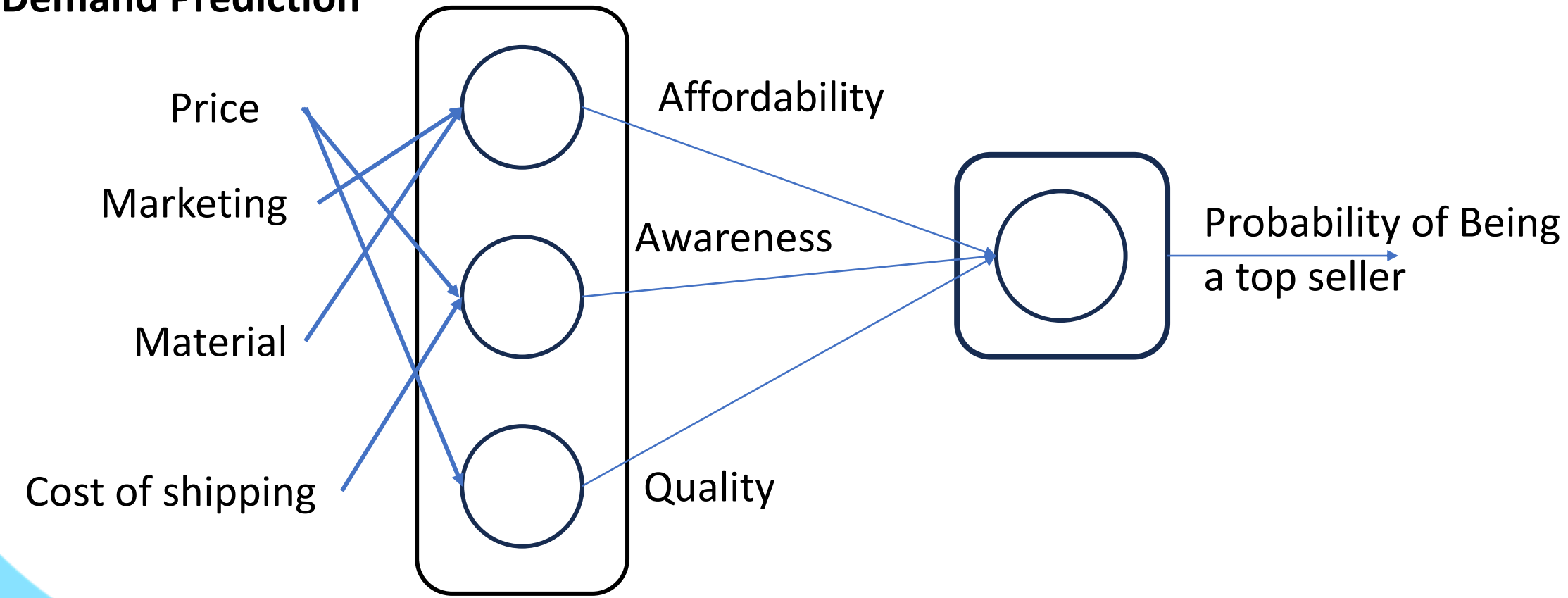
$$activation = f(x) = \frac{1}{1 + e^{-(ax+b)}}$$



