Principles of Machine Learning

Unit-1

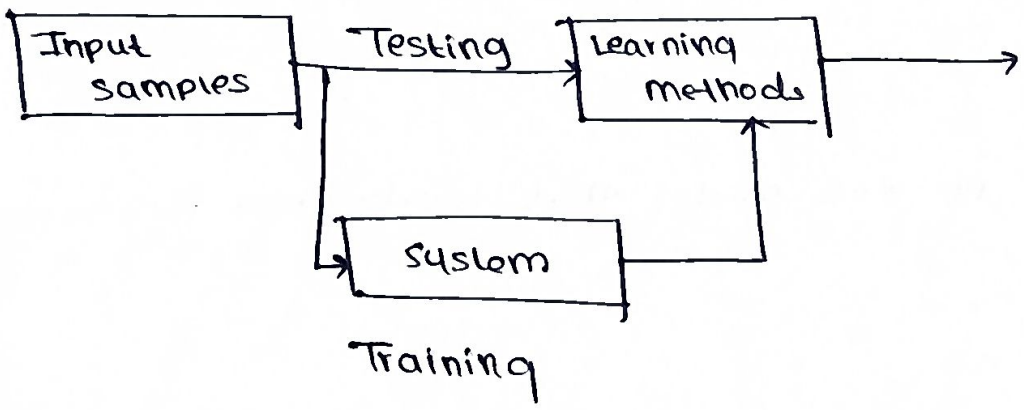
Introduction

Introduction: Machine learning; Examples of machine learning applications: Learning associations - classifications - regression - unsupervised learning - Reinforcement learning; Preliminaria: weight space - curse of dimensionality - Testing machine learning algorithms - Turning data into probabilities - basic statistics - bias - variance - tradeoff

* Machine Learning

→ A branch of artificial intelligence, concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data.

→ The learning system model is as follows:



→ Machine learning is the study of algorithms that

- improve their performance P
- at some task T
- with experience E

A well-defined learning task is given by <P, T, E>

* Why Machine Learning?

→ Develop systems that can automatically adapt and customize themselves to individual users

→ Discover new knowledge from large databases (data mining)

→ Able to mimic humans and replace certain montonous tasks which require some intelligence

→ develop systems that are too difficult / expensive to construct manually because they require specific detailed skills or knowledge tuned to a specific task (knowledge-engineering bottleneck)

* Training Machine Learning Models

→ Training is the process of making the system able to learn

No Free Lunch Theorem - when the performance of all optimization methods is averaged across all conceivable problems, they all perform equally well. It indicates that no one optimum optimization algorithm exists.

→ Essentially, there is no one model that works best for every situation.

→ Additionaly,

- the training and testing set must come from the same distribution
- some assumptions and biases need to be made.

* Performance of ML Models

→ There are several factors affecting the performance:

(i) Types of training provided

(ii) The form and extent of any initial background knowledge

(iii) The type of feedback provided

(iv) The learning algorithms used

(v) Type of modeling and optimization

* ML Algorithms

→ The ML algorithms used control the search to find and build the knowledge structures

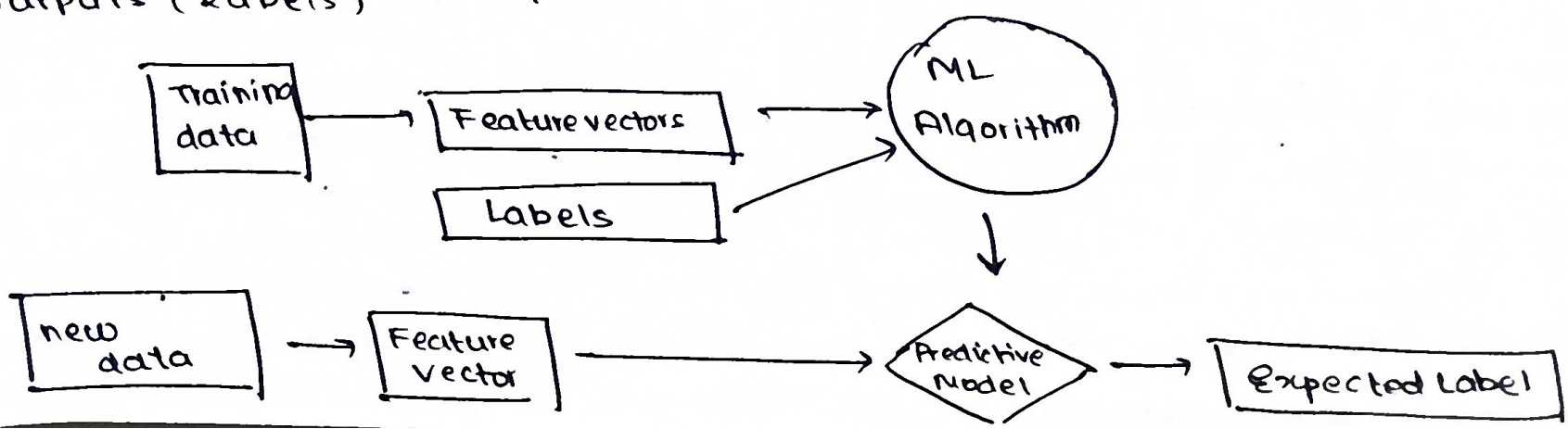
→ should extract useful information from training examples.

Types of Learning Algorithms

A. Supervised Learning

→ of the form $\{x_n \in R^d, y_n \in R^q\}_{n=1}^{N}$

→ also called inductive learning, where training data + desired outputs (labels) are given.



