PRESIDENCY UNIVERSITY

BENGALURU

MODULE 2

SUPERVISED MACHINE LEARNING ALGORITHMS

PRESIDENCY UNIVERSITY

Private University Estd. in Karnataka State by Act No. 41 of 2013

40 YEARS OF ACADEMIC WISDOM

AGENDA

- Introduction to the Machine Learning (ML) Framework
- Types of ML,
- Types of variables/features used in ML algorithms,
- One-hot encoding,
- Simple Linear Regression,
- Multiple Linear Regression,
- Evaluation metrics for regression model

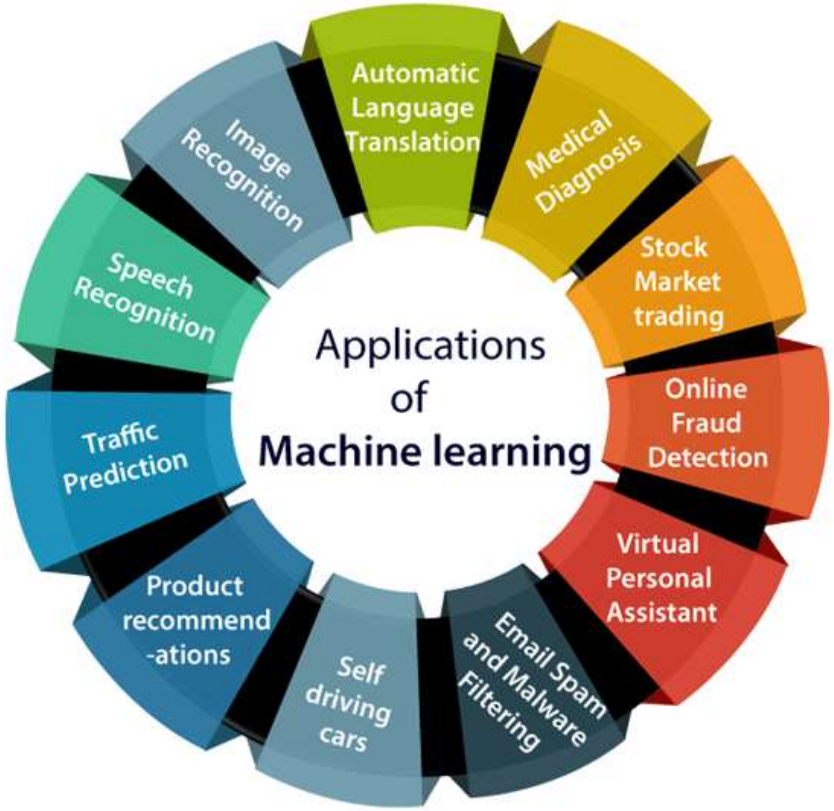
AGENDA

- Classification models
- Decision Tree algorithms using Entropy and Gini Index as measures of node impurity,
- Model evaluation metrics for classification algorithms,
- Cohen's Kappa Statistic,
- Multi-class classification
- Class Imbalance problem.
- Naïve Bayes Classifiers
- Naive Bayes model for sentiment classification

What is Machine Learning

- **Definition**
  - A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it.
  - The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.
- **Importance of machine learning**
  - Finding hidden patterns and extracting useful information from data
  - Solving complex problems and decision making in many fields (applications)
- **Feature of ML: Data-Driven Technology**
  - similar to data mining as it also deals with huge amount of data.
  - uses data to detect various patterns in a given dataset
  - learn from past data and improve automatically.

APPLICATIONS OF MACHINE LEARNING



LIFECYCLE OF MACHINE LEARNING

Gathering Data

- To identify the different data sources, as data can be collected from various sources such as **files**, **database** or **internet**.
- The quantity and quality of the collected data will determine the accuracy of the prediction and efficiency of the output.
- This step includes the below tasks:

➢ Identify various data sources

➢ Collect data

➢ Integrate the data obtained from different sources – This coherent set of data is called dataset

LIFECYCLE OF MACHINE LEARNING

Data preparation: This step can be further divided into two processes:

- **Data exploration:** To understand the characteristics, format, and quality of data to find Correlations, general trends, and outliers for an effective outcome.
- **Data pre-processing:** Cleaning of data is required to address the quality issues: Missing Values, Duplicate data, Invalid data and Noise, which can be solved using filtering techniques.

Data Wrangling

- Reorganizing, mapping and transforming raw, unstructured data to a useable format.
- This step involves data aggregation and data visualization.

LIFECYCLE OF MACHINE LEARNING

Data Analysis

- The aim of this step is to build a machine learning model to analyze the data and review the outcome.

Train Model

- Datasets are used to train the model using various machine learning algorithms – to understand various patterns, rules, and, features.

Test Model

- Tests accuracy of the model with respect to the requirements of project or problem.

Deployment

- Performance of the project is checked with the available data and deployed which is similar to making the final report for a project.

DIFFERENCE BETWEEN AI & ML

Artificial Intelligence	Machine learning
Artificial intelligence is a technology which enables a machine to simulate human behavior.	Machine learning is a subset of AI which allows a machine to automatically learn from past data without programming explicitly.
AI is working to create an intelligent system which can perform various complex tasks.	Machine learning is working to create machines that can perform only those specific tasks for which they are trained.
The main applications of AI are Siri, Expert System, Online game playing, intelligent humanoid robot, etc.	The main applications of machine learning are Online recommender system and Google search algorithms
It includes learning, reasoning, and self-correction.	It includes learning and self-correction when introduced with new data.

MACHINE LEARNING - DATASET

- **A dataset** is a collection of data in which data is arranged in some order. A dataset can contain any data from a series of an array to a database table.
- **Types of data in datasets**
  - **Numerical data:** Such as house price, temperature, etc.
  - **Categorical data:** Such as Yes/No, True/False, Blue/green, etc.
  - **Ordinal data:** These data are similar to categorical data but can be measured on the basis of comparison.
- **Types of Datasets**
  - **Training Dataset**: This data set is used to train the model i.e.; these datasets are used to update the weight of the model.

CONTD...

- **Validation Dataset**
  - It is used to verify that the increase in the accuracy of the training dataset is actually increased if we test the model with the data that is not used in the training.
  - If the accuracy over the training dataset increases while the accuracy over the validation dataset decreases, then this results in the case of high variance i.e., overfitting.
- **Test Dataset**
  - Most of the time when we try to make changes to the model based upon the output of the validation set then unintentionally, we make the model peek into our validation set and as a result, our model might get overfit on the validation set as well.
  - To overcome this issue, we have a test dataset that is only used to test the final output of the model in order to confirm the accuracy.

CONTD...

How to get the datasets / Popular sources for ML dataset

- Kaggle Dataset
- UCI Machine Learning Repository
- Datasets via AWS
- Google's Dataset Search Engine
- Microsoft Dataset
- Awesome Public Dataset Collection
- Government Datasets
- Computer Vision Datasets
- Scikit-learn dataset

MACHINE LEARNING- DATA PREPROCESSING

- **Definition**: Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model.
- **Significance**
  - A real-world data contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models.
  - Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model.
- **Steps**

Getting the dataset

- The data is usually put in CSV file ("**Comma-Separated Values**" files; it is a file format which allows us to save the tabular data, such as spreadsheets. It is useful for huge datasets and can use these datasets in programs).

CONTD...

Importing libraries

- **Numpy:** used for including any type of mathematical operation in the code.
- **Matplotlib:** used to plot any type of charts in Python for the code.
- **Pandas:** used for importing and managing the datasets. It is an open-source data manipulation and analysis library.

Importing datasets

- **read_csv() function:** used to read a csv file
- To distinguish the matrix of features (independent variables) and dependent variables from dataset - **iloc[ ]** method is used to extract the required rows and columns from the dataset.
- To extract dependent variables, again, we will use Pandas.iloc[] method.

CONTD...

- ***Handling Missing Data***

- **By deleting the particular row:** delete the specific row or column which consists of null values. But this way is not so efficient and removing data may lead to loss of information which will not give the accurate output.
- **By calculating the mean:** In this way, we will calculate the mean of that column or row which contains any missing value and will put it on the place of missing value. This strategy is useful for the features which have numeric data such as age, salary, year, etc. Here, we will use this approach.
- To handle missing values, we will use **Scikit-learn** library in our code, which contains various libraries for building machine learning models. Here we will use **Imputer** class of **sklearn.preprocessing** library.

CONTD...

- ***Encoding Categorical Data***

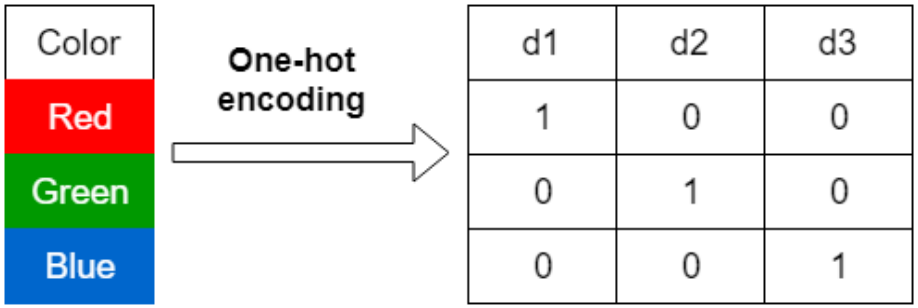
- **LabelEncoder()** class from **preprocessing** library is used for encoding the variables into digits.
- Categorical variables usually have strings for their values. Many machine learning algorithms do not support string values for the input variables. Therefore, we need to replace these string values with numbers. This process is called **categorical variable encoding**.
- Types of encoding:
  - **One-hot encoding**
  - **Dummy encoding**

❖ **One-hot encoding**

- In one-hot encoding, we create a new set of dummy (binary) variables that is equal to the number of categories (k) in the variable.

CONTD...

- For example, let's say we have a categorical variable ***Color*** with three categories called "Red", "Green" and "Blue", we need to use three dummy variables to encode this variable using one-hot encoding. A dummy (binary) variable just takes the value 0 or 1 to indicate the exclusion or inclusion of a category.

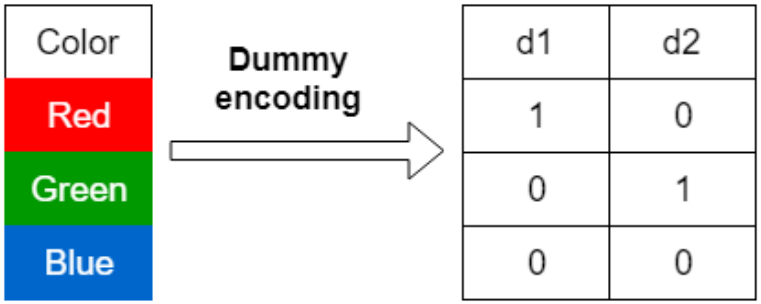


d1	d2	d3
1	0	0
0	1	0
0	0	1

- In one-hot encoding,
- "Red" color is encoded as [1 0 0] vector of size 3.
- "Green" color is encoded as [0 1 0] vector of size 3.
- "Blue" color is encoded as [0 0 1] vector of size 3.

CONTD...

- Dummy encoding
  - Dummy encoding also uses dummy (binary) variables. Instead of creating a number of dummy variables that is equal to the number of categories (k) in the variable, dummy encoding uses k-1 dummy variables.
  - To encode the same Color variable with three categories using the dummy encoding, we need to use only two dummy variables



d1	d2
1	0
0	1
0	0

  - In dummy encoding, "Red" color is encoded as [1 0] vector of size 2, "Green" color is encoded as [0 1] vector of size 2, "Blue" color is encoded as [0 0] vector of size 2.
  - Dummy encoding removes a duplicate category present in the one-hot encoding.

CONTD...

- **Splitting dataset into training, validation and test set**

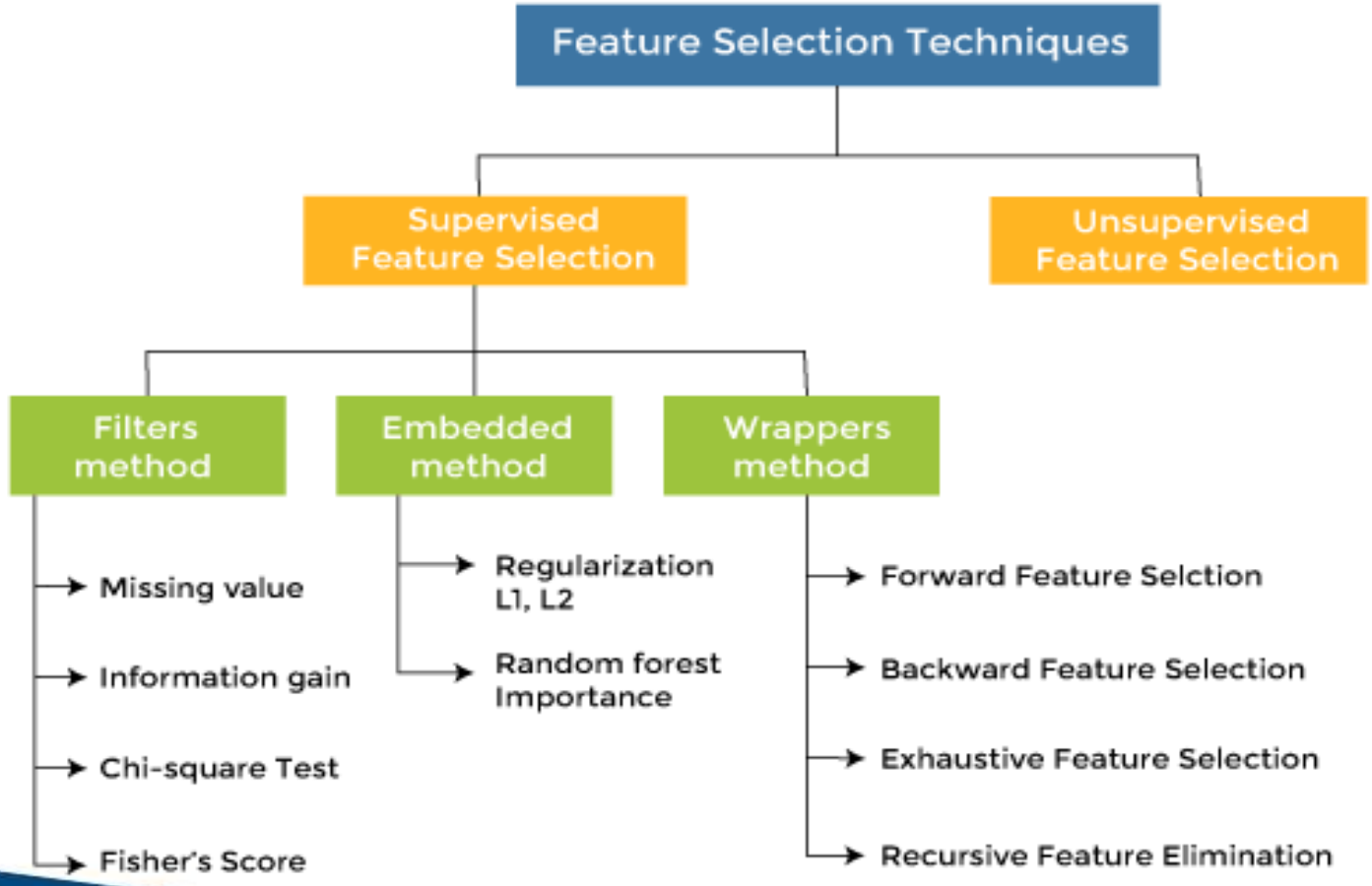
- **Feature scaling**
  - Feature scaling is the final step of data pre-processing in machine learning.
  - It is a technique to standardize the independent variables of the dataset in a specific range.
  - In feature scaling, we put our variables in the same range and in the same scale so that no any variable dominates the other variable.

FEATURE SELECTION TECHNIQUES IN MACHINE LEARNING

FEATURE SELECTION

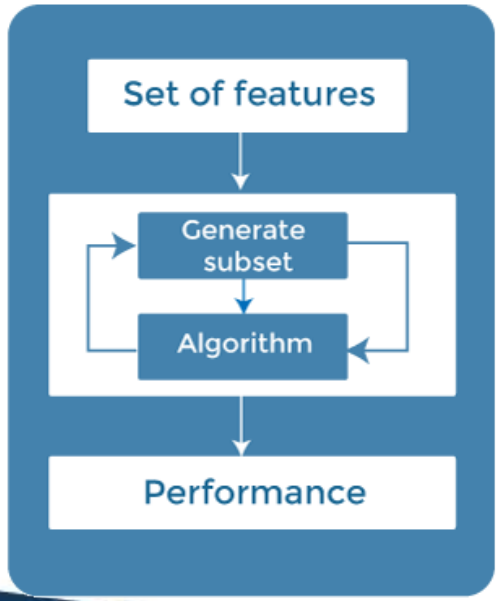
- A **feature** is an attribute that has an impact on a problem or is useful for the problem, and choosing the important features for the model is known as feature selection.
- **Definition**: Feature selection is a way of selecting the subset of the most relevant features from the original features set by removing the redundant, irrelevant, or noisy features.
- **Significance** of Feature Selection:
  - It helps in avoiding the curse of dimensionality.
  - It helps in the simplification of the model so that it can be easily interpreted by the researchers.
  - It reduces the training time.
  - It reduces overfitting hence enhance the generalization.

FEATURE SELECTION TECHNIQUES



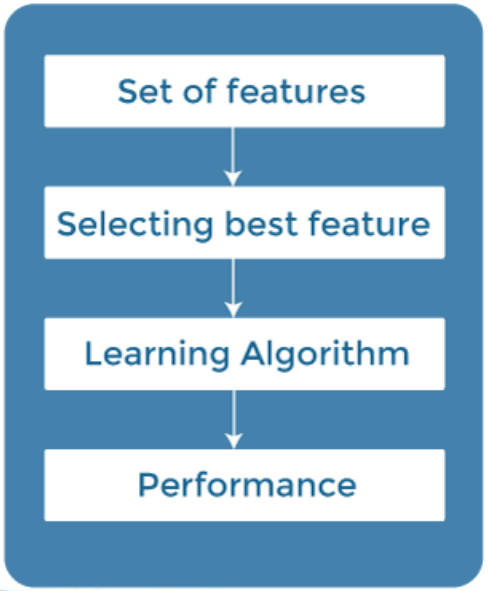
SUPERVISED FEATURE SELECTION TECHNIQUES

- **Wrapper Methods**
  - In wrapper methodology, selection of features is done by considering it as a search problem, in which different combinations are made, evaluated, and compared with other combinations.
  - It trains the algorithm by using the subset of features iteratively.
  - On the basis of the output of the model, features are added or subtracted, and with this feature set, the model has trained again.



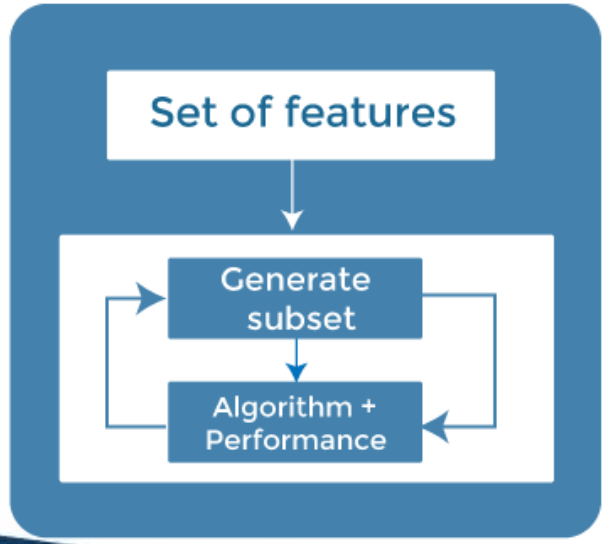
CONTD...

- **Filter Methods**
  - In Filter Method, features are selected on the basis of statistics measures. This method does not depend on the learning algorithm and chooses the features as a pre-processing step.
  - The filter method filters out the irrelevant feature and redundant columns from the model by using different metrics through ranking.
  - The advantage of using filter methods is that it needs low computational time and does not overfit the data.



CONTD...

- **Embedded Methods**
  - Embedded methods combined the advantages of both filter and wrapper methods by considering the interaction of features along with low computational cost. These are fast processing methods similar to the filter method but more accurate than the filter method.
  - These methods are also iterative, which evaluates each iteration, and optimally finds the most important features that contribute the most to training in a particular iteration.



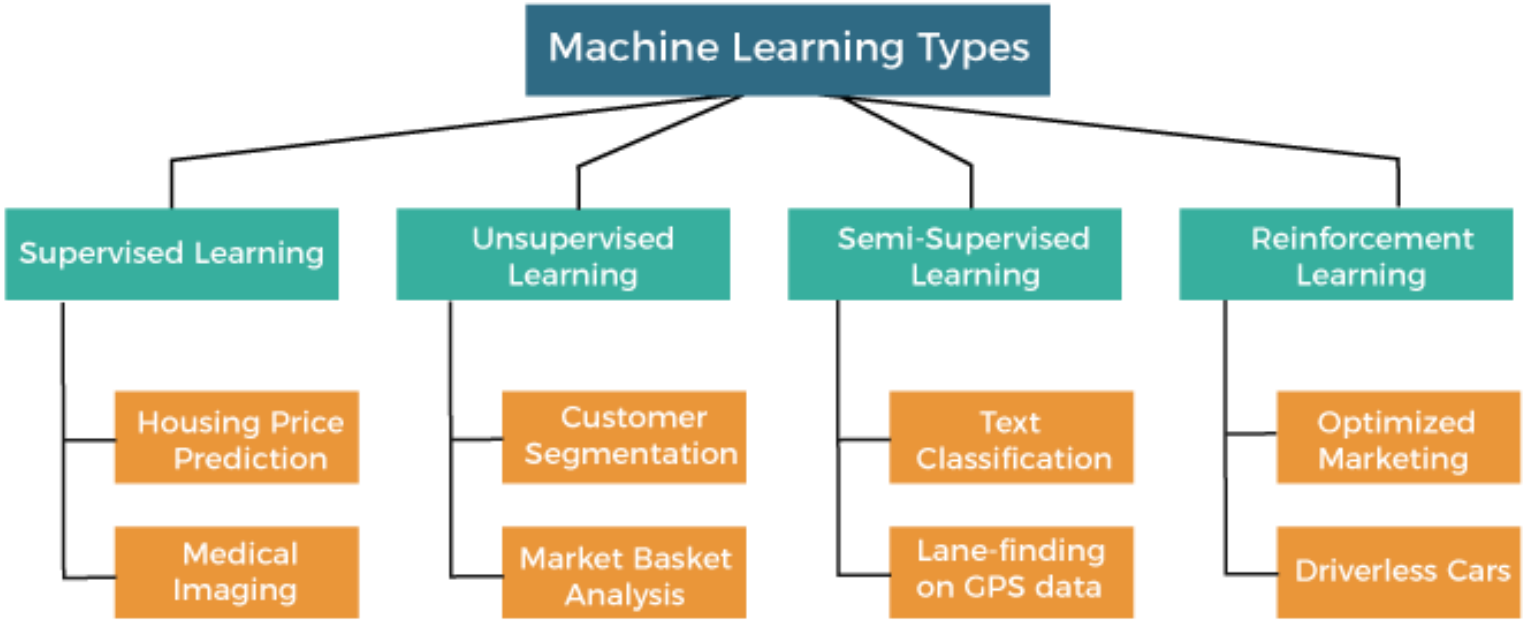
FEATURE ENGINEERING FOR MACHINE LEARNING

- **Feature engineering is the pre-processing step of machine learning, which extracts features from raw data**.
- Feature engineering in ML contains mainly four processes:
  - **Feature Creation:** finding the most useful variables to be used in a predictive model.**,**
  - **Transformations:** This step of feature engineering involves adjusting the predictor variable to improve the accuracy and performance of the model.
  - **Feature Extraction:** Is an automated feature engineering process that generates new variables by extracting them from the raw data
  - **Feature Selection:** Is a way of selecting the subset of the most relevant features from the original features set by removing the redundant, irrelevant, or noisy features

FEATURE ENGINEERING TECHNIQUES FOR ML

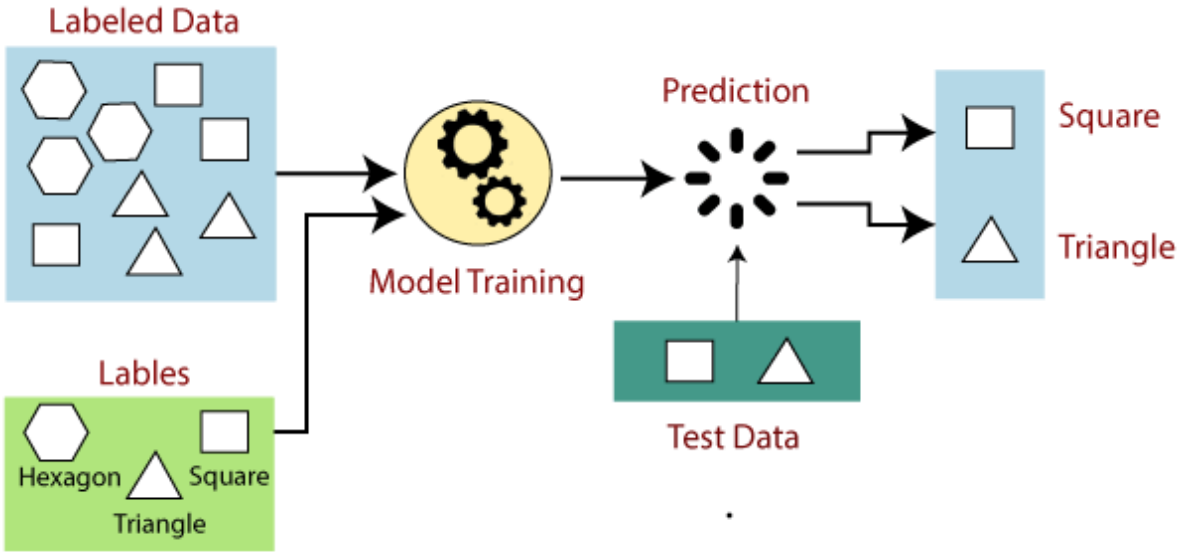
- **Imputation**: Imputation is responsible for handling irregularities within the dataset.
- **Handling Outliers**: Standard deviation can be used to identify the outliers. Z-score can also be used to detect outliers.
- **Log Transform**: helps in handling the skewed data, and it makes the distribution more approximate to normal after transformation.
- **Binning**: used to normalize the noisy data.
- **Feature Split**: is the process of splitting features intimately into two or more parts and performing to make new features.
- **One hot encoding**: It is a technique that converts the categorical data in a form so that they can be easily understood by machine learning algorithms and hence can make a good prediction.

MACHINE LEARNING - TYPES



SUPERVISED LEARNING

- Supervised learning is the types of machine learning in which **machines are trained using well "labelled" training data**, and on basis of that data, machines predict the output. The labelled data means some **input data is already tagged with the correct output.**



TYPES OF SUPERVISED LEARNING

Regression Algorithms

- Are used if there is a relationship between the input variable and the output variable. Example: Weather forecasting, Market Trends, etc.
- Regression algorithms under supervised learning: Linear Regression, Non-Linear Regression, Polynomial Regression, Ridge Regression and Lasso Regression.

Classification Algorithms

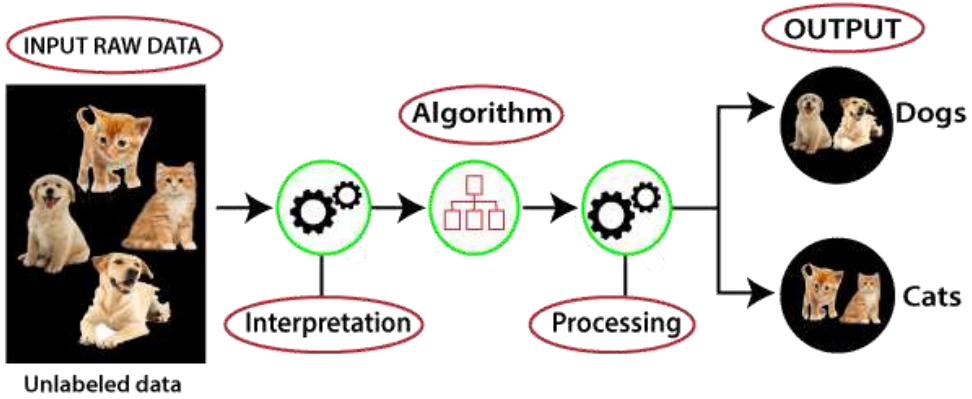
- Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc. Example: Spam Filtering.
- Classification algorithms under supervised learning: Random Forest, Decision Trees, Logistic Regression, Support vector Machines

IMPORTANT TERMINOLOGIES

- **Dependent Variable:** The main factor in Regression analysis which we want to predict or understand is called the dependent variable. It is also called **target variable**.
- **Independent Variable:** The factors which affect the dependent variables or which are used to predict the values of the dependent variables are called independent variable, also called as a **predictor**.
- **Outliers:** Outlier is an observation which contains either very low value or very high value in comparison to other observed values. An outlier may hamper the result, so it should be avoided.
- **Multicollinearity:** If the independent variables are highly correlated with each other than other variables, then such condition is called Multicollinearity.
- **Overfitting:** If our algorithm works well with the training dataset but not well with test dataset, then such problem is called **Overfitting**.
- **Underfitting:** If our algorithm does not perform well even with training dataset, then such problem is called **underfitting**.

UNSUPERVISED MACHINE LEARNING

- **Unsupervised learning** is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision.



- The goal of unsupervised learning is to find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format.

UNSUPERVISED MACHINE LEARNING - TYPES

- **Types:**
  - **Clustering**: Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group.
  - **Association**: An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset.
- **Unsupervised learning algorithms**: K-means clustering, Hierarchal clustering, Anomaly detection, Neural Networks, Principle Component Analysis, Apriori algorithm
- **Advantage of Unsupervised Learning**: "Preferable" as it is easy to get unlabeled data in comparison to labeled data.
- **Disadvantages of Unsupervised Learning**: The result might be less accurate as input data is not labeled, and algorithms do not know the exact output in advance.

SEMI-SUPERVISED LEARNING

- **Semi-Supervised learning** is a type of Machine Learning algorithm that lies between Supervised and Unsupervised machine learning.
- The main aim of semi-supervised learning is to effectively use all the available data, rather than only labelled data like in supervised learning.
- **Advantages:** It is highly efficient and is used to solve drawbacks of Supervised and Unsupervised Learning algorithms.
- **Disadvantages**
  - Iterations results may not be stable.
  - We cannot apply these algorithms to network-level data.
  - Accuracy is low.

REINFORCEMENT LEARNING

- Reinforcement learning works on a feedback-based process, in which an AI agent (A software component) automatically explore its surrounding by hitting & trail, taking action, learning from experiences, and improving its performance.
- Agent gets rewarded for each good action and get punished for each bad action; hence the goal of reinforcement learning agent is to maximize the rewards.
- In reinforcement learning, there is no labelled data like supervised learning, and agents learn from their experiences only.
- Due to its way of working, reinforcement learning is employed in different fields such as **Game theory, Operation Research, Information theory, multi-agent systems.**
- A reinforcement learning problem can be formalized using **Markov Decision Process(MDP).**

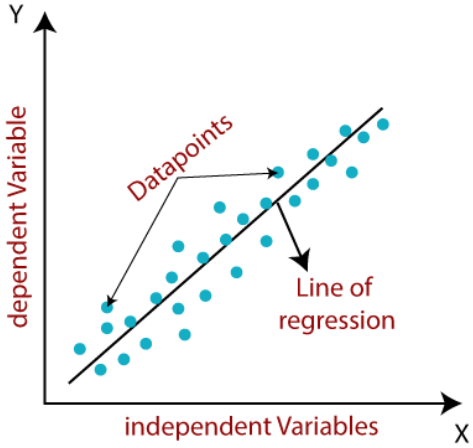
REINFORCEMENT LEARNING

- **Categories of Reinforcement Learning**
  - **Positive Reinforcement Learning:** Specifies increasing the tendency that the required behavior would occur again by adding something.
  - **Negative Reinforcement Learning:** It increases the tendency that the specific behavior would occur again by avoiding the negative condition.
- **Applications**: Robotics, Text Mining, Resource Management, Video Games.
- **Advantages**
  - The learning model of RL is similar to the learning of human beings; hence most accurate results can be found.
  - Helps in achieving long term results.
- **Disadvantages**
  - RL algorithms require huge data and computations.
  - Too much reinforcement learning can lead to an overload of states which can weaken the results.

LINEAR REGRESSION ANALYSIS

- It is a statistical method that is used for predictive analysis.
- Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc.
- Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression.
- Mathematically, we can represent a linear regression as:

  $y= a_0+a_1x+ \varepsilon$



LINEAR REGRESSION ANALYSIS

- **Types of Linear Regression**
  - **Simple Linear Regression:** If a single independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.
  - **Multiple Linear regression:** If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.
- **Model Performance: R-squared method:**
  - R-squared is a statistical method that determines the goodness of fit.
  - The high value of R-square determines the less difference between the predicted values and actual values and hence represents a good model.
  - It can be calculated from the below formula:

$$\text{R-squared} = \frac{\text{Explained variation}}{\text{Total Variation}}$$

SIMPLE LINEAR REGRESSION

- Models the relationship between a dependent variable and a single independent variable. The relationship shown by a Simple Linear Regression model is linear or a sloped straight line.
- Simple Linear regression algorithm has mainly two objectives:
  - Model the relationship between the two variables. Eg: Income and expenditure, experience and Salary, etc.
  - Forecasting new observations. Such as Weather forecasting according to temperature, Revenue of a company according to the investments in a year, etc.
- The Simple Linear Regression model can be represented using the below equation:

$$y = a0 + a1x + \varepsilon$$

a0= It is the intercept of the Regression line (can be obtained putting x=0)

a1= It is the slope of the regression line, which is either increasing or decreasing.

ε = The error term. (For a good model it will be negligible)

MULTIPLE LINEAR REGRESSION

- Multiple Linear Regression is one of the important regression algorithms which models the linear relationship between a single dependent continuous variable and more than one independent variable.
- For MLR, the dependent or target variable(Y) must be the continuous/real, but the predictor or independent variable may be of continuous or categorical form.
- Each feature variable must model the linear relationship with the dependent variable.
- MLR tries to fit a regression line through a multidimensional space of data-points.
- **Example:** Prediction of CO_2 emission based on engine size and number of cylinders in a car.

MULTIPLE LINEAR REGRESSION

- **MLR equation:**
  - In Multiple Linear Regression, the target variable(Y) is a linear combination of multiple predictor variables x_1, x_2, x_3, ...,x_n.

Y= b₀+b₁x₁+ b₂x₂+ b₃x₃+..... bnxn

where, Y= Output/Response variable, b_0, b_1, b_2, b_3 , b_n....= Coefficients of the model, x_1, x_2, x_3, x_4,...= Various Independent/feature variable

- **Assumptions for Multiple Linear Regression:**
  - A **linear relationship** should exist between the Target and predictor variables.
  - The regression residuals must be **normally distributed**.
  - MLR assumes little or **no multicollinearity** (correlation between the independent variable) in data.

EVALUATION METRICS FOR REGRESSION MODEL

- In regression problems, the prediction error is used to define the model performance. The prediction error is also referred to as residuals and it is defined as the difference between the actual and predicted values.
- Residuals are important when determining the quality of a model.
- **Residual = actual value — predicted value**

  $error(e) = y - \hat{y}$

- We can technically inspect all residuals to judge the model's accuracy, but this does not scale if we have thousands or millions of data points. That's why we have summary measurements that take our collection of residuals and condense them into a *single* value representing our model's predictive ability.

EVALUATION METRICS FOR REGRESSION MODEL - Mean Absolute Error (MAE)

- It is the average of the absolute differences between the actual value and the model's predicted value.

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

where, N = total number of data points, Yi = actual value, $\hat{Y}i$ = predicted value.

- A small MAE suggests the model is great at prediction, while a large MAE suggests that your model may have trouble in certain areas. MAE of 0 means that your model is a perfect predictor of the outputs.
- **Advantages of MAE**: It is most Robust to outliers.
- **Disadvantages of MAE**: The graph of MAE is not differentiable so we have to apply various optimizers like Gradient descent which can be differentiable.

EVALUATION METRICS FOR REGRESSION MODEL – Mean Squared Error

- It is the average of the squared differences between the actual and the predicted values. Lower the value, the better the regression model.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

where, **n** = total number of data points, **yi** = actual value, **ŷi** = predicted value

- If you have outliers in the dataset then it penalizes the outliers most and the calculated MSE is bigger.
- **Advantages of MSE** - The graph of MSE is differentiable, so you can easily use it as a loss function.
- **Disadvantages of MSE** - If you have outliers in the dataset then it penalizes the outliers most and the calculated MSE is bigger. So, in short, It is not Robust to outliers which were an advantage in MAE.

EVALUATION METRICS FOR REGRESSION MODEL – Route Mean Squared Error

- It is the average root-squared difference between the real value and the predicted value.
- lower the RMSE value, the better the model is with its predictions.
- A Higher RMSE indicates that there are large deviations between the predicted and actual value.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

where, **n** = total number of data points, **yj** = actual value, **ŷj**= predicted value

- **Advantages of RMSE**: The output value is in the same unit as the required output variable which makes interpretation of loss easy.
- **Disadvantages of RMSE:** It is not that robust to outliers as compared to MAE.

EVALUATION METRICS FOR REGRESSION MODEL – R Squared

- R2 score is a metric that tells the performance of your model, not the loss in an absolute sense that how many wells did your model perform.
- So, with help of R squared we have a baseline model to compare a model which none of the other metrics provides.
- The same we have in classification problems which we call a threshold which is fixed at 0.5. So basically R2 squared calculates how must regression line is better than a mean line.

$$\textbf{R2 Squared} = 1 - \frac{SSr}{SSm}$$

SSr = Squared sum error of regression line

SSm = Squared sum error of mean line

EVALUATION METRICS FOR REGRESSION MODEL – R Squared

- Now, how will you interpret the R2 score? suppose If the R2 score is zero then the above regression line by mean line is equal means 1 so 1-1 is zero.
- So, in this case, both lines are overlapping means model performance is worst, It is not capable to take advantage of the output column.
- Now the second case is when the R2 score is 1, it means when the division term is zero and it will happen when the regression line does not make any mistake, it is perfect. In the real world, it is not possible.
- So we can conclude that as our regression line moves towards perfection, R2 score move towards one. And the model performance improves.
- The normal case is when the R2 score is between zero and one like 0.8 which means your model is capable to explain 80 per cent of the variance of data.

EVALUATION METRICS FOR REGRESSION MODEL – Adjusted R Squared

- The disadvantage of the R2 score is while adding new features in data the R2 score starts increasing or remains constant but it never decreases because It assumes that while adding more data variance of data increases.
- But the problem is when we add an irrelevant feature in the dataset then at that time R2 sometimes starts increasing which is incorrect.
- Hence, To control this situation Adjusted R Squared came into existence.

$$R_a^2 = 1 - \left[\left(\frac{n-1}{n-k-1}\right) \times (1 - R^2)\right]$$

where:

n = number of observations

k = number of independent variables

R_a^2 = adjusted R^2

EVALUATION METRICS FOR REGRESSION MODEL – Adjusted R Squared

- Now as K increases by adding some features so the denominator will decrease, n-1 will remain constant.
- R2 score will remain constant or will increase slightly so the complete answer will increase and when we subtract this from one then the resultant score will decrease.
- So, this is the case when we add an irrelevant feature in the dataset.
- And if we add a relevant feature then the R2 score will increase and 1-R2 will decrease heavily and the denominator will also decrease so the complete term decreases, and on subtracting from one the score increases.