Advanced Learning Algos

Overview of Neural Networks

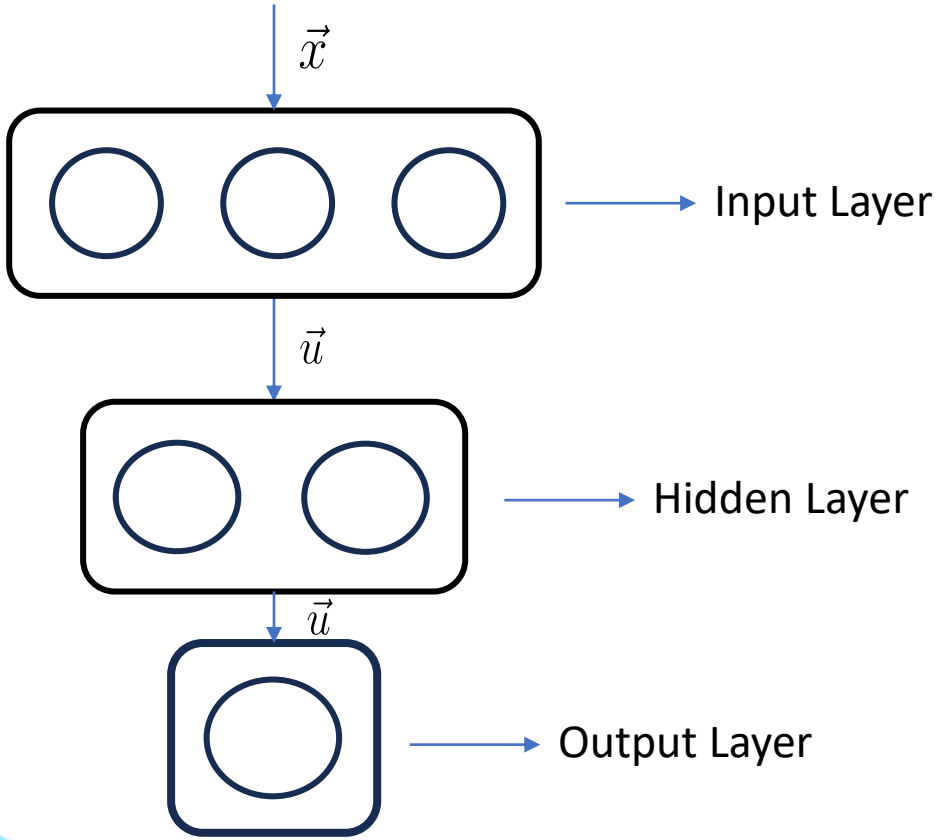
Demand Prediction

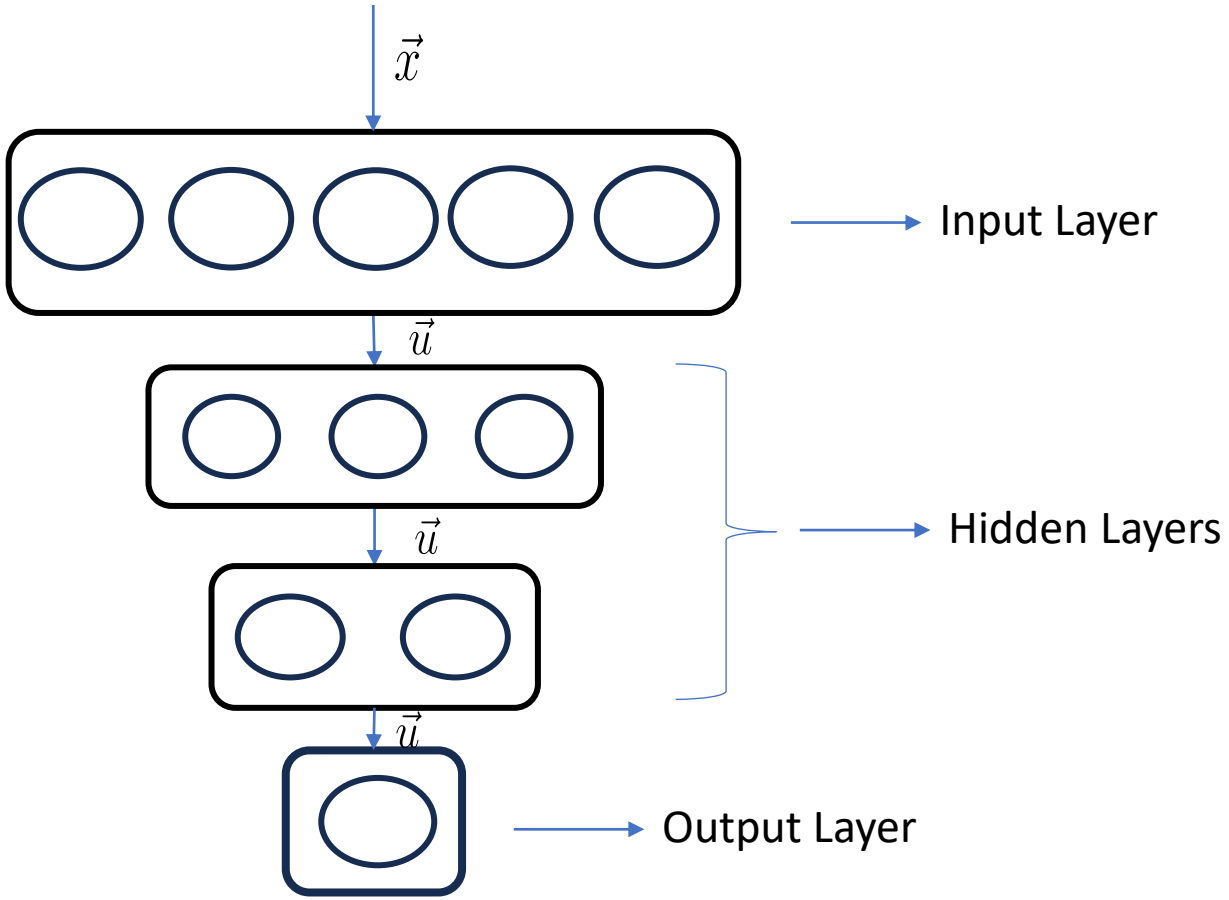


Advanced Learning Algos

Overview of Neural Networks

Multiple Hidden Layers

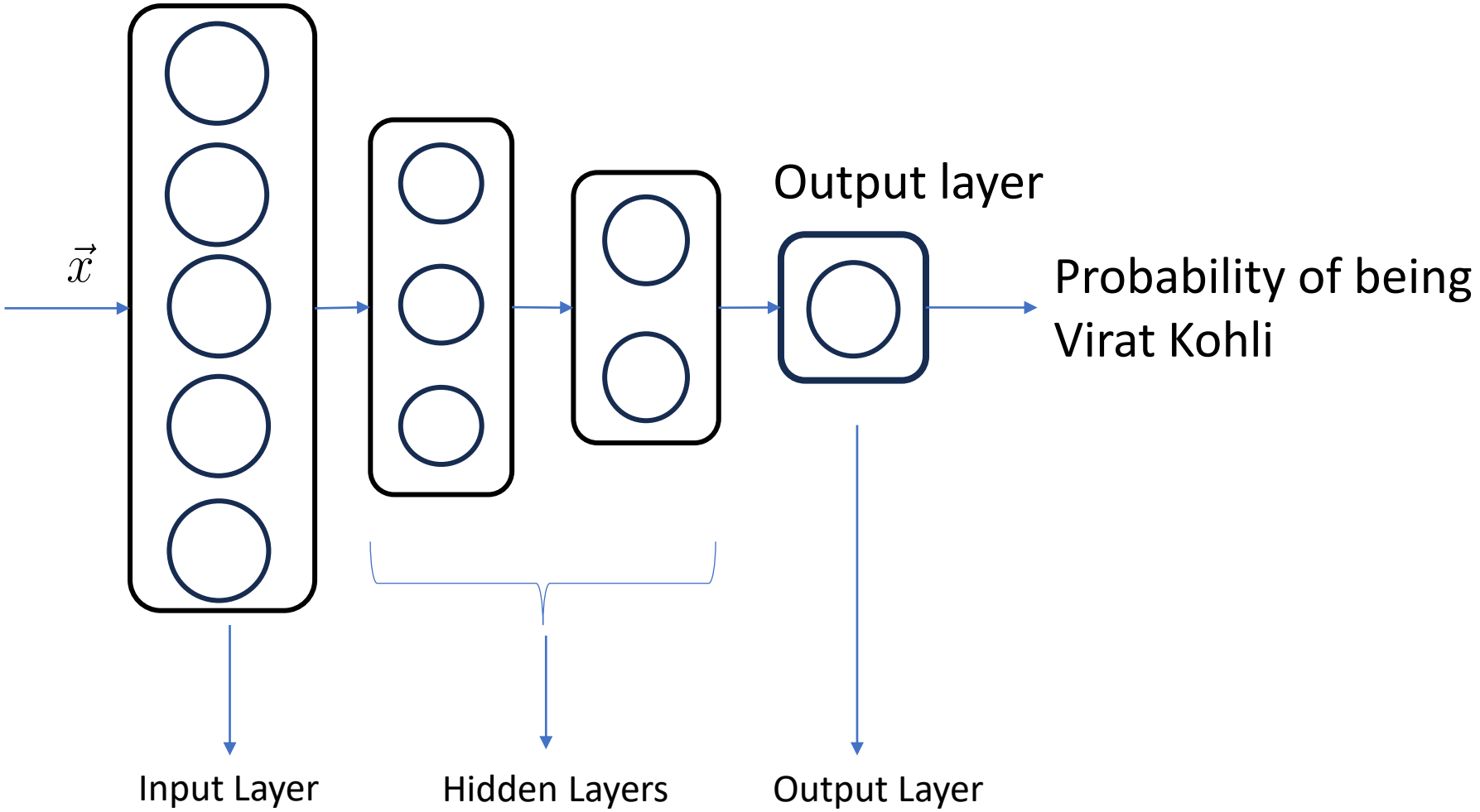




Advanced Learning Algos

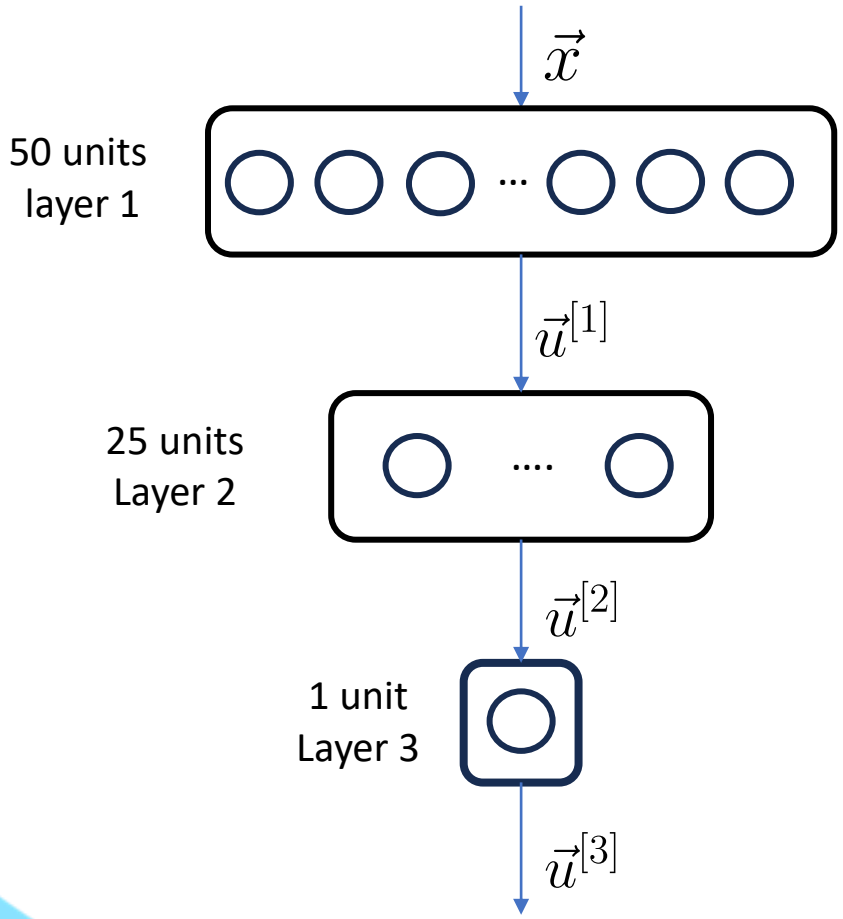
Overview of Neural Networks

Facial Recognition



Advanced Learning Algos

Neural network for binary classification of handwritten digits using TensorFlow



$$\vec{u}^{[1]} = \begin{pmatrix} g(\vec{a}_1^{[1]}.\vec{x} + b_1^{[1]}) \\ \vdots \\ g(\vec{a}_{25}^{[1]}.\vec{x} + b_{25}^{[1]}) \end{pmatrix}$$

$$\vec{u}^{[2]} = \begin{pmatrix} g(\vec{a}_1^{[1]}.\vec{u}^{[1]} + b_1^{[2]}) \\ \vdots \\ g(\vec{a}_{15}^{[1]}.\vec{u}^{[1]} + b_{15}^{[2]}) \end{pmatrix}$$

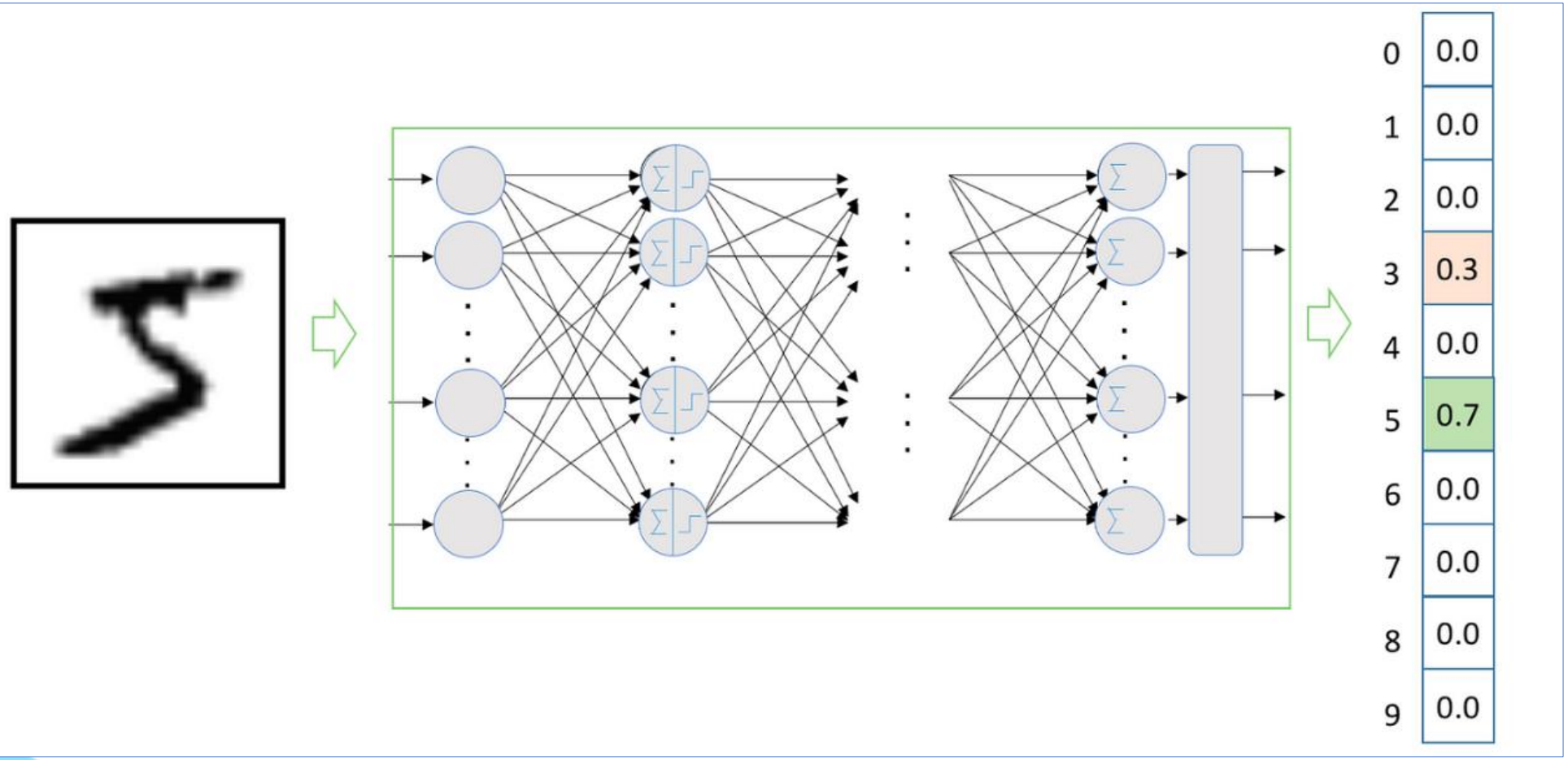
$$\vec{u}^{[3]} = [g(\vec{a}_1^{[3]}.\vec{u}^{[2]} + b_1^{[3]})]$$

Is $u_1^{[3]}$ >= 0.5

- Yes → $\hat{y} = 1$
- No → $\hat{y} = 2$

Advanced Learning Algos

Neural network for binary classification of handwritten digits using TensorFlow



How to build a Deep Learning model in 10 lines – Jordi TORRES.AI

Advanced Learning Algos

Neural network for binary classification of handwritten digits using TensorFlow

Model For Digit classification

```
#Define Input Data (x1)
x1 = np.array([[0.0,…245,…..240…..0]])

#Build the Neural Network Layers(l1,l2,l3)
l1 = Dense(units=50, activation='sigmoid’)
l2 = Dense(units=25, activation='sigmoid’)
l3 = Dense(units=1, activation='sigmoid’)

# The input data x1 is passed to l1 and output is stored in u1
# u1 is passed as an input to l2 and u2 is passed as input to l3
u1 = l1(x1)
u2 = l2(u1)
u3 = l3(u2)
```

```
#Prediction of digit
if u3 >= 0.5:
  yhat = 2

else:
  yhat = 1
```

Advanced Learning Algos

Implementing a neural network in Python from scratch

```
#import libraries
from keras.models import Sequential
from keras.layers import Dense
import tensorflow as tf

#Initialize the model
model = Sequential()

#Add the input Layer
model.add(Dense(units=11,activation="relu",
kernel_initializer="random_uniform"))

#Add the Hidden Layer
model.add(Dense(units=6,activation="relu",
kernel_initializer="random_uniform"))
model.add(Dense(units=6,activation="relu",
kernel_initializer="random_uniform"))
```

```
#Add Output layer
model.add(Dense(units=1,activation="sigmoid",
kernel_initializer="random_uniform"))

#configure the learning process
model.compile(loss="binary_crossentropy",
optimizer="adam",metrics=["accuracy"])

#Fit the model and make predictions
model.fit(x_train,y_train,epochs=100,batch_size=32)
ypred = model.predict(x_test)
```

How neural network computations are “vectorized” to use parallel processing for faster training and prediction