→ Supervised learning can be used for:

(i) prediction

(ii) classification (discrete labels), regression (real values)
true function values

Regression

→ Given $(x_1, y_1), (x_2, y_2) \ldots (x_n, y_n)$

→ Learn a function $f(x)$ to predict y given x

→ y is real-valued in a regression problem.

Classification

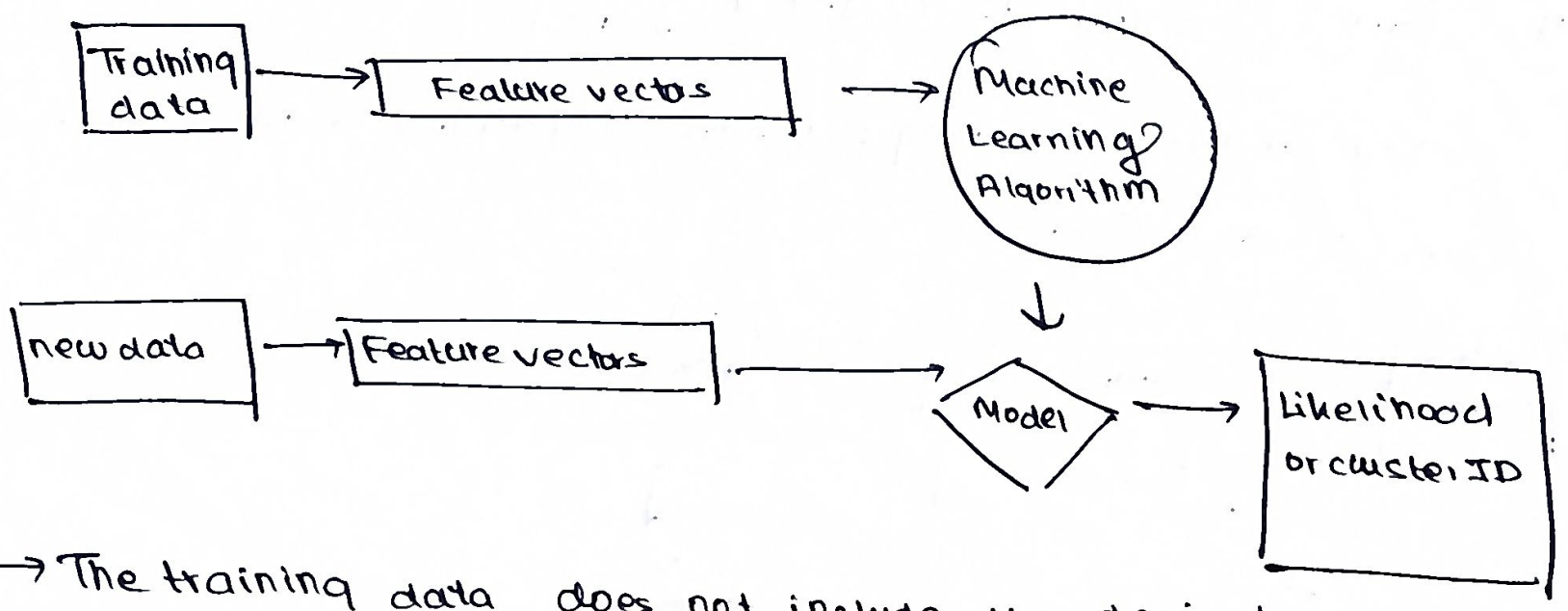
→ Given $(x_1, y_1), (x_2, y_2) \ldots (x_n, y_n)$

→ Learn a function $f(x)$ to predict y given x

→ y is categorical in a classification problem.

B. Unsupervised learning

→ of the form $\{x_n \in R^d\}_{n=1}^{N}$



→ The training data does not include the desired outputs - class labels of data are unknown.

→ Unsupervised learning works with unlabelled data, so there is no test data as such.

→ Given a set of data, the task is to establish the existence of classes or clusters in the data.

→ Unsupervised learning involves:

(i) clustering

(ii) probability distribution estimation

(iii) finding association (in features)