THANK YOU

PRESIDENCY UNIVERSITY

BENGALURU

MODULE 2

SUPERVISED MACHINE LEARNING ALGORITHMS
PART 2 – CLASSIFICATION MODELS

AGENDA

- Classification models
- Decision Tree algorithms using Entropy and Gini Index as measures of node impurity,
- Model evaluation metrics for classification algorithms,
- Cohen's Kappa Statistic,
- Multi-class classification
- Class Imbalance problem.
- Naïve Bayes Classifiers
- Naive Bayes model for sentiment classification – An Introduction

CLASSIFICATION MODELS

WHAT IS CLASSIFICATION ALGORITHM?

- Classification algorithm is a Supervised Learning technique in which a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups. Such as, **Yes or No, 0 or 1, Spam or Not Spam, cat or dog,** etc.
- Classes can be called as targets/labels or categories.
- Unlike regression, the output variable of Classification is a category, not a value, such as "Green or Blue", "fruit or animal".
- Since Classification algorithm is a Supervised learning technique, it takes labeled input data, which means it contains input with the corresponding output.
- In classification algorithm, a discrete output function(y) is mapped to input variable(x), i.e., y=f(x), where y = categorical output

TYPES OF CLASSIFICATIONS

- The algorithm which implements the classification on a dataset is known as a **classifier**.
- There are two types of Classifications:
  - **Binary Classifier:** If the classification problem has only two possible outcomes, then it is called as Binary Classifier. **Examples:** YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, etc.
  - **Multi-class Classifier:** If a classification problem has more than two outcomes, then it is called as Multi-class Classifier. **Example:** Classifications of types of crops, Classification of types of music.

LEARNERS IN CLASSIFICATION PROBLEMS

1. Lazy Learners

- Stores the training dataset and wait until it receives the test dataset.
- In this case, classification is done on the basis of the most related data stored in the training dataset.
- It takes less time in training but more time for predictions.
- **Example:** K-NN algorithm, Case-based reasoning

2. Eager Learners

- Eager Learners develop a classification model based on a training dataset before receiving a test dataset.
- Unlike Lazy learners, Eager Learner takes more time in learning, and less time in prediction.
- **Example:** Decision Trees, Naïve Bayes, ANN.

TYPES OF CLASSIFICATION ALGORITHMS

- **Linear Models**
  - Logistic Regression
  - Support Vector Machines

- **Non-linear Models**
  - K-Nearest Neighbours
  - Kernel SVM
  - Naïve Bayes
  - Decision Tree Classification
  - Random Forest Classification

METHODS FOR EVALUATING A CLASSIFICATION MODEL

Log Loss or Cross-Entropy Loss

- It is used for evaluating the performance of a classifier, whose output is a probability value between the 0 and 1.
- For a good binary Classification model, the value of log loss should be near to 0.
- The value of log loss increases if the predicted value deviates from the actual value.
- The lower log loss represents the higher accuracy of the model.
- For Binary classification, cross-entropy can be calculated as

$$-(y\log(p)+(1-y)\log(1-p))$$

- where y= Actual output, p= predicted output.

METHODS FOR EVALUATING A CLASSIFICATION MODEL

Confusion Matrix

- The confusion matrix provides us a matrix/table as output and describes the performance of the model.
- It is also known as the error matrix.
- The matrix consists of predictions result in a summarized form, which has a total number of correct predictions and incorrect predictions. The matrix looks like as below table:

	Actual Positive	Actual Negative
Predicted Positive	True Positive	False Positive
Predicted Negative	False Negative	True Negative

$$\text{Accuracy} = \frac{TP+TN}{\text{Total Population}}$$

METHODS FOR EVALUATING A CLASSIFICATION MODEL

AUC-ROC curve

- ROC curve stands for **Receiver Operating Characteristics Curve** and AUC stands for **Area Under the Curve**.
- It is a graph that shows the performance of the classification model at different thresholds.
- To visualize the performance of the multi-class classification model, we use the AUC-ROC Curve.
- The ROC curve is plotted with TPR and FPR, where TPR (True Positive Rate) on Y-axis and FPR(False Positive Rate) on X-axis.

USES CASES OF CLASSIFICATION ALGORITHMS

- Email Spam Detection
- Speech Recognition
- Identifications of Cancer tumor cells.
- Drugs Classification
- Biometric Identification, etc.

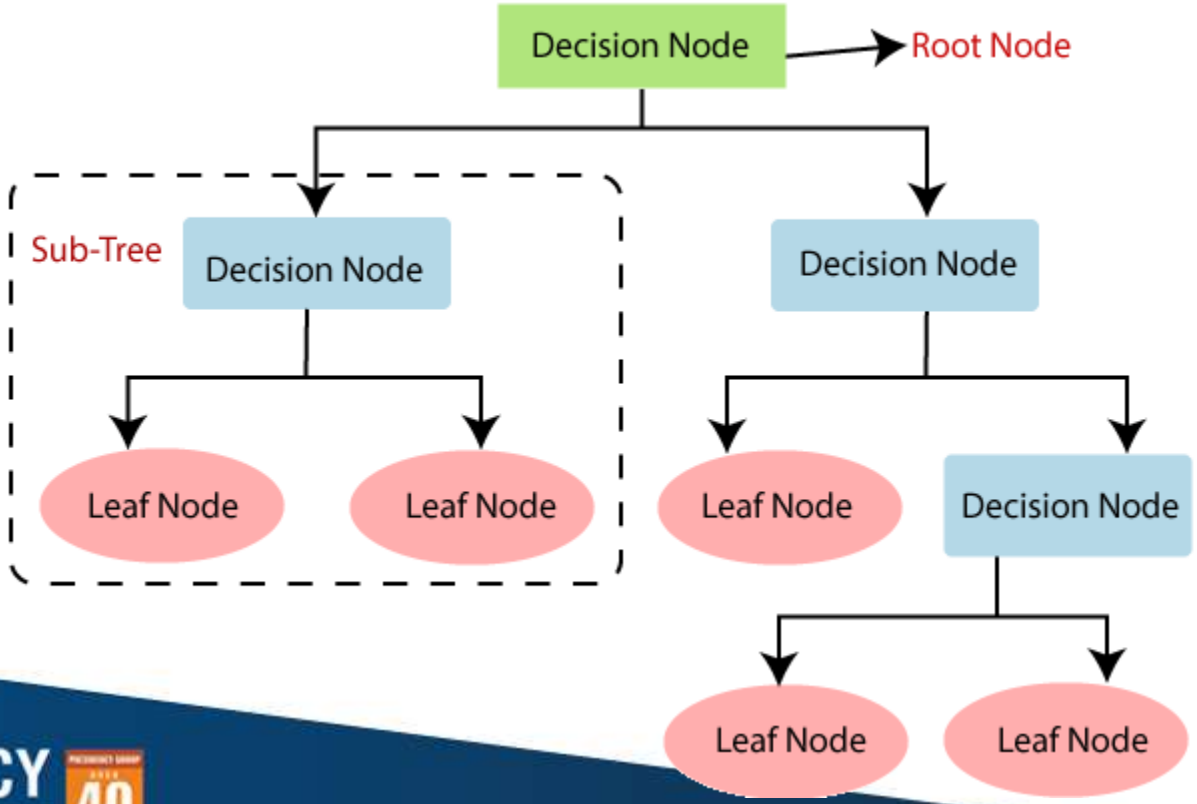
DECISION TREE ALGORITHMS USING ENTROPY AND GINI INDEX

What is Decision Tree?

- A **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems.
- Contains two nodes: **Decision Node** and **Leaf Node.**
- Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
- The decisions or the test are performed on the basis of features of the given dataset.
- **It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.**

CONTD...

- In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees



SIGNIFICANCE OF DECISION TREE

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

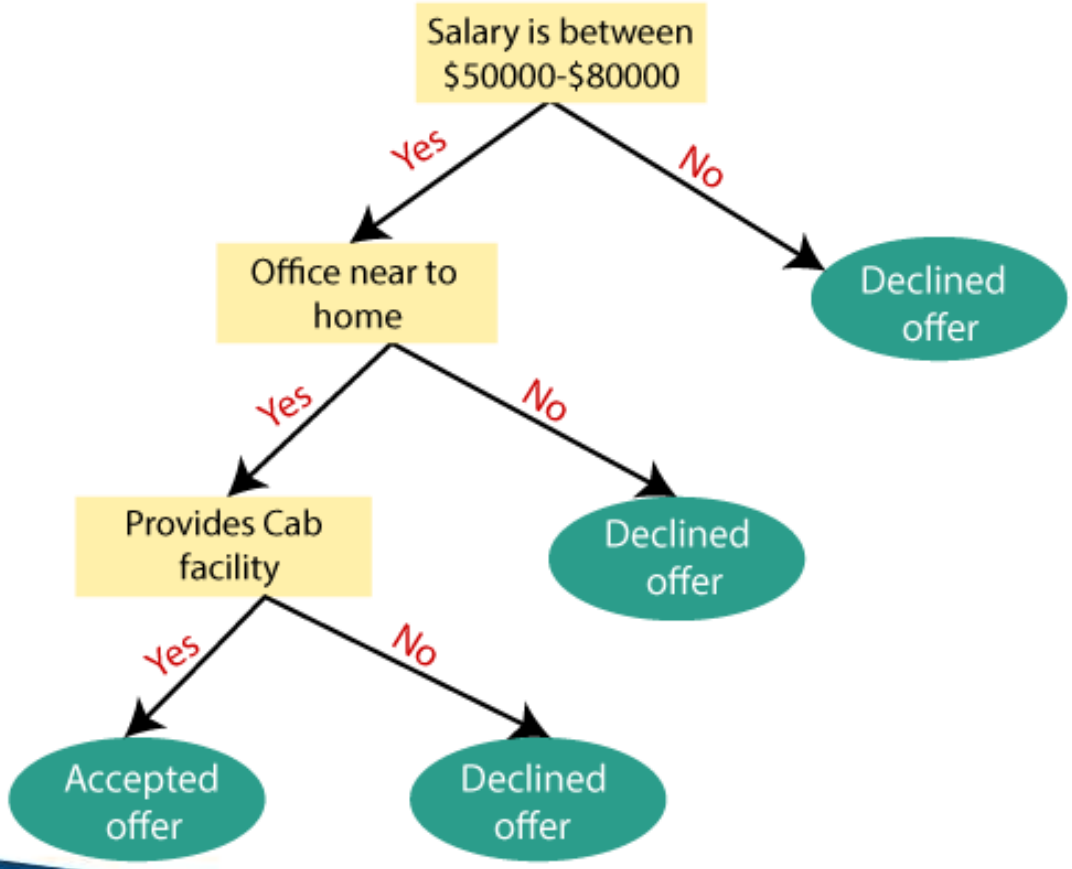
DECISION TREE TERMINOLOGIES

- **Root Node:** Node from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent & Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

DECISION TREE ALGORITHM WORKING

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

ILLUSTRATIVE EXAMPLE



ATTRIBUTE SELECTION MEASURES: ENTROPY

Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

$$\text{Entropy}(s) = -P(\text{yes}) \log_2 P(\text{yes}) - P(\text{no}) \log_2 (\text{no})$$

Where,

- **S= Total number of samples**
- **P(yes)= probability of yes**
- **P(no)= probability of no**

ATTRIBUTE SELECTION MEASURES: INFORMATION GAINS

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)]

ATTRIBUTE SELECTION MEASURES: GINI INDEX

•Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.

•An attribute with the low Gini index should be preferred as compared to the high Gini index.

•It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.

•Gini index can be calculated using the below formula:

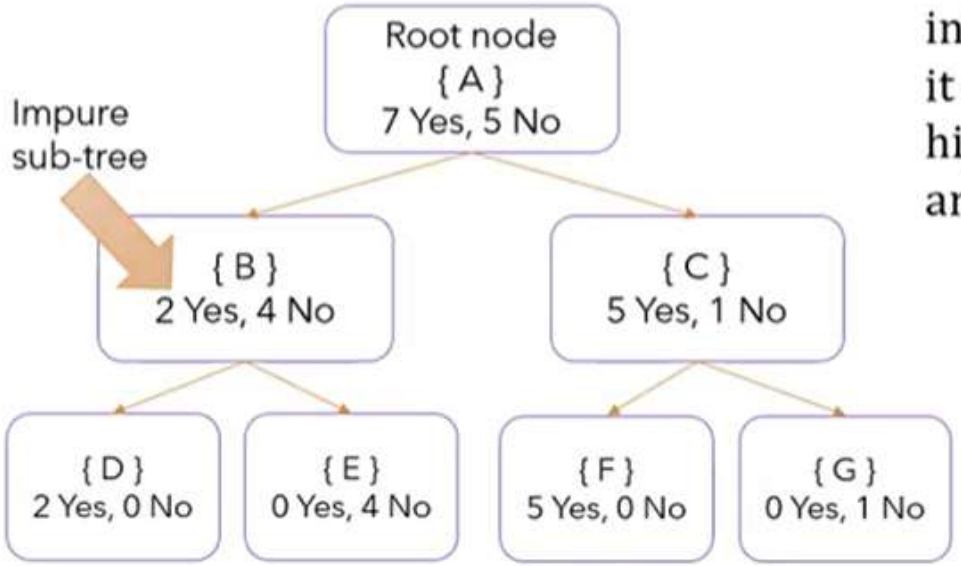
$$\text{Gini Index} = 1 - \sum_j P_j^2$$

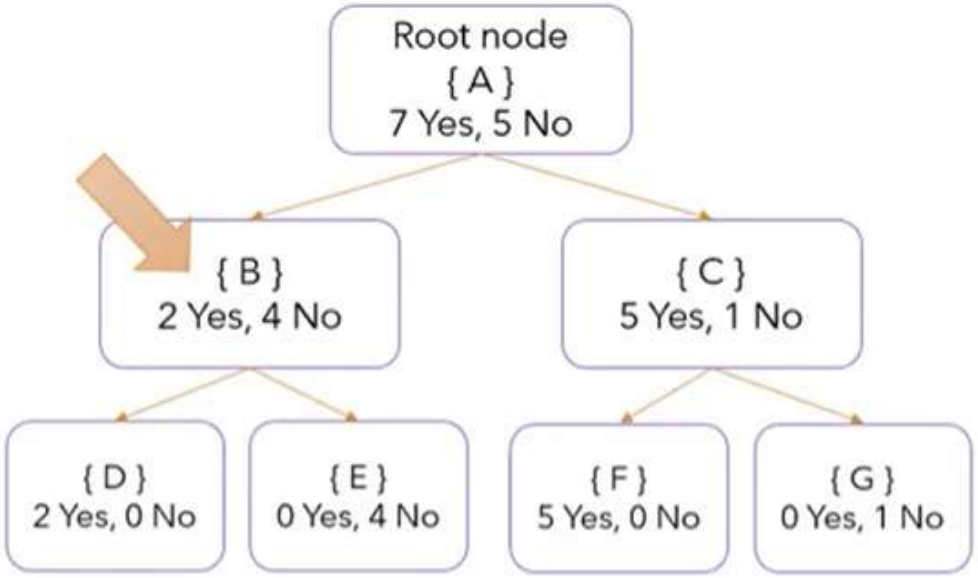
NUMERICAL EXAMPLE – DECISION TREE (ENTROPY, GINI IMPURITY & INFORMATION GAIN)

Input : Feature Attribute				Output
A	B	C	D	O
-	-	-	-	Yes
-	-	-	-	Yes
-	-	-	-	No
-	-	-	-	Yes

Entropy

It measures the randomness in the information being processed or we can say it measures the purity of the split. The higher the **Entropy**, the harder it is to draw any conclusions from that information.





Entropy

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

$p(Yes) = \frac{P(Yes)}{P(Yes)+P(No)}$, for "Yes" component of node A

$$Entropy(B) = -\frac{2}{2+4} \log_2 \left(\frac{2}{2+4}\right) - \frac{4}{4+2} \log_2 \left(\frac{4}{4+2}\right)$$

$$= -\frac{2}{6} \log_2 \left(\frac{2}{6}\right) - \frac{4}{6} \log_2 \left(\frac{4}{6}\right)$$

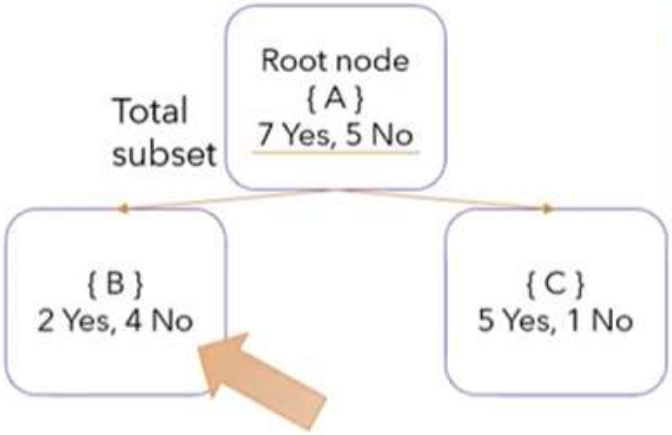
$$= 0.92 \; bits$$

$$Entropy(C) = -\frac{5}{6} \log_2 \left(\frac{5}{6}\right) - \frac{1}{6} \log_2 \left(\frac{1}{6}\right)$$

$$= 0.65 \text{ bits}$$

Entropy ranges from 0 to 1, where 0 consider as pure sub − tree & 1 as most impure sub − tree

Information Gain



$$Entropy(S) = -\frac{7}{12}\ log_2\left(\frac{7}{12}\right) - \frac{5}{12}\ log_2\left(\frac{5}{12}\right)$$

= 0.97 bits

$Entropy(B) = 0.92$ bits

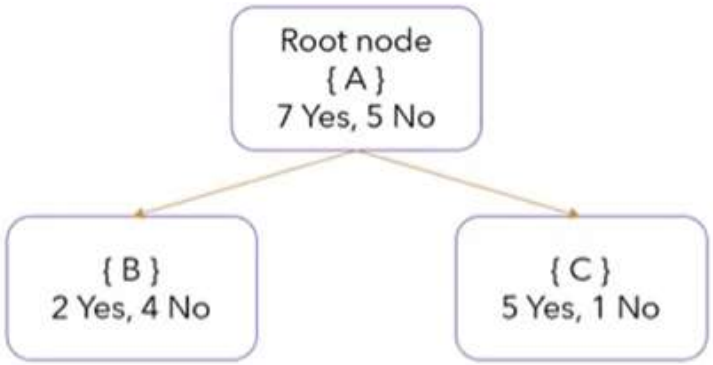
$Entropy(C) = 0.65$ bits

Information gain calculates the reduction in entropy or surprise from transforming a dataset in some way. Specifically, this metrics measure the **quality of a split.** Information gain can also be used for feature selection, by evaluating the gain of each variable in the context of the target variable

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|Sv|}{|S|} Entropy(S_v)$$

$$Gain(S, A) = 0.97 - \frac{|Sv|}{|S|}(0.92) - \frac{|Sv|}{|S|}(0.65)$$

$$Gain(S, A) = 0.97 - \frac{6}{12}(0.92) - \frac{6}{12}(0.65) \quad = 0.185$$



$Entropy(C) = 0.65$ bits

Gini Impurity

Gini Impurity is a measurement of the likelihood of an incorrect classification of a new instance of a random variable, if that new instance were randomly classified according to the distribution of class labels from the data set.

$$GI = 1 - [(\frac{5}{5+1})^2 + (\frac{1}{1+5})^2]$$

$$= 1 - [(\frac{5}{6})^2 + (\frac{1}{6})^2]$$

$$= 0.3$$

$$GI = 1 - \sum_{i=1}^{n} p^2 \Rightarrow GI = 1 - [p(Yes)^2 + p(No)^2]$$

Entropy ranges from 0 to 1 whereas Gini Impurity ranges from 0 to 0.5

COMPUTATION TIME IS REDUCED AS DON'T USE LOGARITHMIC FUNCTION IN GINI IMPURITY

ADVANTAGES & DISADVANTAGES OF THE DECISION TREE

Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems and to generate possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.
- It has overfitting issue - resolved using the Random Forest algorithm.
- For more class labels, the computational complexity of the decision tree may increase.

COHEN'S KAPPA STATISTIC

What is Cohen's Kappa Statistic?

- Cohen's kappa statistic measures interrater reliability (sometimes called interobserver agreement).
- Interrater reliability, or precision, happens when your data raters (or collectors) give the same score to the same data item.
- This statistic should only be calculated when:
  - Two raters each rate one trial on each sample. or
  - One rater rates two trials on each sample.
- Historically, percent agreement (number of agreement scores / total scores) was used to determine interrater reliability.
- However, chance agreement due to raters guessing is always a possibility — in the same way that a chance "correct" answer is possible on a multiple choice test.
- The Kappa statistic takes into account this element of chance.

CONTD...

- The Kappa statistic varies from 0 to 1, where.

  0 = agreement equivalent to chance.

  0.1 – 0.20 = slight agreement.

  0.21 – 0.40 = fair agreement.

  0.41 – 0.60 = moderate agreement.

  0.61 – 0.80 = substantial agreement.

  0.81 – 0.99 = near perfect agreement

  1 = perfect agreement.

- The formula to calculate Cohen's kappa for two raters is: $\kappa = \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e},$
- where:

  P_o = the relative observed agreement among raters.
  P_e = the hypothetical probability of chance agreement

NUMERICAL EXAMPLE

Example Question: The following hypothetical data comes from a medical test where two radiographers rated 50 images for needing further study. The researchers (A and B) either said Yes (for further study) or No (No further study needed). 20 images were rated Yes by both, 15 images were rated No by both. Overall, rater A said Yes to 25 images and No to 25, Overall, Rater B said Yes to 30 images and No to 20. Calculate Cohen's kappa for this data set.

Solution

- **Step 1**: Calculate p_o (the observed proportional agreement):
  20 images were rated Yes by both.
  15 images were rated No by both.
  So, P_o = number in agreement / total = (20 + 15) / 50 = 0.70.
- **Step 2**: Find the probability that the raters would randomly both say Yes.
  Rater A said Yes to 25/50 images, or 50%(0.5).
  Rater B said Yes to 30/50 images, or 60%(0.6).
  The total probability of the raters both saying Yes randomly is: 0.5 * 0.6 = 0.30.

NUMERICAL EXAMPLE

- **Step 3**: Calculate the probability that the raters would randomly both say No.
  Rater A said No to 25/50 images, or 50%(0.5).
  Rater B said No to 20/50 images, or 40%(0.6).
  The total probability of the raters both saying No randomly is:
  $0.5 * 0.4 = 0.20$.
- **Step 4: Calculate P_e** : Add your answers from Step 2 and 3 to get the overall probability that the raters would randomly agree.
  $P_e = 0.30 + 0.20 = 0.50$.
- **Step 5:** Insert your calculations into the formula and solve:

  $k = (P_o - p_e) / (1 - p_e = (0.70 - 0.50) / (1 - 0.50) = 0.40$.

  $k = 0.40$, which indicates fair agreement.

MULTI-CLASS CLASSIFICATION

What is Multi-Class Classification?

- In machine learning and statistical classification, **multiclass classification** or **multinomial classification** is the problem of classifying instances into one of three or more classes (classifying instances into one of two classes is called binary classification).
- While many classification algorithms (notably multinomial logistic regression) naturally permit the use of more than two classes, some are by nature binary algorithms; these can, however, be turned into multinomial classifiers by a variety of strategies.
- Multiclass classification should not be confused with multi-label classification, where multiple labels are to be predicted for each instance.
- The existing multi-class classification techniques can be categorized into **(i) transformation to binary (ii) extension from binary and (iii) hierarchical classification**

MULTI-CLASS CLASSIFICATION TECHNIQUE: Transformation to Binary

- Also called as problem transformation techniques, discusses strategies for reducing the problem of multiclass classification to multiple binary classification problems.
- Categorized into *one vs rest* and *one vs one*.

1. **One vs rest**
   - ➢ This strategy involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives.
   - ➢ The base classifiers produce a real-valued confidence score for its decision, rather than just a class label; discrete class labels alone can lead to ambiguities, where multiple classes are predicted for a single sample.
   - ➢ Disadvantages:
     - ✓ Scale of the confidence values may differ between the binary classifiers.
     - ✓ Even if the class distribution is balanced in the training set, the binary classification learners see unbalanced distributions because typically the set of negatives they see is much larger than the set of positives

MULTI-CLASS CLASSIFICATION TECHNIQUE: Transformation to Binary

2. **one - vs. - one**

- Trains $K\,(K-1)\,/\,2$ binary classifiers for a K-way multiclass problem; each receives the samples of a pair of classes from the original training set, and must learn to distinguish these two classes.
- At prediction time, a voting scheme is applied: all $K\,(K-1)\,/\,2$ classifiers are applied to an unseen sample and the class that got the highest number of "+1" predictions gets predicted by the combined classifier.
- Disadvantage: Like One-vs-Rest, One-vs-One suffers from ambiguities in some regions of its input space which may receive the same number of votes.

MULTI-CLASS CLASSIFICATION TECHNIQUE: Extension from Binary

- Strategies of extending the existing binary classifiers to solve multi-class classification problems - neural networks, decision trees, k-nearest neighbors & Naive Bayes.

1. **Neural Network**
   - Instead of just having one neuron in the output layer, with binary output, one could have N binary neurons leading to multi-class classification.
   - In practice, the last layer of a neural network is usually a softmax function layer, which is the algebraic simplification of N logistic classifiers, normalized per class by the sum of the N-1 other logistic classifiers.
2. **Decision Trees**
   - Tries to infer a split of the training data based on the values of the available features to produce a good generalization.
   - The algorithm can naturally handle binary or multiclass classification problems. The leaf nodes can refer to any of the K classes concerned.

MULTI-CLASS CLASSIFICATION TECHNIQUE: Extension from Binary

3. K-Nearest Neighbors

- One among the oldest non-parametric classification algorithms.
- To classify an unknown example, the distance from that example to every other training example is measured.
- The k smallest distances are identified, and the most represented class by these k nearest neighbours is considered the output class label.

4. Naive Bayes

- Is a successful classifier based upon the principle of maximum a posteriori (MAP).
- This approach is naturally extensible to the case of having more than two classes, and was shown to perform well in spite of the underlying simplifying assumption of conditional independence.

MULTI-CLASS CLASSIFICATION TECHNIQUE: Hierarchical Classification

- Hierarchical classification tackles the multi-class classification problem by dividing the output space i.e. into a tree.
- Each parent node is divided into multiple child nodes and the process is continued until each child node represents only one class.
- Several methods have been proposed based on hierarchical classification.

CLASS IMBALANCE PROBLEM

What is the class imbalance problem?

- It is the problem in machine learning where the total number of a class of data (positive) is far less than the total number of another class of data (negative).
- This problem is extremely common in practice and can be observed in various disciplines including fraud detection, anomaly detection, medical diagnosis, oil spillage detection, facial recognition, etc.

WHY IS IT A PROBLEM?

- Most machine learning algorithms works best when the number of instances of each classes are roughly equal. When the number of instances of one class far exceeds the other, problems arise.
- Example: Given a dataset of transaction data, we would like to find out which are fraudulent and which are genuine ones. For a dataset consisting of 10000 genuine and 10 fraudulent transactions and the classifier classifies fraudulent transactions as genuine transactions.
- The reason for this can be easily explained by the numbers.
- Suppose the machine learning algorithm has two possibly outputs as follows:
    1. Model 1 classified 7 out of 10 fraudulent transactions as genuine transactions and 10 out of 10000 genuine transactions as fraudulent transactions.
    2. Model 2 classified 2 out of 10 fraudulent transactions as genuine transactions and 100 out of 10000 genuine transactions as fraudulent transactions.

Why is it a problem?

- If the classifier's performance is determined by the number of mistakes, then clearly Model 1 is better as it makes only a total of 17 mistakes while Model 2 made 102 mistakes.
- However, as we want to minimize the number of fraudulent transactions happening, we should pick Model 2 instead, which only made 2 mistakes classifying the fraudulent transactions.
- But, this could come at the expense of more genuine transactions being classified as fraudulent transactions, and a general machine learning algorithm will just pick Model 1 instead of Model 2, which is a problem.
- In practice, this means we will let a lot of fraudulent transactions go through although we could have stopped them by using Model 2.
- This translates to unhappy customers and money lost for the company.

How to tell a ML algorithm which is a better solution?

- To tell the machine learning algorithm (or the researcher) that Model 2 is better than Model 1, we need to show that Model 2 above is better than Model 1 above. For that, we will need better metrics than just counting the number of mistakes made.
- We introduce the concept of True Positive, True Negative, False Positive and False Negative:
  - True Positive (TP) – An example that is **positive** and is classified correctly as **positive**
  - True Negative (TN) – An example that is **negative** and is classified correctly as **negative**
  - False Positive (FP) – An example that is **negative** but is classified wrongly as **positive**
  - False Negative (FN) – An example that is **positive** but is classified wrongly as **negative**
  - Based on this above. We will have also the following of True Positive Rate, True Negative Rate, False Positive Rate, False Negative Rate:

CONTD...

Name	Formula	Explanation
True Positive Rate (TP rate)	TP / (TP + FP)	The closer to 1, the better. TP rate = 1 when FP = 0. (No false positives)
True Negative Rate (TN rate)	TN / (TN + FN)	The closer to 1, the better. TN rate = 1 when FN = 0. (No false negatives)
False Positive Rate (FP rate)	FP / (FP + TN)	The closer to 0, the better. FP rate = 0 when FP = 0. (No false positives)
False Negative Rate (FN rate)	FN / (FN + TP)	The closer to 0, the better. FN rate = 0 when FN = 0. (No false negatives)

With these new metrics, let's compare it with the conventional metrics of counting the number of mistakes made with the example above. First, we will use the old metrics to calculate the number of mistakes made (error):

CONTD...

Model 1		Predicted Class	
		+	-
Actual Class	+	3 (TP)	7 (FN)
	-	10 (FP)	9990 (TN)

Error(Model 1)
= (FN + FP) / total dataset size
= (7 + 10) / 10010
= 0.0017 = (0.1% error)

Model 2		Predicted Class	
		+	-
Actual Class	+	8 (TP)	2 (FN)
	-	100 (FP)	9900 (TN)

Error(Model 2)
= (FN + FP) / total dataset size
= (2 + 100) / 10010
= 0.01 (= 1% error)

- As illustrated above, Model 1 looks like it has lower error (0.1% error) than Model 2 (1.0% error) but we know that Model 2 is the better one, as it makes less false negatives (FN) (maximize true positive (TP)).
- Now let's see what the performance of Model 1 and Model 2 are like with the new metrics:

CONTD...

Model 1		Predicted Class	
		+	-
Actual Class	+	3 (TP)	7 (FN)
	-	10 (FP)	9990 (TN)

TP_rate(M1) = 3/(3+7) = 0.3
TN_rate(M1) = 9990/(10+9990) = 0.999
FP_rate(M1) = 10/(10+9990) = 0.001
FN_rate(M1) = 7/(7+3) = 0.7

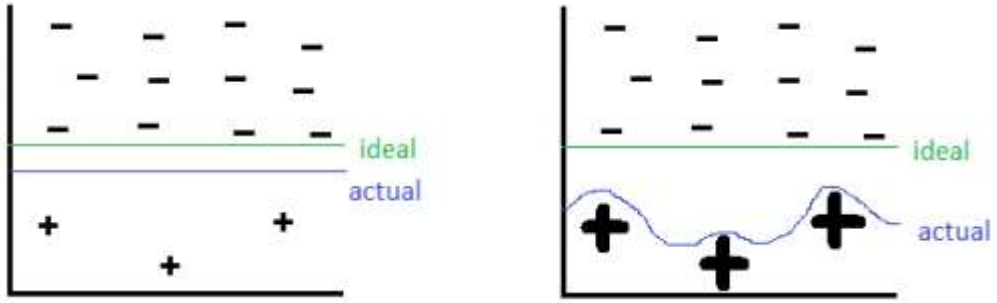
Model 2		Predicted Class	
		+	-
Actual Class	+	8 (TP)	2 (FN)
	-	100 (FP)	9900 (TN)

TP_rate(M2) = 8/(8+2) = 0.8
TN_rate(M2) = 9900/(100+9900) = 0.990
FP_rate(M2) = 100/(100+9900) = 0.01
FN_rate(M2) = 2/(2+8) = 0.2

- Now, we can see that the false negative rate of Model 1 is at 70% while the false negative rate of Model 2 is just at 20%, which is clearly a better classifier.
- This is what we should educate the machine learning algorithm (or us) to use in order to allow it to pick a better algorithm.

SAMPLING BASED APPROACH TO MITIGATE CLASS IMBALANCE PROBLEM

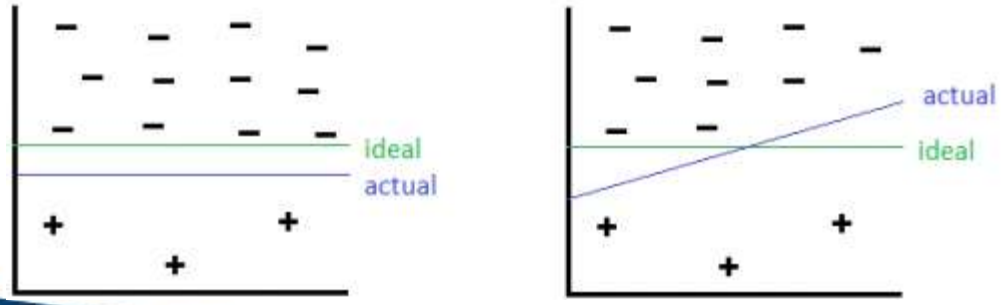
- This can be roughly classified into three categories:
  - Oversampling
  - Undersampling
  - Hybrid, a mix of oversampling and Undersampling
- **Oversampling**
  - By oversampling, just duplicating the minority classes could lead the classifier to overfitting to a few examples, which can be illustrated below:



CONTD...

- On the left hand side is before oversampling, where as on the right hand side is oversampling has been applied. On the right side, The thick positive signs indicate there are multiple repeated copies of that data instance.
- The machine learning algorithm then sees these cases many times and thus designs to overfit to these examples specifically, resulting in a blue line boundary as above.

- **Undersampling**
  - By Undersampling, we could risk removing some of the majority class instances which is more representative, thus discarding useful information. This can be illustrated as follows:



CONTD...

- Here the green line is the ideal decision boundary we would like to have, and blue is the actual result.
- On the left side is the result of just applying a general machine learning algorithm without using Undersampling.
- On the right, we undersampled the negative class but removed some informative negative class, and caused the blue decision boundary to be slanted, causing some negative class to be classified as positive class wrongly.

- **Hybrid approach**
  - By combining Undersampling and oversampling approaches, we get the advantages but also drawbacks of both approaches as illustrated above, which is still a tradeoff.

NAÏVE BAYES CLASSIFIERS

Naïve Bayes algorithm?

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
- The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:
  - Naïve: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
  - Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem.

BAYES' THEOREM

• Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

• The formula for Bayes' theorem is given as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Working of Naïve Bayes' Classifier

- Working of Naïve Bayes' Classifier can be understood with the help of the below example:
- Suppose we have a dataset of weather conditions and corresponding target variable "Play". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions.
- So to solve this problem, we need to follow the below steps:
    1. Convert the given dataset into frequency tables.
    2. Generate Likelihood table by finding the probabilities of given features.
    3. Now, use Bayes theorem to calculate the posterior probability.

CONTD...

Problem: If the weather is sunny, then the Player should play or not?

Solution: To solve this, first consider the given dataset

	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

CONTD...

Frequency table for the Weather Conditions:

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	5

CONTD...

Likelihood table weather condition:

Weather	No	Yes	
Overcast	0	5	5/14= 0.35
Rainy	2	2	4/14=0.29
Sunny	2	3	5/14=0.35
All	4/14=0.29	10/14=0.71	

APPLYING BAYES' THEOREM

- **P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)/P(Sunny)**

P(Sunny|Yes)= 3/10= 0.3

P(Sunny)= 0.35, P(Yes)=0.71

So, P(Yes|Sunny) = 0.3*0.71/0.35= **0.60**

- **P(No|Sunny)= P(Sunny|No)*P(No)/P(Sunny)**

P(Sunny|NO)= 2/4=0.5

P(No)= 0.29, P(Sunny)= 0.35

So P(No|Sunny)= 0.5*0.29/0.35 = **0.41**

As we can see from the above calculation that **P(Yes|Sunny)>P(No|Sunny)**

Hence on a Sunny day, Player can play the game.

ADVANTAGES, DISADVANTAGES AND APPLICATION

- **Advantages**
  - It can be used for Binary as well as Multi-class Classifications and performs well in Multi-class predictions as compared to the other Algorithms.
  - It is widely used for text classification problems.
- **Disadvantages**: Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features.

- **Application**
  - Credit Scoring
  - Medical data classification.
  - In real-time predictions because Naïve Bayes Classifier is an eager learner.
  - Used in Text classification such as Spam filtering and Sentiment analysis.

NAIVE BAYES MODEL FOR SENTIMENT CLASSIFICATION – AN INTRODUCTION

- In Naïve Bayes, probabilities are assigned to words or phrases, segregating them into different labels. Consider the following example:

TRUE SENTIMENT	TEXT
POSITIVE	The food was good.
POSITIVE	The food tasted really good
POSITIVE	The service was good
POSITIVE	The price was reasonable
NEGATIVE	The location was not good

- Here, the model will try to learn how these sentiments are classified using corresponding text. Example: It will see that a sentence having the word "good" has a high probability of being appositive sentiment
- Using such a probabilistic value, a total probability of a test sentiment being positive/ negative can be assigned

THANK YOU

Alternative Splitting Criteria based on INFO

- Entropy at a given node t:

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

- Measures homogeneity of a node.
  - Maximum ($\log n_c$) when records are equally distributed among all classes implying least information
  - Minimum (0.0) when all records belong to one class, implying most information

Examples for computing Entropy

$$Entropy(t) = -\sum_j p(j \mid t) \log_2 p(j \mid t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (5/6) = 0.65

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Splitting Based on INFO...

- Information Gain:

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

Parent Node, p is split into k partitions;

n_i is number of records in partition i

- Measures Reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
- Used in ID3 and C4.5
- Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

ID3 – Iterative Dichotomizer

- Invented by J. Ross Quinlan
- Employs a top-down greedy search through the space of possible decision trees.

  Greedy because there is no backtracking. It picks highest values first.
- Select attribute that is most useful for classifying examples (attribute that has the highest Information Gain).

Entropy

- Entropy measures the impurity of an arbitrary collection of examples.
- For a collection S, entropy is given as:

$$Entropy(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

- For a collection S having positive and negative examples

  Entropy(S) = $-p_+ \log_2 p_+ - p_- \log_2 p_-$

where p_+ is the proportion of positive examples

and p_- is the proportion of negative examples

In general, Entropy(S) = 0 if all members of S belong to the same class.

Entropy(S) = 1 (maximum) when all members are split equally.

Information Gain

- Measures the expected reduction in entropy. The higher the IG, more is the expected reduction in entropy.