Advanced Learning Algos

```
import numpy as np
from tensorflow.keras import layers

# Sample input data (single data point for illustration)
x1 = np.array([[200.0, 17.0]])  # Shape: (1, 2)

# Build the neural network layers
l1 = layers.Dense(units=50, activation='sigmoid')
l2 = layers.Dense(units=25, activation='sigmoid')
l3 = layers.Dense(units=1, activation='sigmoid')

# Apply the first layer to the input, vectorizing the computations
u1 = l1(x1)

# Apply the second layer, vectorized for efficient computation
u2 = l2(u1)

# Apply the output layer, vectorized for prediction
u3 = l3(u2)

# thresholding for binary classification
if u3 >= 0.5:
    yhat = 1
else:
    yhat = 0

# Print the predicted probability
print(u3)
```

Output:

```
tf.Tensor([[0.8]], shape=(1, 1), dtype=float32)
```

Build a neural network to perform multi-class classification of handwritten digits in TensorFlow, using categorical cross-entropy loss functions and the SoftMax activation

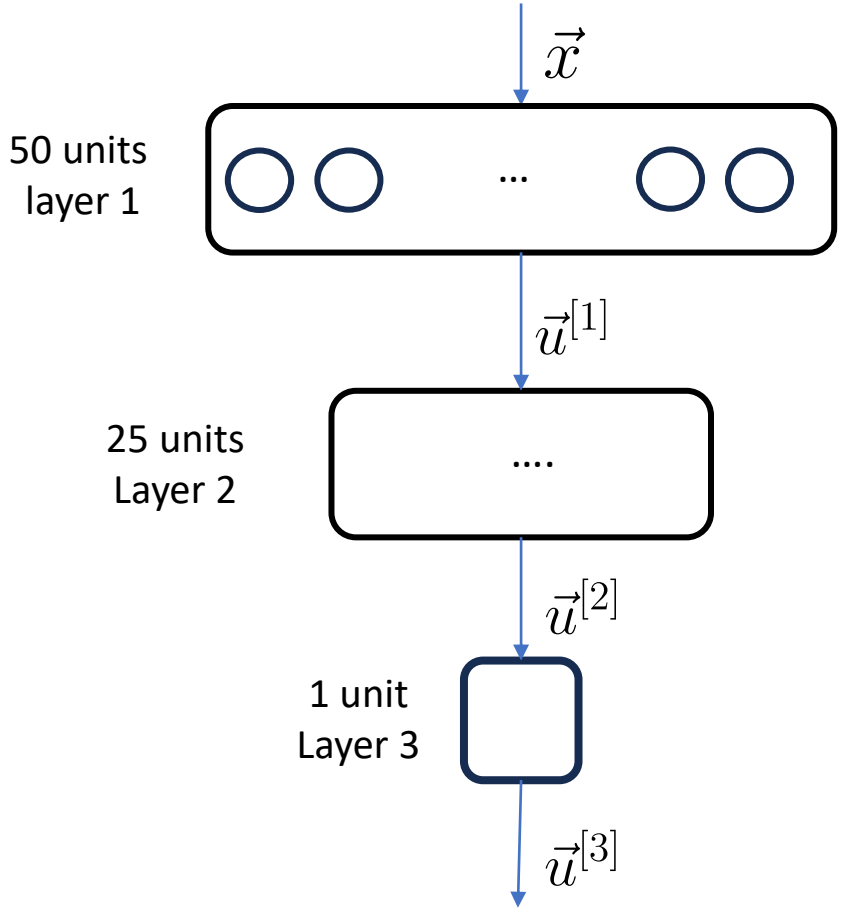
Advanced Learning Algos

```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense

#Initialize the model and build the layers simultaneously
model = Sequential([
        Dense(units=50, activation='relu'),
        Dense(units=25, activation='relu'),
        Dense(units=10, activation='softmax')])

#Compile the model using categorical cross-entropy loss functions
from tensorflow.keras.losses import SparseCategoricalCrossentropy
model.compile(loss= SparseCategoricalCrossentropy() )

#Train the model
model.fit(X,Y,epochs=100)
```



Where to use different activation functions (ReLu, linear, sigmoid, softmax) in a neural network, depending on the task you want your model to perform.

Advanced Learning Algos

Where to use different activation functions in a neural network

- **Activation functions** are a critical part of the design of a neural network.
- The choice of activation function in the hidden layer will control how well the network model learns the training dataset. The choice of activation function in the output layer will define the type of predictions the model can make.
- An activation function in a neural network defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network.
- A network may have three types of layers: input layers that take raw input from the domain, **hidden layers** that take input from another layer and pass output to another layer, and **output layers** that make a prediction.

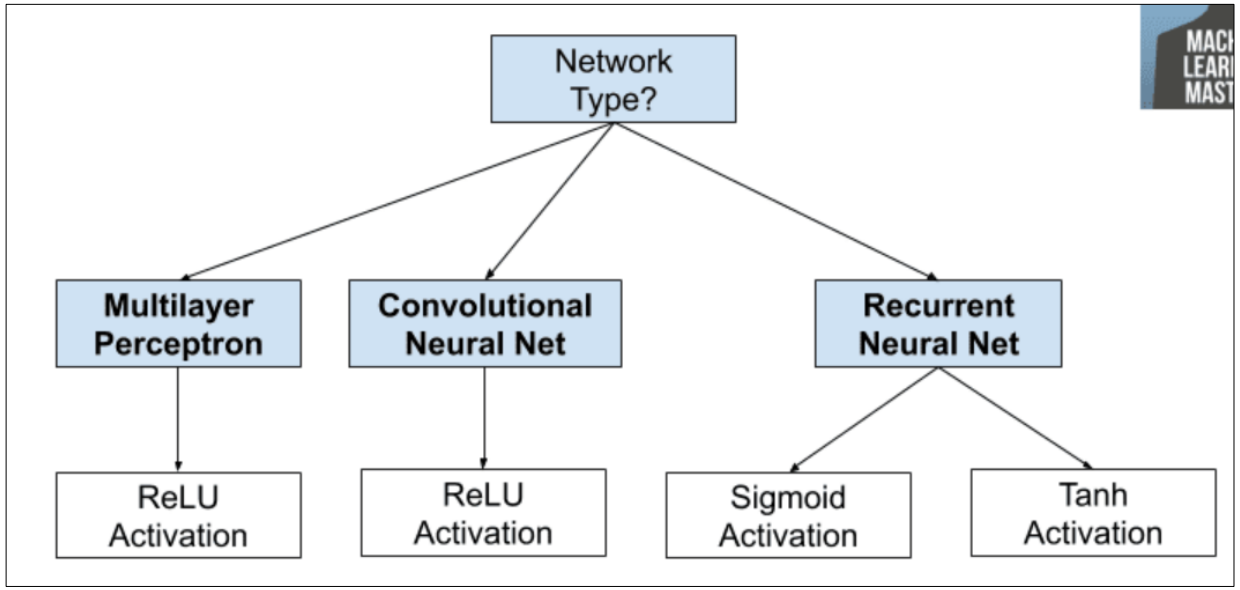
Advanced Learning Algos

Where to use different activation functions in a neural network

- All hidden layers typically use the same activation function. The output layer will typically use a different activation function from the hidden layers and is dependent upon the type of prediction required by the model.

There are perhaps three activation functions you may want to consider for use in hidden layers; they are:

- Rectified Linear Activation (**ReLU**)
- Logistic (**Sigmoid**)
- Hyperbolic Tangent (**Tanh**)



Advanced Learning Algos

Where to use different activation functions in a neural network

The output layer is the layer in a neural network model that directly outputs a prediction.

All feed-forward neural network models have an output layer.

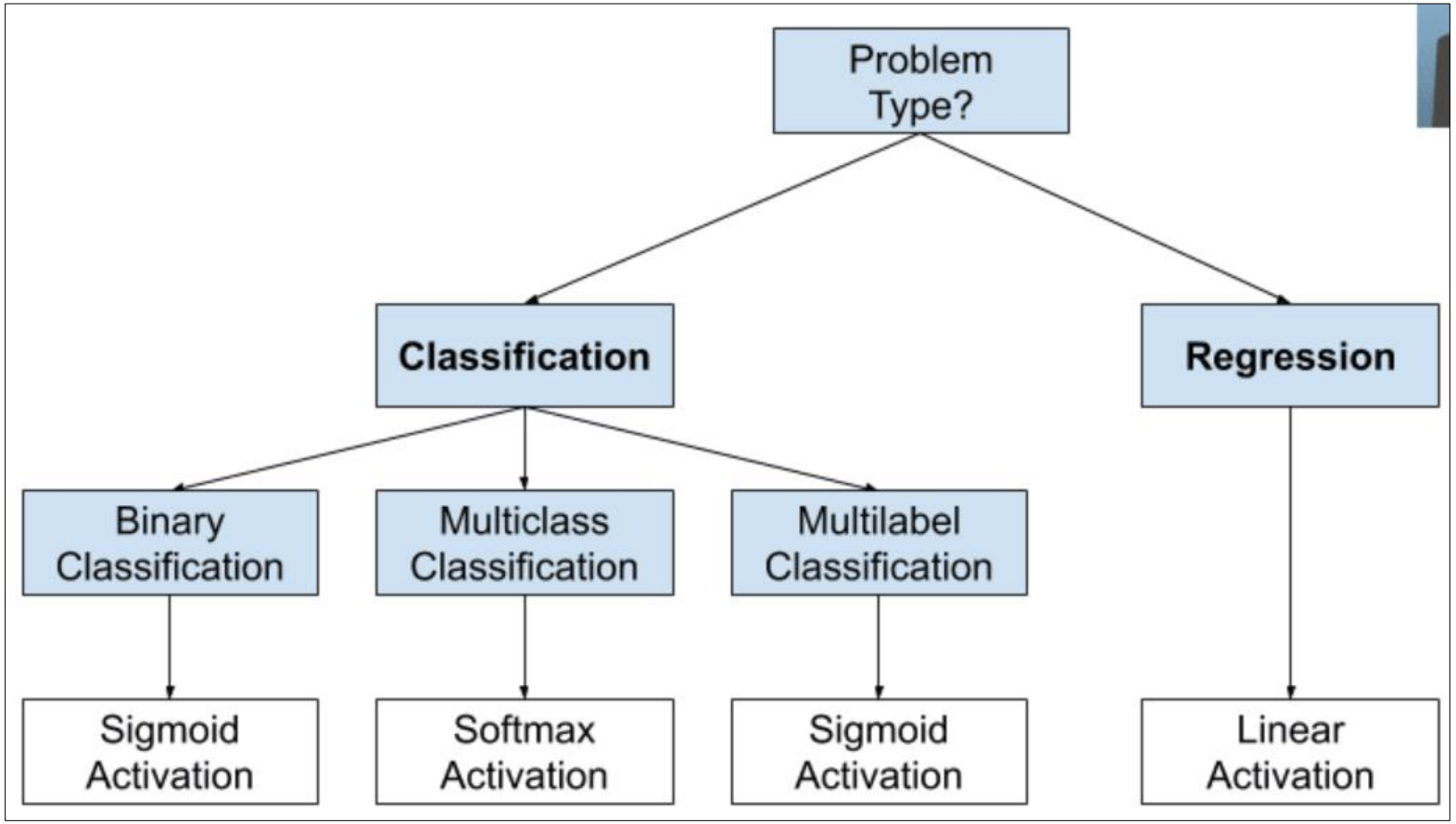
There are perhaps three activation functions you may want to consider for use in the output layer; they are:

- Linear
- Logistic (Sigmoid)
- Softmax

Advanced Learning Algos

Where to use different activation functions in a neural network

How to Choose an Output Activation Function



[How to Choose an Activation Function for Deep Learning - MachineLearningMastery.com]

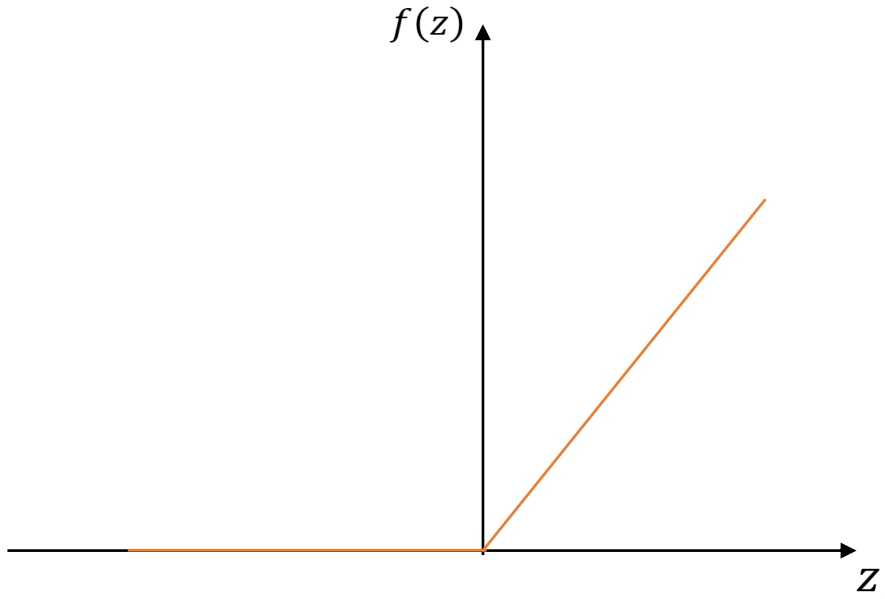
Advanced Learning Algos

Where to use different activation functions in a neural network

ReLU (Rectified Linear Unit)

- In most cases ReLU is used for hidden layers
- ReLU reduces the vanishing gradient problem.
- It's non-linear and allows the network to learn complex representations.

ReLU is defined as: $f(z) = \max(0, z)$



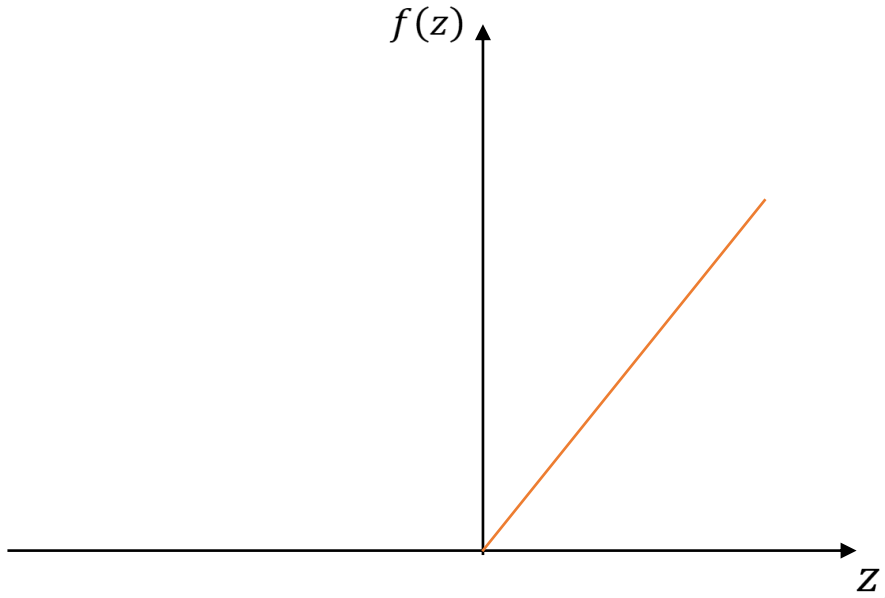
Advanced Learning Algos

Where to use different activation functions in a neural network

Linear Activation

- Linear Activation is used for regression tasks (predicting continuous values).
- It doesn't introduce non-linearity and directly outputs the weighted sum of inputs.

Linear Activation is defined as: $f(z) = z$



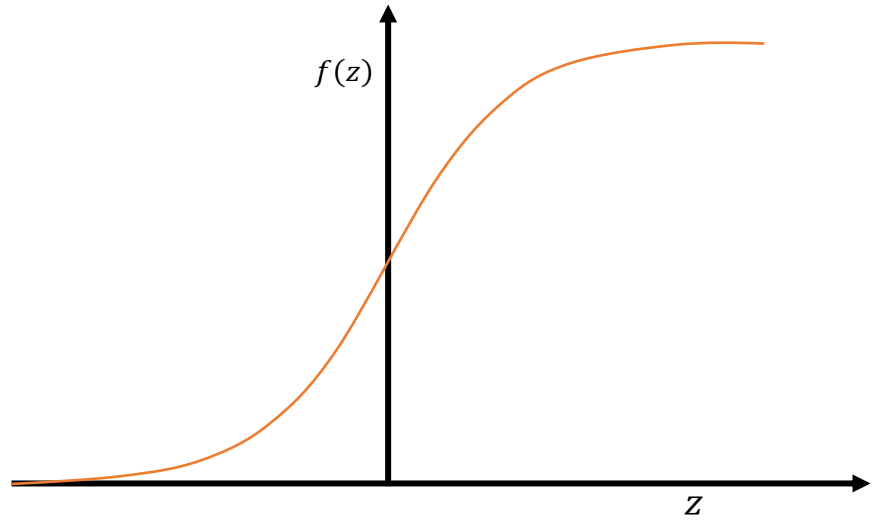
Advanced Learning Algos

Where to use different activation functions in a neural network

Sigmoid Activation

- Sigmoid Activation is used for binary classification problems (output between 0 and 1).
- It squashes values into the range [0, 1].

Sigmoid Activation is defined as: $f(z) = \frac{1}{1 + e^{-z}}$



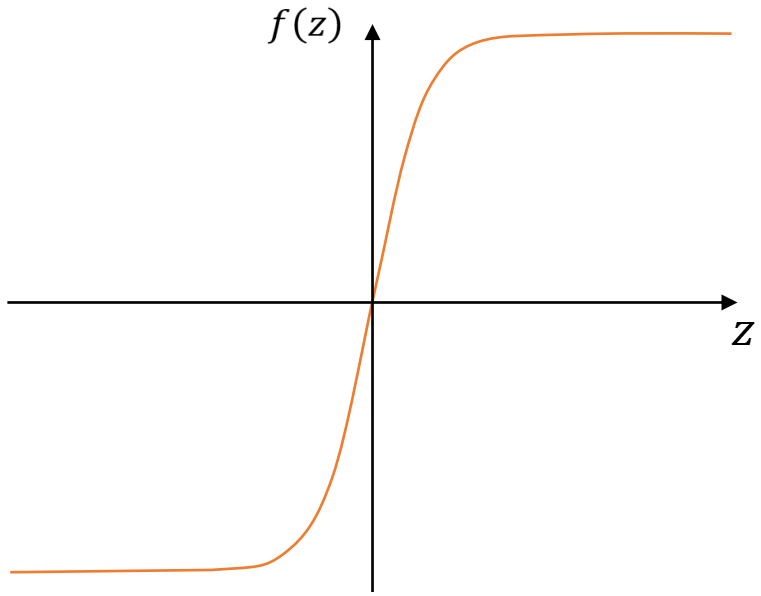
Advanced Learning Algos

Where to use different activation functions in a neural network

Softmax Activation:

- Softmax Activation is used multi class classification problems (when you have more than two classes).
- It converts raw scores into probability distributions.

Softmax Activation is defined as: $f(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$



Advanced Learning Algos

Use the advanced “Adam optimizer” to train your model more efficiently

The Adam optimizer, short for “Adaptive Moment Estimation,” is an iterative optimization algorithm used to minimize the loss function during the training of neural networks.

Adam can be looked at as a combination of RMSprop and Stochastic Gradient Descent with momentum.

Adam optimizer is like a smart helper for training neural networks. It helps adjust the network’s settings (called parameters) to make it better at its job, like recognizing images or understanding text.

Reference: What is Adam Optimizer? - Analytics Vidhya

Advanced Learning Algos

Use the advanced "Adam optimizer" to train your model more efficiently

- Adam stands for **Adaptive Moment estimation**
- Adam optimizer doesn't use one global learning rate a, instead it uses different alpha for every parameter of the model
- Suppose, our model has 3 parameters then Adam optimizer uses four a values.

$$a_1 = a_1 - \alpha_1 \frac{\partial}{\partial a_1} J(\vec{a}, b)$$

$$\vdots$$

$$a_3 = a_3 - \alpha_3 \frac{\partial}{\partial a_3} J(\vec{a}, b)$$

$$b = b - \alpha_4 \frac{\partial}{\partial b} J(\vec{a}, b)$$

Advanced Learning Algos

Use the advanced “Adam optimizer” to train your model more efficiently

Adam leverages concepts from two other optimizers:

Stochastic Gradient Descent (SGD) with Momentum:

- Considers an **average of past gradients** (momentum) along with the current gradient. This "momentum" helps updates move in a more consistent direction, potentially escaping local minima in the loss function.

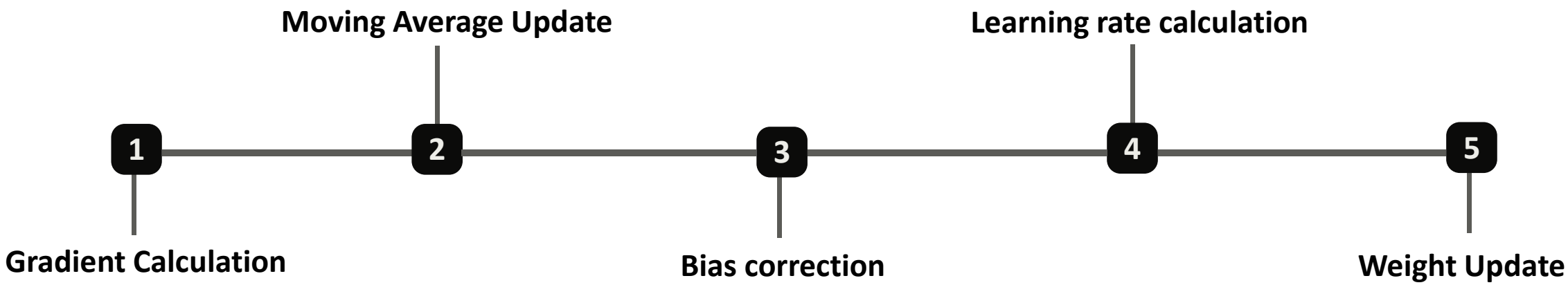
Root Mean Square Prop (RMSprop):

- Focuses on **historical squared gradients** to normalize updates. This reduces oscillations in gradient updates, especially for features that rarely change.

Advanced Learning Algos

Use the advanced “Adam optimizer” to train your model more efficiently

Steps in Adam Optimization



Advanced Learning Algos

Use the advanced “Adam optimizer” to train your model more efficiently

Knowledge Check:

What is an essential consideration when utilizing the Adam optimizer?