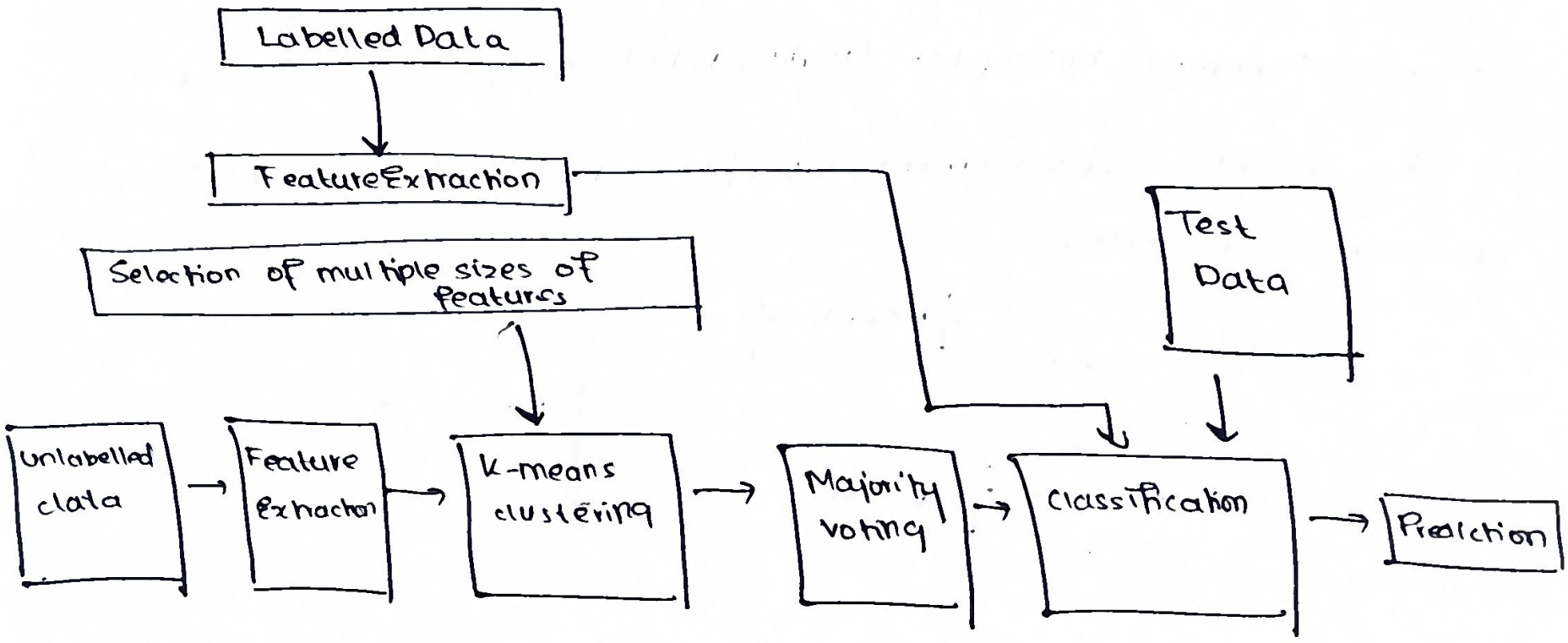
(iv) dimension reduction

c. **Semi - Supervised Learning**

→ The training data includes a few desired outputs.

→ The data is cheap, but labelled data is expensive.

→ Use small labelled training set to build an initial model, which is then refined using the unlabelled data.



Assumptions made in Semi-Supervised Learning

1. Continuity - points closer to each other are likely to have the same output label

2. Cluster Assumption - data points in the same cluster created are more likely to share an output label

3. Manifold Assumption - The data lies approximately on a manifold of a much lower dimension than the input space

D. Reinforcement Learning

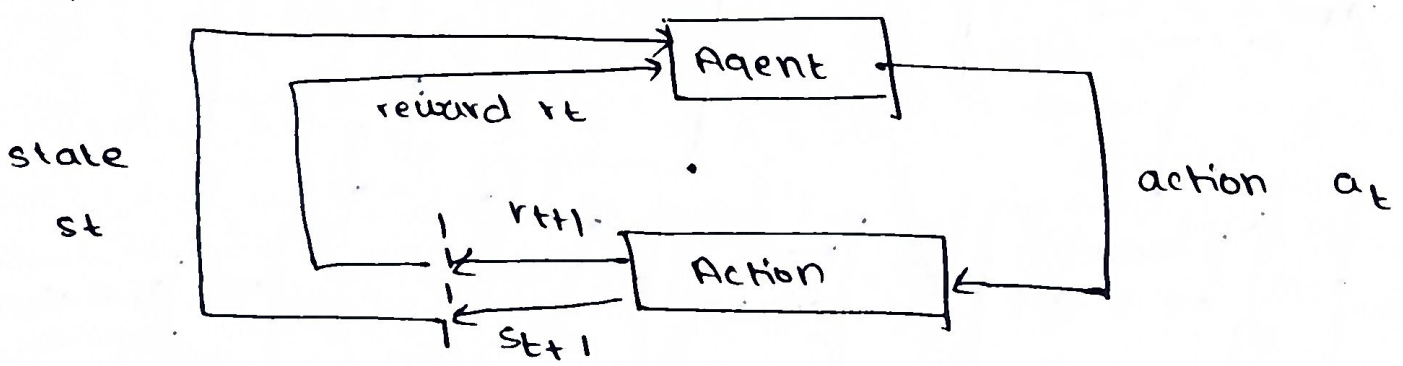
→ In such a framework, provide only a reward function which indicates to the learning algorithm that when it is doing correct and when it is doing poorly.

→ faces the exploration vs. exploitation allemma — explore new states but maximize reward

→ Then it is the job of the learning algorithm to figure out how to choose actions over time so as to obtain large rewards

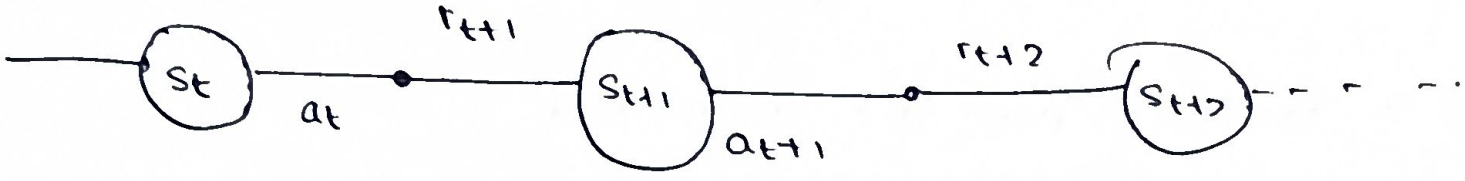
→ eg. autonomous helicopter flight, robot locomotion, chess play etc.

→ The agent - environment interface in a reinforcement learning problem is as follows:



→ The agent and environment interact at discrete time intervals: $t = 0, 1, 2, \ldots$

- Agent observes state at step t:
- Agent produces an action at step $t = a$
- Gets a resulting reward: r_{t+1}
- and moves to the resulting next state: s_{t+1}



Types of Reinforcement Learning?