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where Values(A) is the set of all possible values for attribute A,

S_v is the subset of S for which attribute A has value v.

Example : Determine whether an animal lays eggs using entropy

Independent/Condition attributes					Dependent/ Decision attributes
Animal	**Warm-blooded**	**Feathers**	**Fur**	**Swims**	**Lays Eggs**
Ostrich	Yes	Yes	No	No	Yes
Crocodile	No	No	No	Yes	Yes
Raven	Yes	Yes	No	No	Yes
Albatross	Yes	Yes	No	No	Yes
Dolphin	Yes	No	No	Yes	No
Koala	Yes	No	Yes	No	No

$$Entropy(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Entropy(4Y,2N): $-(4/6)\log_2(4/6) - (2/6)\log_2(2/6)$

$= 0.91829$

Now, we have to find the IG for all four attributes
Warm-blooded, Feathers, Fur, Swims

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

For attribute 'Warm-blooded':

Values(Warm-blooded) : [Yes,No]

S = [4Y,2N]

S_{Yes} = [3Y,2N] $E(S_{Yes})$ = 0.97095

S_{No} = [1Y,0N] $E(S_{No})$ = 0 (all members belong to same class)

Gain(S,Warm-blooded) = 0.91829 – [(5/6)*0.97095 + (1/6)*0]

= 0.10916

For attribute 'Feathers':

Values(Feathers) : [Yes,No]

S = [4Y,2N]

S_{Yes} = [3Y,0N] $E(S_{Yes})$ = 0

S_{No} = [1Y,2N] $E(S_{No})$ = 0.91829

Gain(S,Feathers) = 0.91829 – [(3/6)*0 + (3/6)*0.91829]

= 0.45914

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

For attribute 'Fur':

Values(Fur) : [Yes,No]

S = [4Y,2N]

S_{Yes} = [0Y,1N] $E(S_{Yes})$ = 0

S_{No} = [4Y,1N] $E(S_{No})$ = 0.7219

Gain(S,Fur) = 0.91829 – [(1/6)*0 + (5/6)*0.7219]

= 0.3167

For attribute 'Swims':

Values(Swims) : [Yes,No]

S = [4Y,2N]

S_{Yes} = [1Y,1N] $E(S_{Yes})$ = 1 (equal members in both classes)

S_{No} = [3Y,1N] $E(S_{No})$ = 0.81127

Gain(S,Swims) = 0.91829 – [(2/6)*1 + (4/6)*0.81127]

= 0.04411

Gain(S,Warm-blooded) = 0.10916

Gain(S,Feathers) = 0.45914

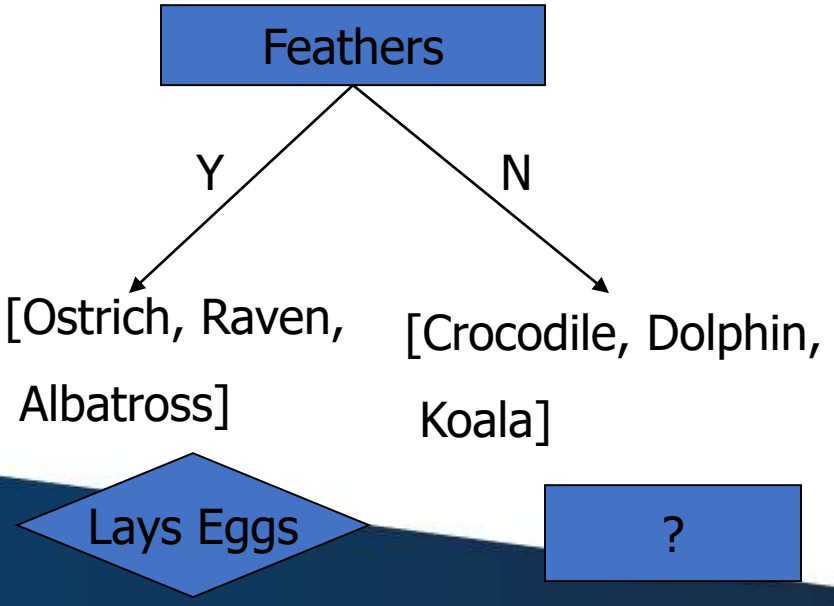
Gain(S,Fur) = 0.31670

Gain(S,Swims) = 0.04411

Gain(S,Feathers) is maximum, so it is considered as the root node

Animal	Warm-blooded	Feathers	Fur	Swims	Lays Eggs
Ostrich	Yes	Yes	No	No	Yes
Crocodile	No	No	No	Yes	Yes
Raven	Yes	Yes	No	No	Yes
Albatross	Yes	Yes	No	No	Yes
Dolphin	Yes	No	No	Yes	No
Koala	Yes	No	Yes	No	No

The 'Y' descendant has only positive examples and becomes the leaf node with classification 'Lays Eggs'



Animal	Warm-blooded	Feathers	Fur	Swims	Lays Eggs
Crocodile	No	No	No	Yes	Yes
Dolphin	Yes	No	No	Yes	No
Koala	Yes	No	Yes	No	No

We now repeat the procedure,

S: [Crocodile, Dolphin, Koala]

S: [1+,2-]

$$Entropy(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Entropy(S) = $-(1/3)\log_2(1/3) - (2/3)\log_2(2/3)$

= 0.91829

- **For attribute ‘Warm-blooded’:**

Values(Warm-blooded) : [Yes,No]

S = [1Y,2N]

S_{Yes} = [0Y,2N] $E(S_{Yes}) = 0$

S_{No} = [1Y,0N] $E(S_{No}) = 0$

Gain(S,Warm-blooded) = 0.91829 – [(2/3)*0 + (1/3)*0] = **0.91829**

- **For attribute ‘Fur’:**

Values(Fur) : [Yes,No]

S = [1Y,2N]

S_{Yes} = [0Y,1N] $E(S_{Yes}) = 0$

S_{No} = [1Y,1N] $E(S_{No}) = 1$

Gain(S,Fur) = 0.91829 – [(1/3)*0 + (2/3)*1] = **0.25162**

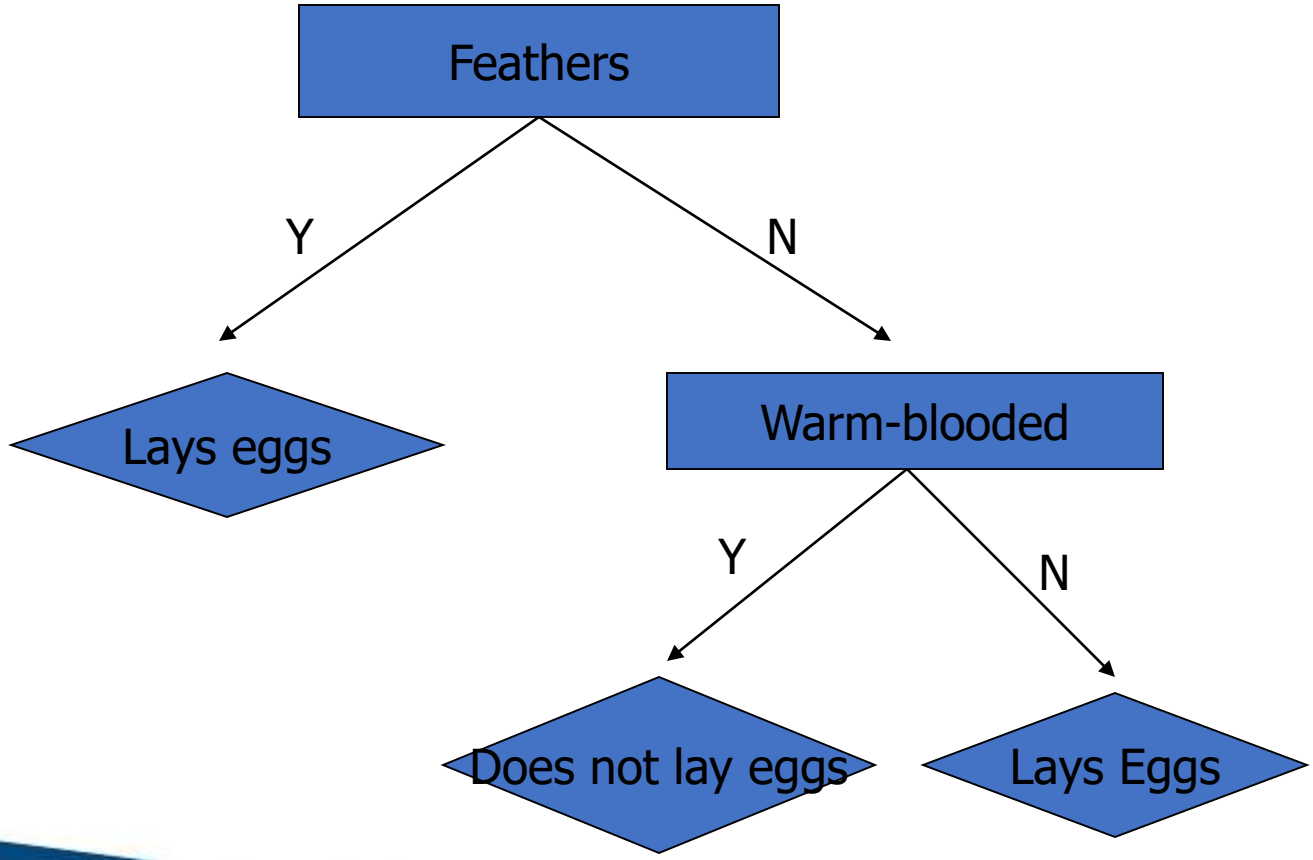
For attribute 'Swims':

Values(Swims) : [Yes,No]
S = [1Y,2N]
S_{Yes} = [1Y,1N] $E(S_{Yes}) = 1$
S_{No} = [0Y,1N] $E(S_{No}) = 0$

Gain(S,Swims) = 0.91829 – [(2/3)*1 + (1/3)*0] =
0.25162

Gain(S,Warm-blooded) is maximum

The final decision tree will be:



What is Information Gain?

Information Gain is used to determine which feature/attribute gives us the maximum information about a class. It is based on the concept of entropy, which is the degree of uncertainty, impurity or disorder. It aims to reduce the level of entropy starting from the root node to the leave nodes.

Formula for Entropy

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

'*p*', denotes the probability and E(S) denotes the entropy. Entropy is not preferred due to the 'log' function as it increases the computational complexity.

What is Gini Index?

Gini index or Gini impurity measures the degree or probability of a particular variable being wrongly classified when it is randomly chosen. But what is actually meant by 'impurity'? If all the elements belong to a single class, then it can be called pure. The degree of Gini index varies between 0 and 1, where 0 denotes that all elements belong to a certain class or if there exists only one class, and 1 denotes that the elements are randomly distributed across various classes. A Gini Index of 0.5 denotes equally distributed elements into some classes.

Formula for Gini Index

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

where p_i is the probability of an object being classified to a particular class.
While building the decision tree, we would prefer choosing the attribute/feature with the least Gini index as the root node.

Past Trend	Open Interest	Trading Volume	Return
Positive	Low	High	Up
Negative	High	Low	Down
Positive	Low	High	Up
Positive	High	High	Up
Negative	Low	High	Down
Positive	Low	Low	Down
Negative	High	High	Down
Negative	Low	High	Down
Positive	Low	Low	Down
Positive	High	High	Up

Example of Gini Index

Let's start by calculating the Gini Index for 'Past Trend'.
P(Past Trend=Positive): 6/10
P(Past Trend=Negative): 4/10
If (Past Trend = Positive & Return = Up), probability = 4/6
If (Past Trend = Positive & Return = Down), probability = 2/6
Gini index = 1 - ((4/6)^2 + (2/6)^2) = 0.45
If (Past Trend = Negative & Return = Up), probability = 0
If (Past Trend = Negative & Return = Down), probability = 4/4
Gini index = 1 - ((0)^2 + (4/4)^2) = 0
Weighted sum of the Gini Indices can be calculated as follows:
Gini Index for Past Trend = (6/10)*0.45 + (4/10)*0 = 0.27

Past Trend	Open Interest	Trading Volume	Return
Positive	Low	High	Up
Negative	High	Low	Down
Positive	Low	High	Up
Positive	High	High	Up
Negative	Low	High	Down
Positive	Low	Low	Down
Negative	High	High	Down
Negative	Low	High	Down
Positive	Low	Low	Down
Positive	High	High	Up

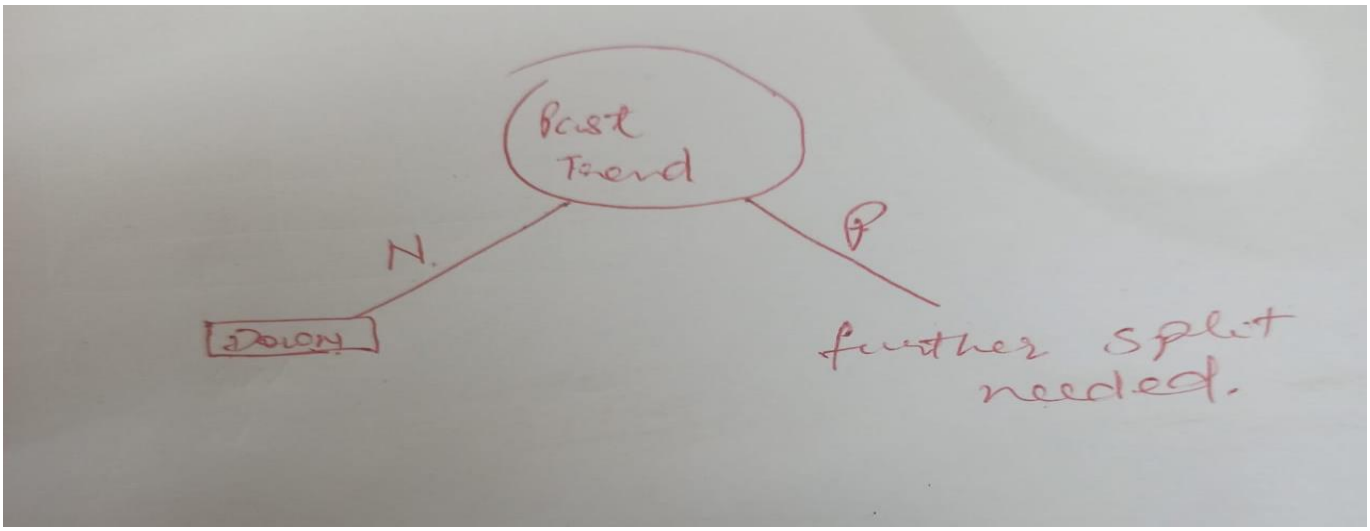
Calculation of Gini Index for Open Interest
P(Open Interest=High): 4/10
P(Open Interest=Low): 6/10
If (Open Interest = High & Return = Up), probability = 2/4
If (Open Interest = High & Return = Down), probability = 2/4
Gini index = 1 - ((2/4)^2 + (2/4)^2) = 0.5
If (Open Interest = Low & Return = Up), probability = 2/6
If (Open Interest = Low & Return = Down), probability = 4/6
Gini index = 1 - ((2/6)^2 + (4/6)^2) = 0.45
Weighted sum of the Gini Indices can be calculated as follows:
Gini Index for Open Interest = (4/10)*0.5 + (6/10)*0.45 = 0.47

Past Trend	Open Interest	Trading Volume	Return
Positive	Low	High	Up
Negative	High	Low	Down
Positive	Low	High	Up
Positive	High	High	Up
Negative	Low	High	Down
Positive	Low	Low	Down
Negative	High	High	Down
Negative	Low	High	Down
Positive	Low	Low	Down
Positive	High	High	Up

Calculation of Gini Index for Trading Volume
P(Trading Volume=High): 7/10
P(Trading Volume=Low): 3/10
If (Trading Volume = High & Return = Up), probability = 4/7
If (Trading Volume = High & Return = Down), probability = 3/7
Gini index = 1 - ((4/7)^2 + (3/7)^2) = 0.49
If (Trading Volume = Low & Return = Up), probability = 0
If (Trading Volume = Low & Return = Down), probability = 3/3
Gini index = 1 - ((0)^2 + (1)^2) = 0
Weighted sum of the Gini Indices can be calculated as follows:
Gini Index for Trading Volume = (7/10)*0.49 + (3/10)*0 = 0.34

Attributes/Features	Gini Index
Past Trend	0.27
Open Interest	0.47
Trading Volume	0.34

From the above table, we observe that ‘Past Trend’ has the lowest Gini Index and hence it will be chosen as the root node for how decision tree works.
We will repeat the same procedure to determine the sub-nodes or branches of the decision tree.



We will calculate the Gini Index for the 'Positive' branch of Past Trend as follows:

Past Trend	Open Interest	Trading Volume	Return
Positive	Low	High	Up
Positive	Low	High	Up
Positive	High	High	Up
Positive	Low	Low	Down
Positive	Low	Low	Down
Positive	High	High	Up

Calculation of Gini Index of Open Interest for Positive Past Trend

P(Open Interest=High): 2/6
P(Open Interest=Low): 4/6
If (Open Interest = High & Return = Up), probability = 2/2
If (Open Interest = High & Return = Down), probability = 0
Gini index = 1 - (sq(2/2) + sq(0)) = 0
If (Open Interest = Low & Return = Up), probability = 2/4
If (Open Interest = Low & Return = Down), probability = 2/4
Gini index = 1 - (sq(0) + sq(2/4)) = 0.50
Weighted sum of the Gini Indices can be calculated as follows:
Gini Index for Open Interest = (2/6)*0 + (4/6)*0.50 = 0.33

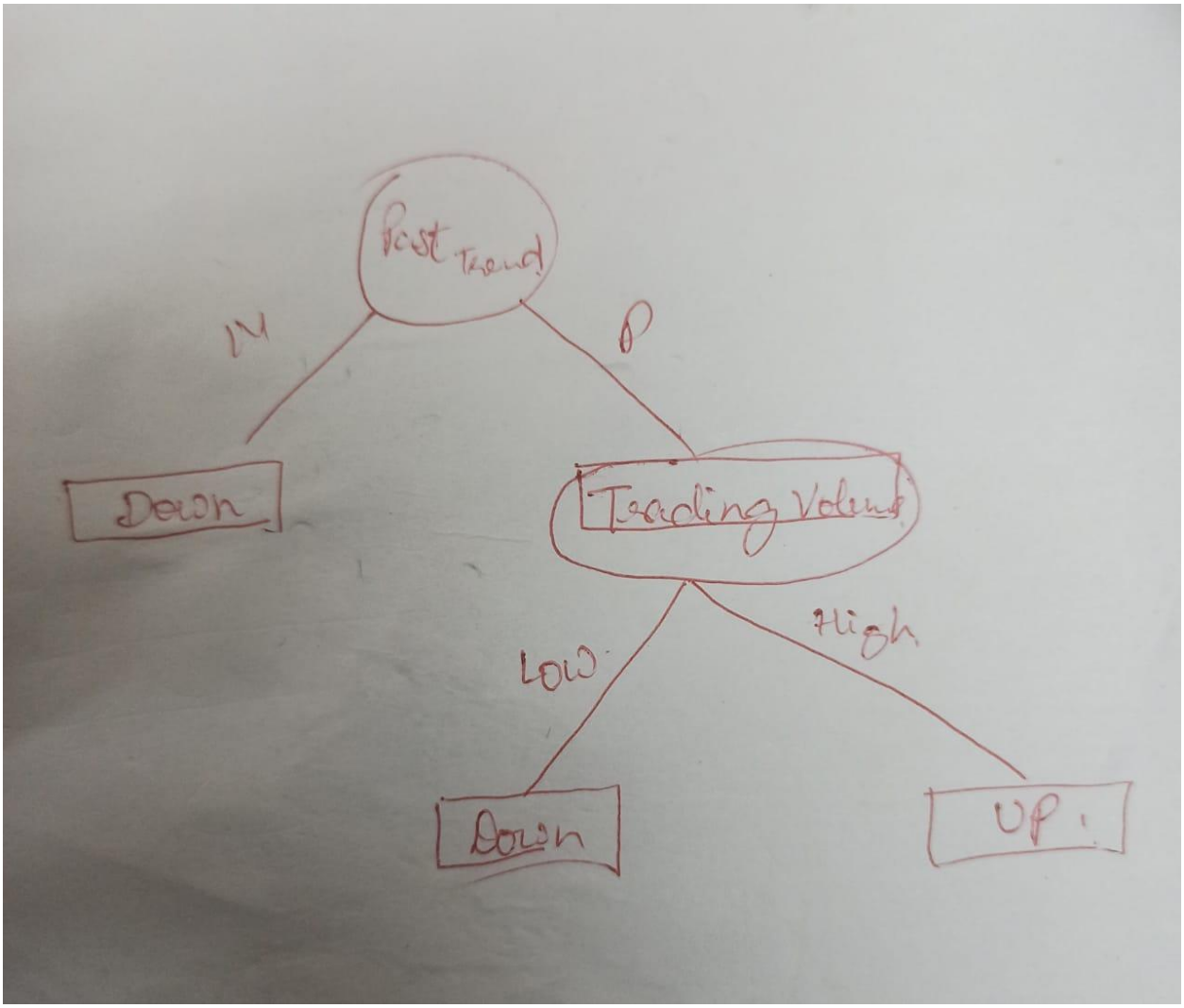
Past Trend	Open Interest	Trading Volume	Return
Positive	Low	High	Up
Positive	Low	High	Up
Positive	High	High	Up
Positive	Low	Low	Down
Positive	Low	Low	Down
Positive	High	High	Up

Calculation of Gini Index for Trading Volume

P(Trading Volume=High): 4/6
P(Trading Volume=Low): 2/6
If (Trading Volume = High & Return = Up), probability = 4/4
If (Trading Volume = High & Return = Down), probability = 0
Gini index = 1 - (sq(4/4) + sq(0)) = 0
If (Trading Volume = Low & Return = Up), probability = 0
If (Trading Volume = Low & Return = Down), probability = 2/2
Gini index = 1 - (sq(0) + sq(2/2)) = 0
Weighted sum of the Gini Indices can be calculated as follows:
Gini Index for Trading Volume = (4/6)*0 + (2/6)*0 = 0

Attribute/Feature	Gini Index
Open Interest	0.33
Trading Volume	0

We will split the node further using the ‘Trading Volume’ feature, as it has the minimum Gini index.

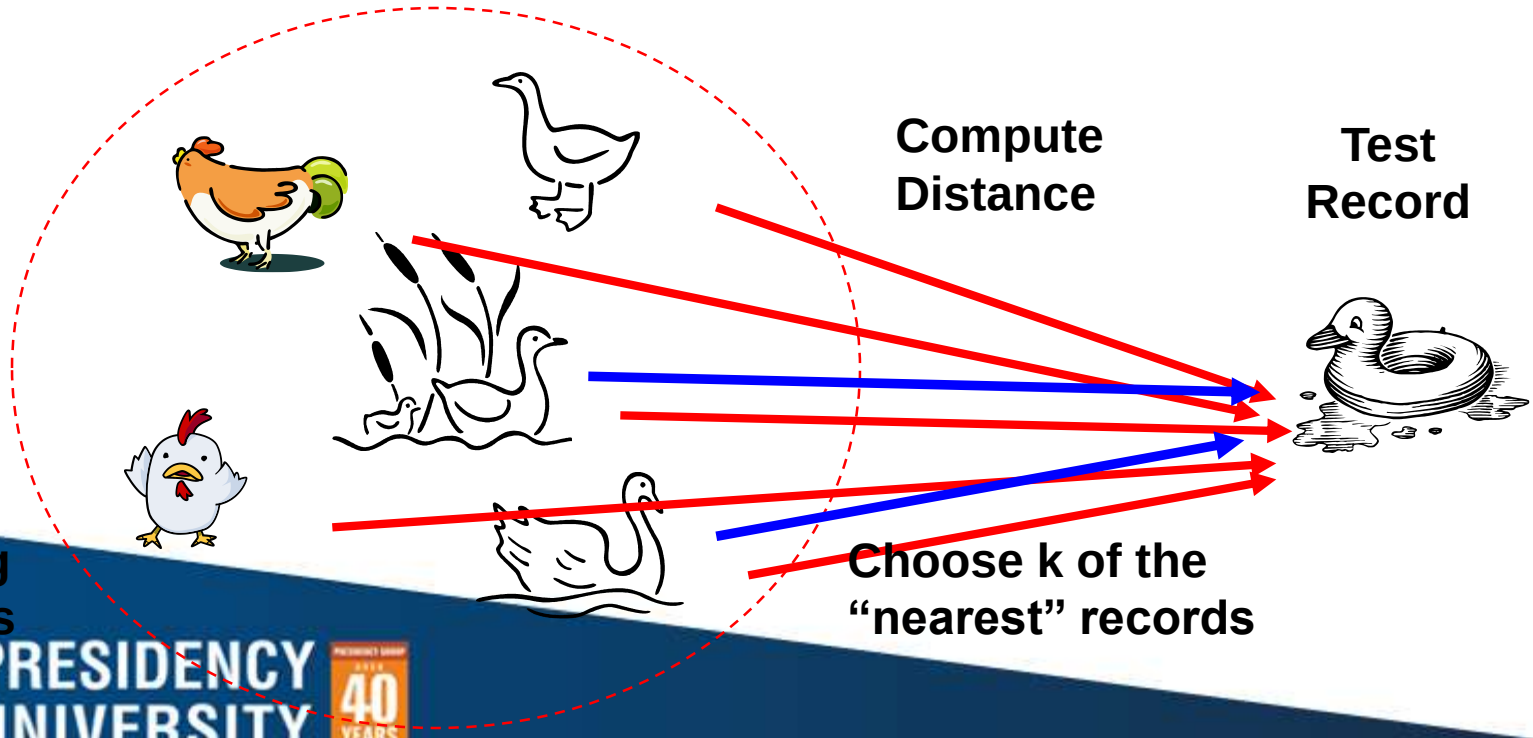

Past Trend
N
P
Down
Trading Volume
Low
High
Down
Up

Agenda

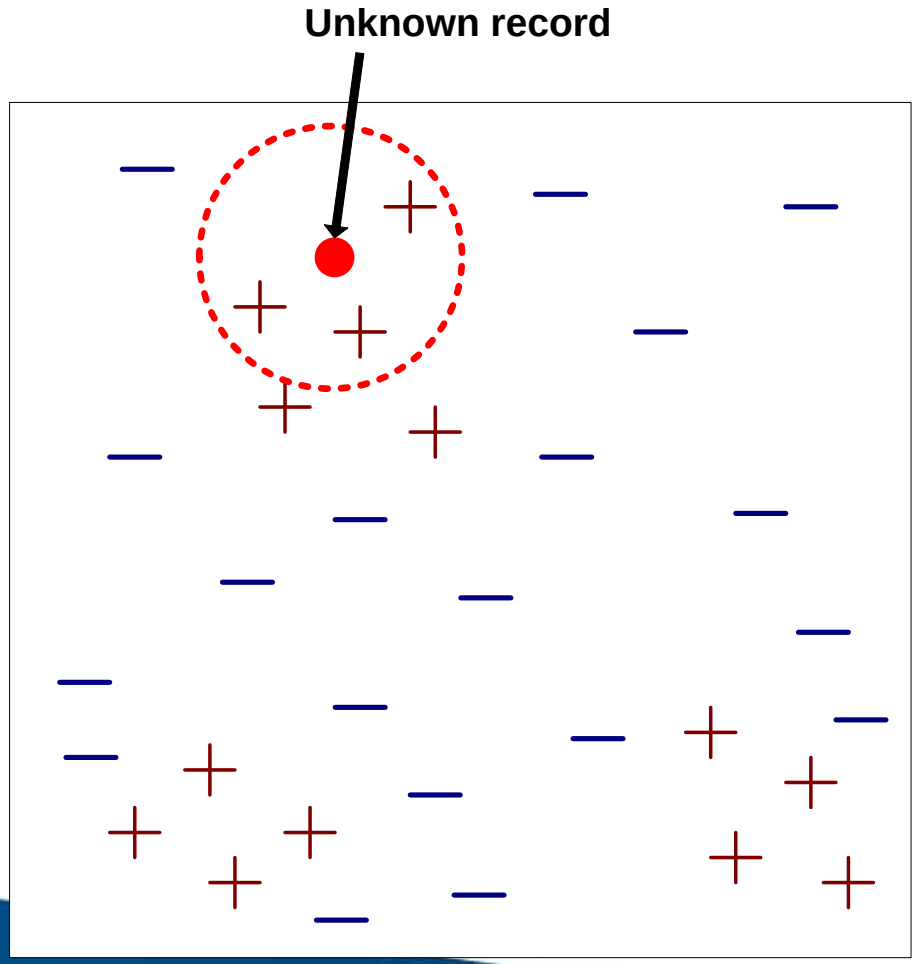
- Nearest Neighbor techniques
- Cost functions and Optimization Technique
- introduction to Gradient Descent, its applications on Linear Regression.
- Ensemble Learning algorithms – Bagging (Random Forest), Boosting(AdaBoost)

Nearest Neighbor Classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck



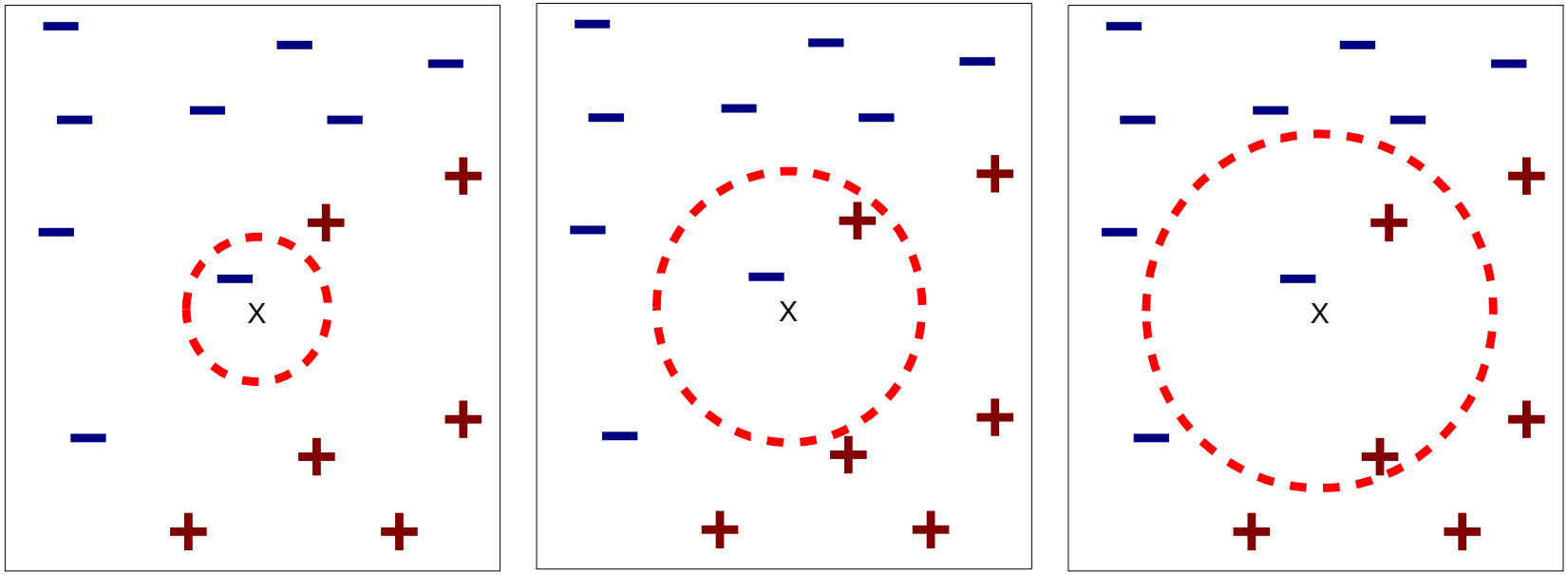
Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Definition of Nearest Neighbor



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

Nearest Neighbor Classification

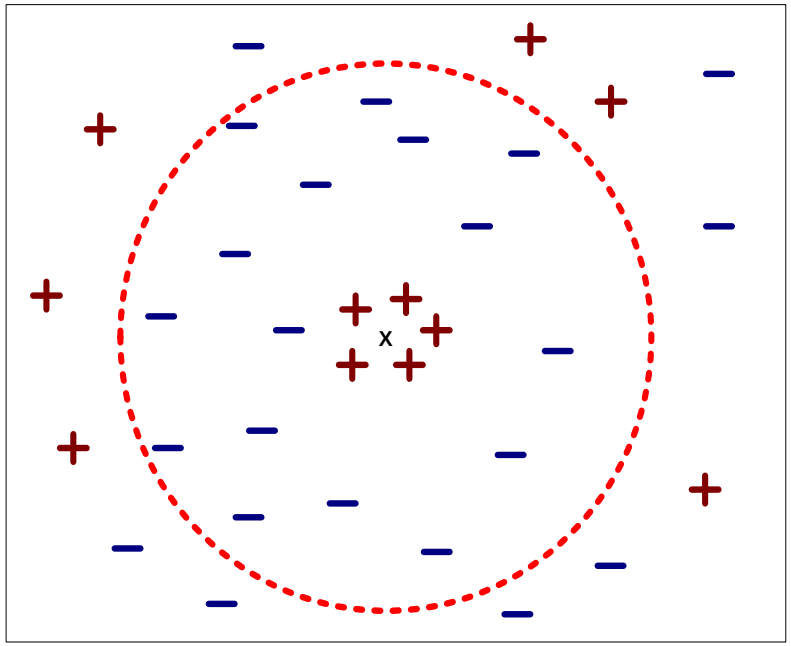
- Compute distance between two points:
  - Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, $w = 1/d^2$

Nearest Neighbor Classification…

- Choosing the value of k:
  - If k is too small, susceptible to overfitting, due to noise points in the training data.
  - If k is too large, neighborhood may include points from other classes.



Nearest Neighbor Classification…

- Scaling issues
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 90lb to 300lb
    - income of a person may vary from $10K to $1M
  - Solution: Normalize the vectors to unit length

Nearest Neighbor Classification…

1. Let *k* be the no. of nearest neighbors and ***D*** be the set of training examples.

2. *for each test example z = (x', y') do*

2.1 compute *d(x', x),* the distance between *z* and every example $(x,y) \in \boldsymbol{D}$.

2.2 Select $D_z \subseteq \boldsymbol{D}$, the set of *k* closest training examples to *z*.

2.3

$$y' = \underset{v}{\operatorname{argmax}} \sum_{(x_i, y_i) \in D_z} I(v = y_i)$$

2.4 end for

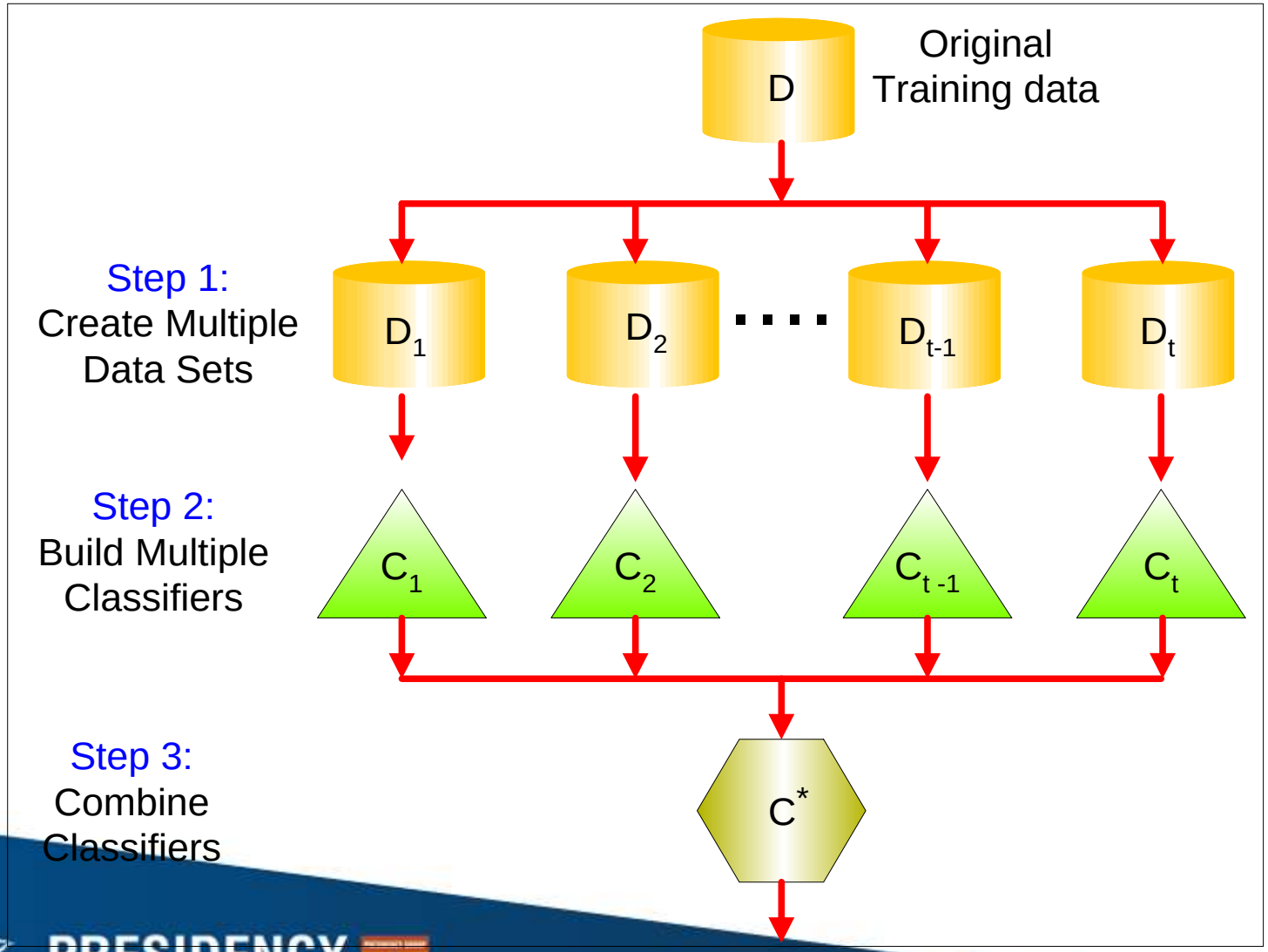
Nearest neighbor Classification…

- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Unlike eager learners such as decision tree induction and rule-based systems
  - Classifying unknown records are relatively expensive

Ensemble Methods/Learning

- Construct a set of classifiers from the training data
- Predict class label of previously unseen records by aggregating predictions made by multiple classifiers
- Improves the classification accuracy
- Predicted output of the base classifiers is combined by majority voting
- Build different experts and let them vote.

General Idea



General Procedure for ensemble methods

1. Let D …. The original training data, k… no. of base classifiers, T….. Test data
2. For i = 1 to k do
    1. Create training set D_i from D
    2. Build a base classifier C_i from D
3. End for
4. For each test record $x \in$ T do
5. $C^*(x) = vote\ (C_1\,(\text{x}),\ C_1\,(\text{x}),\ldots\ldots\ldots,\ C_k\,(\text{x})$
6. Each C_i returns its class prediction.
7. End for

Bagging

- Sampling with replacement
- Bootstrap samples $D_{i,}$ each with 63% of original data

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Build classifier on each bootstrap sample
- Each sample has probability $(1 - 1/n)^n$ of being selected in each D_i

Example of Bagging

- Refer notes for a numerical example.
- Data Set used to construct an ensemble of bagging classifiers

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all N records are assigned equal weights, 1/N
  - Unlike bagging, weights may change at the end of boosting round
  - With each boosting sample, a classifier is induced(iteratively) and is used to classify all training examples.
  - Misclassified examples are assigned more weights for the next round.