(a) Adam may not be universally suitable for all problem types, particularly those involving highly noisy data.

(b) Adam invariably necessitates a large dataset to function effectively.

(c) Adam exhibits significant sensitivity to the initial learning rate selection.

Advanced Learning Algos

Use the advanced “Adam optimizer” to train your model more efficiently

Knowledge Check:

What is an essential consideration when utilizing the Adam optimizer?

(a) Adam may not be universally suitable for all problem types, particularly those involving highly noisy data.

(b) Adam invariably necessitates a large dataset to function effectively.

(c) Adam exhibits significant sensitivity to the initial learning rate selection.

Advanced Learning Algos

The value of separating your data set into training, cross-validation, and test sets.

Training Set (60-80%):

- The largest portion of the data used to "train" the model.
- The model identifies patterns and relationships within this data to learn and make predictions.

Validation Set (10-20%):

- A held-out set used specifically to monitor model performance during training.
- Evaluates the model's ability to perform well on unseen data from the original dataset.
- Acts as an early warning system for overfitting.

Test Set (10-20%):

- The final, unbiased evaluation of the model's generalizability.
- Composed of entirely new, unseen data not used in training or validation.
- Provides a realistic picture of how the model will perform in real-world applications.

Advanced Learning Algos

The value of separating your data set into training, cross-validation, and test sets.

- The train-test-validation split is fundamental in machine learning and data analysis, particularly during model development. It involves dividing a dataset into three subsets: training, testing, and validation. Train test split is a model validation process that allows you to check how your model would perform with a new data set.

- The train validation test split helps assess how well a machine learning model will generalize to new, unseen data. It also prevents overfitting, where a model performs well on the training data but fails to generalize to new instances. By using a validation set, practitioners can iteratively adjust the model's parameters to achieve better performance on unseen data.

Advanced Learning Algos

The value of separating your data set into training, cross-validation, and test sets.

- **Overfitting Prevention**

Overfitting occurs when a model learns the training data well, capturing noise and irrelevant patterns. The validation set acts as a checkpoint, allowing for the detection of overfitting. By evaluating the model's performance on a different dataset, you can adjust model complexity, techniques, or other hyperparameters to prevent overfitting and enhance generalization.

- **Performance Evaluation**

The testing set is essential to a machine learning model's performance. After training and validation, the model faces the testing set, which checks real-world scenarios. A well-performing model on the testing set indicates that it has successfully adapted to new, unseen data. This step is important for gaining confidence in deploying the model for real-world applications.

A Comprehensive Guide to Train-Test-Validation Split in 2024 - Analytics Vidhya

Choose from various versions of your model using a cross-validation dataset, and evaluate its ability to generalize to real-world data using a test dataset

Advanced Learning Algos

- Consider, We have ten models and we have to choose a model out of them,

1. $f_{\vec{a},b}(\vec{x}) = a_1 x + b$
2. $f_{\vec{a},b}(\vec{x}) = a_1 x + a_2 x^2 + b$
3. $f_{\vec{a},b}(\vec{x}) = a_1 x + a_2 x^2 + a_3 x^3 + b$

.

.

.

.

10. $f_{\vec{a},b}(\vec{x}) = a_1 x + a_2 x^2 + ... + a_{10} x^{10} + b$

Advanced Learning Algos

- Split the dataset into 3 sets: **Training set**, **Cross Validation set** and **Testing Set**
- Training set is denoted by:

$$(x^{(1)}, y^{(1)})$$
$$\vdots$$
$$(x^{(m_{train})}, y^{(m_{train})})$$

- Cross Validation set is denoted by:

$$(x_{cv}^{(1)}, y_{cv}^{(1)})$$
$$\vdots$$
$$(x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$$

- Testing set is denoted by:

$$(x_{test}^{(1)}, y_{test}^{(1)})$$
$$\vdots$$
$$(x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$$

Advanced Learning Algos

- Compute the Training error, Cross validation error and Test error
- Training error:

$$J_{train}(\vec{a}, b) = \frac{1}{2m_{train}} \left[\sum_{i=1}^{m_{train}} (f_{\vec{a},b}(\vec{x}^{(i)}) - y^{(i)})^2 \right]$$

- Cross Validation error:

$$J_{cv}(\vec{a}, b) = \frac{1}{2m_{cv}} \left[\sum_{i=1}^{m_{cv}} (f_{\vec{a},b}(\vec{x}^{(i)}) - y^{(i)})^2 \right]$$

- Test error:

$$J_{test}(\vec{a}, b) = \frac{1}{2m_{test}} \left[\sum_{i=1}^{m_{test}} (f_{\vec{a},b}(\vec{x}^{(i)}) - y^{(i)})^2 \right]$$

Advanced Learning Algos

1. $f_{\vec{a},b}(\vec{x}) = a_1 x + b$ $J_{cv}(a^{<1>}, b^{<1>})$
2. $f_{\vec{a},b}(\vec{x}) = a_1 x + a_2 x^2 + b$ $J_{cv}(a^{<2>}, b^{<2>})$
3. $f_{\vec{a},b}(\vec{x}) = a_1 x + a_2 x^2 + a_3 x^3 + b$ $J_{cv}(a^{<3>}, b^{<3>})$

.

.

.

.

10. $f_{\vec{a},b}(\vec{x}) = a_1 x + a_2 x^2 + ... + a_{10} x^{10} + b$ $J_{cv}(a^{<10>}, b^{<10>})$

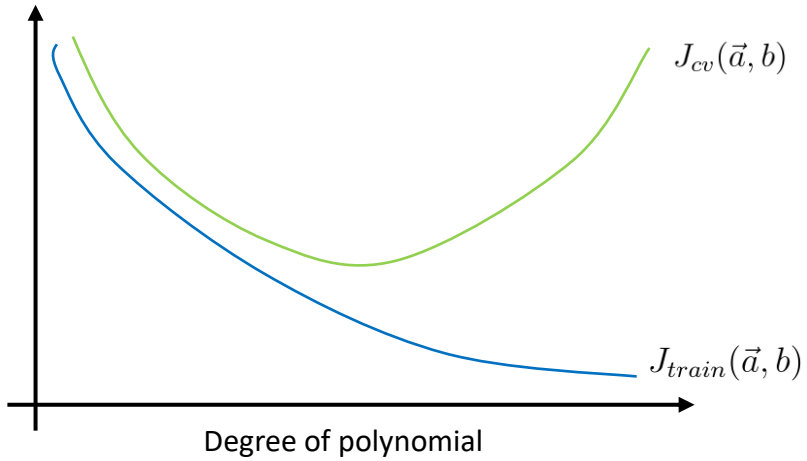
- For all the models, compute the **Cross Validation error** and choose the model with the lowest Cross validation error.

Use “learning curves” to determine if your model is experiencing high bias or high variance (or both), and learn which techniques to apply (regularization, adding more data, adding or removing input features) to improve your model’s performance

Advanced Learning Algos

High variance and High Bias

- How to find out if a algorithm has a bias or variance problem:



High bias

J_{train} will be high

$J_{train} \approx J_{cv}$

High Variance

$J_{cv} \gg J_{train}$

J_{train} may be low

High Variance and High Bias

J_{train} will be high

$J_{cv} \gg J_{train}$

Advanced Learning Algos

Learning Curves to determine High variance and Bias

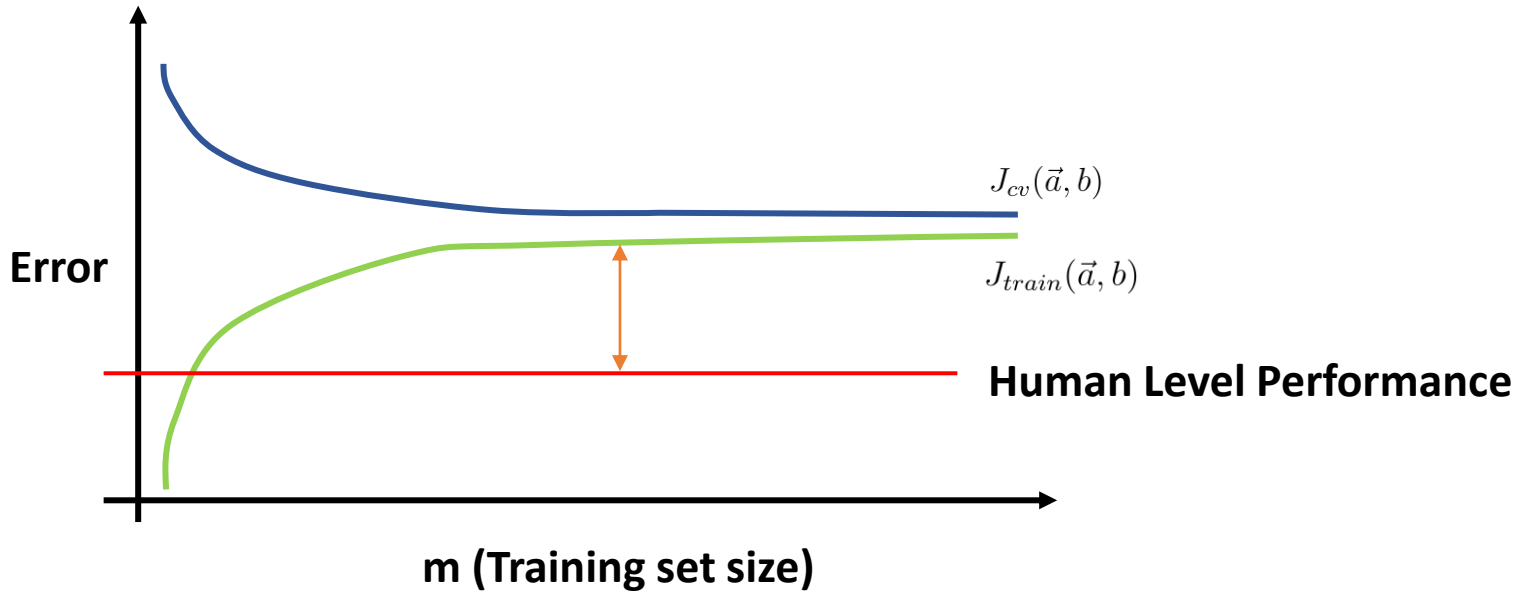
- Learning curves are a valuable tool for diagnosing bias and variance issues in machine learning models
- When there is a **high bias**, The training and validation curves will be relatively flat and low across increasing training set sizes. This indicates the model is underfitting the data.
- When there is a **high variance**, The training curve will increase rapidly with increasing training set size, but the validation curve will lag behind and may even fluctuate significantly.