① Positive - increase the strength and frequency of the behavior that impacts positively on the action taken by the agent

② Negative - strengthening of behavior that occurs because of a negative condition that should have been stopped or avoided

Implementation of Reinforcement Learning?

1. Value-based - aim to maximize a value function $v(s)$
2. Policy-based - come up with a policy that the action performed in every state helps it gain the maximum reward in the future.

Policy-based methods can be:

(i) deterministic - for any state, the same action is produced by the policy π

(ii) stochastic - every action has a certain probability.

3. Model-based - create a virtual model for each environment

The agent learns to perform in that specific environment.

Formulating a RL Problem

The following terms define the basic element of an RL problem:

1. Environment - physical world in which the agent operates
2. State - current situation of the agent
3. Reward - feedback from the environment
4. Policy - method to map agent's state to actions
5. Value - future reward that an agent would receive by taking an action in a particular state.

Mathematics of RL Problems

→ Markov Decision Processes (MDPs) are mathematical frameworks to describe an environment in RL.

→ An MDP consists of a finite set of environment states, a set of possible actions A(s) in each state, a real-valued reward function R(s) and a transition model P(s', s|a)

RL Algorithms

① Q-Learning - off-policy method in which the agent learns the value based on action derived from another policy.

② SARSA (State-Reward-Action-State-Action) - on-policy method

where it learns the value based on its current action a derived from its current policy.

(3) Deep Q-Networks (DQNs) - use neural networks to estimate Q-values. DQNs can only handle discrete, low-dimensional action spaces.

(4) Deep Deterministic Policy Gradient (DDPG) - uses learning policies in high dimensions.

Advantages and Disadvantages of RL

Advantages	Disadvantages
(i) doesn't require large labelled datasets	(i) not preferable for simple problems
(ii) can come up w/ new solutions not known to humans	(ii) requires lots of data and a lot of computation
(iii) goal-oriented	(iii) highly dependent on the quality of the reward function
(iv) adaptable - doesn't require retraining	(iv) difficult to debug and interpret

* Objectives of ML Algorithms

1. Supervised Algorithms : (i) Low E_{out}

or

(ii) maximize probabilistic terms

$$\text{error} = \frac{1}{N} \sum_{n=1}^{N} [y_n \neq g(x_n)]$$

E_{in} = for training set $E_{out}(g) \approx E_{in}(g)$

E_{out} = for testing set

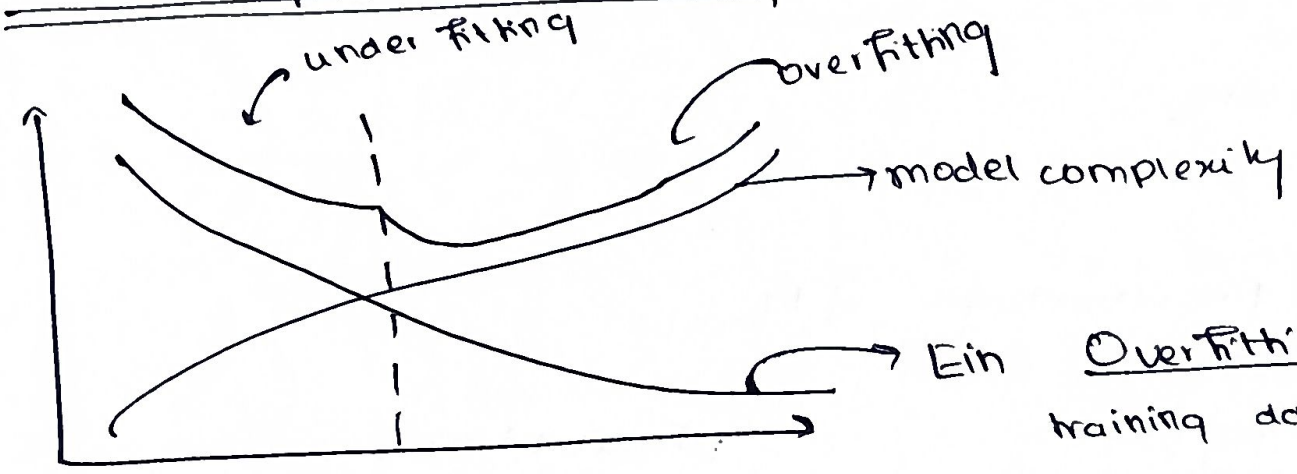
2. Unsupervised Algorithms : (i) minimize quantization error

(ii) minimize distance

(iii) MAP

(iv) MLE (maximum likelihood estimation)

* Overfitting and Underfitting



Overfitting - learns to match training data too well, does not generalize well to new data

Underfitting - model is too simplistic to capture the underlying structure of the data.

* Learning Techniques

A. Supervised Learning Techniques

1. Linear Classifier
   - based on numerical functions
   - find optimal decision boundary

2. Parametric (Probabilistic Functions)
   - Naive Bayes
   - Gaussian Discriminant Analysis
   - Hidden Markov Models
   - Probabilistic Graphical Model

3. Non-parametric (Instance-based functions)
   - K-nearest neighbors
   - Kernel regression
   - Kernel density estimation
   - Local regression

4. Non-metric - classification and regression tree (CART)

- Decision tree (a flow chart-like tree structure)

internal node = test on an attribute
branch = outcome of test
leaf = class labels / class distribution

5. Aggregation - Bagging (bootstrap aggregation)

- Adaboost

- Random Forest

6. Lazy Learners - generalization beyond the training data is delayed until a new instance is provided to the system.

eg. kNN

B. Unsupervised Learning Techniques

1. Clustering

- k-means clustering
- spectral clustering

2. Density estimation - Gaussian mixture model (GMM)

- graphical models

3. Dimensionality reduction - principal component analysis (PCA)

- factor analysis

* Preliminaries for Machine Learning?