Boosting

- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

- **Example 4 is hard to classify**
- **Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds**
- **Final ensemble is an aggregate of the base classifiers got from each boosting round.**

How Boosting Works?

- Weights are assigned to each training tuple.
- A series of k classifiers is iteratively learnt
- After a classifier M_i is learnt, the weights are updated to allow the subsequent classifier M_{i+1}, to pay more attention to the training tuples that were misclassified by M_i.
- The final M^* combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy.

Basic Idea

- Suppose there are just 5 training examples {1, 2, 3, 4, 5}
- Initially each example has 0.2 (1/5) probability, of being sampled.
- If the boosting samples for the first round are {2,4,4,3,2}, a base classifier is built from this.
- Suppose 2,3,5 are correctly predicted by this classifier and 1,4 are wrongly predicted:
  - Weight of 1,4 is increased
  - Weight of 2,3,5 is decreased.
- Second round of boosting , again 5 samples, but now 1,4 are more likely to be sampled.

Boosting

- Is an iterative procedure.
- The distribution of training examples are adaptively changed, so that the base classifiers in the next iteration, focus more on examples that are wrongly predicted in the previous iteration.
- Boosting assigns a weight to each example.
- Weights are adaptively changed at the end of each boosting round.
- Weights assigned to the training examples are used in the following ways:-

Boosting

1. To draw a set of bootstrap samples from the original data
2. Can be used by the base classifier to learn a model that is biased towards heigher weight examples.

Steps:-

1. Initially wt of all examples are same 1/N.
2. A sample is drawn as per the sampling distbn of the training examples to get a new training set.
3. A classifier is induced from this training set.

Boosting

4. All examples of the original data are classified using this classifier.

5. Wrongly classified examples … increase in weight

Correctly classified examples …. decrease in weight.

So, wrongly classified examples will be focussed more in subsequent iterations.

6. Repeat steps 2 to 5 for k times(k = no. of base classifiers)

Boosting

7. As boosting round proceeds, wrongly classified examples become more prevalent.

8. Final ensemble is got by aggregating base classifiers got from each boosting round.

Several implementations of the boosting algorithm have been developed. They all differ in terms of

1) How the weights of the examples are updated.
2) How the predictions of the base classifiers are combined.

Example: AdaBoost

1. Let $\{(x_j, y_j) \mid j = 1,2,3,4\ldots. N\}$ is a set of N training examples.

- Base classifiers: $C_1, C_2, \ldots, C_T$

2. Error rate of a base classifier:

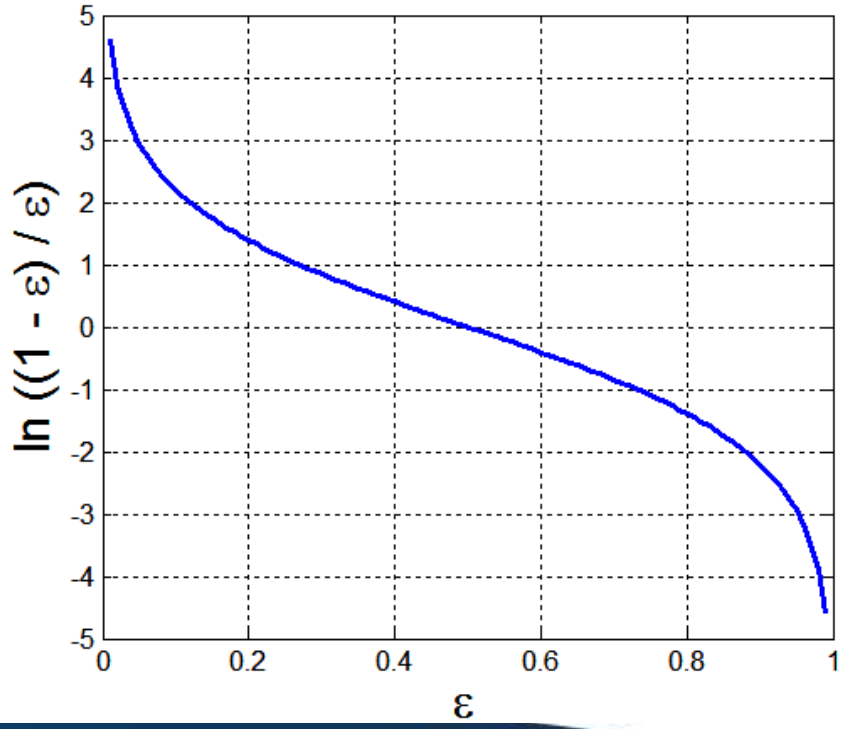
$$\varepsilon_i = \frac{1}{N}\sum_{j=1}^{N} w_j \delta\left(C_i(x_j) \neq y_j\right)$$

where $\delta(P) = 1 \; if \; P \; is \; true$

$= 0$ otherwise

- Importance of a classifier:

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$



Example: AdaBoost

3. $\propto_i$ is also used to update the weight of training examples.

Weight update:

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

Z_j is used to ensure that $\sum w_i^{j+1} = 1$

where Z_j is the normalizat ion factor

- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to 1/n and the resampling procedure is repeated

Ada Boost

4. Instead of majority voting, the prediction made by each C_j is weighted according to α_j. The weighted average of this is the final ensemble.

Classification

$$C^*(x) = \arg\max_y \sum_{j=1}^{T} \alpha_j \delta\left(C_j(x) = y\right)$$

[Refer book for the algorithm

Refer transparencies for a numerical example]

KNN Numerical Example (hand computation)

Numerical Example of K Nearest Neighbor Algorithm

Here is step by step on how to compute K-nearest neighbors KNN algorithm:

1. Determine parameter K = number of nearest neighbors
2. Calculate the distance between the query-instance and all the training samples
3. Sort the distance and determine nearest neighbors based on the K-th minimum distance
4. Gather the category y of the nearest neighbors
5. Use simple majority of the category of nearest neighbors as the prediction value of the query instance

Example

We have data from the questionnaires survey (to ask people opinion) and objective testing with two attributes (acid durability and strength) to classify whether a special paper tissue is good or not. Here is four training samples

X1 = Acid Durability (seconds)	X2 = Strength (kg/square meter)	Y = Classification
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good

Now the factory produces a new paper tissue that pass laboratory test with X1 = 3 and X2 = 7. Without another expensive survey, can we guess what the classification of this new tissue is?

1. Determine parameter K = number of nearest neighbors

Suppose use K = 3

2. Calculate the distance between the query-instance and all the training samples

Coordinate of query instance is (3, 7), instead of calculating the distance we compute square distance which is faster to calculate (without square root)

2

X1 = Acid Durability (seconds)	X2 = Strength (kg/square meter)	**Square Distance to query instance (3,7)**
7	7	$(7-3)^2+(7-7)^2=16$
7	4	$(7-3)^2+(4-7)^2=25$
3	4	$(3-3)^2+(4-7)^2=9$
1	4	$(1-3)^2+(4-7)^2=13$

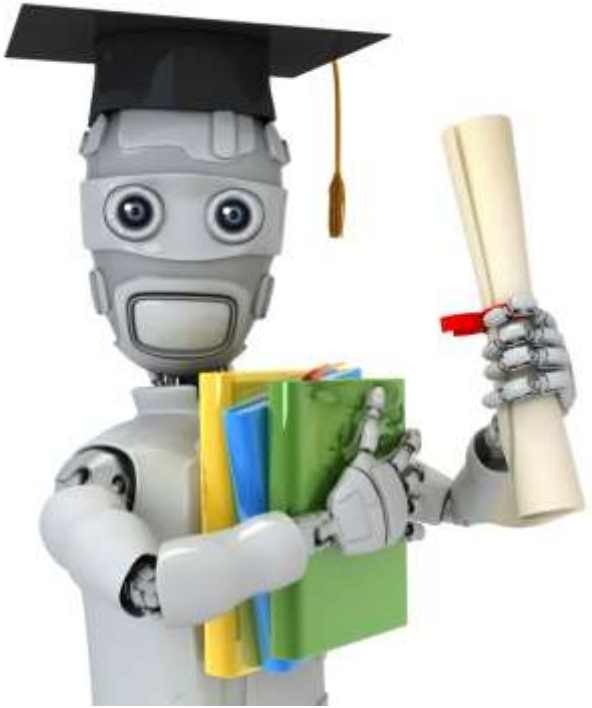
3. *Sort the distance and determine nearest neighbors based on the K-th minimum distance*				
X1 = Acid Durability (seconds)	X2 = Strength (kg/square meter)	Square Distance to query instance (3, 7)	**Rank minimum distance**	**Is it included in 3-Nearest neighbors?**
7	7	$(7-3)^2+(7-7)^2=16$	**3**	**Yes**
7	4	$(7-3)^2+(4-7)^2=25$	**4**	**No**
3	4	$(3-3)^2+(4-7)^2=9$	1	**Yes**
1	4	$(1-3)^2+(4-7)^2=13$	**2**	**Yes**

4. Gather the category Y of the nearest neighbors. Notice in the second row last column that the category of nearest neighbor (Y) is not included because the rank of this data is more than 3 (=K).

X1 = Acid Durability	X2 =Strength	Square Distance to query instance (3, 7)	Rank minimum	Is it included in 3-Nearest neighbors?	**Y = Category of nearest Neighbor**
(seconds)	(kg/square meter)		distance		
7	7	$(7-3)^2+(7-7)^2=16$	3	yes	**Bad**
7	4	$(7-3)^2+(4-7)^2=25$	4	no	**-**
3	4	$(3-3)^2+(4-7)^2=9$	1	yes	**good**
1	4	$(1-3)^2+(4-7)^2=13$	2	yes	**good**

5. *Use simple majority of the category of nearest neighbors as the prediction value of the query instance*

We have 2 good and 1 bad, since 2>1 then we conclude that a new paper tissue that pass laboratory test with X1 = 3 and X2 = 7 is included in **Good** category.

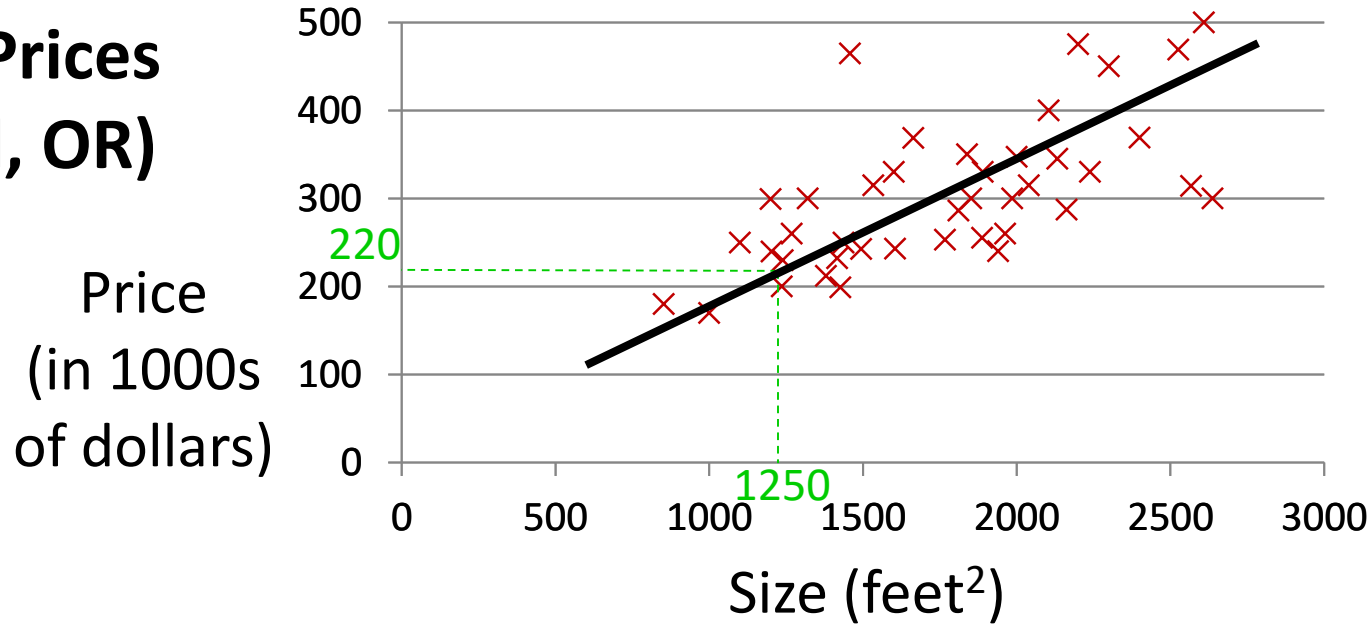


Machine Learning

Linear regression with one variable

Model representation

Housing Prices (Portland, OR)



Supervised Learning

Given the "right answer" for each example in the data.

Regression Problem

Predict real-valued output

Classification : Discrete-valued output

Training set of housing prices (Portland, OR)

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

m

Notation:

m = Number of training examples

x's = "input" variable / features

y's = "output" variable / "target" variable

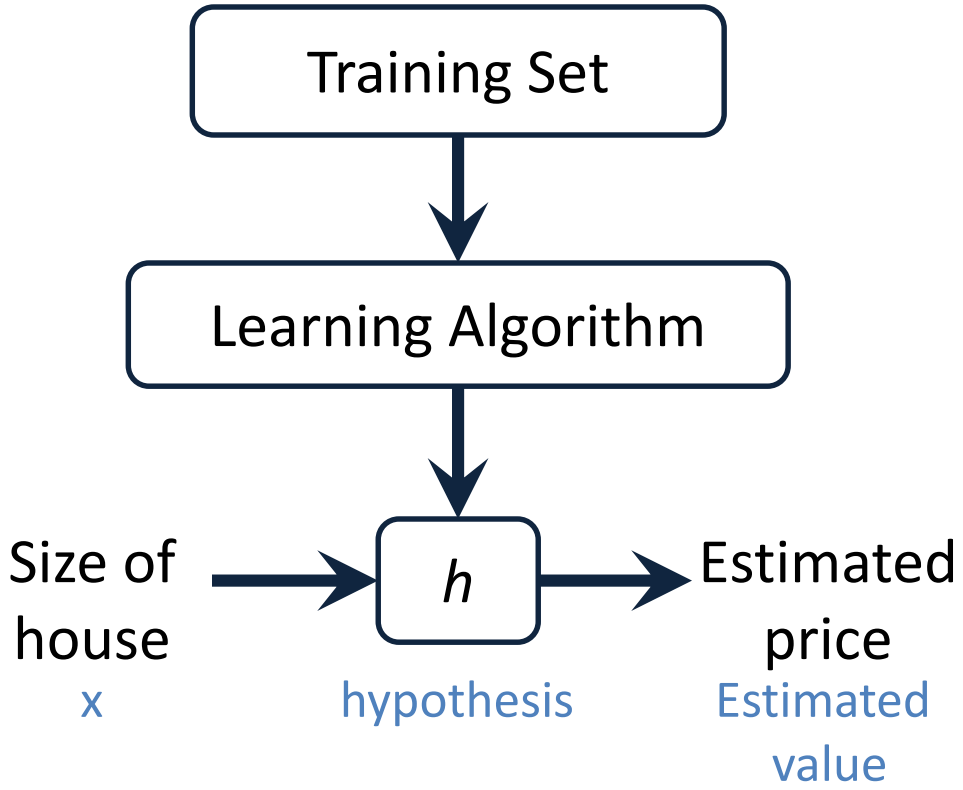
$x^{(1)} = 2104$

$x^{(2)} = 1416$

$y^{(1)} = 460$

(x, y) – one training example

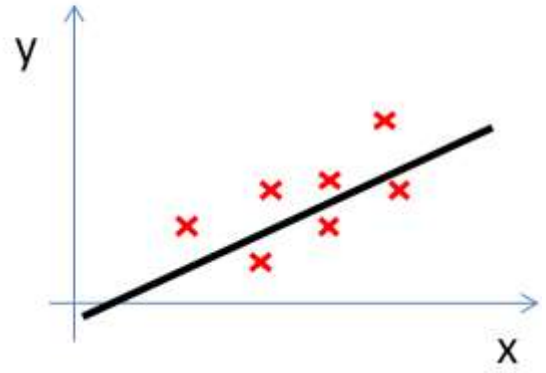
$(x^{(i)}, y^{(i)})$ – ith trainingg example



h maps from x's to y's

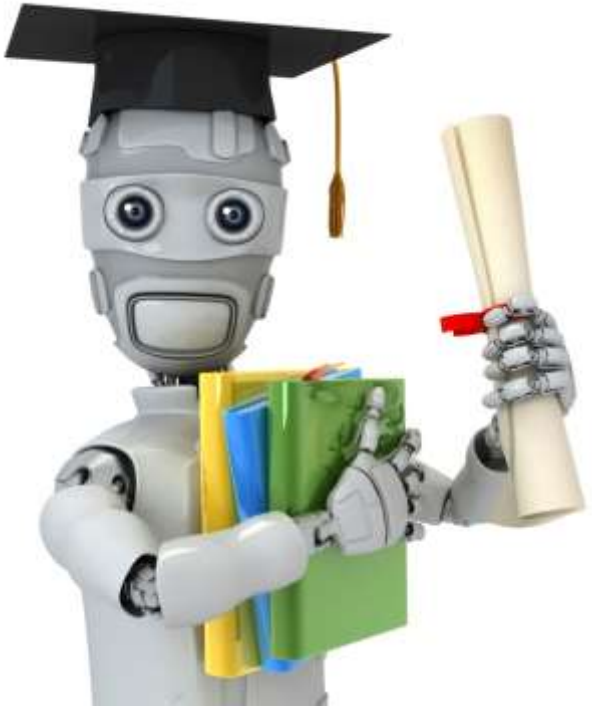
How do we represent *h* ?

$h_\Theta(x) = \Theta_0 + \Theta_1 x$



Linear regression with one variable.
Univariate linear regression.

One variable



Machine Learning

Linear regression with one variable

Cost function

Training Set

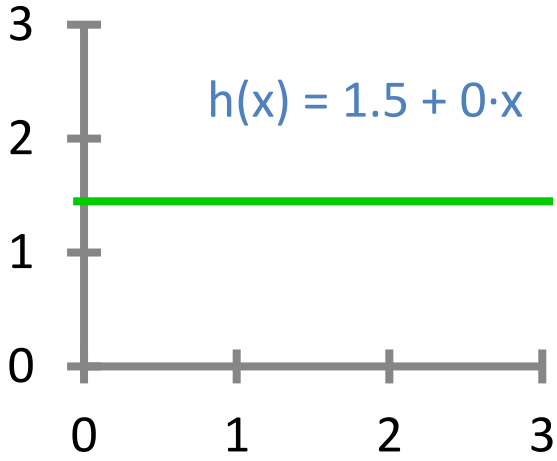
Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

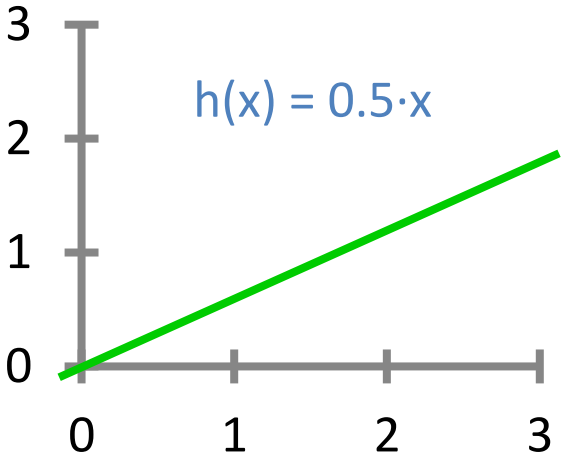
θ_i's: Parameters

How to choose θ_i's ?

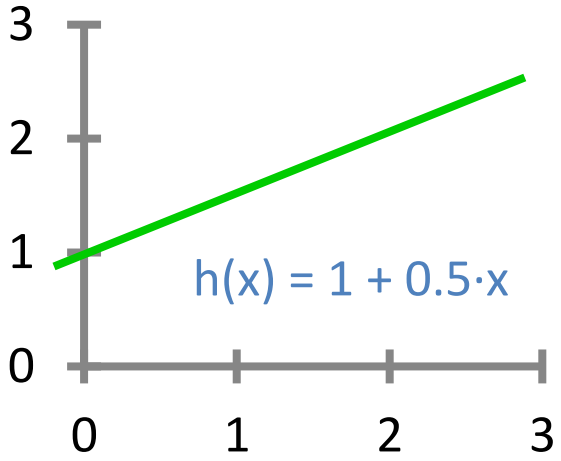
$$h_\theta(x) = \theta_0 + \theta_1 x$$



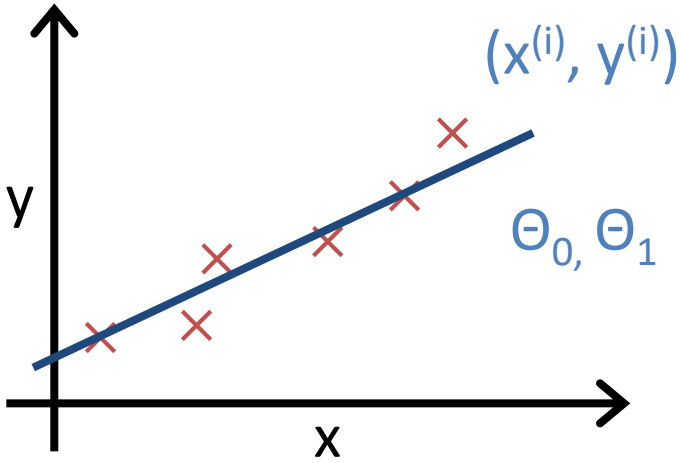
$\theta_0 = 1.5$
$\theta_1 = 0$



$\theta_0 = 0$
$\theta_1 = 0.5$



$\theta_0 = 1$
$\theta_1 = 0.5$



$$\underset{\Theta_0\,\Theta_1}{\text{minimize}} \frac{1}{2m}\sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)})^2$$

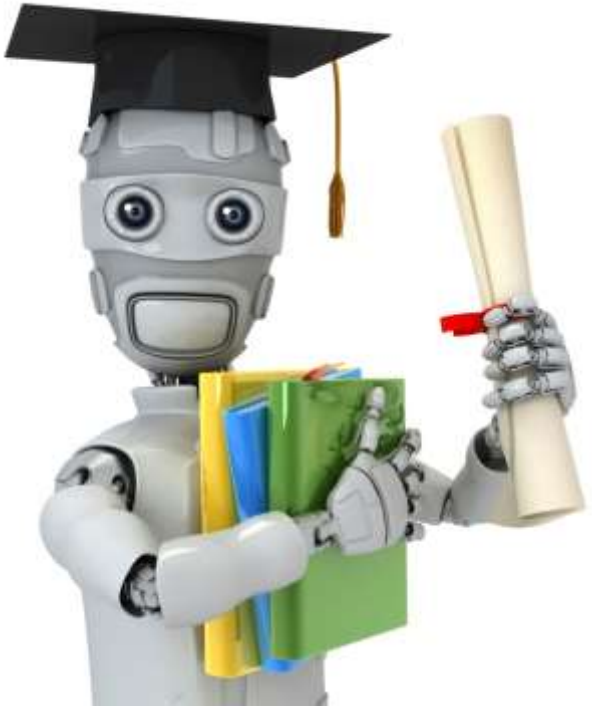
$$h(x) = \Theta_0 + \Theta_1 x^{(i)}$$

$$J(\Theta_0, \Theta_1) = \frac{1}{2m}\sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)})^2$$

Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y)

$$\underset{\Theta_0\,\Theta_1}{\text{Minimize}}\ J(\Theta_0, \Theta_1) : \text{Cost Function}$$

Squared error function



Machine Learning

Linear regression with one variable

Cost function intuition I

Simplified

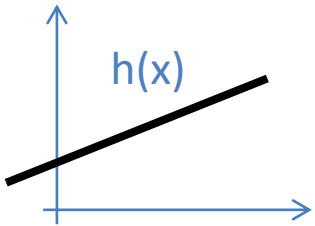
Hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

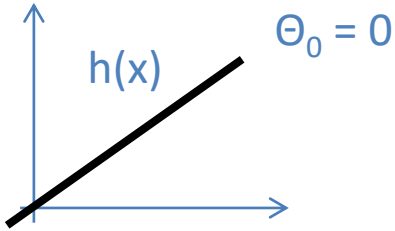
$$h_\theta(x) = \theta_1 x$$

Parameters:

$$\theta_0, \theta_1$$



$$\theta_1$$



Cost Function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

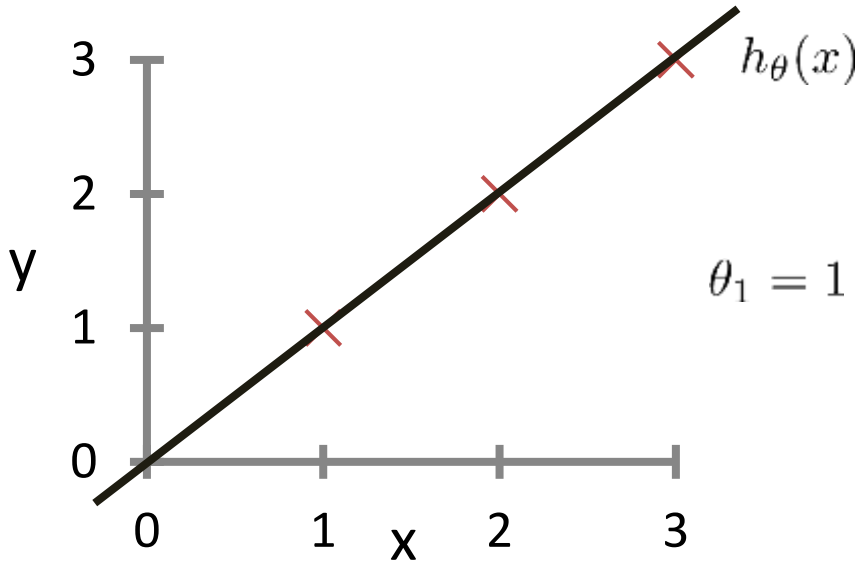
$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

$\underset{\theta_1}{\text{minimize}}\ J(\theta_1)$

$h_\theta(x)$

(for fixed θ_1, this is a function of x)



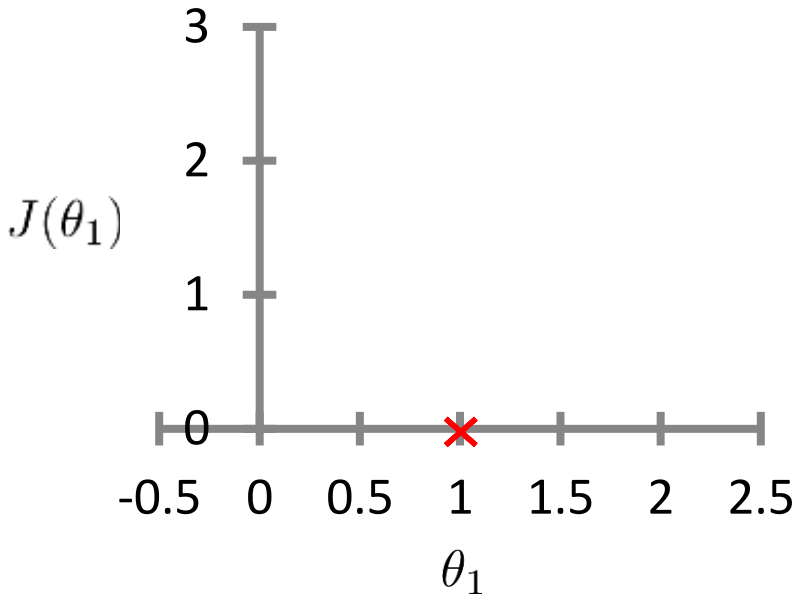
$J(\Theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\Theta(x^{(i)}) - y^{(i)})^2$

$= \frac{1}{2m}\sum_{i=1}^{m}(\Theta_1 x^{(i)} - y^{(i)})^2$

$= \frac{1}{2m}(0^2 + 0^2 + 0^2)$

$J(\theta_1)$

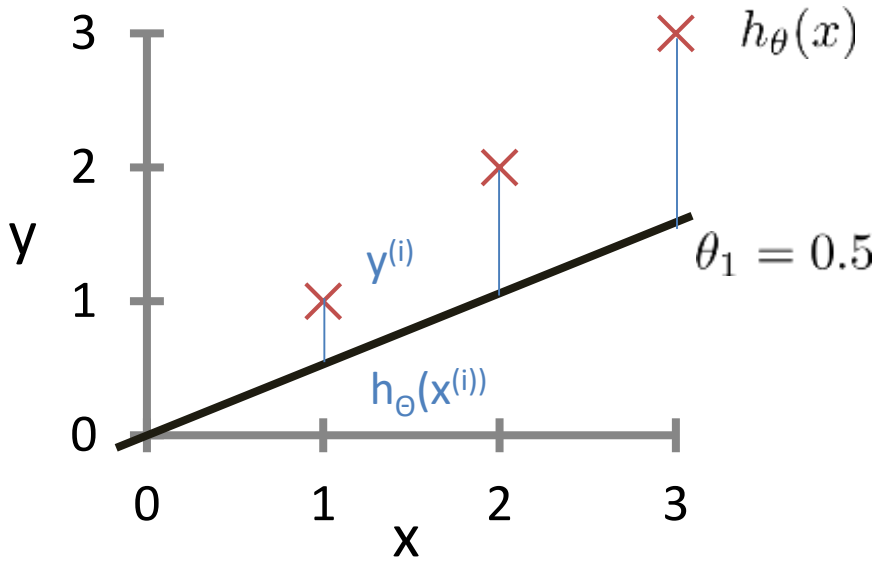
(function of the parameter θ_1)



$J(1) = 0$

$h_\theta(x)$

(for fixed θ_1, this is a function of x)

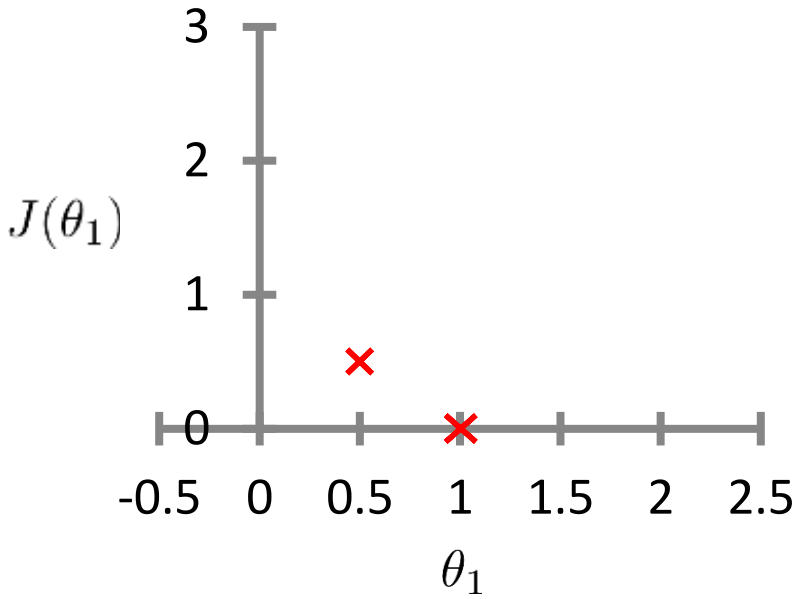


$J(\theta_1)$ = $\frac{1}{2\cdot 3}\sum_{i=1}^{3}[(0.5-1)^2+(1-2)^2+(1.5-3)^2]$

= $\frac{1}{6}\cdot(3.5) = 0.58$

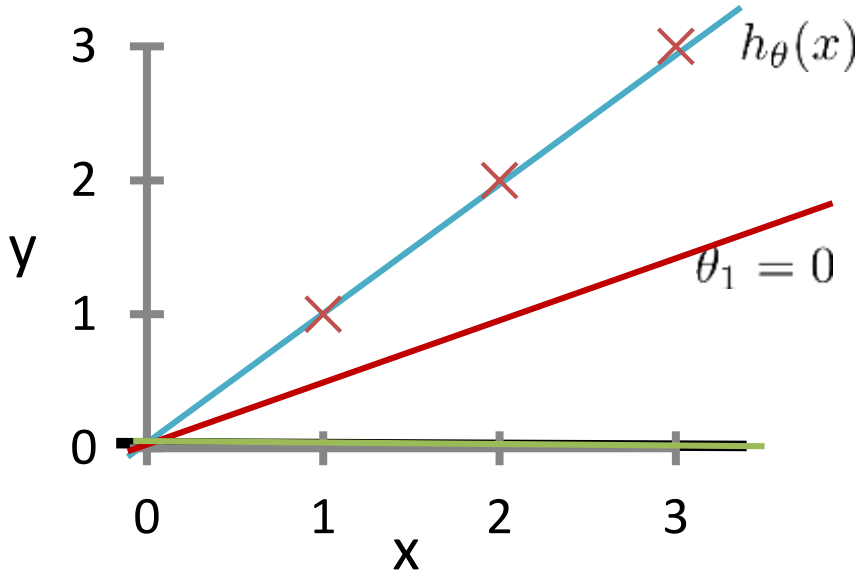
$J(\theta_1)$

(function of the parameter θ_1)



$h_\theta(x)$

(for fixed θ_1, this is a function of x)

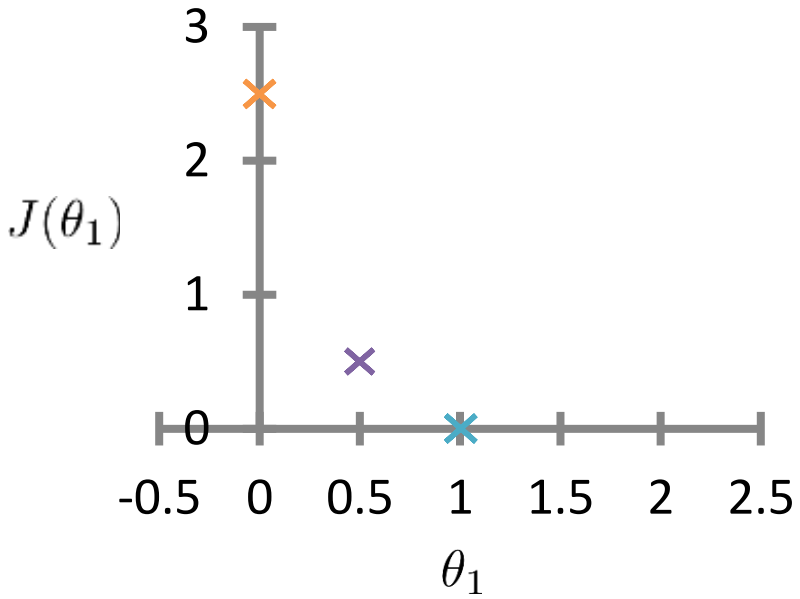


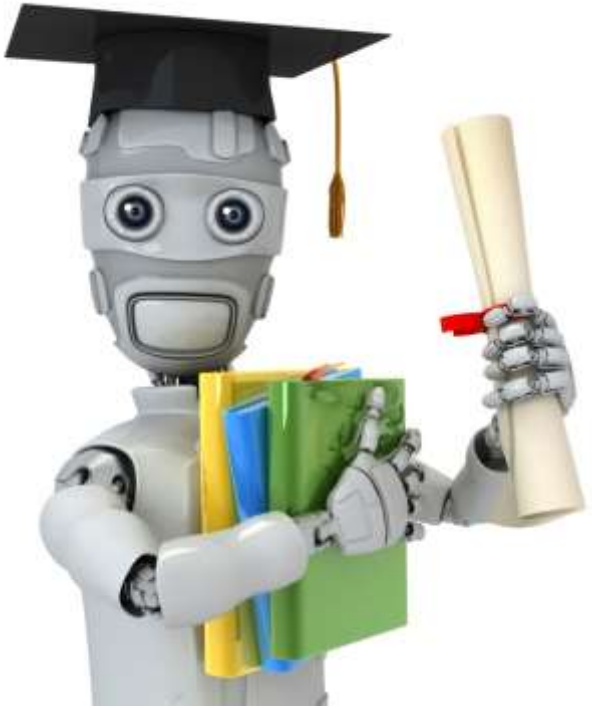
$J(0) = \frac{1}{2 \cdot 3} \sum_{i=1}^{3} [1^2 + 2^2 + 3^2]$

$= \frac{1}{6} \cdot 14 = 2.3$

$J(\theta_1)$

(function of the parameter θ_1)





Machine Learning

Linear regression with one variable

Cost function intuition II

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

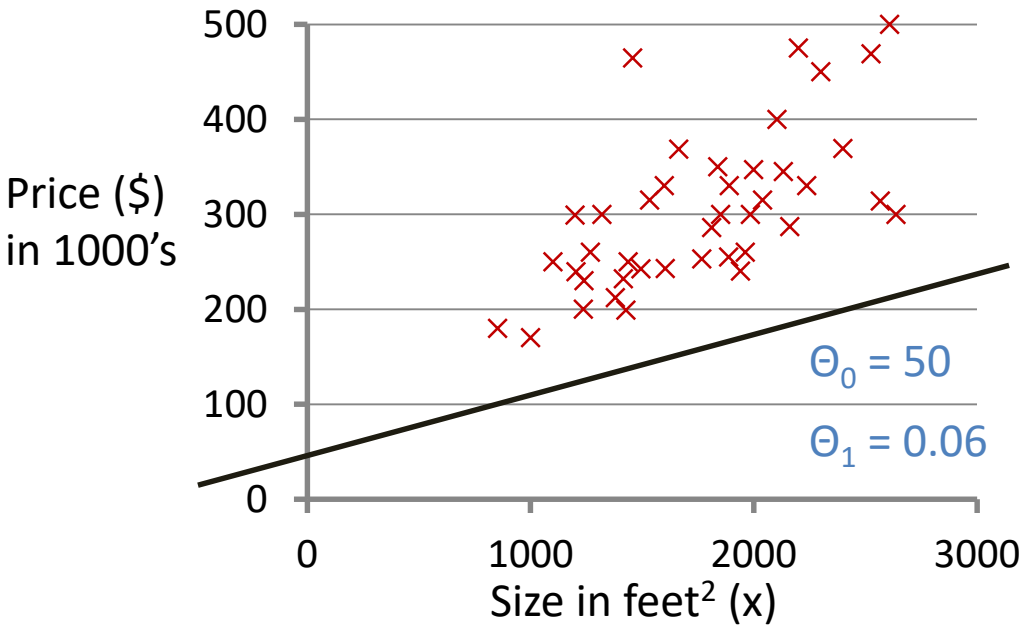
Parameters: θ_0, θ_1

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

$h_\theta(x)$

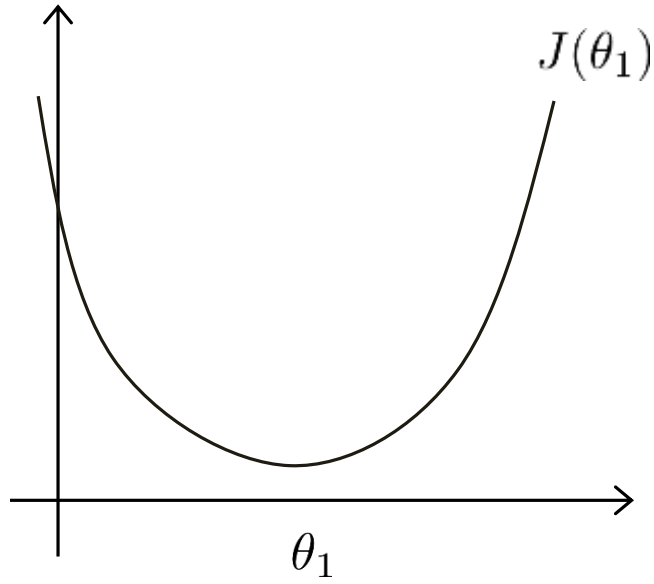
(for fixed θ_0, θ_1, this is a function of x)