Advanced Learning Algos

Learning Curves to determine High variance and Bias

Learning Curve for a High Bias Model

Consider, a high bias model: $f_{\vec{a},b}(\vec{x}) = a_1 x + b$

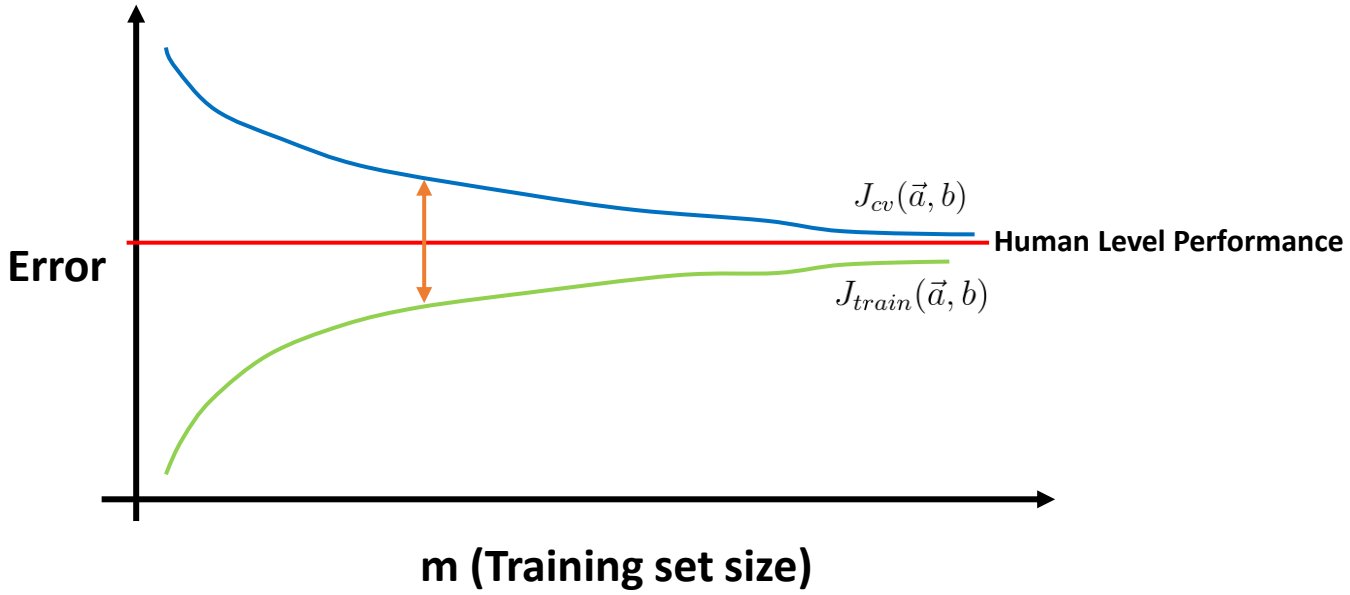


Advanced Learning Algos

Learning Curves to determine High variance and Bias

Learning Curve for a High Variance Model

Consider, a high variance model: $f_{\vec{a},b}(x) = a_1x + a_2x^2 + a_3x^3 + a_4x^4 + b$



Advanced Learning Algos

Knowledge Check:

Which scenario in a learning curve indicates a model with high variance (overfitting)?

(a) A small gap between training and validation error, with both decreasing rapidly.

(b) A large gap between training and validation error, with the validation error increasing after a point.

(c) A steady decrease in error rate for both training and validation data, eventually reaching a similar low point.

Advanced Learning Algos

Knowledge Check:

Which scenario in a learning curve indicates a model with high variance (overfitting)?

(a) A small gap between training and validation error, with both decreasing rapidly.

(b) A large gap between training and validation error, with the validation error increasing after a point.

(c) A steady decrease in error rate for both training and validation data, eventually reaching a similar low point.

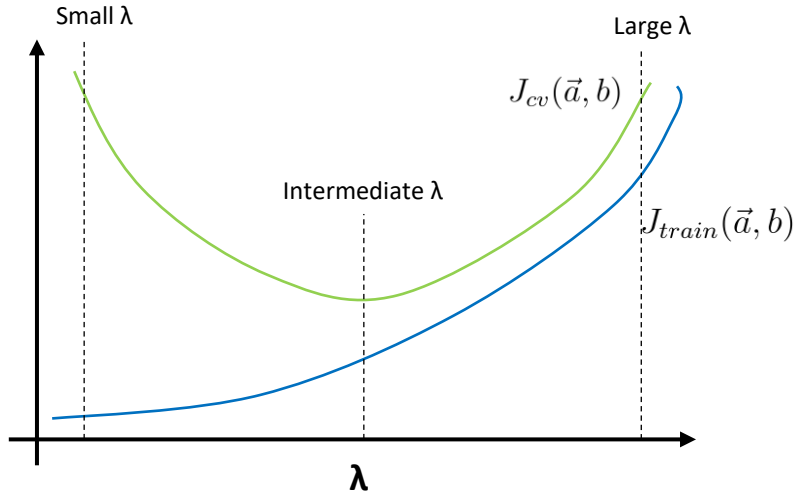
Advanced Learning Algos

Bias and Variance - Regularization

- Consider,

$$J(\vec{a}, b) = \frac{1}{2m} \sum_{i=1}^{m} (f_{\vec{a},b}(\vec{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^{n} a_j^2$$

- Choose an intermediate value for λ, where the cross validation error is low, this would yield a good model for application.



Advanced Learning Algos

Fixing High Bias and High Variance

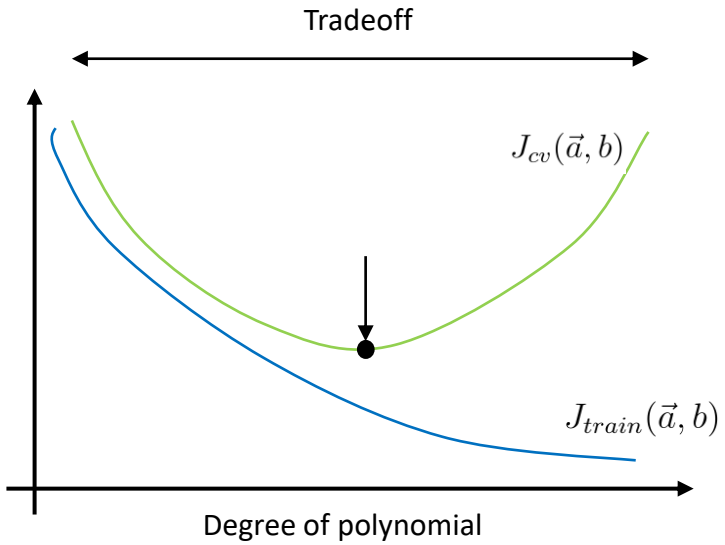
Solution	Fixes High Bias	Fixes High Variance
more training data	✗	✓
Small set of features	✗	✓
Addition of features	✓	✗
Addition of polynomial features	✓	✗
$\lambda \uparrow$	✗	✓
$\lambda \downarrow$	✓	✗

The “bias-variance trade-off” is different in the age of deep learning, and apply Andrew Ng’s advice for handling bias and variance when training neural networks

Advanced Learning Algos

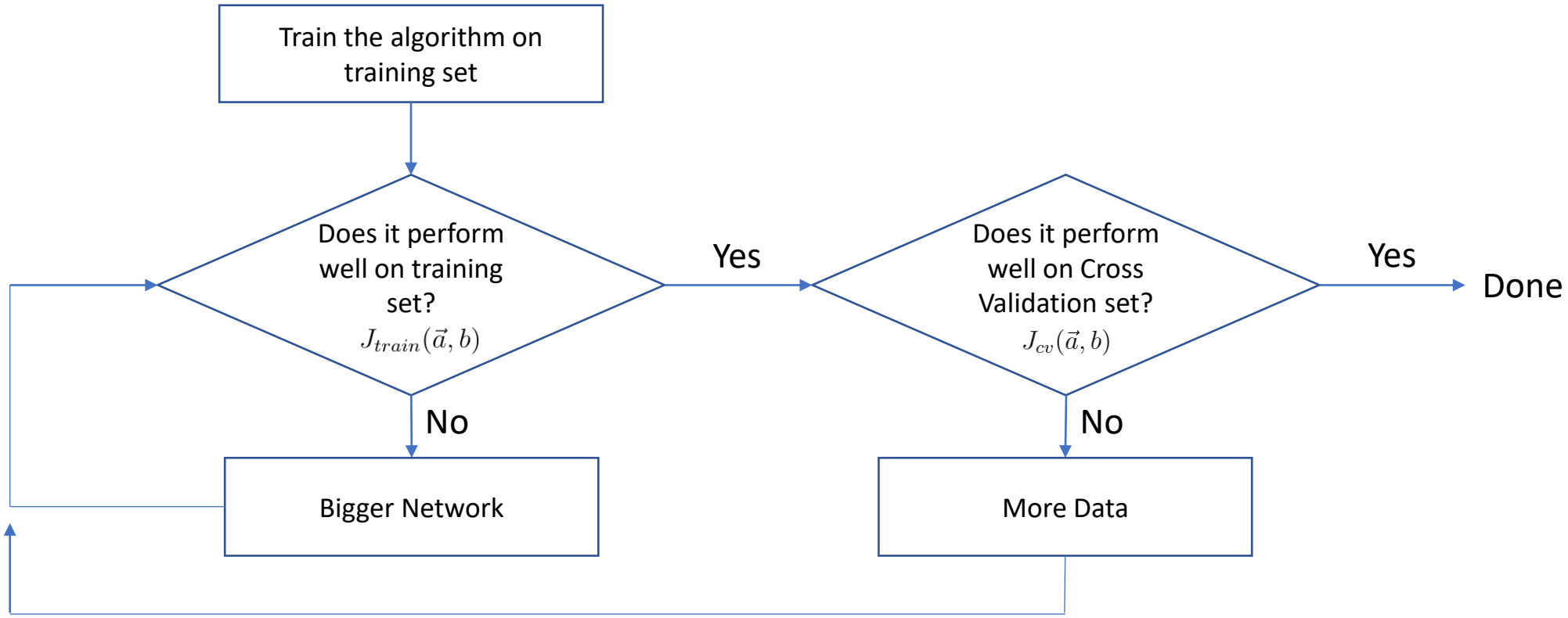
Bias Variance Tradeoff

- There's a trade-off between bias and variance.
- Simpler models tend to have lower variance (less sensitive to training data) but higher bias (underfit).
- More complex models tend to have lower bias (better fit to training data) but higher variance (overfit).
- The goal is to find a model that balances these two errors for optimal performance on unseen data.