(1) Input - an input vector is the data given as one input to the algorithm

- denoted by x (x_i i goes from 1 to no. of input dimensions m)

② Weights - w_{ij} = weighted connections between nodes i and j.

- weights are arranged into a matrix W.

③ Outputs - The output vector y, with elements y_j where (y_j j = 1 to n output dimensions)

Output = $y(x, W)$

④ Targets - target vector t (t_j j = 1 to n output dimensions)

- extra data that is needed for supervised learning

⑤ Activation Function - $g(\cdot)$ is a mathematical function in neural networks, that describes the firing of the neuron as a response to the weighted inputs.

⑥ Error : E, a function that computes the inaccuracies of the network as a function of the outputs y and targets t

* Weight Space

→ In supervised learning models, weight space refers to the space of all possible values for the parameters or weights of the model.

For eg. in a linear regression model:

$y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$

$w_0, w_1, \ldots w_n$ are the parameters / weights of the model.

→ The weight space would be the n-dimensional space formed by all possible combinations of the weights $w_0, w_1, w_2 \ldots w_n$

→ Each point in this space represents a particular set of weight values, and each set of weight values corresponds to a particular linear model.

→ The process of training a machine learning model involves searching through this weight space to find the optimal set of weights that minimizes a chosen loss function

* Training, Testing and Validation Sets

→ The sets are used as follows:

(i) training set to actually train the algorithm

(ii) the validation set to keep track of how well it is doing as it learns

(iii) the test set to produce the final results.

→ The learning process must be stopped before the algorithm overfits, which means that one needs to know how well it generalizes at each timestep.

→ Training data cannot be used for this, since it would not detect overfitting, but testing data cannot be used either, since it it is to be used for the final tests.

→ A third dataset is used for this purpose, called the validation set, since it is used to validate the learning so far.

↳ called cross-validation in statistics

→ The usual proportion of training : validation : testing is:

(i) 50 : 25 : 25 (with plenty of data)

(ii) 60 : 20 : 20

Training vs. Test Distribution

→ The general assumption is that the training and test samples are independently drawn from the same overall distribution of data.

called as IID - independent and identically distributed.

→ If examples are not independent, it requires collective classification.

→ If the test distribution is different, it requires transfer learning

Validation Types

(a) K-cross fold validation - dataset is divided into K subsets of equal size

- train model K times - each time use one subset as training data, and the part as validation data

- average performance over K iterations

(b) Leave-One-Out-Cross-Validation - a case of K-cross fold val, where K = no. of samples in the dataset.

- one point is held out for validation, rest used for training
- computationally expensive for large datasets

(c) Leave-P-Out Cross-Validation - leave out P (some) samples for validation, use the rest for training

(d) Multi-fold-cross-validation - any combination of (a), (b) or (c) w/ multiple iterations

* Curse of Dimensionality

High-Dimensional Data - data in which the number of features (p), are close to or larger than the number of observations (n) (data points)

→ The curse of dimensionality encapsulates the difficulties that arise as the number of dimensions in a dataset increases.

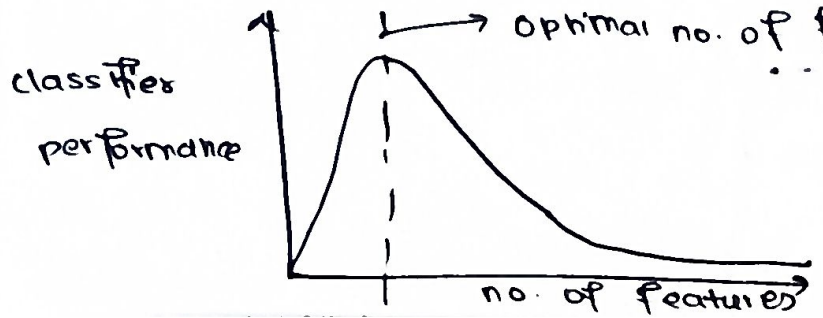
→ The curse of dimensionality causes the following problems:

(i) models become overly complex, and may capture noise rather than the underlying pattern, leading to poor generalization

(ii) more computationally complex

(iii) visualization of more than 3 dimensions is not straightforward

(iv) In high dimensional spaces, traditional distance metrics like Euclidean distance learning can become less meaningful.



* Dimensionality Reduction

→ reduce the dimensionality of the space by projecting the d-dimensional points into a k-dimensional space such that:

- $k << d$
- preserve distances as much as possible

→ some methods used include: (i) PCA
(ii) SVD - singular value decomposition

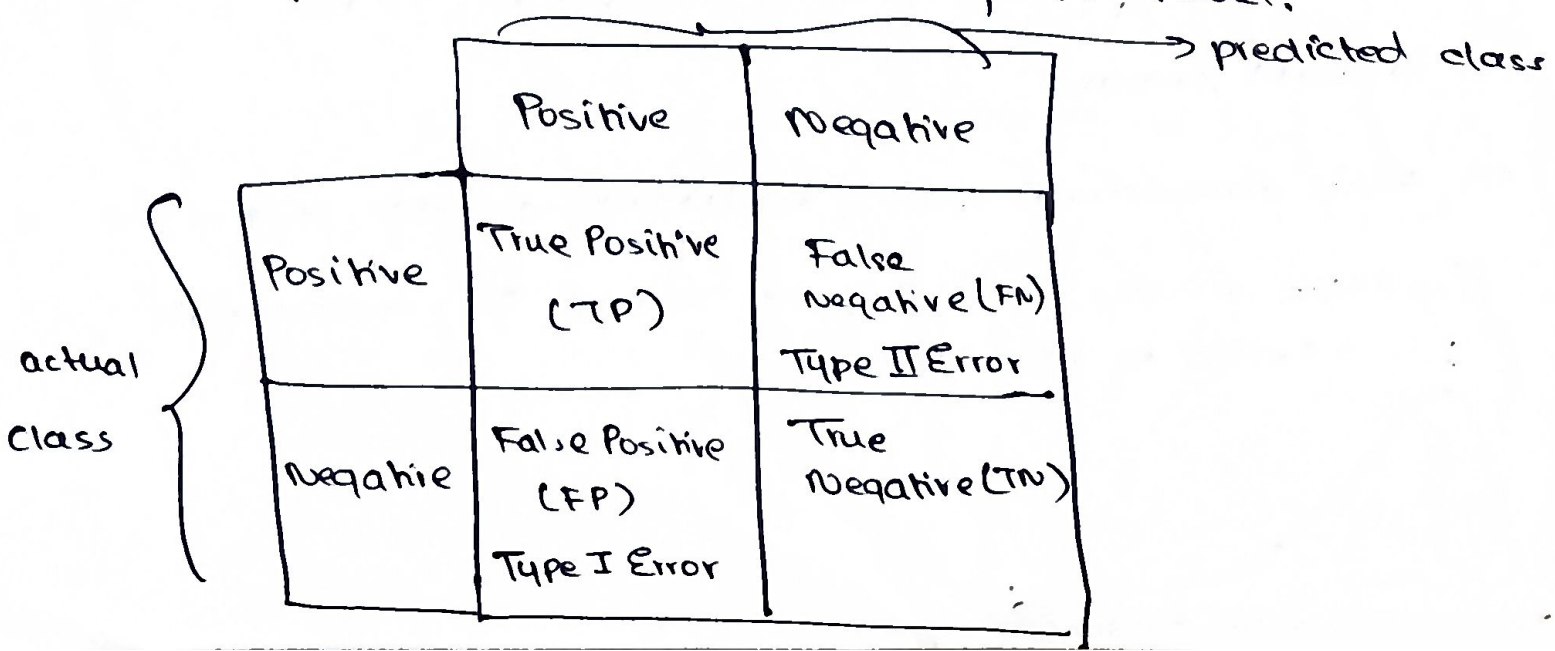
* Testing Machine Learning Algorithms

A. Performance Metrics for Classification Tasks

(i) accuracy
(ii) sensitivity
(iii) specificity
(iv) precision
(v) recall
(vi) F1 measure
(vii) True positive rate (TPR)
(viii) False negative rate (FNR)
(ix) ROC
(x) AUC, mcc

* Confusion Matrix

→ A square matrix used to compare the actual labels of a dataset with the predicted labels generated by the model.



Calculating metrics from the Confusion Matrix

		Predicted class: Positive	Predicted class: Negative	
Actual Class	Positive	True Positive (TP) ★	False Negative (FN)	Sensitivity $\frac{TP}{TP+FN}$
	Negative	False Positive (FP)	True Negative (TN) ★	Specificity $\frac{TN}{FP+TN}$
		Precision $\frac{TP}{TP+FP}$	Negative Predictive Value $\frac{TN}{FN+TN}$	Accuracy $\frac{TP+TN}{TP+TN+FP+FN}$

Sensitivity: also called the True Positive Rate - ratio of the no. of correct positive examples to the number classified as positive

$$= \frac{TP}{TP+FN}$$ (also called as recall) or true positive rate

Specificity: ratio of the no. of correct negative examples to the no. classified as negative

$$= \frac{TN}{TN+FP}$$

Precision: ratio of correct positive examples to the number of actual positive examples

$$= \frac{TP}{TP+FP}$$

* F1 measure

→ Precision and recall are somewhat inversely related ; in a manner that if the no. of false positives increases, the no. of false negatives often decreases and vice versa.

→ They can be combined to give a single measure, which can be written in terms of precision and recall as:

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

* TPR and FNR

$$\text{True positive rate} = \frac{TP}{TP + FN}$$

$$\text{False negative rate} = \frac{FN}{TP + FN}$$

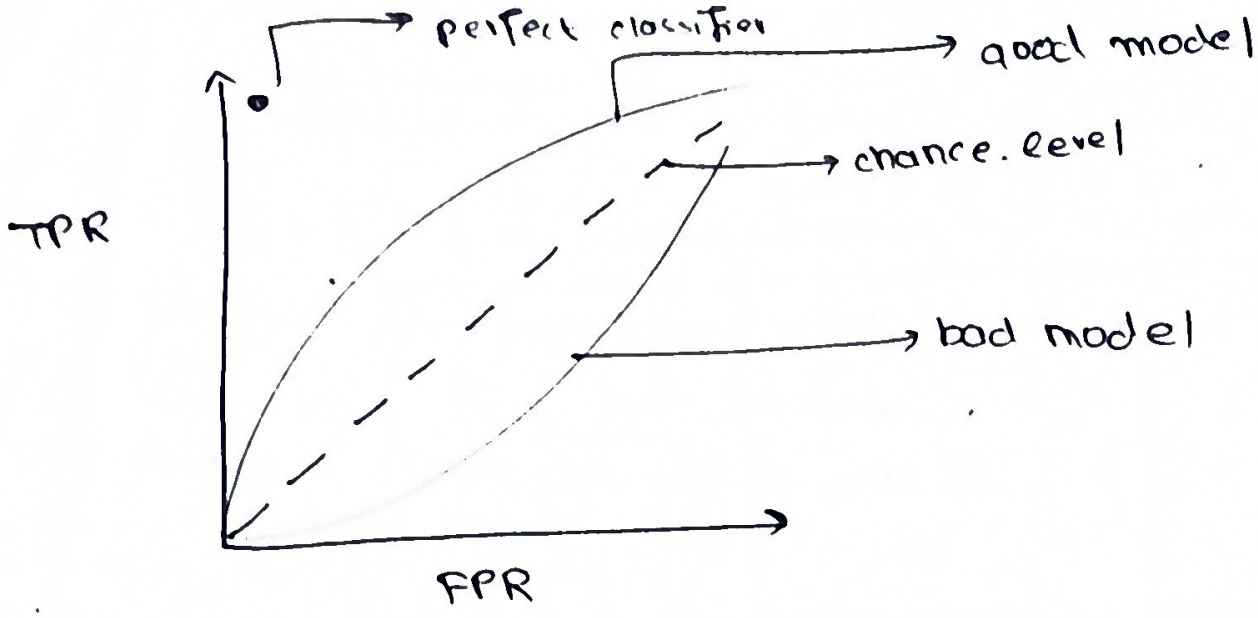
* Receiver Operator Characteristics (ROC) Curve

→ a graphical representation that plots the True Positive Rate against the False Positive Rate.

→ The TPR is plotted against the FPR at various threshold values.

→ A diagonal line from the origin ($x = y$), represents a random classifier with no predictive power.

→ The goal is for the ROC curve to be as close to the top left corner as possible, indicating high TPR and low FPR across different threshold settings.



* ROC - AUC

→ With imbalanced datasets, the Area under the Curve is calculated from the ROC.

→ The AUC indicates if the curve is above or below the diagonal

→ values range from 0 - 1 (100% correct predictions. ⇒ AUC = 1.0)

* Precision and Trueness

Precision - measure of variability of the algorithm, and a measure of how repeatable the predictions that the algorithm makes are.

Trueness - measure of how well the algorithm's predictions match reality - can be defined as the average distance between the correct output and the prediction

* Accuracy for Unbalanced Datasets

→ The accuracy computed from the confusion matrix assumes that the no. of +ve and -ve samples in the dataset are the same.

→ For an unbalanced dataset, the Matthew's Correlation Coefficient can be used.

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

if any in the dr. are 0 - set dr to 1

B. Performance Metrics for Regression Tasks

(i) Mean Absolute Error (MAE)

(ii) Mean Squared Error (MSE)

(iii) Root Mean Squared Error (RMSE)

(iv) Mean Absolute Percentage Error (MAPE)

(v) R^2 Score

(vi) Adjusted R^2 Score

* Residuals

→ It is not possible for a regression model to accurately predict the value of a continuous variable - it can also predict values that are lower or higher.

→ Residuals are the difference between the actual and predicted value.

$$e_i = y_i - \hat{y_i}$$

$\hat{y_i}$ → predicted value

y_i → actual value

e_i → residual

(i) Mean Absolute Error

→ sum of all the residuals divided by the total no. of points in the dataset

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|$$

→ When to use MAE - to find the model's ~~abso~~ average absolute distance.

(ii) Mean Square Error

→ square of the distances between the actual and predicted values.

$$MSE = \frac{1}{n} \sum_{i=1}^{N} (x_i - y_i)^2$$

(x_i: actual, y_i: predicted)

(iii) Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - y_i)^2}$$

→ When to use RMSE : (i) When there is a concern about large errors

(iv) R^2 Score

→ a statistical measure that tells how well the model is making all its predictions on a scale of zero to one

$$R^2 = 1 - \frac{RSS}{TSS}$$

RSS = ~~squared~~ sum of squares of residuals

TSS = total sum of squares

$$RSS = \sum (y_i - \hat{y}_i)^2$$

↓ act ↓ pred

$$TSS = \sum (y_i - \bar{y})^2$$

↓ act ↳ mean value

→ When to use RSS - get the accuracy of the model on a % scale like in a classification model.

(v) Mean Absolute Percentage Error

→ measures the absolute percentage diff. between the actual and predicted values

$$MAPE = \frac{1}{n} \sum_{i=1}^{N} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100$$

(vi) Adjusted R^2 - Score

→ The adjusted R^2 takes into account the number of independent variables in the model and penalizes the R^2 score for adding variables that do not improve the model's performance

$$\text{Adjusted } R^2 = 1 - \left(\frac{(1-R2)(n-1)}{(n-p-1)} \right)$$

R^2 = coeff. of determination

n = no. of observations

p = no. of independent var

Adjusted R^2 will be < R^2 score

* Methods for Performance Evaluation

→ The performance of a model may depend on other factors other than the learning algorithm.

These include : (i) class distribution
(ii) cost of misclassification
(iii) size of training and testing set

→ Some methods for performance evaluation are:

(i) Holdout Samples (can use random subsampling)

(ii) Cross validation

(iii) Bootstrapping (sampling w/ replacement)

→ Class imbalance can be handled by:

(i) sample from the larger class so that the size of the two class is the same.

(ii) replicate the data of the class of interest so that the classes are balanced (may lead to overfitting)

* Bias-Variance Tradeoff

Bias - The difference between a model's predictions and the actual distribution of the value it tries to predict.