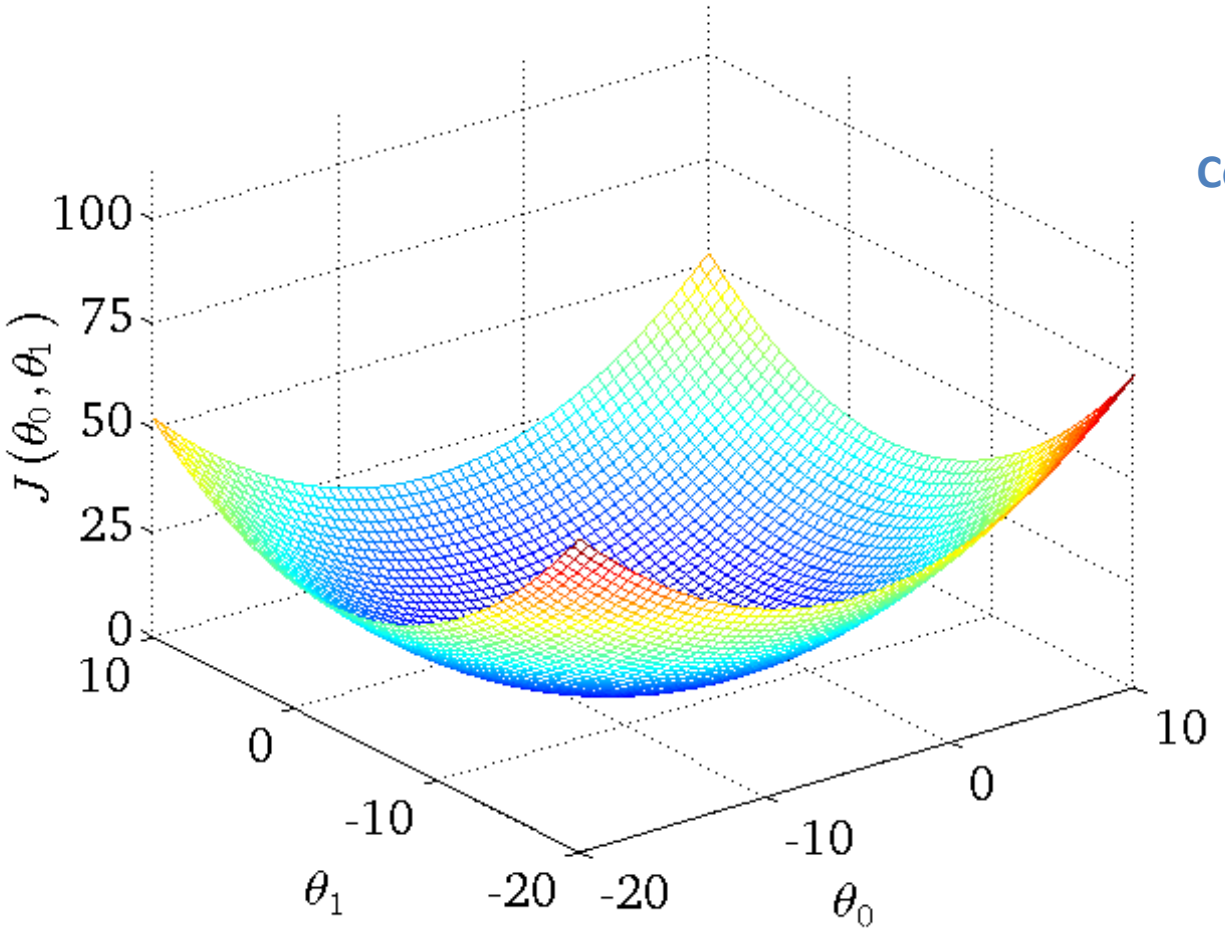
$$h_\theta(x) = 50 + 0.06x$$

$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

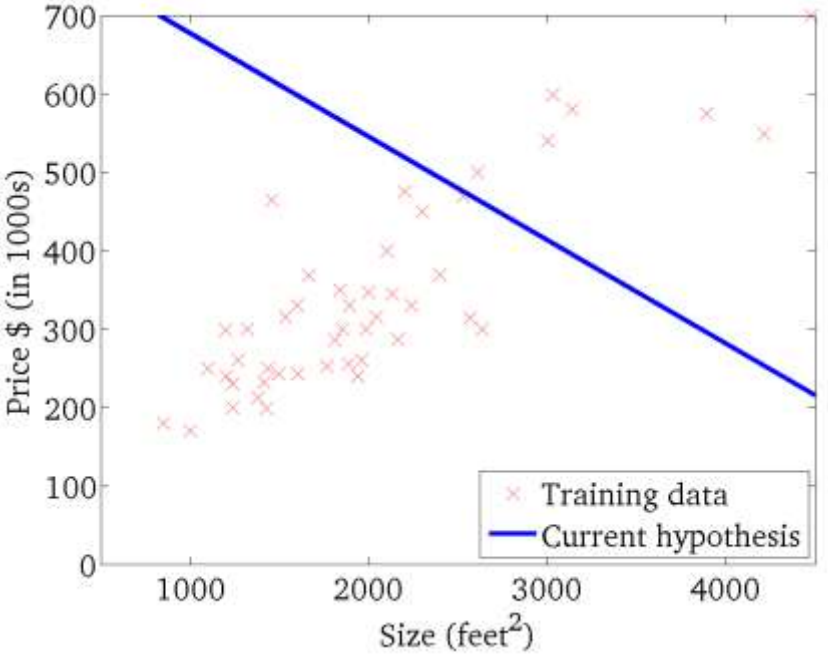


Contour plots



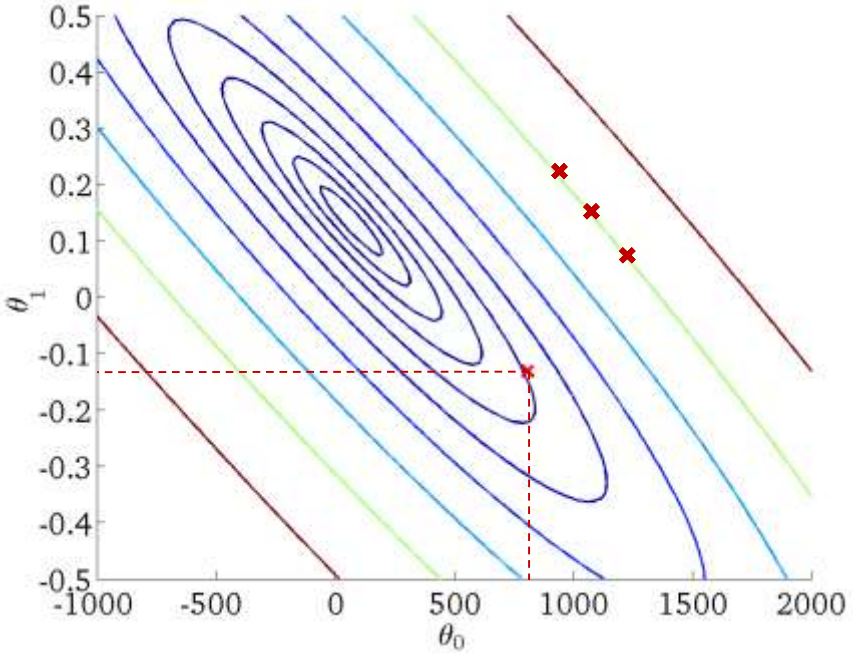
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



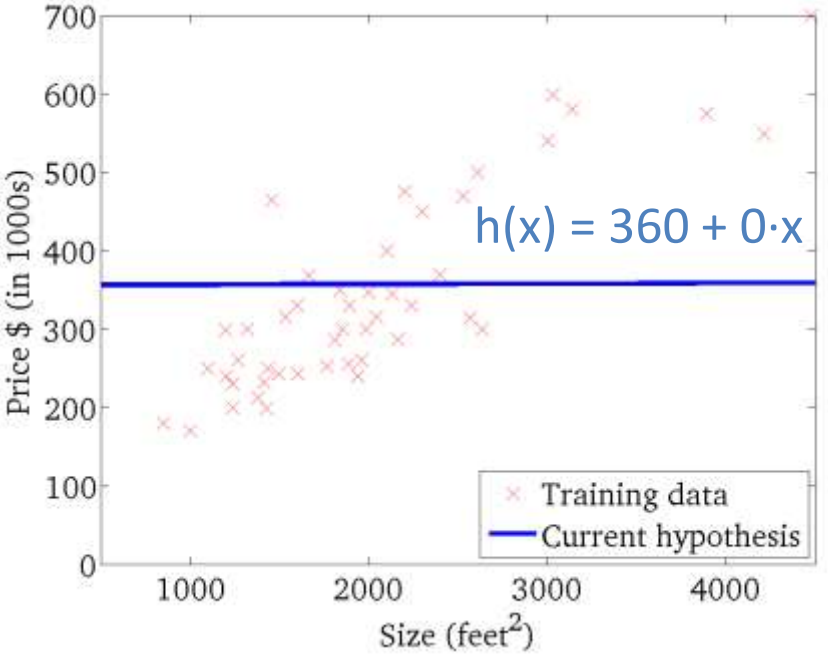
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



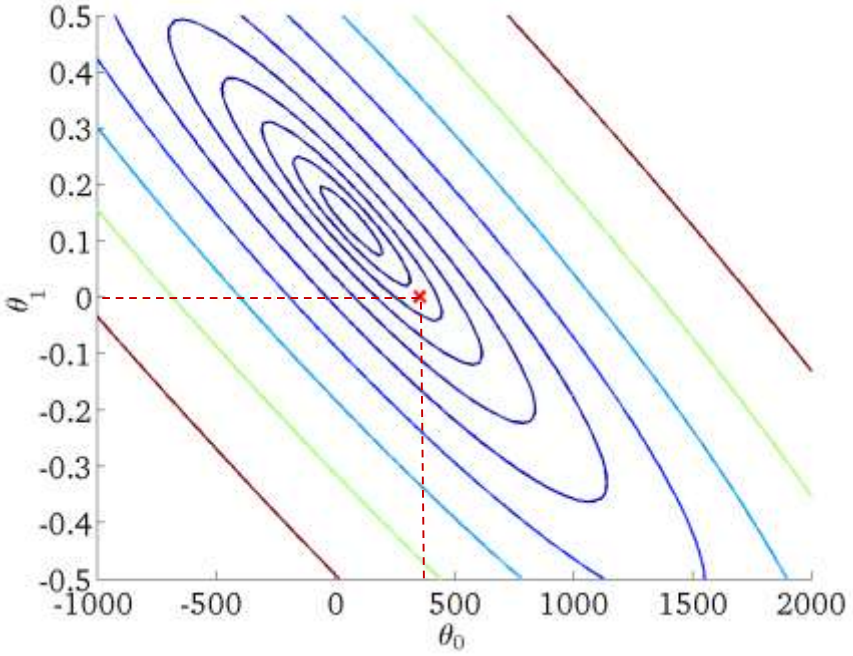
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

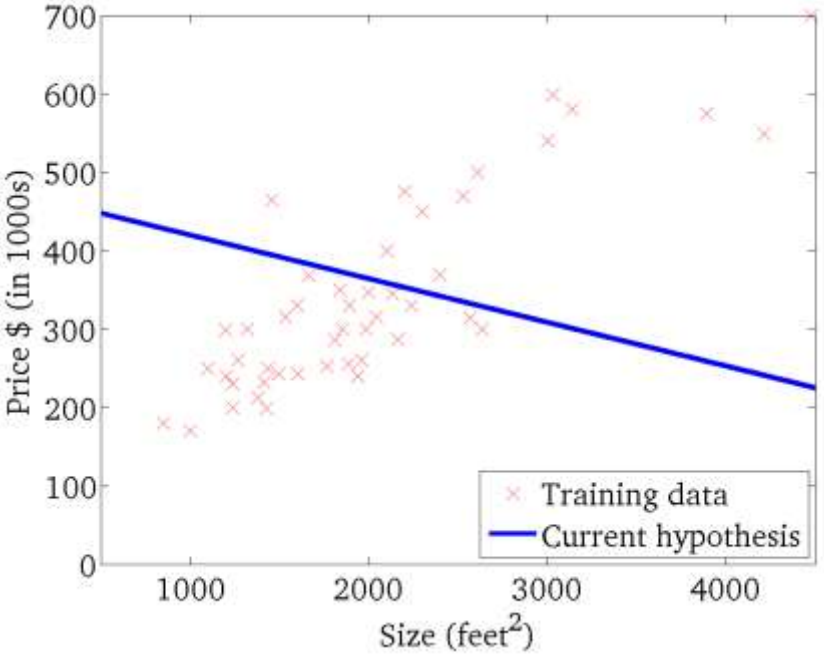


$\Theta_0 = 360$

$\Theta_1 = 0$

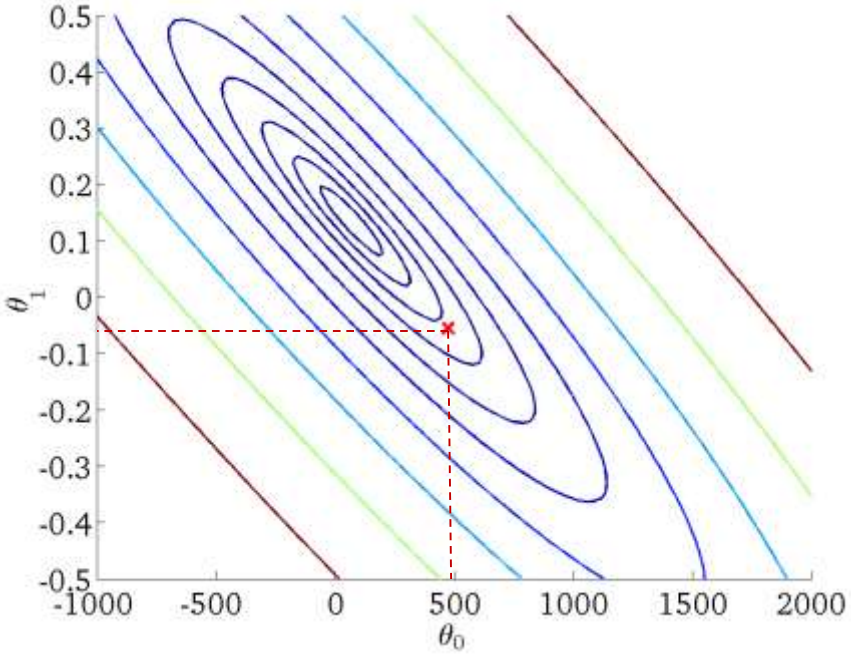
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



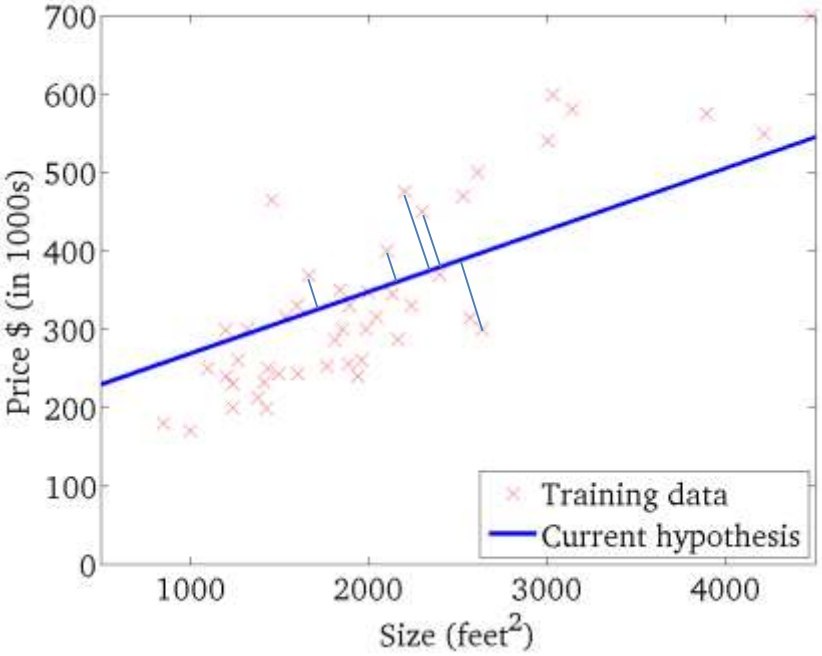
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



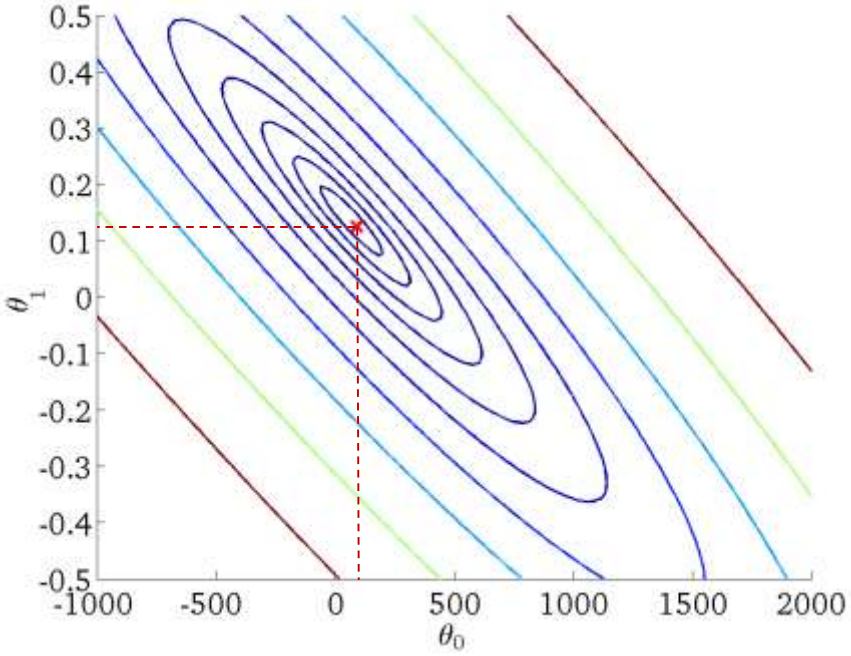
$h_\theta(x)$

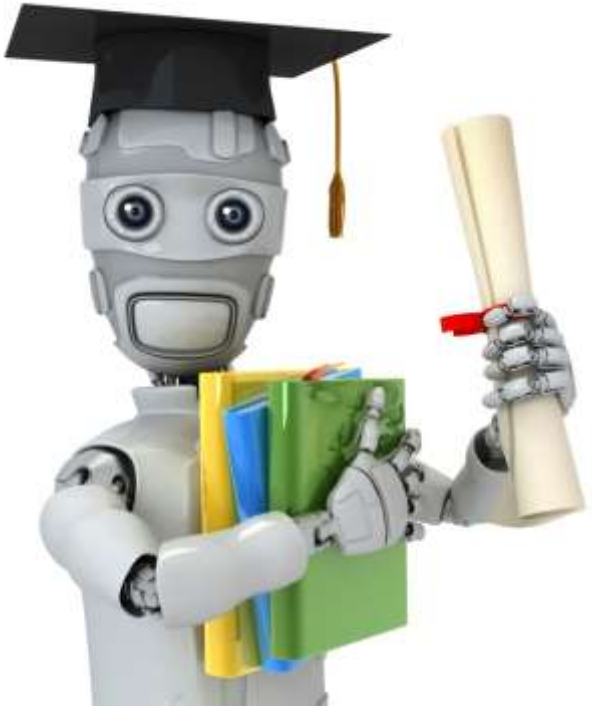
(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)





Machine Learning

Linear regression with one variable

Gradient descent

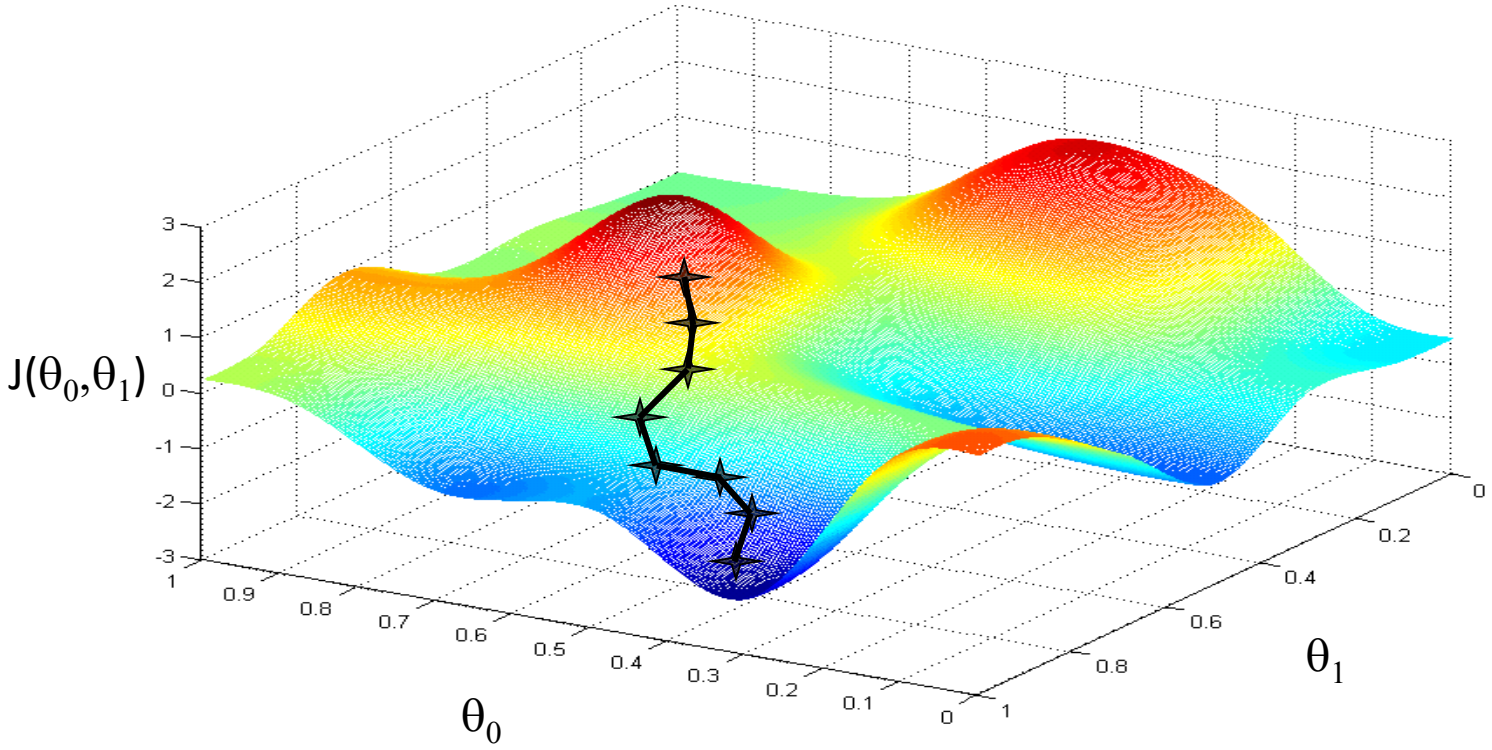
Have some function $J(\theta_0, \theta_1)$

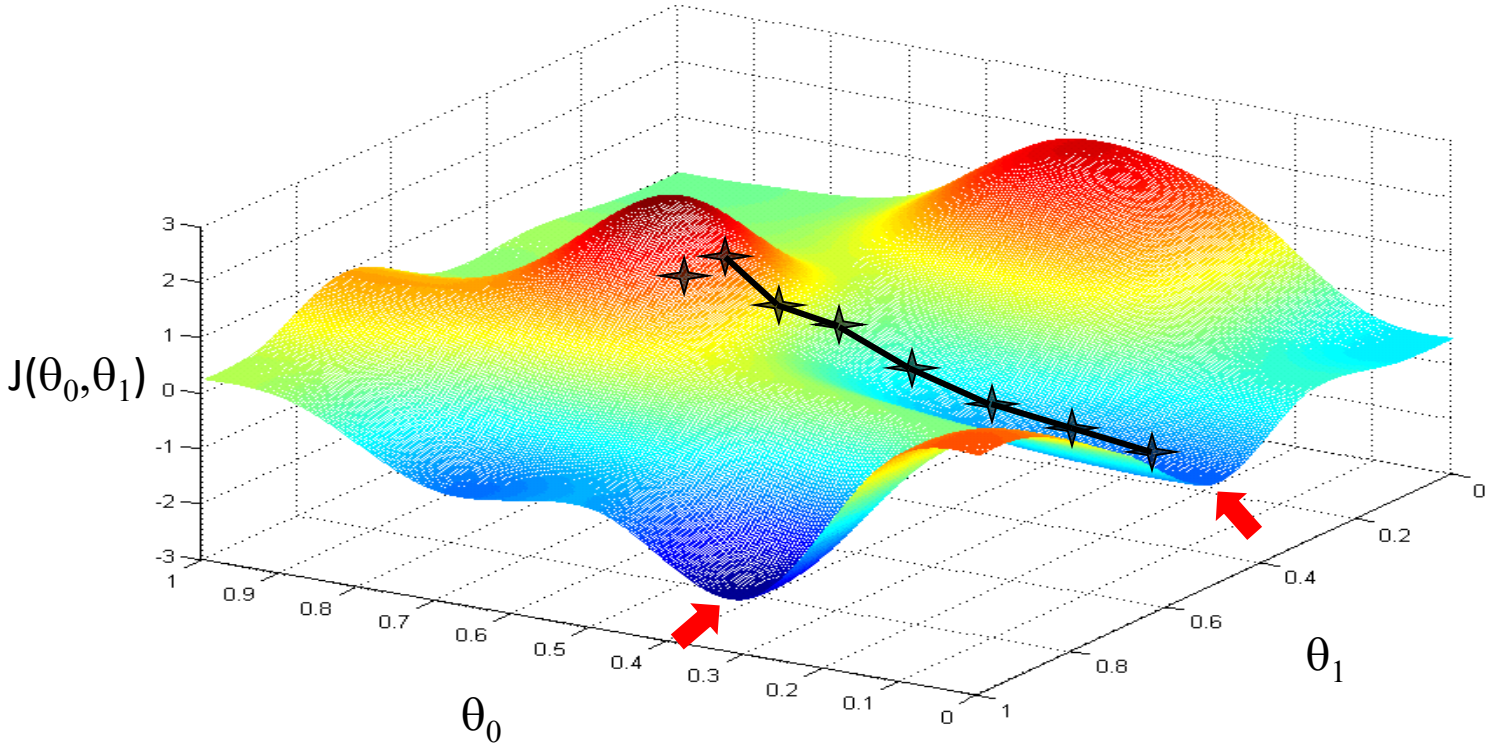
Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$

  until we hopefully end up at a minimum


$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1


$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1

Gradient descent algorithm

assignment
a:=b

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{for } j = 0 \text{ and } j = 1)$$

}

Learning rate

Simultaneously update
Θ_0 & Θ_1

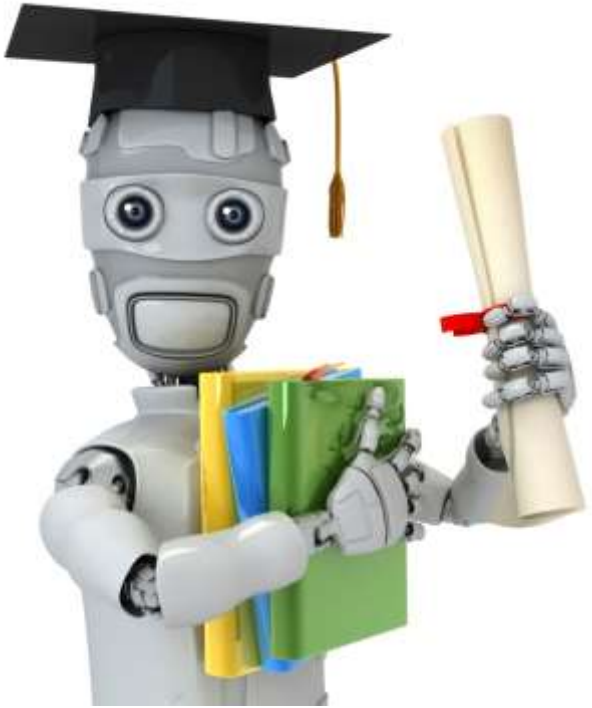
Correct: Simultaneous update

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\theta_1 := \text{temp1}$$

Incorrect:

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_1 := \text{temp1}$$

Andrew Ng



Machine Learning

Linear regression with one variable

Gradient descent intuition

Gradient descent algorithm

repeat until convergence {

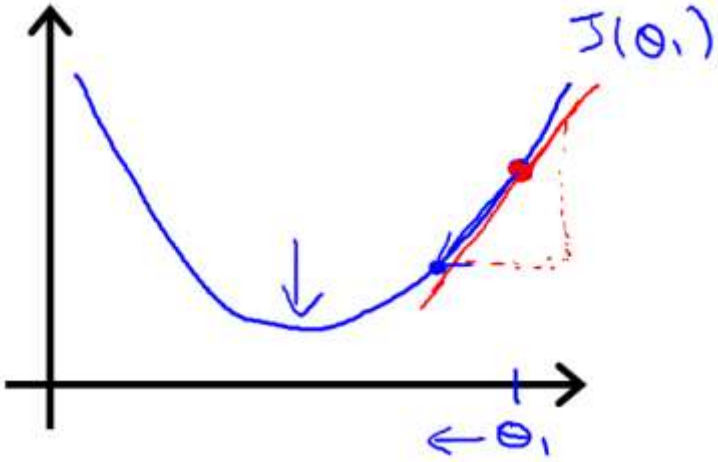
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

(simultaneously update $j = 0$ and $j = 1$)

}

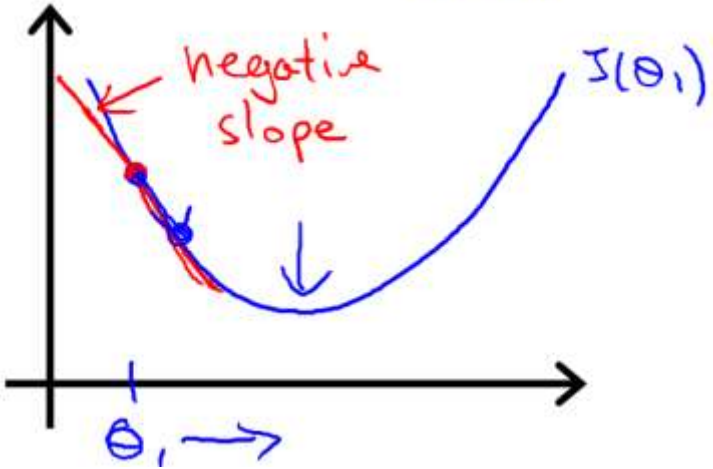
Learning rate

derivative



$$\Theta_1 := \Theta_1 - \alpha \underbrace{\frac{d}{d\theta_1} J(\theta_1)}_{\geq 0}$$

$$\Theta_1 := \Theta_1 - \alpha \cdot J(positive\ number)$$

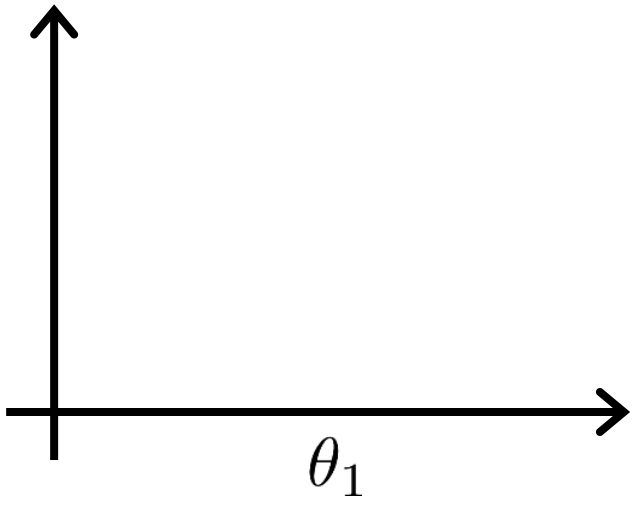


$$\Theta_1 := \Theta_1 - \alpha \underbrace{\frac{d}{d\theta_1} J(\theta_1)}_{\leq 0}$$

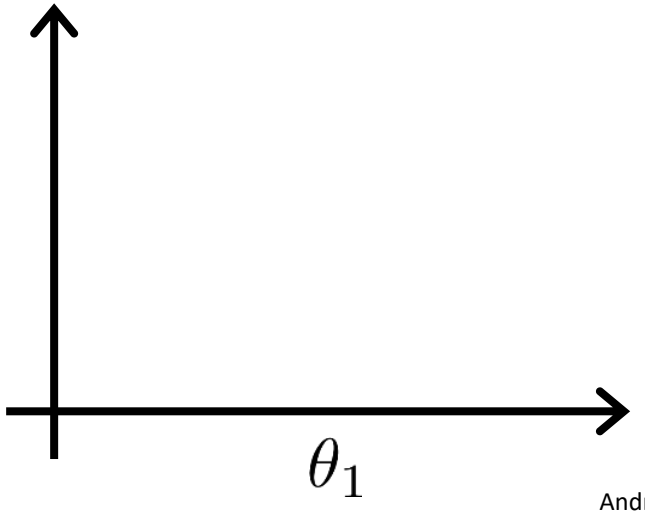
$$\Theta_1 := \Theta_1 - \alpha \cdot J(negative\ number)$$

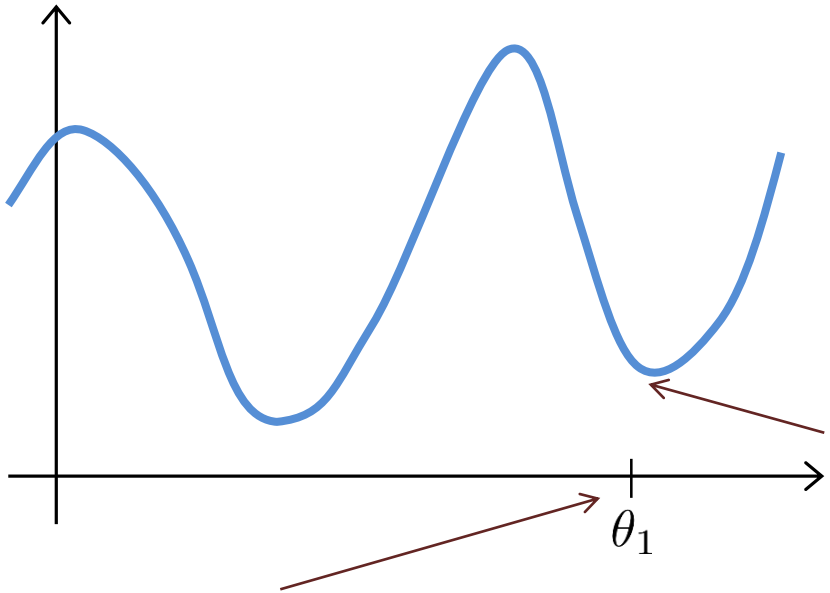
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

If α is too small, gradient descent can be slow.



If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.





θ_1 at local optima

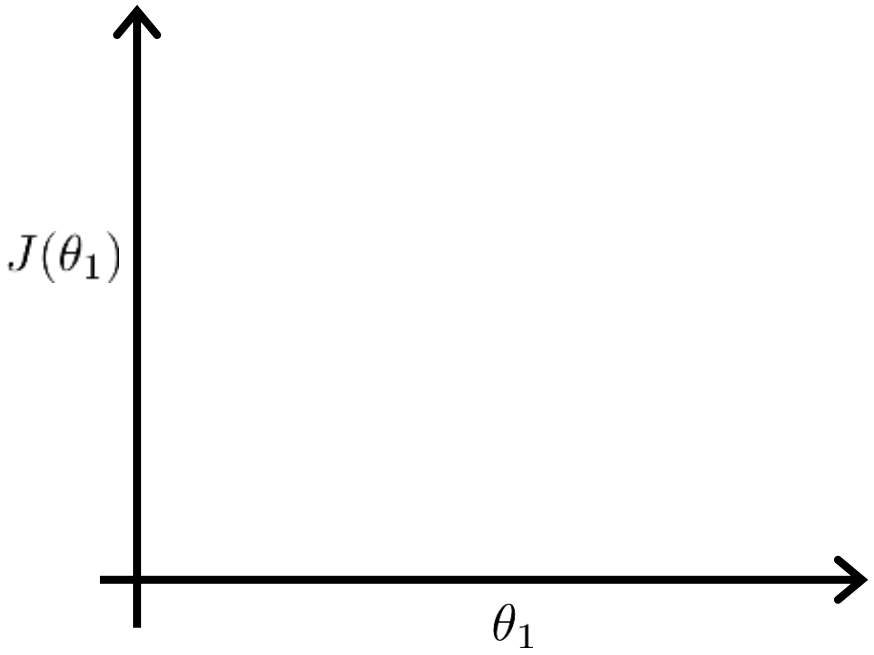
Current value of θ_1

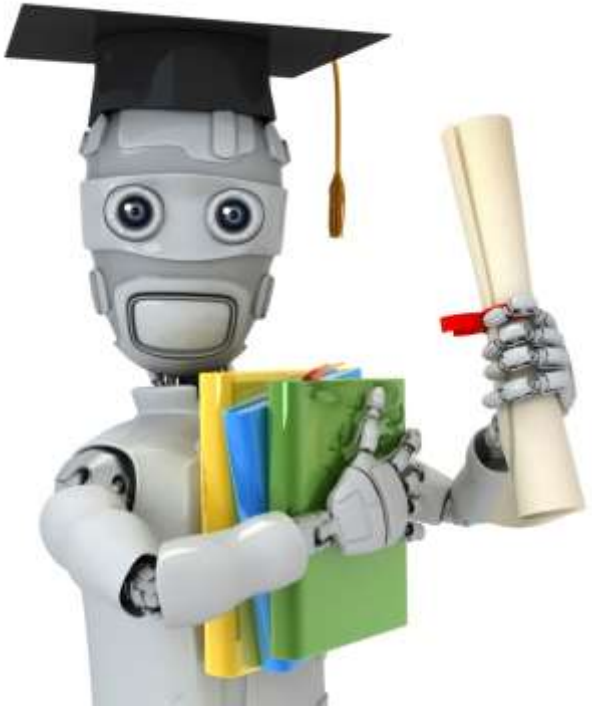
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

Gradient descent can converge to a local minimum, even with the learning rate α fixed.