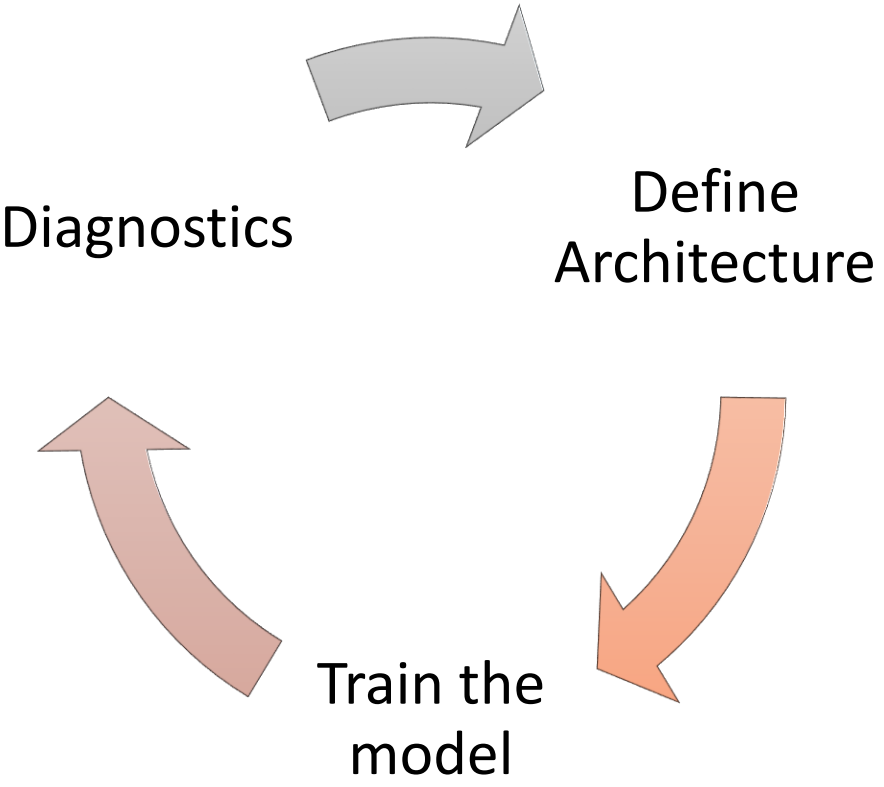
Advanced Learning Algos

Handling Bias and Variance when training neural network



Advanced Learning Algos

Apply the “iterative loop” of machine learning development to train, evaluate, and tune your model



Advanced Learning Algos

Apply the “iterative loop” of machine learning development to train, evaluate, and tune your model

1. **Define Architecture:**
   - Choose a machine learning model
   - Select the data to use for training
   - Decide on hyperparameters

2. **Train the model:**
   - Build the model based on your chosen architecture
   - Train the model on the selected data

Advanced Learning Algos

Apply the “iterative loop” of machine learning development to train, evaluate, and tune your model

3. Diagnostics:

- Evaluate the model's performance using techniques like:
  - Bias-variance analysis, Error analysis etc
- Analyse the results to understand the model's strengths and weaknesses.

Based on the diagnostics, make adjustments to the architecture and go back to step 1 with the updated architecture and repeat the loop until the model achieves the desired performance.

Advanced Learning Algos

Apply the “iterative loop” of machine learning development to train, evaluate, and tune your model

Knowledge Check:

When does the iterative loop in machine learning development typically stop?

(a) After a fixed number of iterations, regardless of model performance.

(b) When a specific accuracy threshold is achieved on the training data.

(c) When the model achieves satisfactory performance on a held-out validation or test set.

Advanced Learning Algos

Apply the “iterative loop” of machine learning development to train, evaluate, and tune your model

Knowledge Check:

When does the iterative loop in machine learning development typically stop?

(a) After a fixed number of iterations, regardless of model performance.

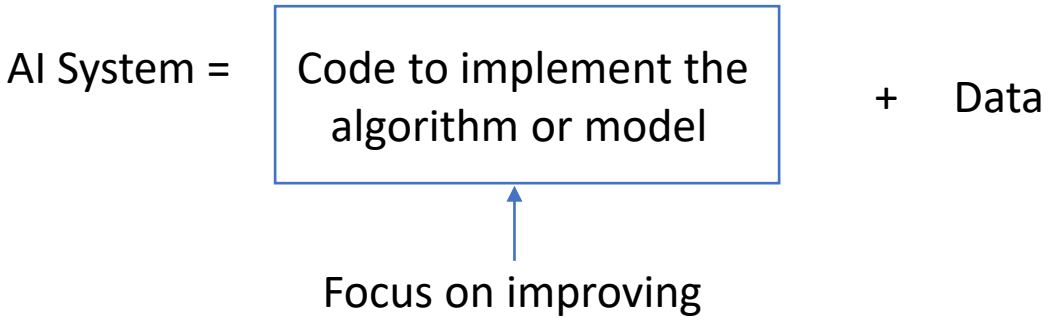
(b) When a specific accuracy threshold is achieved on the training data.

(c) When the model achieves satisfactory performance on a held-out validation or test set.

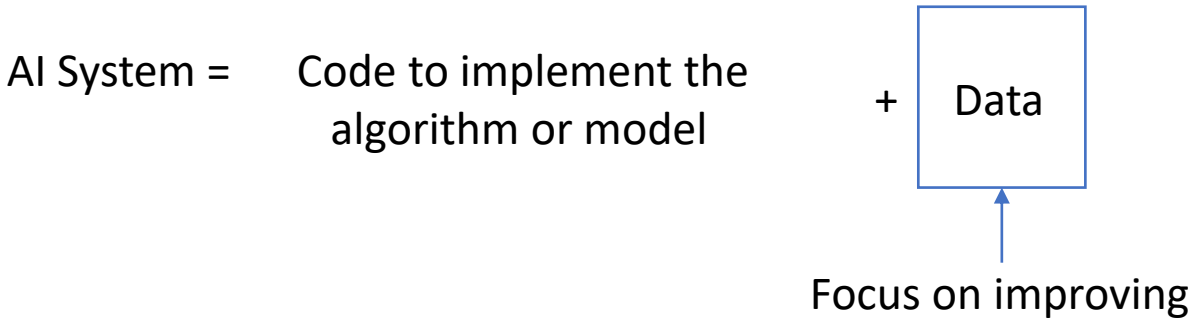
Apply “data-centric AI” to not only tune your model but tune your data (using data synthesis or data augmentation) to improve your model’s performance

Advanced Learning Algos

In Conventional Model Centric approach,



In Data Centric approach,



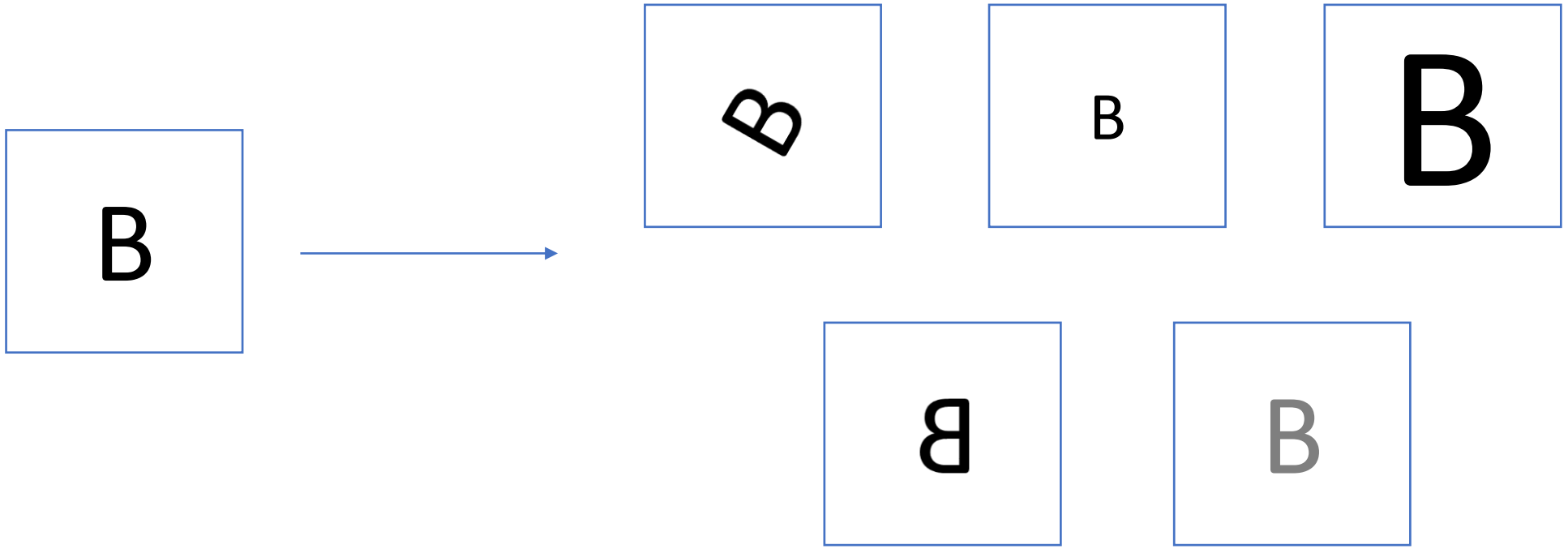
Advanced Learning Algos

Data Augmentation

- This technique involves modifying existing training examples to create new ones. For instance, in image recognition, you can rotate, enlarge, shrink, or change the contrast of an image to create new variations while preserving the original label.
- The modifications should be realistic and similar to the distortions the model might encounter in real-world applications. Adding random noise that doesn't reflect real-world scenarios would not be helpful.

Advanced Learning Algos

Data Augmentation



Advanced Learning Algos

Data synthesis

- This technique involves creating entirely new training examples from scratch.
- In photo OCR (Optical Character Recognition), where the goal is to train a computer to read text in images, you can use a computer's text editor with various fonts and colors to generate new images containing random text. This creates more training data for the OCR algorithm.

Advanced Learning Algos

Data synthesis – Photo OCR

Image Source: https://www.publicdomainpictures.net/en/view-image.php?image=5745&picture=times-square

Advanced Learning Algos

Data synthesis – Photo OCR

Image Source: https://www.publicdomainpictures.net/en/view-image.php?image=5745&picture=times-square

Build decision trees and tree ensembles, such as random forest and XGBoost (boosted trees) to make predictions

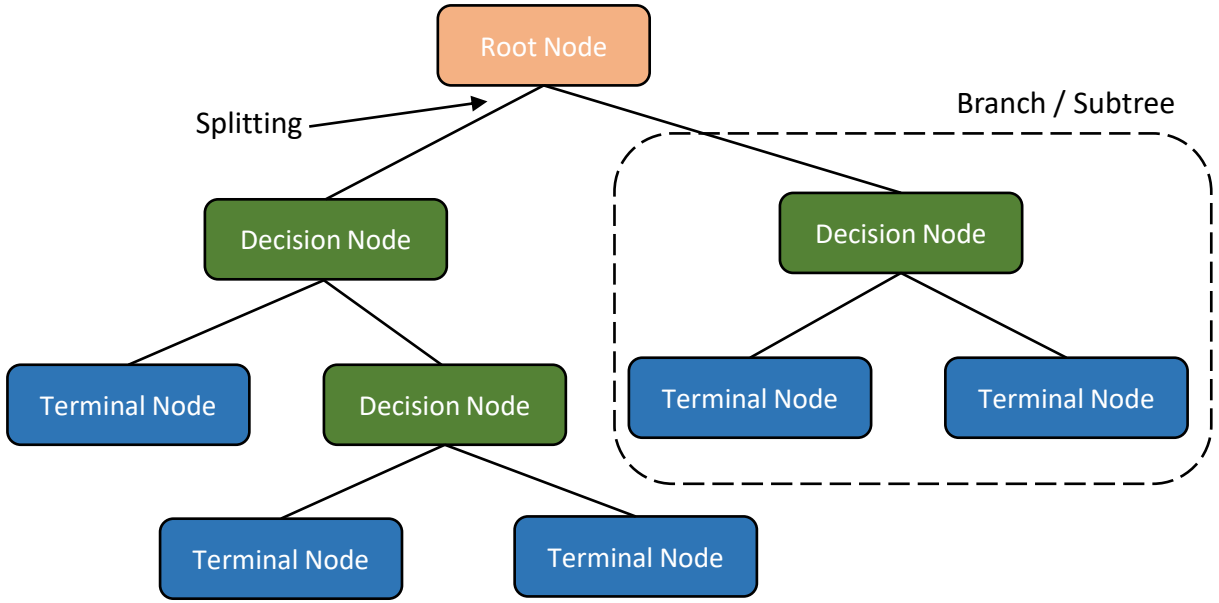
Advanced Learning Algos

Decision Trees

- Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression
- It uses a tree-like model of decisions.
- Growing a tree involves deciding on which features to choose and what conditions to use for splitting, along with knowing when to stop.