- Models with high bias oversimplify the data distribution rule/fn, resulting in high errors.
- Bias can be measured using:
  (i) Mean Squared Error (MSE)
  (ii) Mean Absolute Error (MAE)

Features of models with high bias

1. underfitting - oversimplify the solution, cannot grasp underlying features
2. Low training accuracy.
3. Oversimplification

Overcoming High Bias

1. Incorporating additional features
2. Increasing the no. of training iterations
3. Avoiding high-bias algorithm · like linear, Logistic regression & instead using non-linear algos like KNN, SVM, decision trees
4. Decreasing regularization to prevent underfitting

Variance - measures how much a distribution on several sets of data values differs from each other

- can be measured using cross-validation expts.

→ A model w/ high variance depends heavily on the training data and has a limited ability to generalize.

→ This is often seen in non-linear algos, due to their high flexibility.

Features of models with high variance

1. Low Testing accuracy
2. Overfitting
3. Overcomplexity

Overcoming High variance

1. Reducing the no. of features
2. Replacing model with a simpler one.
3. Increasing training data diversity
4. Avoiding high-variance algos.

+

Cross validation
Feature selection
Regularization
Ensemble methods
Early Stopping

Bias - Variance Tradeoff

→ The bias-variance tradeoff considers two opposing scenarios:

Scenario1 - The model works perfectly w/ the data it was trained on.
- provides incorrect predictions w/ new data

Scenario2 - The model struggles to grasp the intricacies of the data and thus fails to make an accurate prediction

→ Striking a balance between accuracy and the ability to make predictions beyond the training data is called the bias-variance tradeoff.

→ The possible cases of bias & variance are possible.

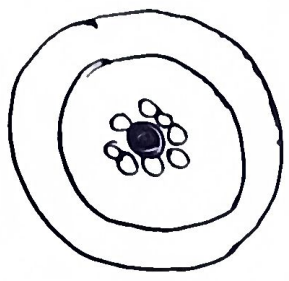
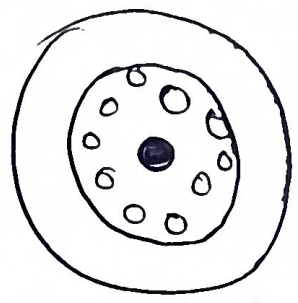
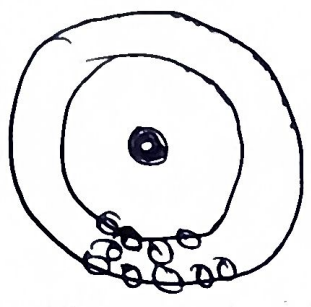
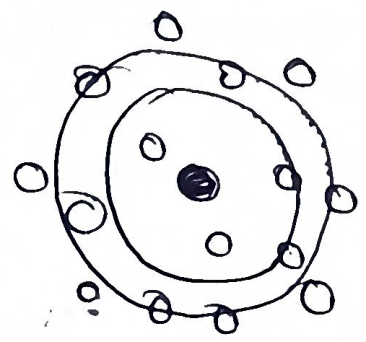
(i) Low Bias, Low Variance - ideal scenario

(ii) Low Bias, High Variance - overfitting

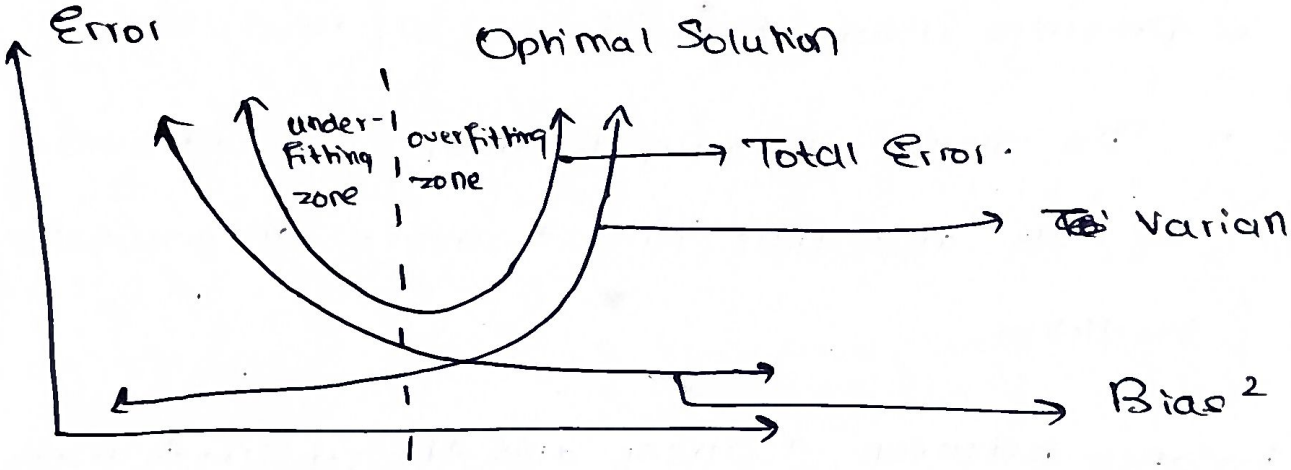
(iii) High Bias, Low Variance - underfitting (model doesn't learn)

(iv) High Bias, High Variance - Incorrect predicts & inconsistent

→ The 4 scenarios are depicted as follows:

	Low Variance	High variance
Low Bias		
High Bias		

→ Graphically,



Methods of Adjusting Bias-variance Balance

for ↓ bias, ↑ variance

1. kNN - increase the value of k

   ↑ bias, ↓ variance

2. SVM - for ↓ bias, ↑ variance

   change the C parameter

   ↑ bias, ↓ variance