$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time.





Machine Learning

Linear regression with one variable

Gradient descent for linear regression

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

(for $j = 1$ and $j = 0$)

}

Linear Regression Model

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{d}{d\theta_j} \cdot \frac{1}{2m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)})^2$$

$$= \frac{d}{d\theta_j} \cdot \frac{1}{2m} \sum_{i=1}^{m} (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})^2$$

$$j = 0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)})$$

$$j = 1 : \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

Gradient descent algorithm

repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

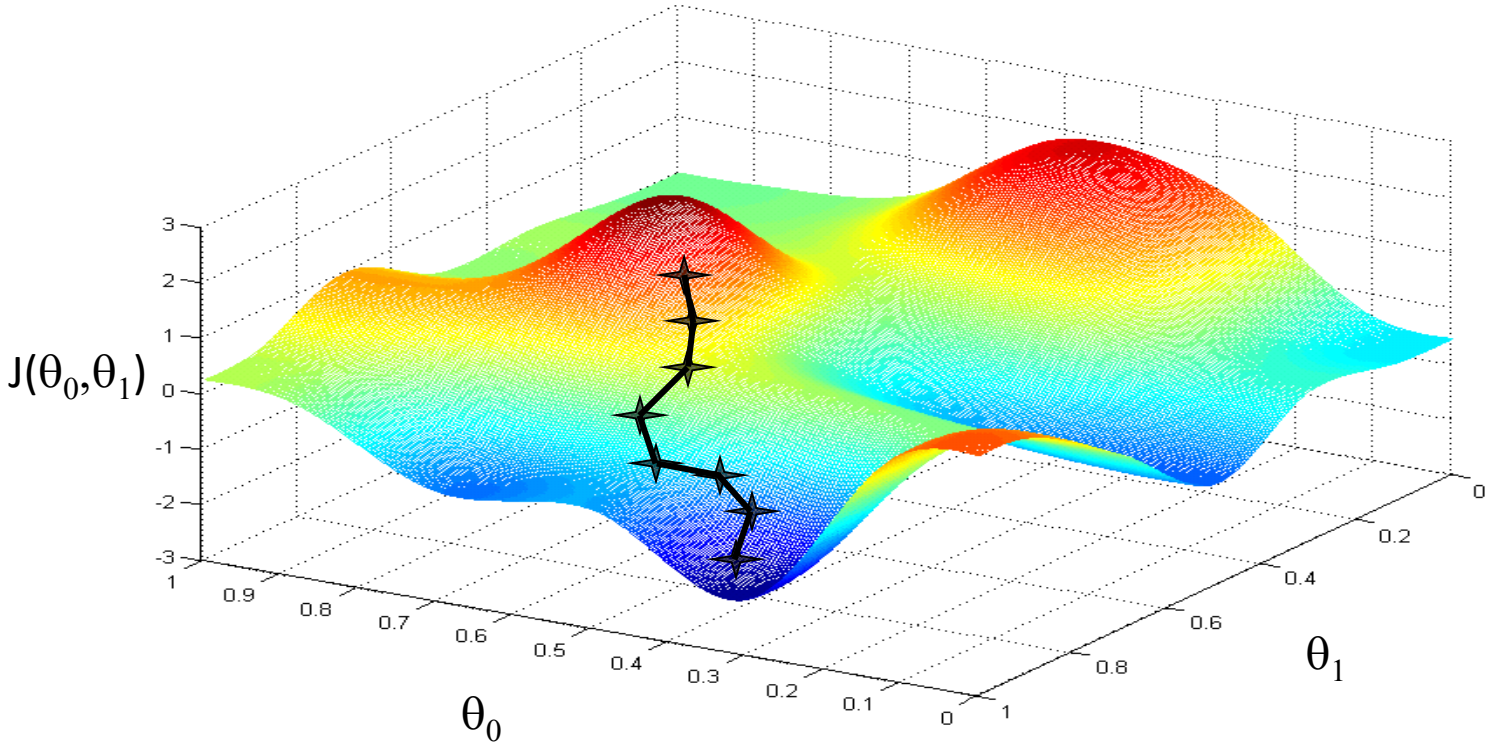
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$$

}

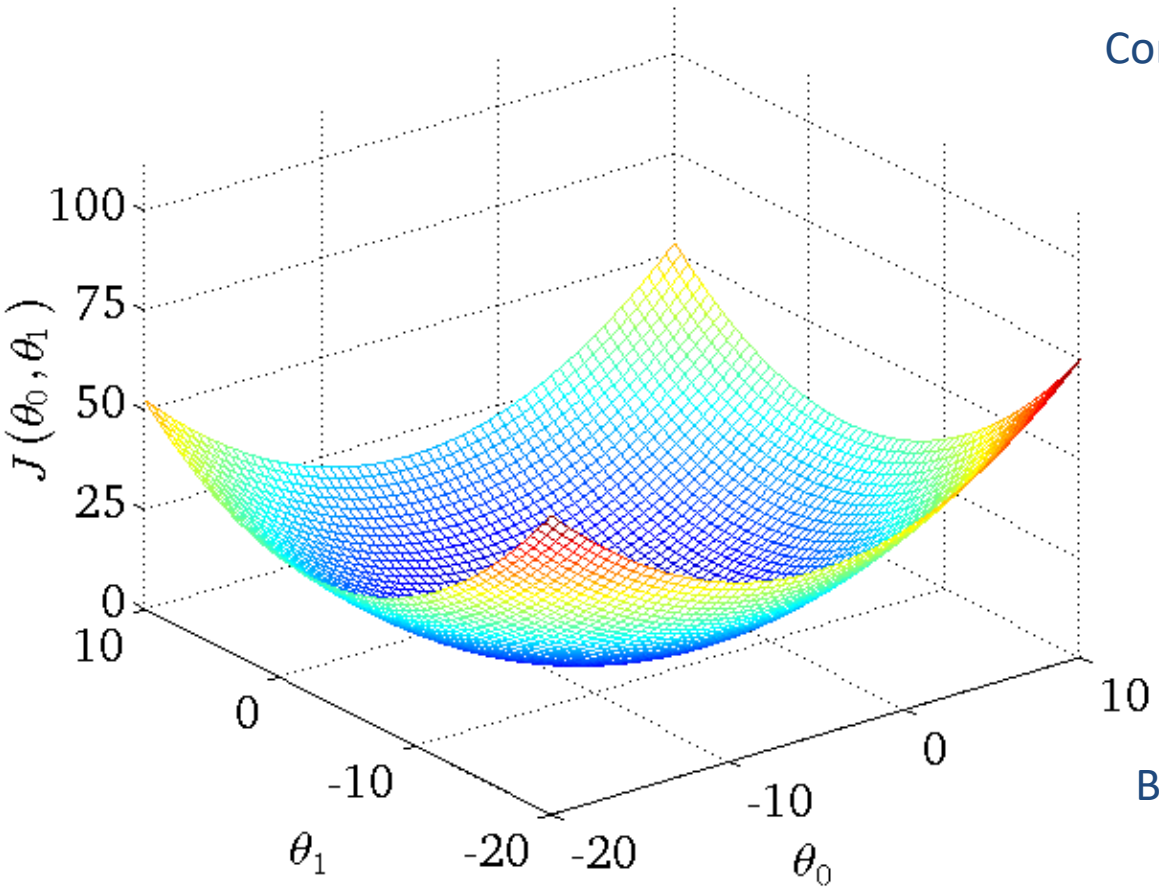
$\frac{d}{d\theta_0} \cdot J(\theta_0, \theta_1)$

$\frac{d}{d\theta_1} \cdot J(\theta_0, \theta_1)$

update θ_0 and θ_1 simultaneously


$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1

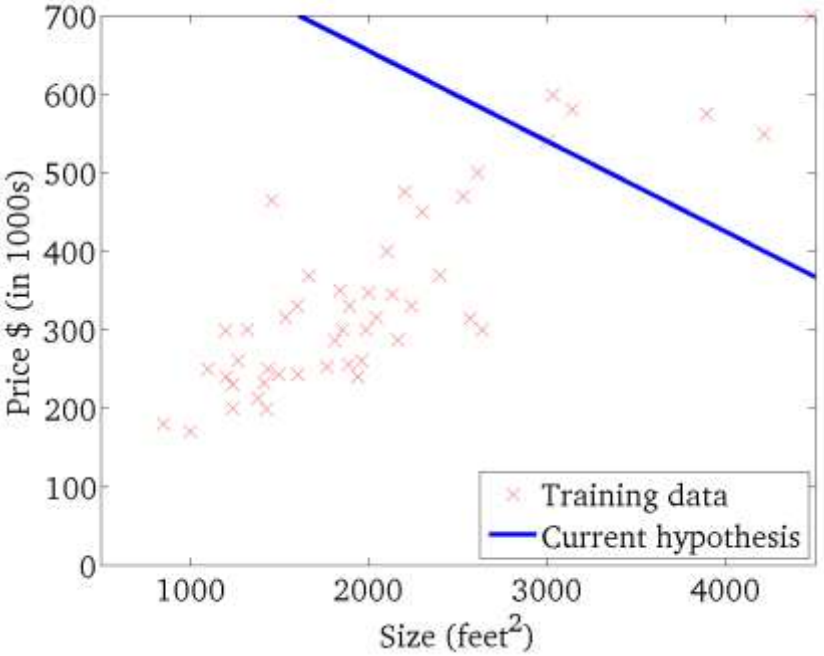
Convex function



Bowl-shaped

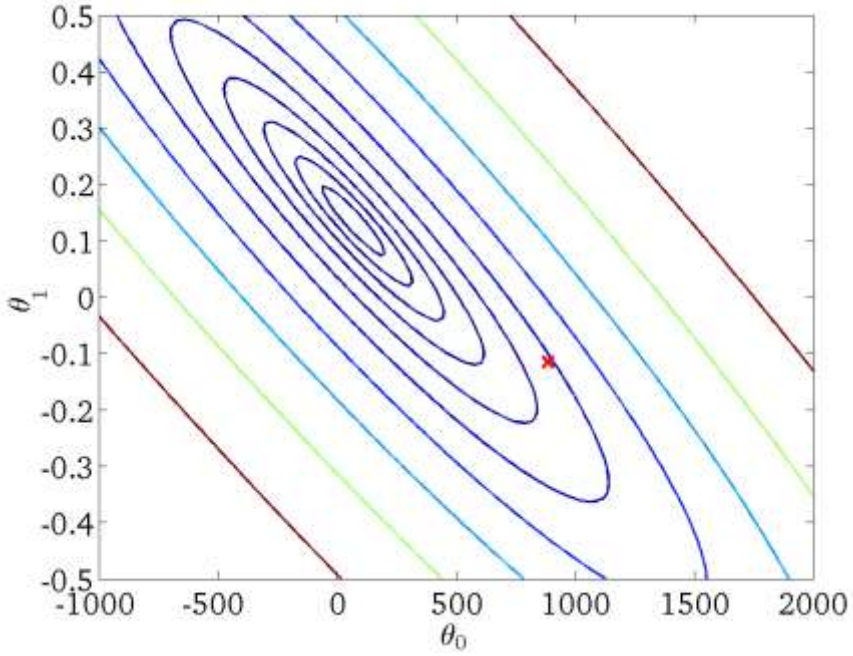
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



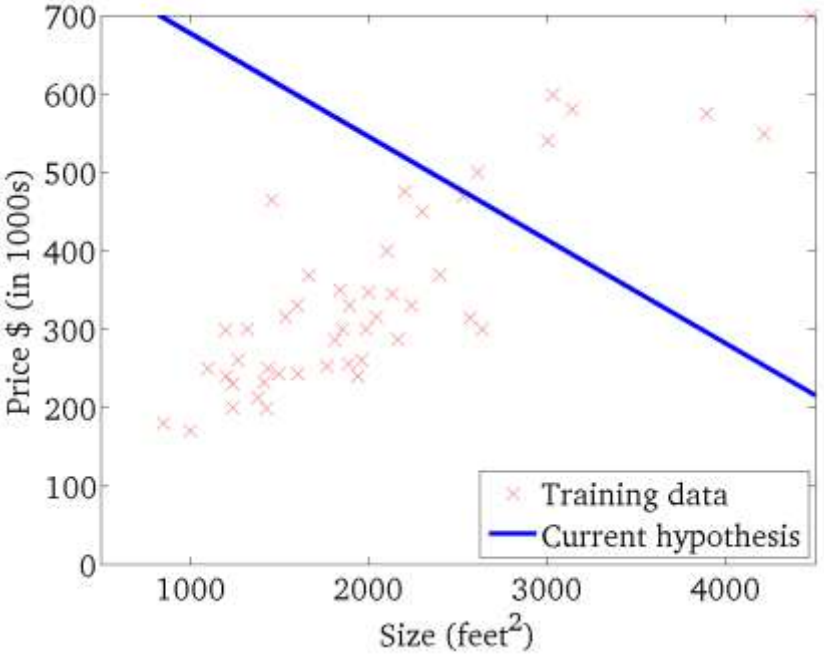
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



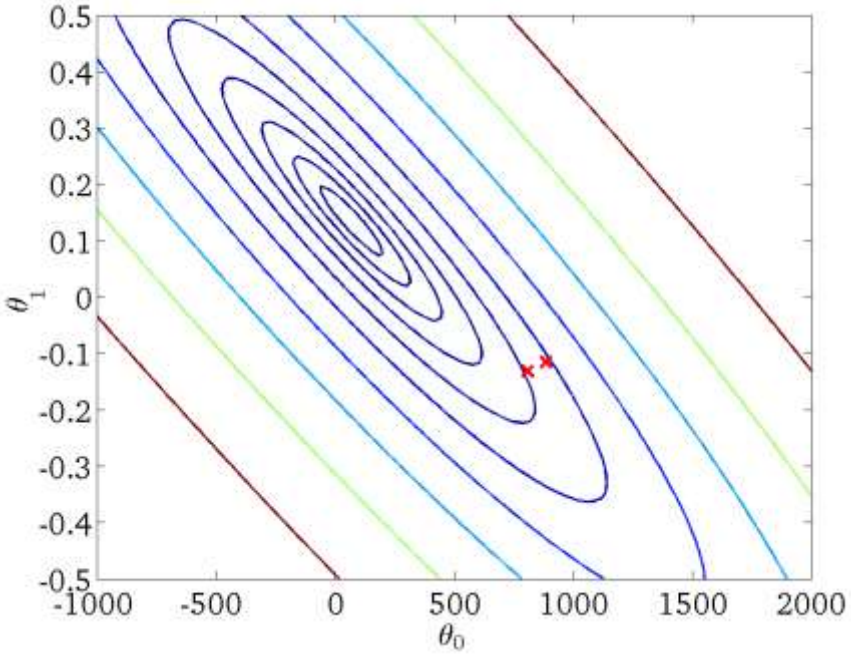
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



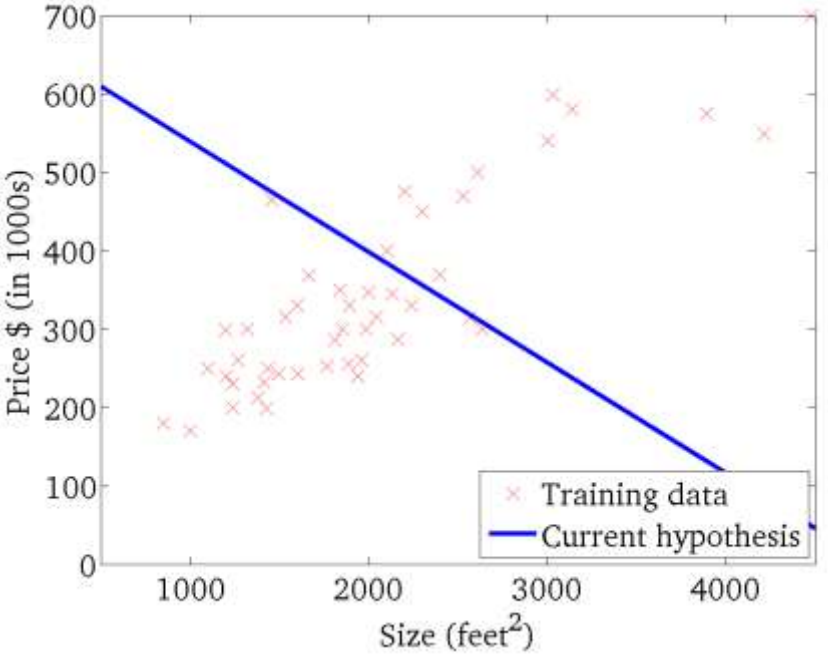
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



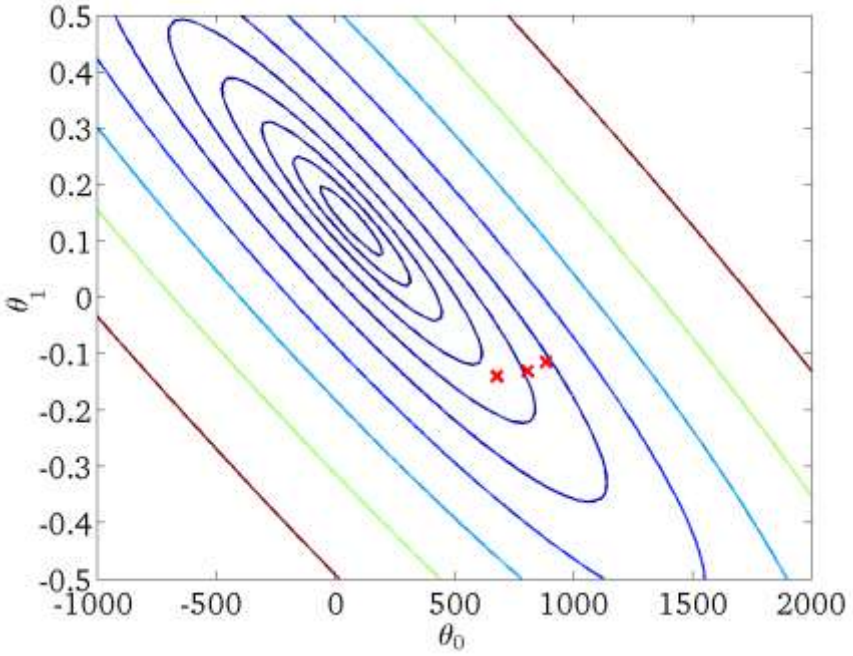
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



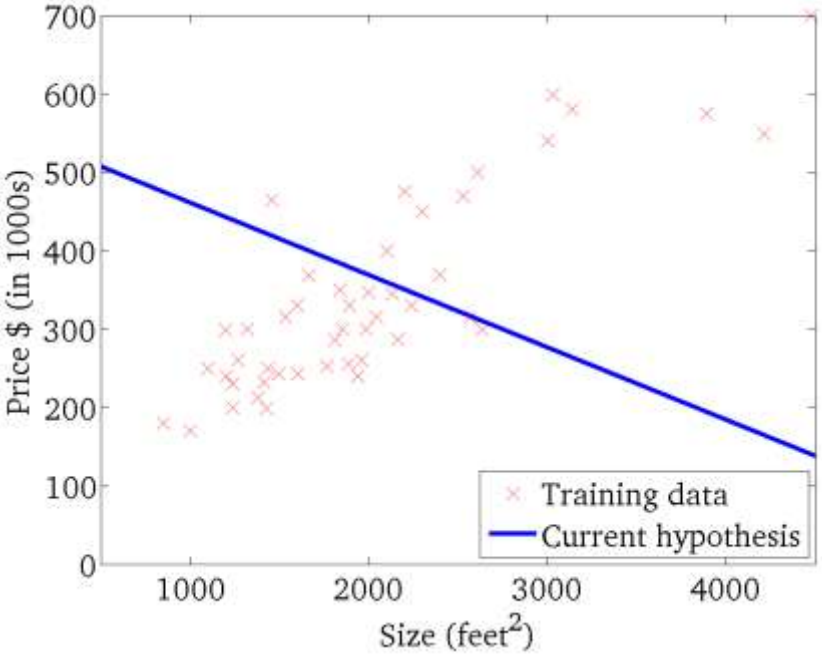
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



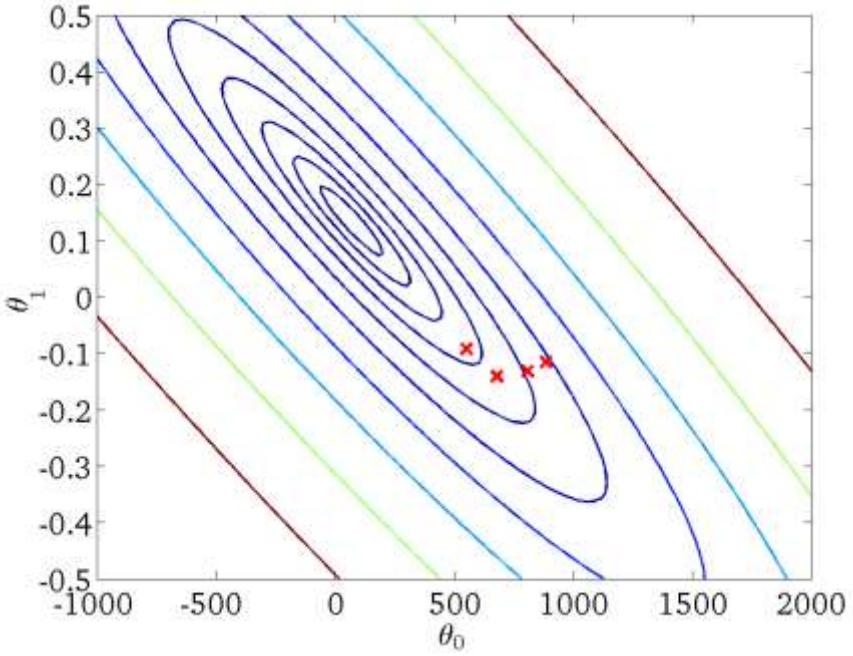
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



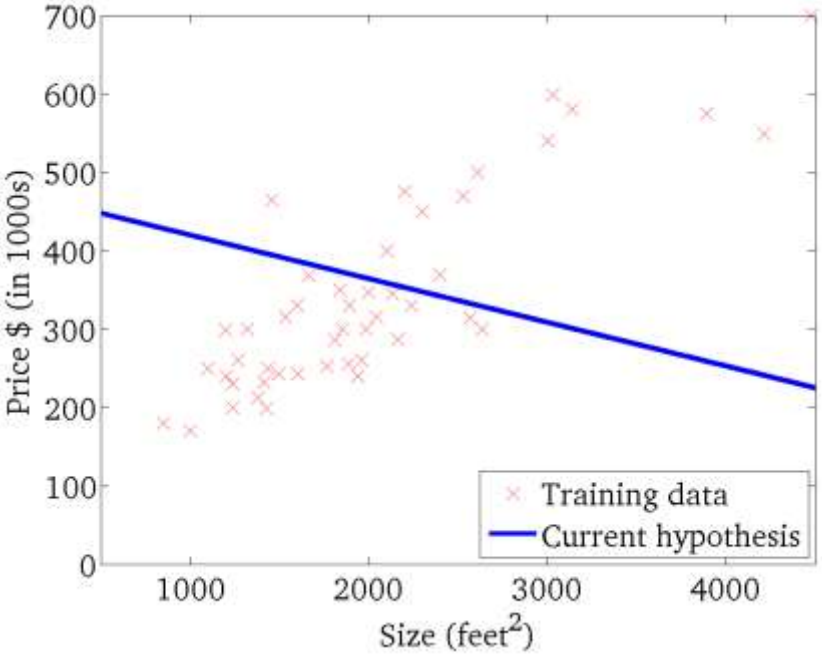
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



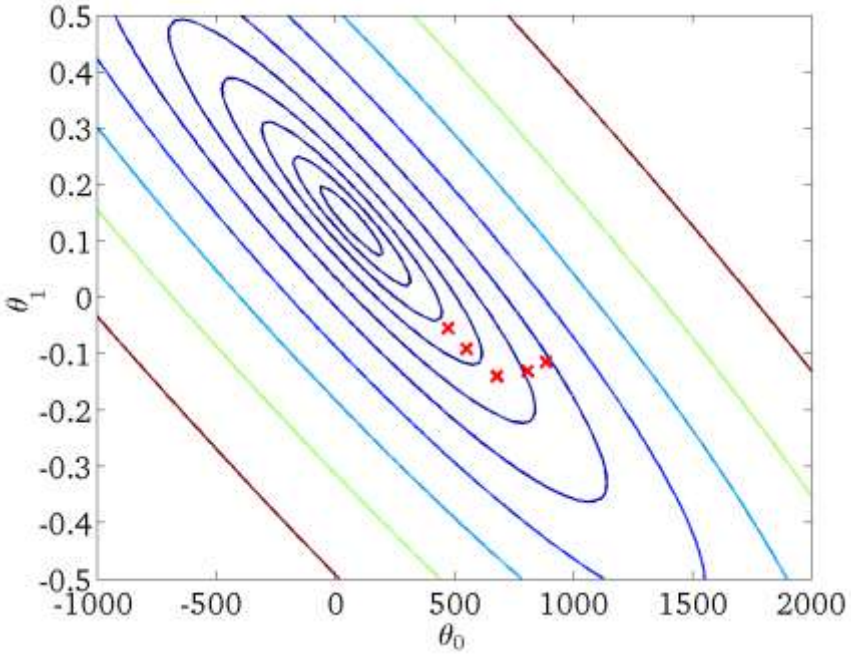
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



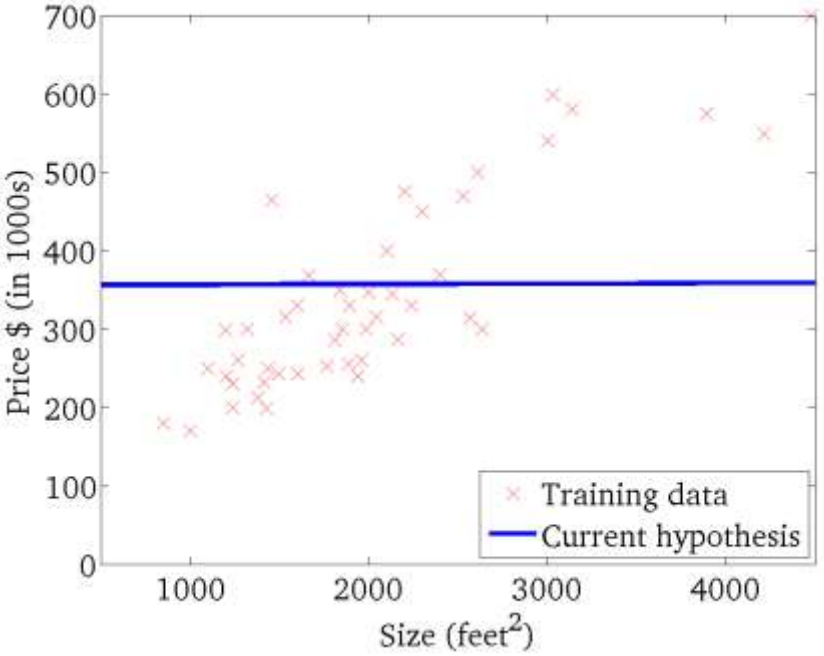
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



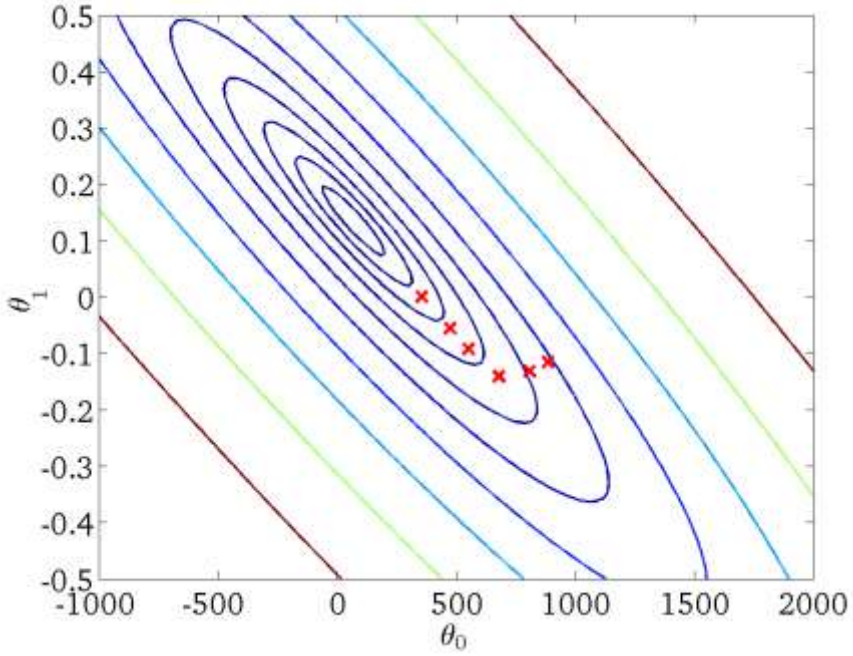
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



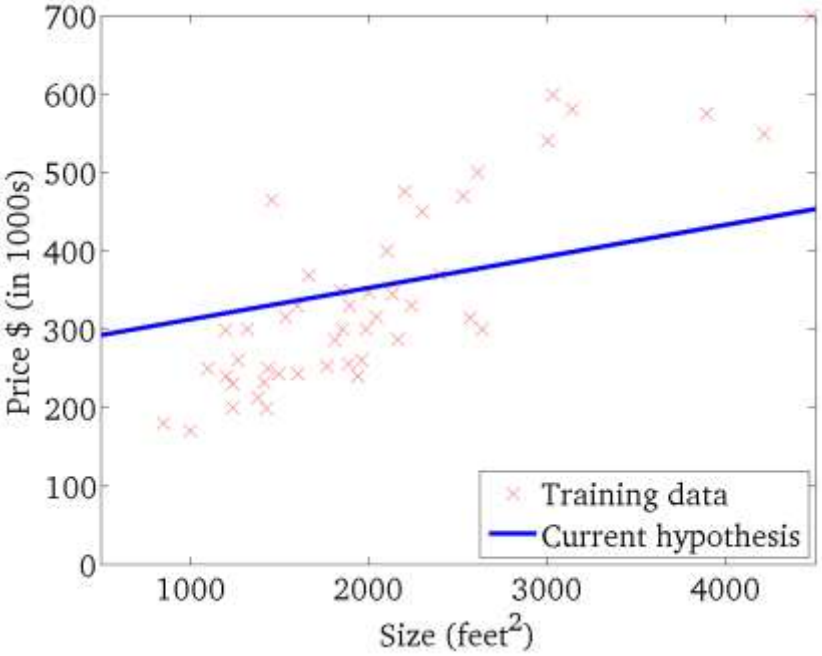
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



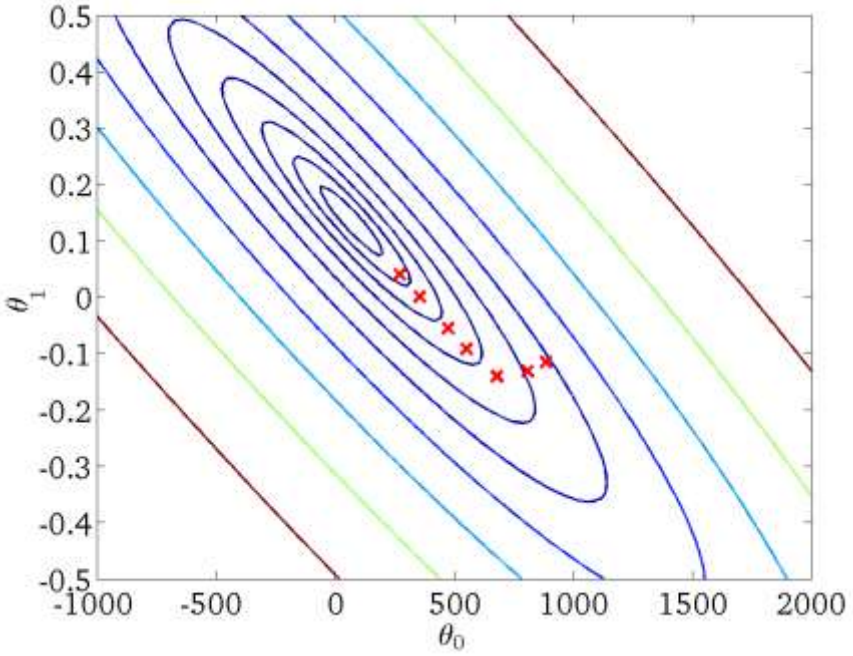
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



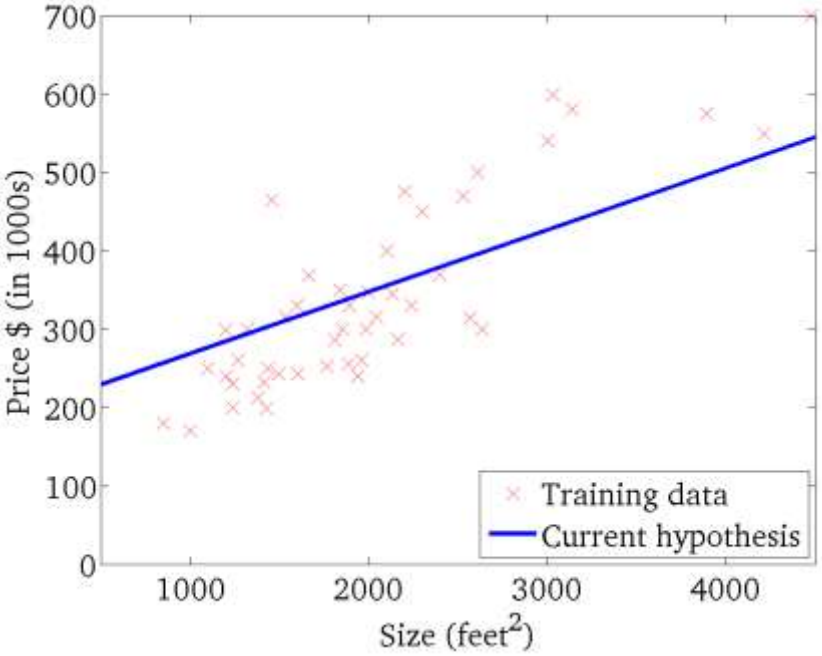
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



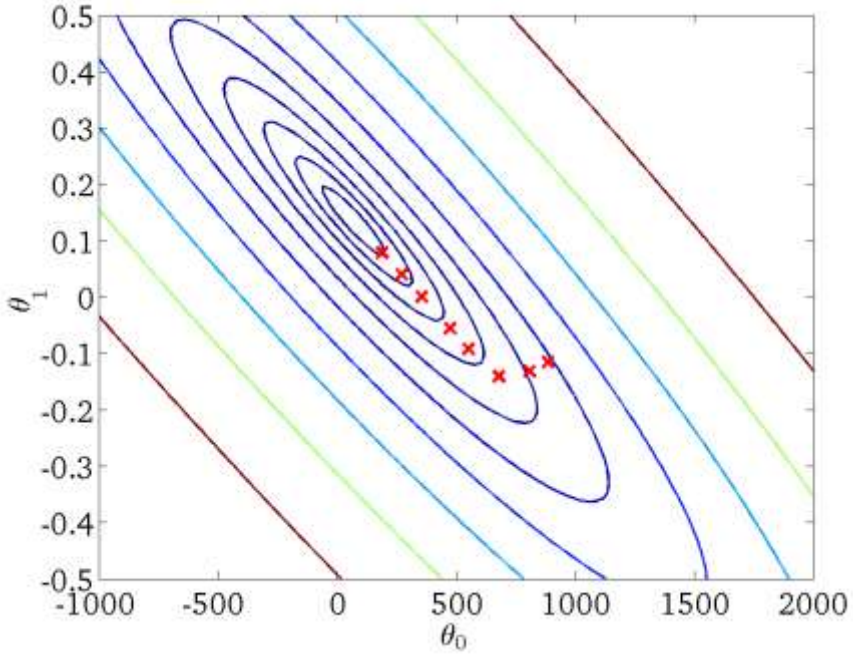
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



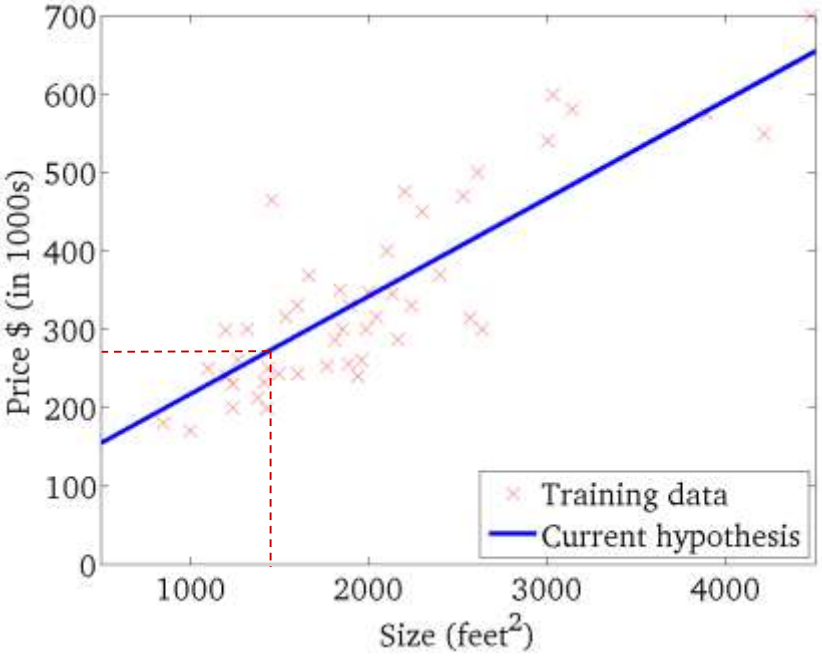
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



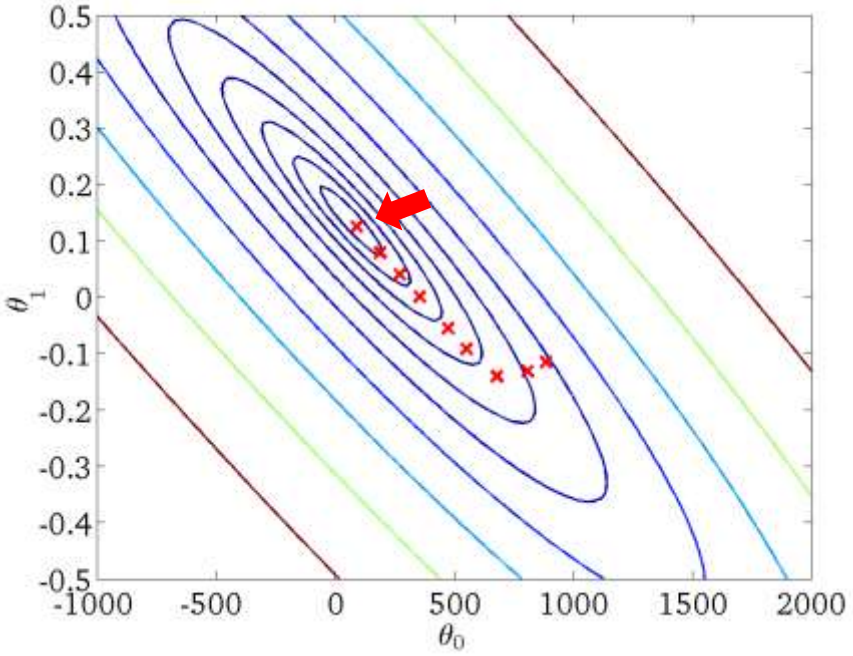
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



“Batch” Gradient Descent

“Batch”: Each step of gradient descent uses all the training examples.