Advanced Learning Algos

Decision Trees



Advanced Learning Algos

Decision Trees

Working of Decision Trees:

- **Selecting the Best Attribute:** the algorithm identifies the most informative feature (attribute) to separate the data.
- **Splitting the Dataset:** Based on the chosen attribute, the data gets divided into subsets.
- **Repeating the Process:** This process of selecting an attribute, splitting, and asking questions continues for each resulting subset. This recursive process stops when a certain condition is met.

Advanced Learning Algos

Decision Trees

Metrics for splitting Decision Trees

Gini Impurity:

Measures the likelihood of an incorrect classification of a new instance if it was randomly classified according to the distribution of classes in the dataset.

$$\text{Gini index} = 1 - \sum_{i=1}^{n} (p_i)^2$$

Advanced Learning Algos

Decision Trees

Metrics for splitting Decision Trees

Entropy :

Measures the amount of uncertainty or impurity in the dataset

$$\text{Entropy} = -\sum_{i=1}^{n} p_i log_2(p_i)$$

Advanced Learning Algos

Decision Tree

Knowledge Check:

How does a decision tree algorithm typically choose the best splitting feature at each internal node?

(a) By selecting the feature with the highest correlation coefficient to the target variable.

(b) By choosing the feature that results in the purest separation of classes or the greatest reduction in impurity for the target variable.

(c) By randomly selecting a feature at each node.

Advanced Learning Algos

Decision Tree

Knowledge Check:

How does a decision tree algorithm typically choose the best splitting feature at each internal node?

(a) By selecting the feature with the highest correlation coefficient to the target variable.

(b) By choosing the feature that results in the purest separation of classes or the greatest reduction in impurity for the target variable.

(c) By randomly selecting a feature at each node.

Advanced Learning Algos

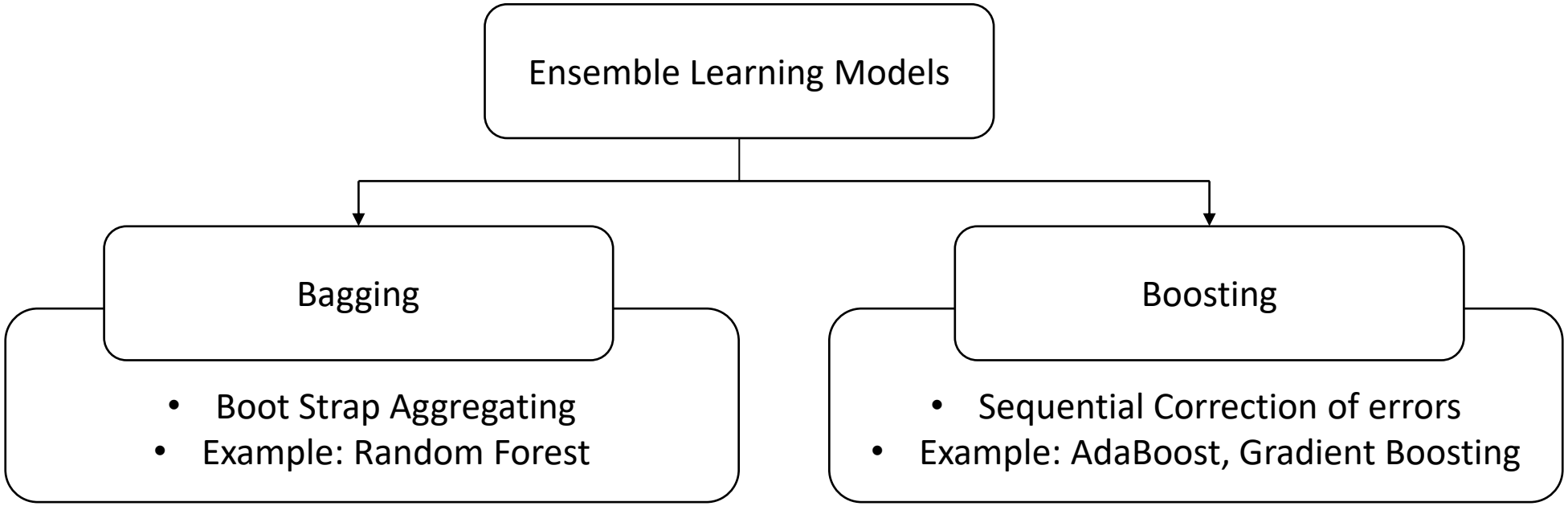
Tree ensembles

- Ensemble learning is a technique in machine learning where multiple models are trained to solve the same problem and combined to get better results.
- Why use ensemble learning models?
  - Reduces overfitting.
  - Handles complex data better.
  - Improves prediction accuracy.
  - Increases robustness against noise and outliers.

Advanced Learning Algos

Tree ensembles

Types of Ensemble Learning Models



Advanced Learning Algos

Tree ensembles

Random Forest

- Random Forest algorithm is a powerful tree learning technique in Machine Learning.
- It works by creating a number of Decision Trees during the training phase.
- Each tree is constructed using a random subset of the data set to measure a random subset of features in each partition.

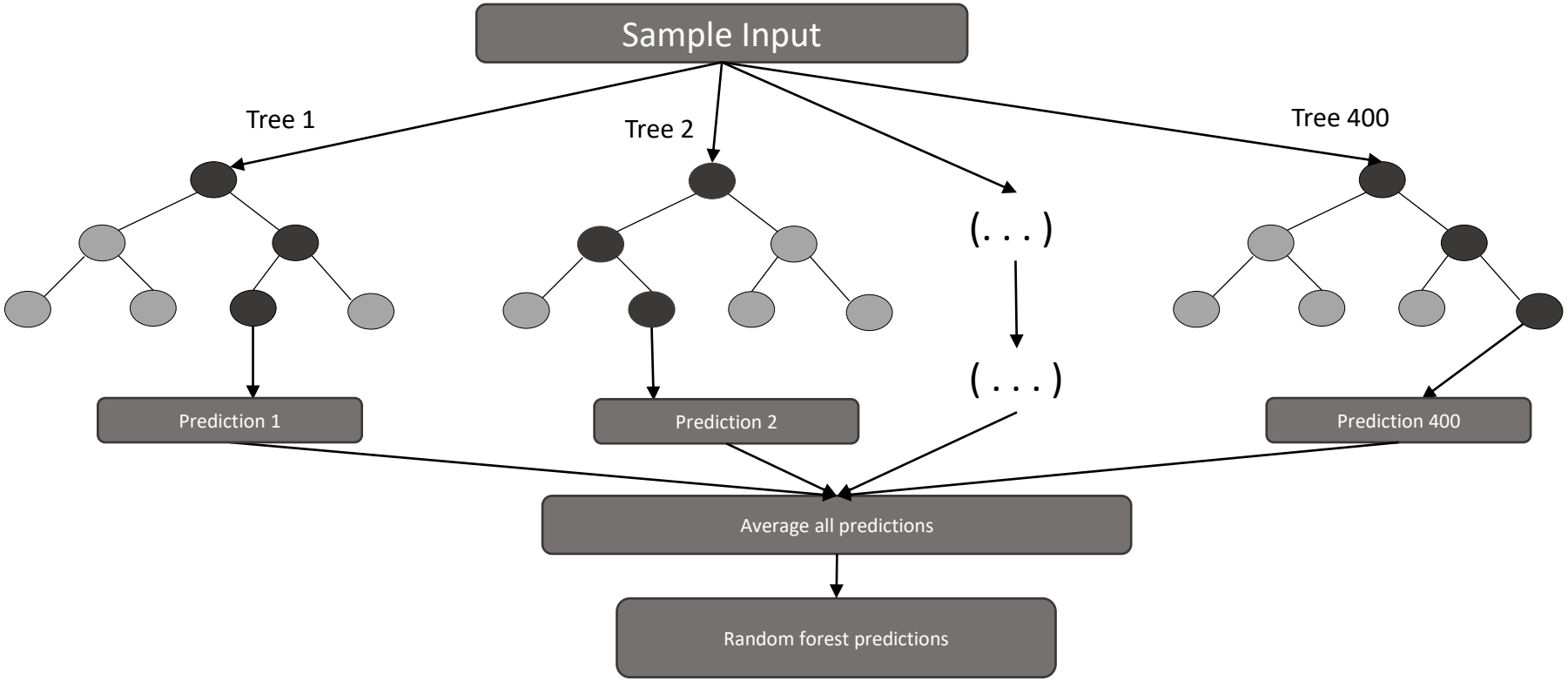
Advanced Learning Algos

Tree ensembles

Random Forest

- This randomness introduces variability among individual trees, reducing the risk of overfitting and improving overall prediction performance.
- In prediction, the algorithm aggregates the results of all trees, either by voting (for classification tasks) or by averaging (for regression tasks).
- This collaborative decision-making process, supported by multiple trees with their insights, provides an example stable and precise results.

Advanced Learning Algos



Advanced Learning Algos

Tree ensembles

XGBoost (Extreme Gradient Boosting)

- XGBoost is an algorithm for building ensembles of decision trees,
- Incorporates a technique called boosting which focuses on improving the model's performance on examples that previous decision trees in the ensemble struggled with.
- XGBoost can be used for both classification and regression tasks by using the appropriate class (XGBoostClassifier or XGBRegressor) during model initialization.

Advanced Learning Algos

Tree ensembles

XGBoost

XGBoost Classifier

```
from xgboost import XGBClassifier

classifier_xgb = XGBClassifier(
                    objective = "Specify objective for classification",
                    n_estimators="Number of trees in the ensemble",
                    learning_rate="Learning rate",
                    max_depth= "Maximum depth of the trees"
)

classifier_xgb.fit(X_train, y_train)

preds= classifier_xgb.predict(X_test)
```

XGBoost Regressor

```
from xgboost import XGBRegressor

regressor_xgb = XGBRegressor(
                    n_estimators="Number of trees in the ensemble",
                    learning_rate="Learning rate",
                    max_depth= "Maximum depth of the trees"
)

regressor_xgb.fit(X_train, y_train)

preds= regressor_xgb.predict(X_test)
```

Use neural network or tree ensemble models for your task, as these are the two most commonly used supervised learning models in practice today

Advanced Learning Algos

Why Neural network is a most commonly used supervised learning model in practice today

Neural network offers,

Data Versatility: effectively handle all data types, including structured, unstructured, and even mixed formats.

Unstructured Data Mastery: They are particularly adept at extracting patterns from unstructured data like images and text.

Advanced Learning Algos

Why Neural network is a most commonly used supervised learning model in practice today

Neural network offers,

Transfer Learning: Neural networks can leverage pre-trained models on extensive datasets (transfer learning) to enhance performance even with limited training data.

Scalability: Integrating multiple neural networks within a system is often more straightforward than with decision trees.

Advanced Learning Algos

Why tree ensembles is a most commonly used supervised learning model in practice today

Tree ensembles offers,

Computational Efficiency: Tree ensembles boast rapid training times, enabling faster model iteration and performance optimization.

Interpretability: Particularly for smaller trees, the decision-making logic remains comprehensible, providing valuable insights into model behavior.

Structured Data: They excel at handling tabular data (numerical and categorical features) in both classification and regression tasks.

?

Thank You !!!