L2.2 - Cost / Function and Optimization Techniques

The supervised m/c learning algorithms learn using a function called the loss fn / cost fn / Error fn. This is a function of the predicted output and the desired output.

If $h(x_i)$ is the predicted o/p,

y_i is the desired / actual o/p, then

the loss function is

$$L = \frac{1}{n} \sum_{i=1}^{n} [h(x_i) - y_i]^2 \quad \text{--- (i)}$$

where 'n' is the no. of records. For regression models, (SSE) Sum of Squared error is the loss function. The function 'L' defined in equation (i) is the SSE.

∴ The objective of a supervised M/L algorithm is to learn the values of the model parameters / model coefficients / feature weights that minimizes the cost function

The ML algorithms use optimization techniques for minimizing the loss functions. The most widely used optimization technique is Gradient Descent.

The Gradient Descent Algorithm for Regression

The functional form of a SLR model is given by:-

$$y_i = mx_i + b + \varepsilon_i$$

where b is the intercept and m is the slope or feature weight, ε_i is the error in prediction.

The predicted value y_i is written as $\hat{y_i} = \hat{m} x_i + \hat{b}$ where $\hat{m}$ and $\hat{b}$ are the estimated values of the parameters.

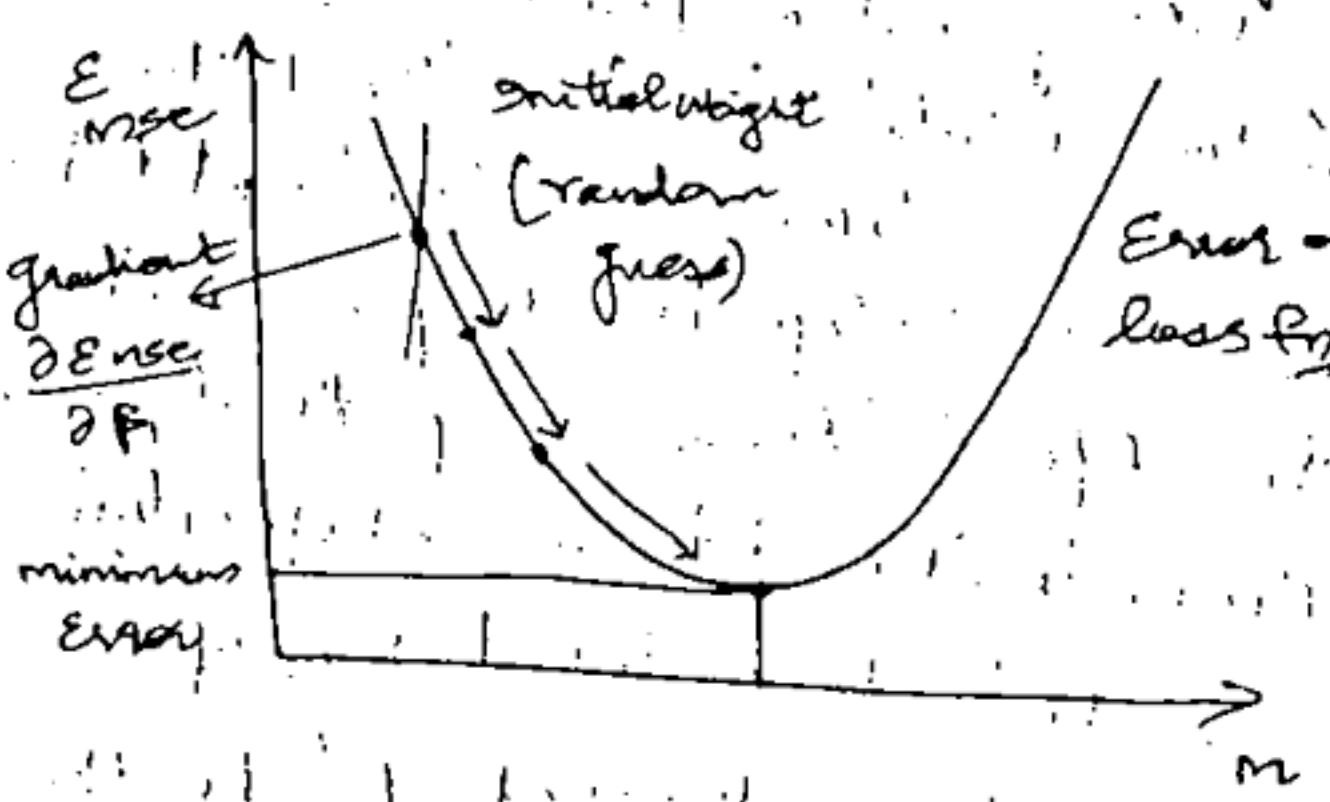
The Error is given by,

$$\varepsilon_i = y_i - \hat{y_i} = (y_i - \hat{b} - \hat{m} x_i)$$

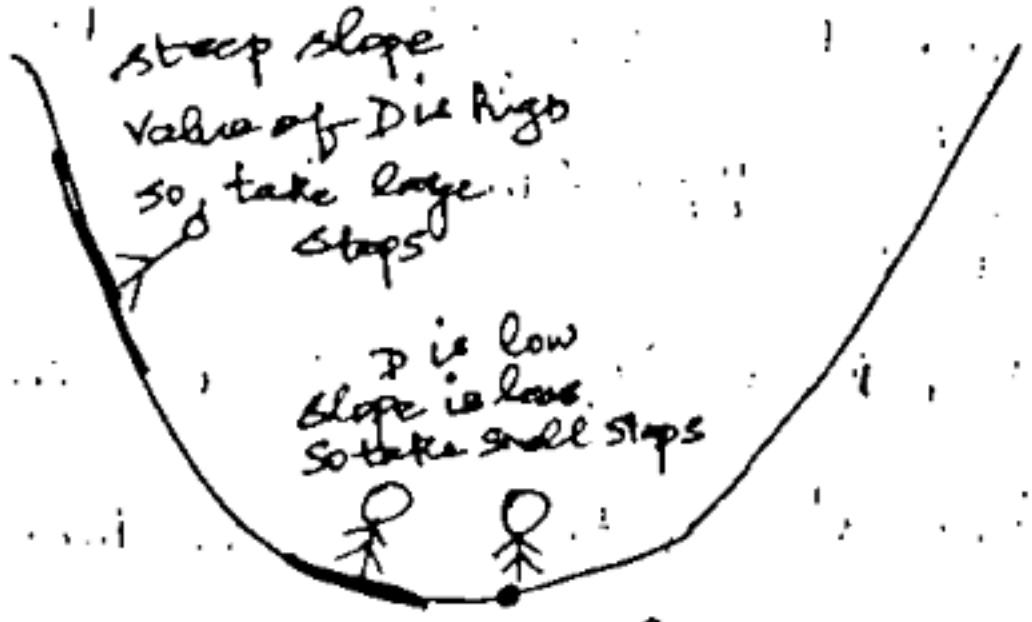
The cost function for linear regression is the mean-squared error MSE given by

$$MSE = \frac{1}{n} \sum_{i=1}^{n} \left[y_i - (mx_i + b) \right]^2$$

Error is a function of b and m and it is a pure convex function, which has a global minimum as in the figure below -

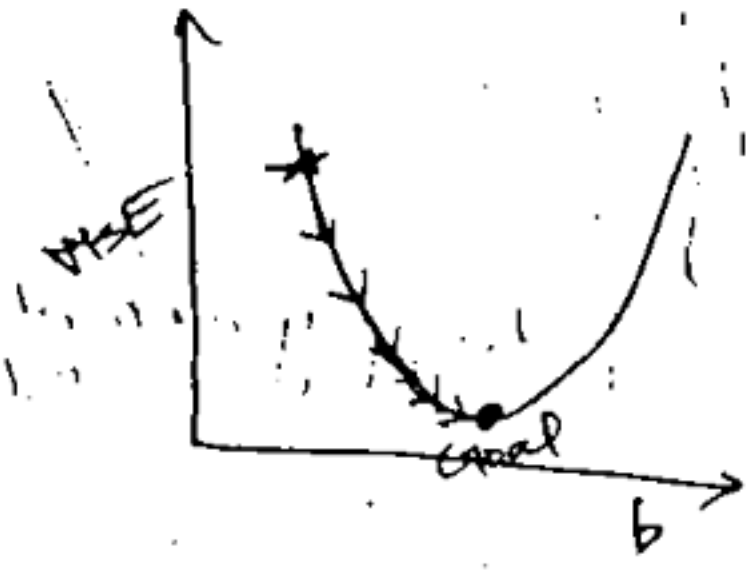


Gradient Descent is an iterative optimization algorithm to find the minimum of a function. Here that function is our Loss function.



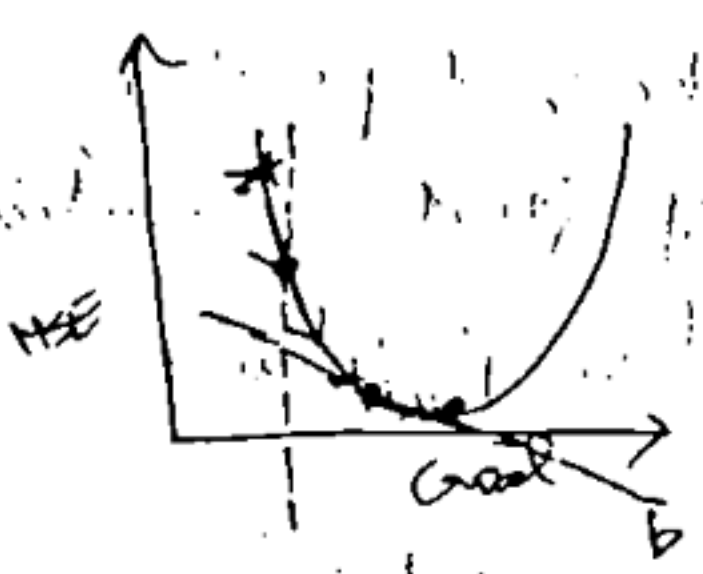
Imagine a valley and a person with no sense of direction wants to get to the bottom of the valley. He goes down the slope and takes large steps when the slope is steep & small steps when the slope is less steep. He decides his next position based on his current position and stops when he gets to the bottom of the valley which was his goal. Let 'L' be the speed at which he moves called the 'Learning Rate'. If he takes fixed steps he will miss his goal.

The "size" of the steps taken should depend on the curvature. When he is



far away from the slope, he should take longer steps, and when he is near the goal, he should take smaller steps.

One way to decide the size of the steps and their direction is to find the slope/ gradient / derivative at each point.



The slope at each point is the partial derivative of 'b' and 'm', with respect to the cost function.

Examples of derivatives :-

To find derivative of a function with one variable, $f(x) = x^3$ is given by

$$\therefore \frac{d}{dx} x^3 = 3x^2$$

The derivative of a function with two variables, $f(x,y) = x^3 + y^2$

Partial derivative of f with respect to x is

$$\partial f/\partial x = 3x^2 + 0 \quad [\text{keep y as constant}]$$
$$= 3x^2$$

Partial derivative of f with repat to y is

$$\partial f/\partial y = 0 + 0 + 2y = 2y \quad [\text{keep x as constant}]$$

To find the size of the steps for gradiat descent, find the partial derivatives of the Loss function with respect to m and b.

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - (mx_i + b))^2$$

$$\therefore \partial/\partial m = -\frac{2}{n}\sum_{i=1}^{n} x_i (y_i - (mx_i + b))$$

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

The value of m and b at this point are the optimum values.

Example -1 :-

Assume I/P $x = [1, 2, 3, 4, 5]$

O/P $y = [5, 7, 9, 11, 13]$

find the prediction function for y using x.

Our actual prediction is $y = 2x + 3$

$\downarrow$ m $\downarrow$ b

∴ The optimum values given by the Gradient Descent should be almost $m = 2$ and $b = 3$.

Refer Lab6 - Gradient - DescentEx1.ipynb for the implementation.

Ex: Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

Weights of the training records.

Round	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

Decision stump is the base classifier.

Round	Split point	Left class	Right class	α
1	0.75	-1	1	1.738
2	0.05	1	1	2.7784
3	0.3	1	-1	4.1195

Details of Round 1:-

Original training set:-

x	y	wt	P(x)	
0.1	1	0.1	-1	x
0.2	1	0.1	-1	x
0.3	1	0.1	-1	x
0.4	-1	0.1	-1	
0.5	-1	0.1	-1	
0.6	-1	0.1	-1	
0.7	-1	0.1	-1	
0.8	+1	0.1	1	
0.9	1	0.1	1	
1	1	0.1	1	

$\therefore$ 3 wrong predictions

$\therefore$

$$\varepsilon_i = \frac{1}{10}\left[3 \times 0.1\right]$$

$$= 0.03$$

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-0.03}{0.03}\right)$$

$$= \frac{1}{2}\ln(32.33)$$

$$= \frac{1}{2} \times 3.48$$

$$= \boxed{1.738}$$

wt updation:

$\alpha = 1.738$

$\exp(1.738) = 5.69$

x	WP/CP	old wt	New wt =	Normal wt
0.1	X	0.1	0.1 × 5.69	0.311
0.2	X	0.1	0.1 × 5.69	
0.3	X	0.1	0.1 × 5.69	
0.4	✓	0.1	0.1 × 0.176	0.01
0.5	✓	0.1	0.1 × 0.176	
0.6	✓	0.1	0.1 × 0.176	
0.7	✓	0.1	"	
0.8	✓	0.1	"	
0.9	✓	0.1	"	
1	✓	0.1	"	

0.569

0.0176 = 0.02

α	Exp(α)	Exp($-\alpha$)
1.738	5.69	0.17587

Normalizing the wts:

Sum of the wts. $3 \times 0.569 + 7 \times 0.0176$

$= 1.707 + 0.1232$

$= 1.8302$

Divide each wt by the sum of wts.

Combining classifiers

Round	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
1	-1	-1	-1	-1	-1	-1	-1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
Sum	5.16	5.16	5.16	-3.08	-3.08	-3.08 →		0.397 →		
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.8	0.9	1	1	1
y	1	1	1	-1	-1	1	1	1	1	1

x <= 0.65 ==> y = 1
x > 0.65 ==> y = 1

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.3 ==> y = 1
x > 0.3 ==> y = -1

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

x <= 0.05 ==> y = -1
x > 0.05 ==> y = 1

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1
True Class	1	1	1	-1	-1	-1	-1	1	1	1

Module 3 - Cluster Analysis

From

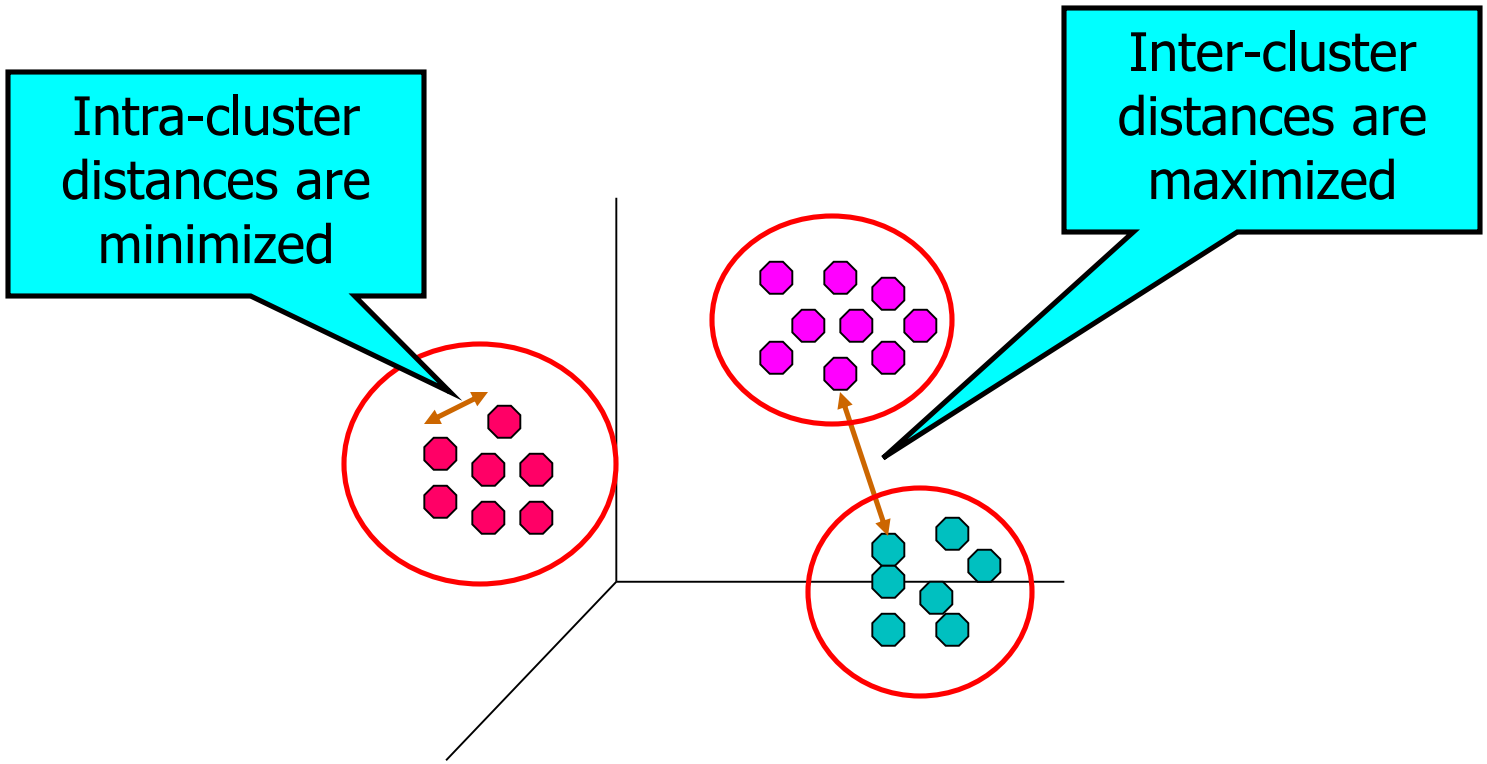
Introduction to Data Mining

by

Tan, Steinbach, Kumar

What is Cluster Analysis?

- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



Clustering

- **Clustering of data is a method by which large sets of data is grouped into clusters of smaller sets of similar data**
- **Objects in one cluster have high similarity to each other and are dissimilar to objects in other clusters**
- **An example of unsupervised learning**
- **Group objects that share common characteristics**

Examples of Clustering Applications

- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- Land use: Identification of areas of similar land use in an earth observation database
- Insurance: Identifying groups of motor insurance policy holders with a high average claim cost
- City-planning: Identifying groups of houses according to their house type, value, and geographical location
- Earth-quake studies: Observed earth quake epicenters should be clustered along continent faults

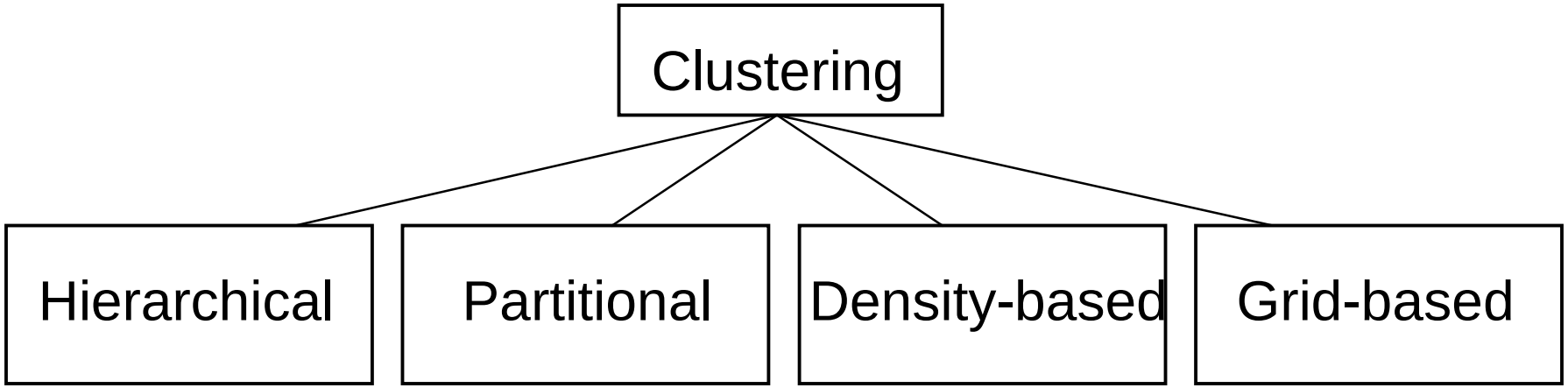
What Is Good Clustering?

- A good clustering method will produce high quality clusters with
  - high intra-class similarity
  - low inter-class similarity
- The quality of a clustering result depends on both the similarity measure used by the method and its implementation.
- The quality of a clustering method is also measured by its ability to discover some or all of the hidden patterns.

Requirements of Clustering in Data Mining

- Scalability
- Ability to deal with different types of attributes
- Discovery of clusters with arbitrary shape
- Minimal requirements for domain knowledge to determine input parameters
- Able to deal with noise and outliers
- Insensitive to order of input records
- High dimensionality
- Incorporation of user-specified constraints
- Interpretability and usability

Clustering Approaches

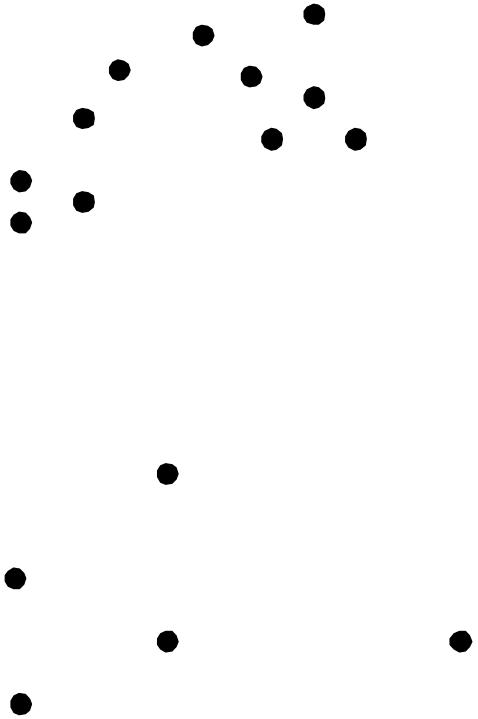


Partitioning Methods

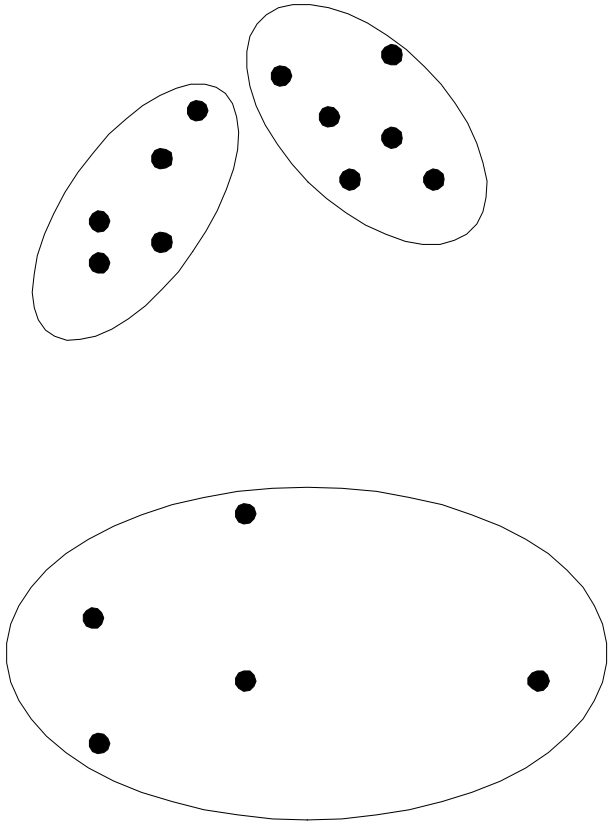
Given a DB of *n* objects, a partitioning method constructs *k* partitions of the data, where each partition represents a cluster and *k*<=*n* such that

1. Each group must contain atleast one object, and
2. Each object must belong to exactly one group

Partitional Clustering



Original Points



A Partitional Clustering

Density-based Methods

- Most partitioning-based methods cluster objects based on distances between them
- Can find only spherical-shaped clusters
- Density-based clustering
- Continue growing a given cluster as long as the density in the ‘neighborhood’ exceeds some threshold.

Hierarchical Clustering

- Hierarchical partition of the objects into clusters.
- It doesn't require the no. of clusters as input.
- Need to specify a termination condition.
- A dendrogram shows how the clusters are merged/split hierarchically.

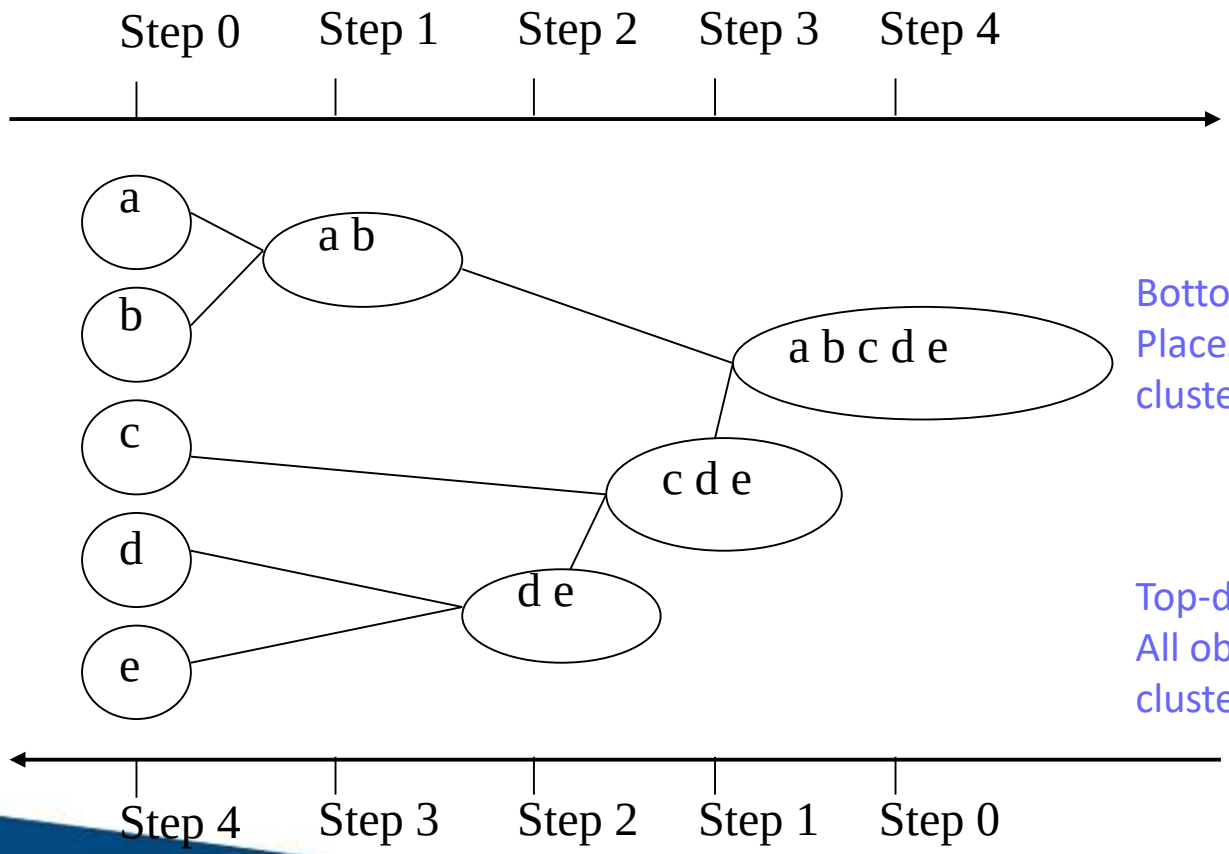
a) Agglomerative : (Bottom Up)

- Start with each object in a cluster.
- Combine similar clusters into larger and larger clusters.

b) Divisive: (Top Down)

- Start with one large cluster and split into smaller and smaller clusters

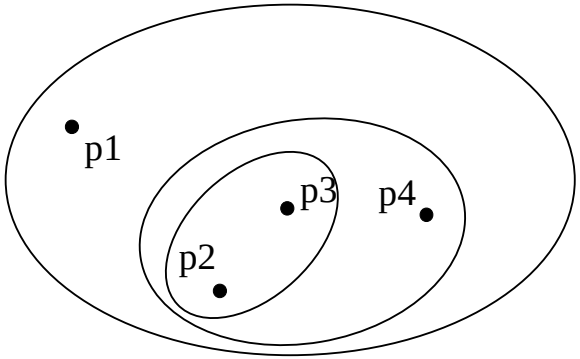
Hierarchical Clustering



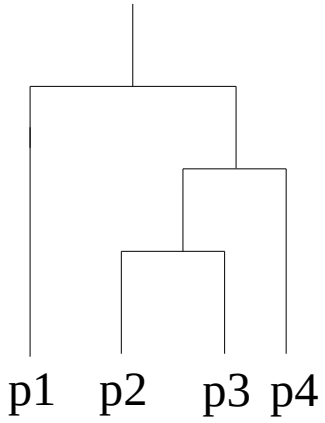
Bottom-up
Places each object in its own cluster & then merges

Top-down
All objects in a single cluster & then subdivides

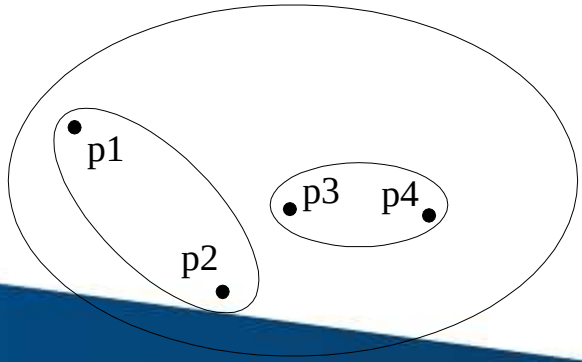
Hierarchical Clustering



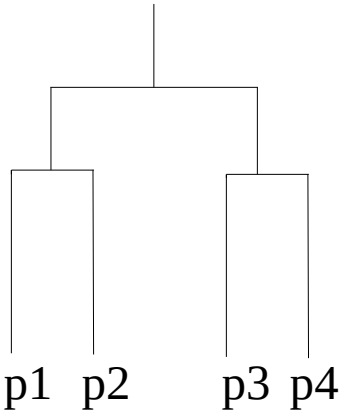
Traditional Hierarchical Clustering



Traditional Dendrogram



Non-traditional Hierarchical Clustering



Non-traditional Dendrogram

Partitioning Algorithms: Basic Concept

- Partitioning method: Construct a partition of a database *D* of *n* objects into a set of *k* clusters
- Given a *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic methods: *k-means* and *k-medoids* algorithms
    - *k-means* (MacQueen'67): Each cluster is represented by the center of the cluster
    - *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

K-means Clustering

1. Select the number of clusters, **k**.
2. Pick **k** seeds randomly as centroids of the **k** clusters.
3. Compute the Euclidean distance of each object from each of the centroids.
4. Allocate each object to its nearest cluster.
5. Compute the new centroids of the clusters
   - Mean of the attribute values of the objects in each cluster is the cluster centroid.
6. Check if the stopping condition has been met. If yes, stop else go to step 3.
   - No change in the cluster membership.

K-means Example 1

- For simplicity, 1-dimension objects and k=2.
  - Numerical difference is used as the distance
- Objects: 1, 2, 5, 6,7
- K-means:
  - Randomly select 5 and 6 as centroids;
  - End of Iteration 1 : Two clusters {1,2,5} and {6,7}; meanC1=8/3, meanC2=6.5
  - End of Iteration 2 : {1,2}, {5,6,7}; meanC1=1.5, meanC2=6
  - End of Iteration 3 : no change.
- EXAMPLE WORKED IN PDF

K-means Clustering – Details

- Initial centroids are often chosen randomly.
  - Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- 'Closeness' is measured by Euclidean distance, cosine similarity, correlation, etc.
- K-means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to 'Until relatively few points change clusters'
- Complexity is O(n * K * I * d)
  - n = number of points, K = number of clusters,
    I = number of iterations, d = number of attributes

Evaluating K-means Clusters

- Most common measure is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest cluster
  - To get SSE, we square these errors and sum them.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

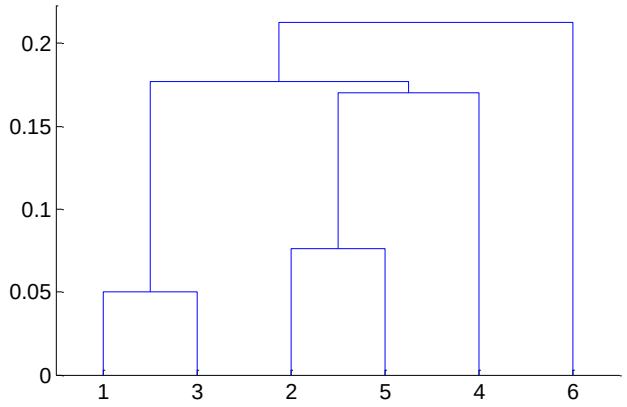
  - x is a data point in cluster C_i and m_i is the representative point for cluster C_i
    - can show that m_i corresponds to the center (mean) of the cluster
  - Given two clustering results, we can choose the one with the smallest error
  - One easy way to reduce SSE is to increase K, the number of clusters
    - A good clustering with smaller K can have a lower SSE than a poor clustering with higher K

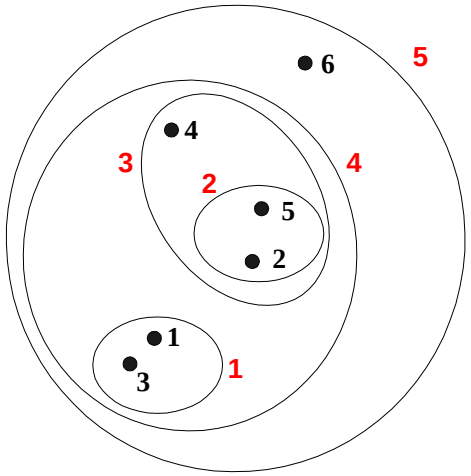
Updating Centers Incrementally

- In the basic K-means algorithm, centroids are updated after all points are assigned to a cluster
- An alternative is to update the centroids after each assignment of a point to a cluster (incremental approach)
  - all clusters start with a single point.
  - Each assignment updates zero or two centroids
  - Never get an empty cluster
  - More expensive
  - Introduces an order dependency

Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
  - A tree like diagram that records the sequences of merges or splits





Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by ‘cutting’ the dendogram at the proper level

Hierarchical Clustering

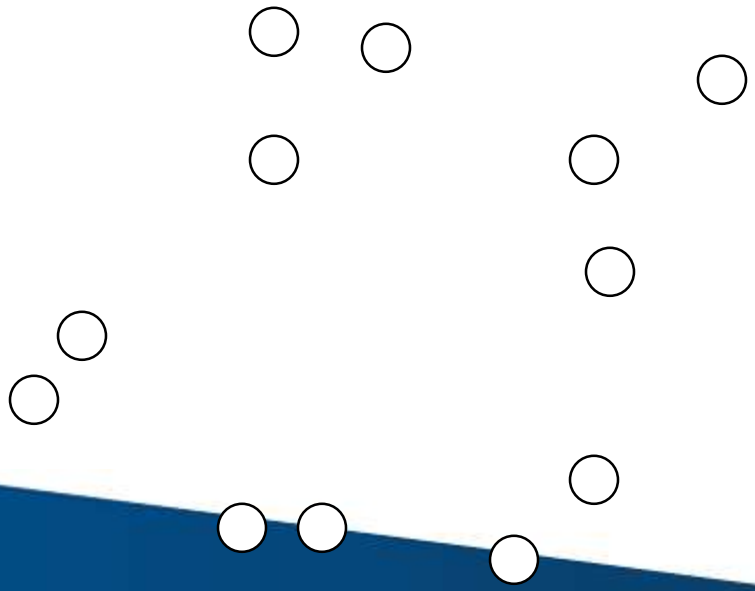
- Two main types of hierarchical clustering
  - Agglomerative:
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
  - Divisive:
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are k clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
  - Merge or split one cluster at a time

Agglomerative Clustering Algorithm

- More popular hierarchical clustering technique
- Basic algorithm is straightforward
  1. Compute the proximity matrix
  2. Let each data point be a cluster
  3. **Repeat**
  4. Merge the two closest clusters
  5. Update the proximity matrix
  6. **Until** only a single cluster remains
- Key operation is the computation of the proximity of two clusters
  - Different approaches to defining the distance between clusters distinguish the different algorithms

Starting Situation

- Start with clusters of individual points and a proximity matrix



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

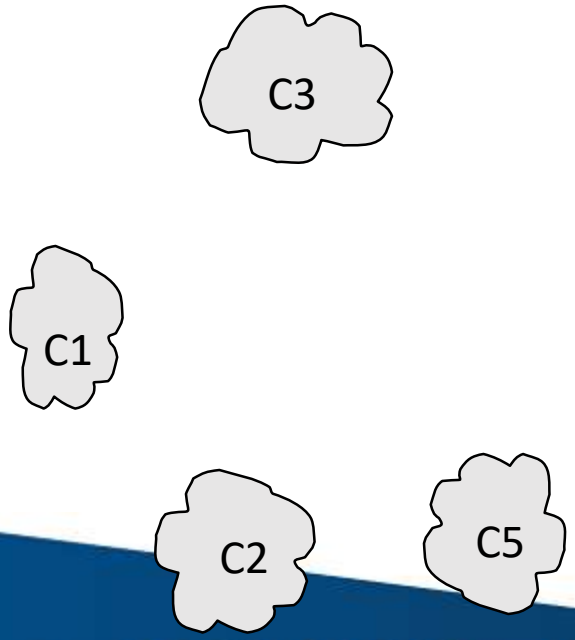
Proximity Matrix

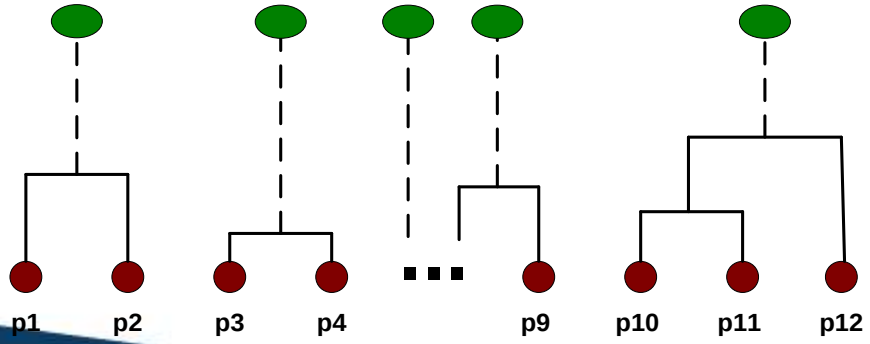
Intermediate Situation

- After some merging steps, we have some clusters

	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



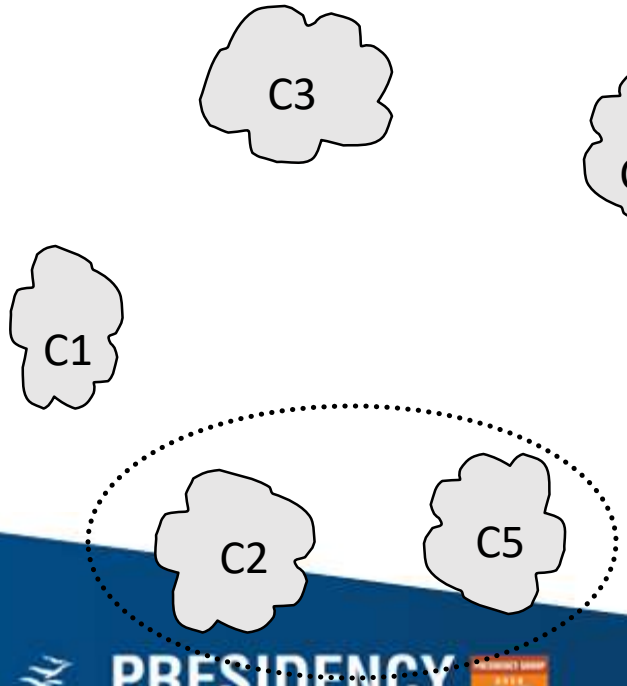


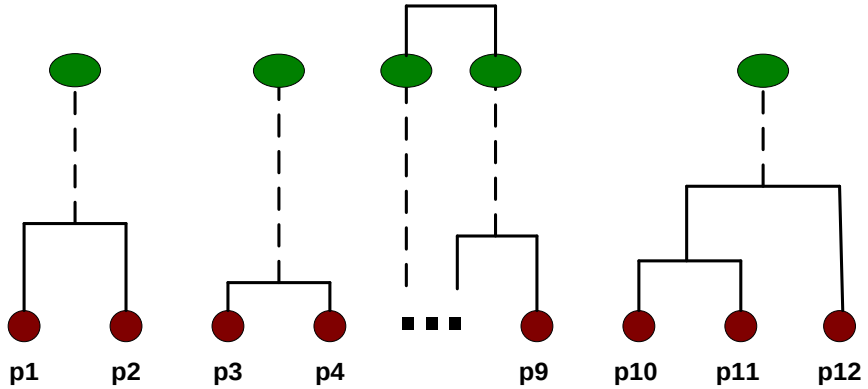
Intermediate Situation

- We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.

	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

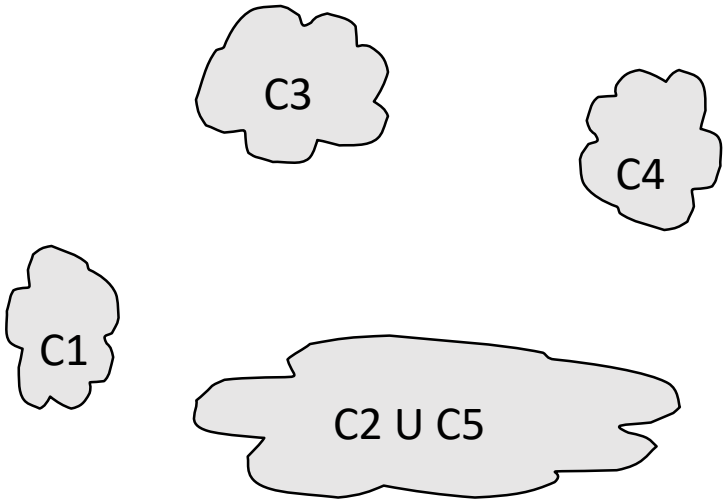
Proximity Matrix





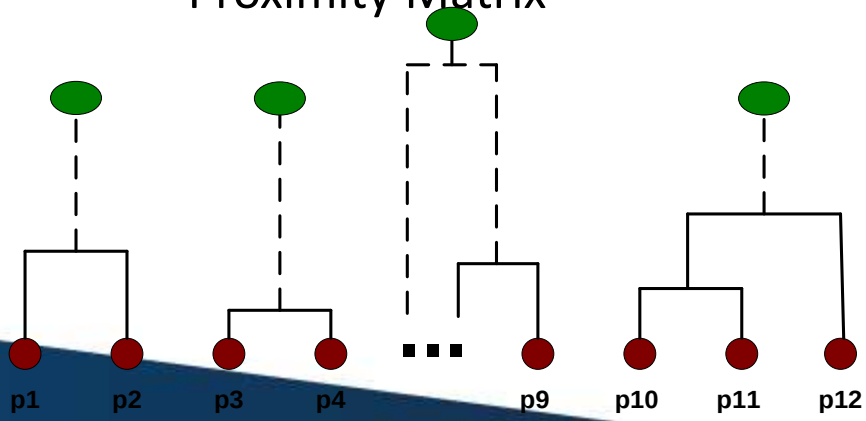
After Merging

- The question is "How do we update the proximity matrix?"

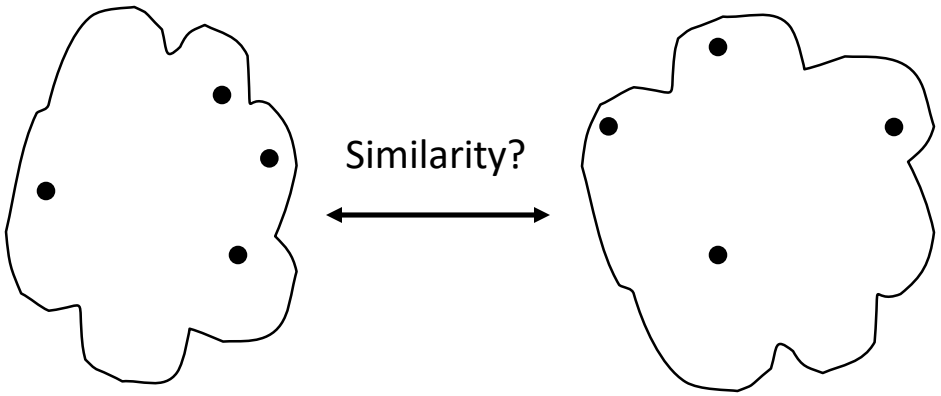


	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



How to Define Inter-Cluster Similarity

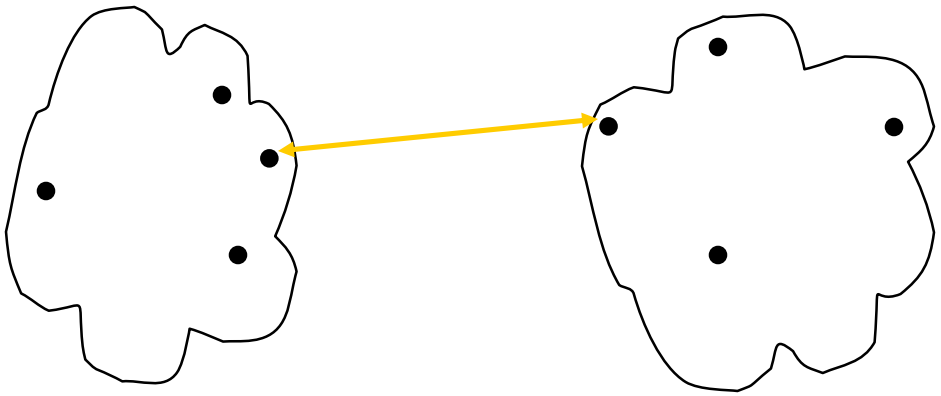


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity

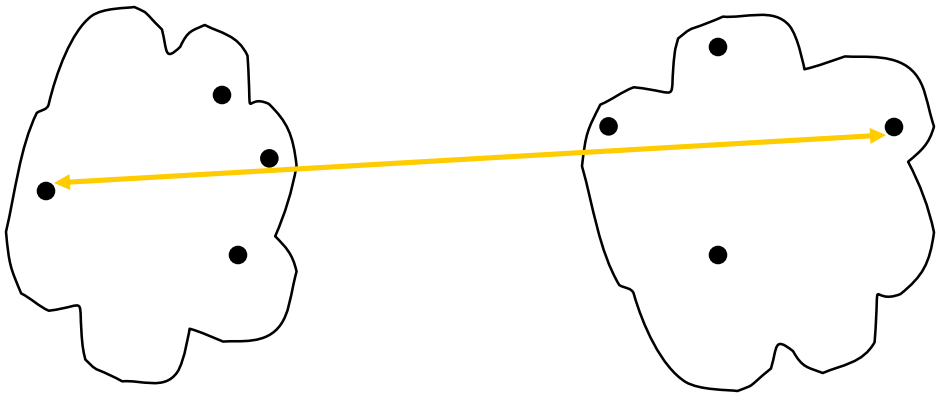


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity

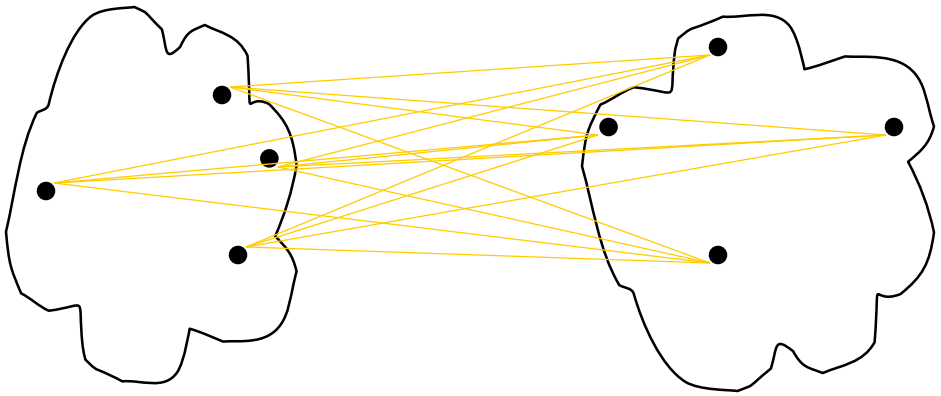


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity

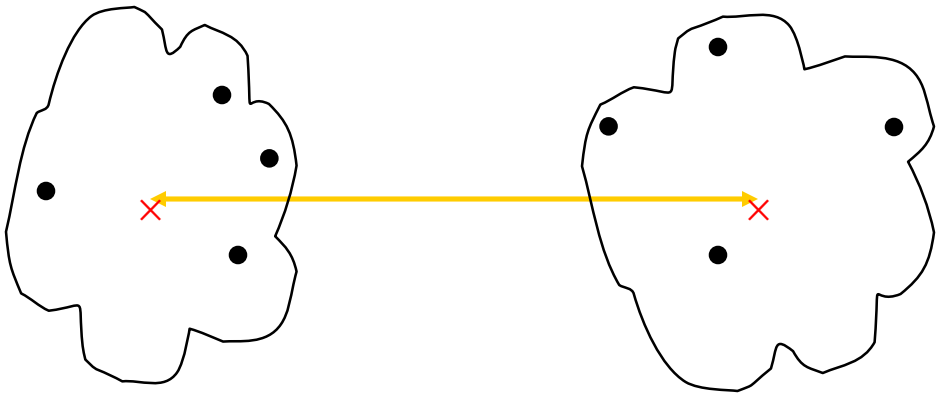


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

- NUMERICAL EXAMPLES IN HANDWRITTEN NOTES

Cluster Validity – SSE - Cohesion

- SSE = average pairwise distance between the points in a cluster.
- Cluster SSE = $\sum_{x \in C_i} dist(C_i, x)^2$

$$= \frac{1}{2m_i} \sum_{x \in C_i} \sum_{y \in C_i} dist(x, y)^2$$

Where m_i is the centroid of cluster i

x and y are points in cluster i.

- measures how closely related are points in a cluster.

Internal measures – Separation

- How distinct or well separated is a cluster from other clusters.
- **Ex: cohesion** :- within cluster sum of squares

$$\text{WSS} = \sum_i \sum_{x \in C_i} (x - m_i)^2$$

Separation : between clusters sum of squares

$$\text{BSS} = \sum |C_i| \, (m - m_i)^2$$

where $|C_i|$ is the size of cluster i

m is the overall centroid.

External measures: Entropy and purity

- **Entropy of a cluster C_j:-**
- Find the probability of each class i in C_j, where

$$p_{ij} = \frac{m_{ij}}{m_j}$$

where m_j = no. of values in cluster C_j.

m_{ij} = no. of values of class i in cluster C_j.

Entropy of C_j: –

$$e_i = \sum_{i=1}^{L} p_{ij} \log_2 p_{ij}$$

Entropy of a clustering with k no. of clusters

- $e = \sum_{i=1}^{k} \frac{m_j}{m} e_j$

Where e_j = entropy of j^{th} cluster

m_j = size of the j^{th} cluster

m = total no. of data points.

Purity :-

1) Purity of a cluster $\boldsymbol{C_j}$:-

$$purity_j = \max P_{ij}$$

2) Purity of a clustering:-

$$\text{purity} = \sum_{i=1}^{k} \frac{m_i}{m} purity_i$$

K Means Clustering for multi-Dimensional Data

Example : Given Data and k =2

Sample No.	X	Y
1	185	72
2	170	56
3	168	60
4	179	68
5	182	72
6	188	77

Step 1: Initial Centroid

Cluster	X	Y
k1	185	72
k2	170	56

Step 2: Consider Euclidean distance for calculating distances.

Distance[(x,y),(a,b)] = sqrt[$(x-a)^2+(y-b)^2$]

Iteration 1:

Step 3: Find the Euclidean distance for the Sample no 1 (185,72)

Distance from Cluster 1(185,72) = sqrt[$(185-185)^2+(72-72)^2$] = **0**

Cluster-2(170,56) = sqrt[$(170-185)^2+(56-72)^2$]= sqrt[$(-15)^2+(-16)^2$]= sqrt[255 + 256]

= sqrt[481] = **21.93**

Find the Euclidean distance for the Sample no 2 (170,56)

Cluster-1(185,72) = sqrt[$(185-170)^2+(72-56)^2$]= sqrt[$(15)^2+(16)^2$]= sqrt[255 + 256]

= sqrt[481] = **21.93**

Cluster-2(170,56)= sqrt[$(170-170)^2+(56-56)^2$] = **0**

Step 4 : Cluster Assignment

Sample	Euclidean Distance		Cluster Assignment
	Cluster 1	Cluster 2	
(185,72)	0	21.93	1
(170,56)	21.93	0	2

Step 5: Centroid Update

Cluster	Centroid	
	X	Y
k1	185	72
k2	170	56

Iteration 2: repeat (step 3, 4 & 5)

Step 3 : Calculate Euclidean distance for the next dataset --Sample 3(168,60)

Distance from Cluster 1(185,72) = sqrt[$(168-185)^2$+$(60-72)^2$]= 20.808

Distance from Cluster 2(170,56) = sqrt[$(168-170)^2$+$(60-56)^2$]= 4.472

Step 4: Cluster Assignment

Sample	Euclidean Distance		Cluster Assignment
	Cluster 1	Cluster 2	
(168,60)	20.808	4.472	2

Step 5: Centroid Update

Cluster	Centroid	
	X	Y
k1	185	72
k2	=(170+168)/2 =169	=(56+60)/2 =58

Iteration 3: repeat (step 3, 4 & 5)

Step 3: Calculate Euclidean distance for the next dataset- sample 4(179,68)

Distance from Cluster 1(185,72) = sqrt[$(179-185)^2$+$(68-72)^2$]= 7.211103

Distance from Cluster 2(169,58) = sqrt[$(179-169)^2$+$(68-58)^2$]= 14.14214

Step 4: Cluster Assignment

Sample	Euclidean Distance		Cluster Assignment
	Cluster 1	Cluster 2	
(179,68)	7.211103	14.14214	1

Step 5: Centroid Update

Cluster	Centroid	
	X	Y
k1	=(185+179)/2 =182	= (72+68)/2 =70
k2	169	58

Iteration 4: repeat (step 3, 4 & 5)

Step 3: Calculate Euclidean distance for the next dataset- sample 5(182, 72)

Distance from Cluster 1(182,70) = sqrt[$(182-182)^2$+$(72-70)^2$]= 2

Distance from Cluster 2(169,58) = sqrt[$(182-169)^2$+$(72-58)^2$]= 19.10

Step 4: Cluster Assignment

Sample	Euclidean Distance		Cluster Assignment
	Cluster 1	Cluster 2	
(182,72)	2	19.10	1

Step 5: Centroid Update

Cluster	Centroid	
	X	Y
k1	=(182+182)/2 =182	=(70+72)/2 =71
k2	169	58

Iteration 5: repeat (step 3, 4 & 5)

Step 3: Calculate Euclidean distance for the next dataset- sample 5(188, 77)

Distance from Cluster 1(182,71) = sqrt[$(188-182)^2$+$(77-71)^2$]= 8.4852

Distance from Cluster 2(169,58) = sqrt[$(188-169)^2$+$(77-58)^2$]= 26.87

Step 4: Cluster Assignment

Sample	Euclidean Distance		Cluster Assignment
	Cluster 1	Cluster 2	
(188,77)	8.4852	26.87	1

Step 5: Centroid Update

Cluster	Centroid	
	X	Y

k1	=(182+188)/2 =185	=(71+77)/2 =74
k2	169	58

Step 6: FINAL ASSIGNMENT TABLE

Sample No.	X	Y	Assignment
1	185	72	1
2	170	56	2
3	168	60	2
4	179	68	1
5	182	72	1
6	188	77	1

Ex: for cluster validity.

Assume we have a test collection D of 900 documents from three topics (or 3 classes) Science, Sports and Politics. Each class has 300 documents. Each document in D is labelled with one of the topic (class). These documents are grouped into 3 clusters. Measure the effectiveness of the clustering.

Cluster	Science	Sports	Politics	→	Total
1	250	20	10		280
2	20	180	80		280
3	30	100	210		340
Total	300	300	300		

Solution:

Step 1: Calculate the total in each cluster.

cluster 1 → 250 + 20 + 10 = 280

Step 2: Find out the probability of each cluster.

Cluster	Science	Sports	Politics	Purity
1	0.893	0.0714	0.035	0.893
2	0.0714	0.643	0.286	0.643
3	0.0882	0.2941	0.6176	0.617
Total	300	300	300	0.711

C1: Prob (Science) = 250/280 = 0.893, Prob (Politics) = 0.0357

Prob (Sports) = 20/280 = 0.0714

Step 3: Calculation of Purity, by considering the Maximum probability. = Max (Prob).

C1 : Max (0.893, 0.0714, 0.035) = 0.893

Similarly for all other clusters.

Step 4:
Purity of the clustering = $\sum_{i=1}^{3} \frac{m_i}{m}$ Purity.

$$= \left[\frac{280}{900} \times 0.893 + \frac{280}{900} \times 0.643 + \frac{340}{900} \times 0.617\right]$$

$$= 0.277 + 0.2000 + 0.2330$$

$$= 0.711$$

Step 5: Calculate the Entropy of each cluster.

$$C1 : -\left(\frac{250}{280}\log_2\frac{250}{280} + \frac{20}{280}\log_2\frac{20}{280} + \frac{10}{280}\log_2\frac{10}{280}\right)$$

$$= -(0.893\log_2 0.893 + 0.0714\log_2 0.0714 + 0.035 + \log_2 0.035)$$

$$= -(-0.14579 + 0.271886 - 0.1692775)$$

$$= 0.587$$

Similarly for all the clusters calculate the Entropy.

Prob	Science	Sports	Politics	Entropy	Purity.
1	0.893	0.0714	0.035	0.587	0.893
2	0.0714	0.643	0.286	1.198	0.643
3	0.088	0.294	0.617	1.257	0.617
Total	300	300	300	1.0301	0.711

$$-\left(\frac{20}{280}\log_2\frac{20}{28} + \frac{180}{280}\log_2\frac{180}{280} + \frac{80}{280}\log_2\frac{80}{280}\right)$$

$$-(0.071\log_2 0.071 + 0.642\log_2 0.642 + 0.285\log_2 0.285)$$

$$-(-0.27093 - 0.41046 - 0.51612) = 1.198$$

$$-\left(\frac{30}{340}\log_2\frac{30}{340} + \frac{100}{340}\log_2\frac{100}{340} + \frac{210}{340}\log_2\frac{210}{340}\right)$$

$$-(0.0882\log_2 0.0882 + 0.294\log_2 0.294 + 0.617\log_2 0.617)$$

$$= -(-0.30897 - 0.5192 - 0.4298)$$

$$= 1.257$$

Step 6: Calculate the entropy of clustering.

$$= \sum_{i=1}^{k} \frac{m_i}{m} e_i$$

$$= \frac{280}{900} \times 0.587 + \frac{280}{900} \times 1.198 + \frac{340}{900} \times 1.257$$

$$= 0.1826 + 0.3727 + 0.4748$$

$$= 1.0301$$

Data Mining
Association Analysis: Basic Concepts and Algorithms

Lecture Notes for Chapter 6

Introduction to Data Mining

by

Tan, Steinbach, Kumar

Why Mining Association Rules?

- Objective:
  - Finding interesting co-occurring items (or objects, events) in a given data set.
- Examples:
  - Given a database of transactions, each transaction is a list of items (purchased by a customer in a visit), you may find:

    computer → financial_management_software
    [support=2%, confidence=60%]
  - From a student database, you may find
    - major(x, "CS") ^ gpa(x, "A") → has_taken(x, "DB") [1%, 75%]

What Kind of Databases?

Transaction database TDB

TID	Items
100	f, a, c, d, g, i, m, p
200	a, b, c, f, l,m, o
300	b, f, h, j, o
400	b, c, k, s, p
500	a, f, c, e, l, p, m, n

- Itemset: a set of items
- A *transaction* is a tuple (tid, X)
  - Transaction ID tid
  - Itemset X
- A *transaction database* is a set of transactions
  - In many cases, a transaction database can be treated as a set of itemsets (ignore TIDs)
- Association rule from TDB (relates two itemsets):
  - {a, c, m} → {l} [support=40%, confidence=66.7%]

Association Rule Mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Association Rules

{Diaper} → {Beer},
{Milk, Bread} → {Eggs,Coke},
{Beer, Bread} → {Milk},

Implication means co-occurrence, not causality!

Definition: Association Rule

- **Association Rule**
  - An implication expression of the form $X \rightarrow Y$, where X and Y are itemsets
  - Example:
    {Milk, Diaper} → {Beer}

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

- **Rule Evaluation Metrics**
  - Support (s)
    - Fraction of transactions that contain both X and Y
  - Confidence (c)
    - Measures how often items in Y appear in transactions that contain X

Example:

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Mining Association Rules

- Problem statement

  Given a *minimum support* (*min_sup*), also called *support threshold*, and a *minimum confidence* (*min_conf*), also called *confidence threshold*, find all association rules that satisfy both *min_sup* and *min_conf* from a data set *D*.

Definition: Frequent Itemset

- **Itemset**
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- **Support count (σ)**
  - Frequency of occurrence of an itemset
  - E.g. σ({Milk, Bread,Diaper}) = 2
- **Support**
  - Fraction of transactions that contain an itemset
  - E.g. s({Milk, Bread, Diaper}) = 2/5
- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a *minsup* threshold

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Mining Association Rules

- Two-step approach:
  1. Frequent Itemset Generation
     - Generate all itemsets whose support $\geq$ minsup
  2. Rule Generation
     - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset
- Frequent itemset generation is still computationally expensive

$\text{confidence}(X \rightarrow Y) = \text{P}(Y|X)$

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Relative frequency is used to estimate the Probability.

- $A \rightarrow C$
- $C \rightarrow A$
- $A \,\&\, C \rightarrow B$
- $A \,\&\, B \rightarrow E$

Support and Confidence: Example

$\text{support}(A \rightarrow B) = \text{P}(A \cup B)$

$\text{confidence}(A \rightarrow B) = \text{P}(B|A)$

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

- $A \rightarrow C$ (50%, 66.7%)
- $C \rightarrow A$ (50%, 100%)
- $A \,\&\, C \rightarrow B$ (25%, 50%)
- $A \,\&\, B \rightarrow E$ (0%, 0%)

How to Mine Association Rules

- A two-step process:
  - Find all frequent itemsets ---- the key step
  - Generate strong association rules from frequent itemsets.
- Example: given min_sup=50% and min_conf=50%

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Frequent Itemset	Support
{A}	75%
{B}	50%
{C}	50%
{A, C}	50%

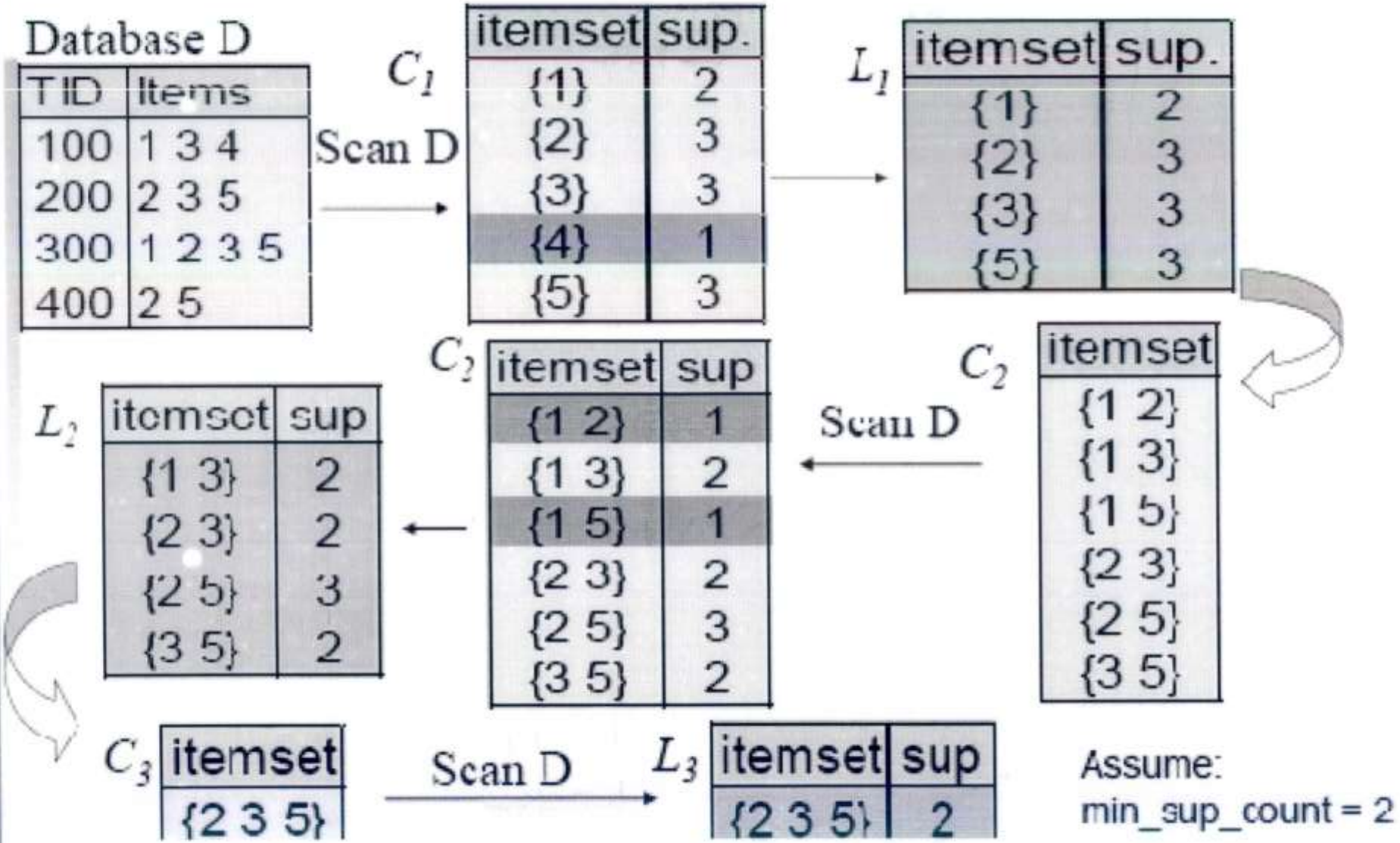
- Generate strong rules:
  - {A} → {C} [support=50%, confidence=66.6%]
  - {C} → {A} [support=50%, confidence=100%]

(12)

Finding Frequent Itemsets

- Objective: find the itemsets that satisfy the minimum support count.
- Algorithms
  - Apriori
  - FP-Growth
  - H-Mine
  - Partition
  - CLOSET
  - CHARM
  - etc.

The Apriori Algorithm — Example



Apriori

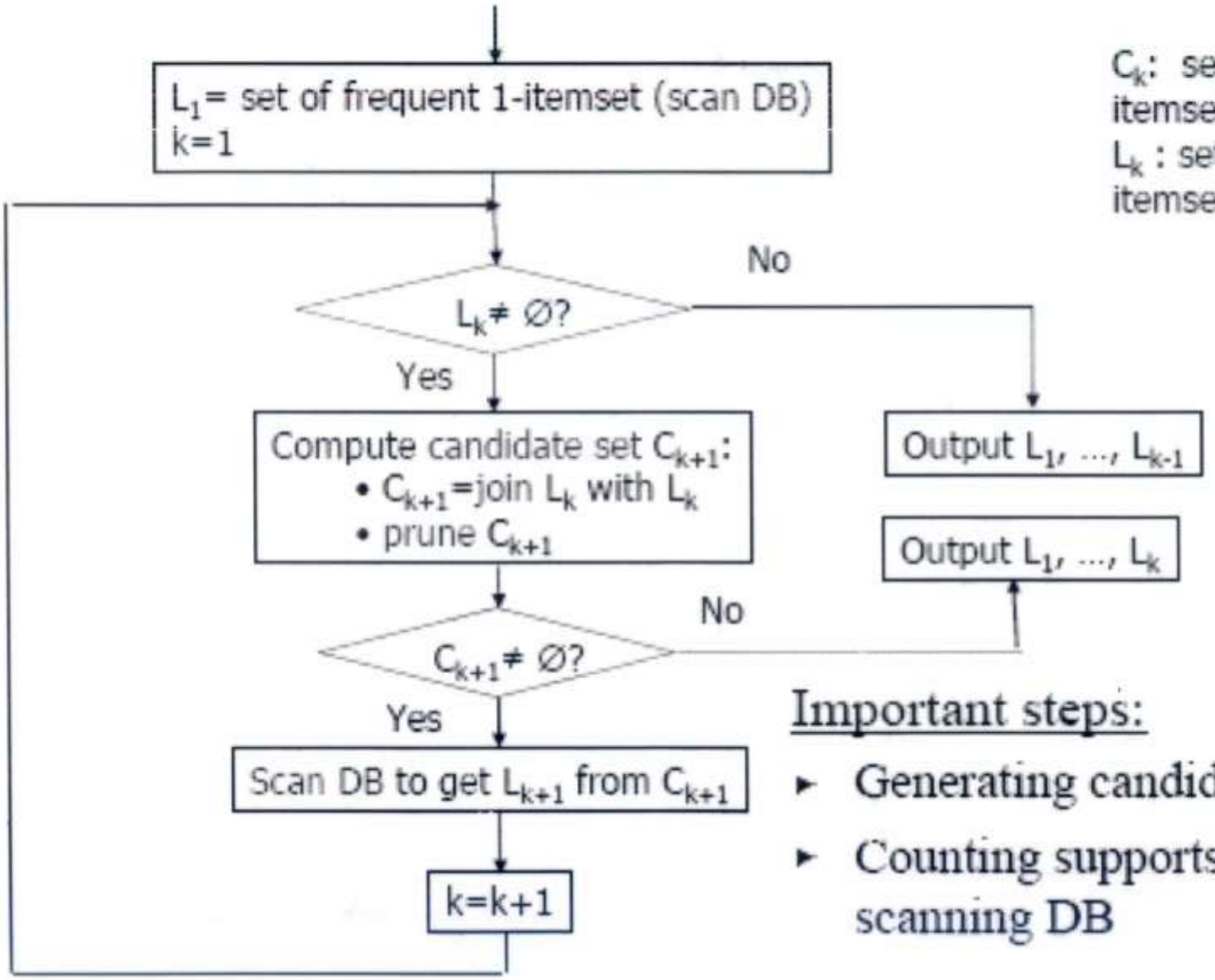
- The Apriori property (an anti-monotone property), also called *downward closure* property:

  Any nonempty subset of a frequent itemset must be frequent

  - i.e., if {*A,B,C*} is a frequent itemset, {A,B}, {A,C}, {B,C}, {*A*}, {*B*} and {C} must also be frequent.
  - This is because a transaction containing {A, B, C} also contains {A,B}, {B,C}, …

- No superset of any infrequent itemset should be checked.
  - Many item combinations can be pruned from the search space.
- Apriori-based mining (level-wise iterations)
  - Generate length (k+1) candidate itemsets from frequent k-itemsets
  - Test the candidates against DB

Apriori Algorithm (Flow Chart)



C_k: set of candidate k-itemsets
L_k : set of frequent k-itemsets

Important steps:

- Generating candidates
- Counting supports of candidates by scanning DB

The Apriori Algorithm

- Based on the Apriori property, use *iterative level-wise approach* and *candidate generation-and-test*
- Pseudo-code:

C_k: a set of candidate itemsets of size k
L_k : the set of frequent itemsets of size k

L_1 = {frequent items}; (*Scan database to find all frequent 1-itemsets*)
for (k = 1; L_k !=∅; k++) **do begin**
  C_{k+1} = candidates generated from L_k;
  for each transaction t in database **do**
    increment the count of all candidates in C_{k+1} that are contained in t
  L_{k+1} = candidates in C_{k+1} satisfying min_support
  end
return $\cup_k L_{k-1}$;

Scan database to calculate support for each itemset in C_{k+1}

Level-wise Generation process: $L_k \rightarrow C_{k+1} \rightarrow L_{k+1}$

How to Generate Candidates?
(i.e., How to Generate C_{k+1} from L_k)

- Given L_k = the set of frequent k-itemsets
- List the items in each itemset of L_k in an order

L_3 =

{1 2 3}
{1 2 4}
{1 3 4}
{1 3 5}
{2 3 4}

- Given L_k, generate C_{k+1} in two steps:
  - Join Step: Join L_k with L_k by joining two k-itemsets in L_k. Two k-itemsets are joinable if their first (k-1) items are the same ~~and the last item in the first itemset is smaller than the last item in the second itemset~~ (the condition for joining two members of L_k).

    Now, C_4={{1 2 3 4}, {1 3 4 5}}
  - Prune Step: Delete all candidates in C_{k+1} that have a non-frequent subset by checking all length-k subsets of a candidate
    - Now, C_4={{1 2 3 4}}

(RD

Example of Candidate-generation

- L_4={*abcd, abcg, abdg, abef, abeh, acdg, bcdg*}
- Self-joining: L_4*L_4
  - *abcdg* from *abcd* and *abcg*
  - *abefh* from *abef* and *aceh*
- Pruning:
  - *abefh* is removed because *abfh or aefh or befh* is not in L_4
- C_5={*abcdg*}

Generate Association Rules from Frequent Itemsets

- Naïve algorithm:

for each frequent itemset *l* do
    for each nonempty proper subset *s* of *l* do
        if (support(*l*)/support(*s*) >= min_conf)
            output the rule $s \rightarrow l\text{-}s$, with
                support =support(l) and
                confidence=support(l)/support(s)

Generate Association Rules from Frequent Itemsets

- Example:
  - Given a frequent itemset: l={I1, I2, I5}
  - nonempty subsets of l are {I1,I2}, {I1,I5}, {I2,I5}, {I1}, {I2}, {I5}
  - resulting association rules:

$$I1 \wedge I2 \rightarrow I5, \quad confidence = 50\%$$
$$I1 \wedge I5 \rightarrow I2, \quad confidence = 100\% \checkmark$$
$$I2 \wedge I5 \rightarrow I1, \quad confidence = 100\% \checkmark$$
$$I1 \rightarrow I2 \wedge I5, \quad confidence = 33\%$$
$$I2 \rightarrow I1 \wedge I5, \quad confidence = 29\%$$
$$I5 \rightarrow I1 \wedge I2, \quad confidence = 100\% \checkmark$$

If minimum confidence threshold is 70%, only 3 rules are output.

Example for Association Rule Mining.

For the following TDB, find all association rules with 50% support and 75% Confidence.

TID	Items
100	Bread, Cheese, Eggs, Juice
200	Bread, Cheese, Juice
300	Bread, Milk, Yogurt
400	Bread, Juice, Milk
500	Cheese, Juice, Milk

Assume Minimum Support Count = 3

Solution:

C_1

Item	Sup
Bread	4
Cheese	3
Juice	4
Milk	3
* Eggs	1
* Yogurt	1

$\Rightarrow$ L_1:

Item	Sup
Bread	4
Cheese	3
Juice	4
Milk	3

C_2:

Itemset	Sup. count	
(Bread, Cheese)	2	X
(Bread, Juice)	3	
(Bread, Milk)	2	X
(Cheese, Juice)	3	
(Cheese, Milk)	1	X
(Juice, Milk)	2	X

$\Rightarrow$

L_2:

Itemset	Sup
(Bread, Juice)	3
(Cheese, Juice)	3

No C_3

Rule Generation:

		Confidence
L_2: (Bread, Juice)	Bread → Juice	$\frac{3}{4} = 75\%$
	Juice → Bread	$\frac{3}{4} = 75\%$
(Cheese, Juice)	Cheese → Juice	$3/3 = 100\%$
	Juice → Cheese	$3/4 = 75\%$

∴ All have minimum 75% confidence, all qualify.

Collaborative Filtering

Collaborative Filtering is a technique or a method to predict a user's taste and find the items that a user might prefer on the basis of information collected from various other users having similar tastes or preferences. It takes into consideration the basic fact that if person X and person Y have a certain reaction for some items then they might have the same opinion for other items too.

The two most popular forms of collaborative filtering are:

- **User Based:** Here, we look for the users who have rated various items in the same way and then find the rating of the missing item with the help of these users.
- **Item Based:** Here, we explore the relationship between the pair of items (the user who bought Y, also bought Z). We find the missing rating with the help of the ratings given to the other items by the user.

Let's talk about Item-Based Collaborative Filtering in detail. It was first invented and used by Amazon in 1998. Rather than matching the user to similar customers, item-to-item collaborative filtering matches each of the user's purchased and rated items to similar items, then combines those similar items into a recommendation list. Now, let us discuss how it works.

Item to Item Similarity: The very first step is to build the model by finding similarity between all the item pairs. The similarity between item pairs can be found in different ways. One of the most common methods is to use cosine similarity.

Formula for Adjusted Cosine Similarity:

The rating based similarity (S) between i and j is given in following formula.

$$S = \frac{\sum_{u \epsilon U_{ij}} (r_{u,i} - \bar{r}_u) \times (r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \epsilon U_{ij}} (r_{u,i} - \bar{r}_u)^2} \times \sqrt{\sum_{u \epsilon U_{ij}} (r_{u,j} - \bar{r}_u)^2}}$$

Where $r_{u,i}$ and $r_{u,j}$ are user u's rating on items i and j. $\bar{r}_u$ is the average rating of all items which is rated by user u. The range of S is [-1, 1]. So it is normalized to get the range [0,1] using the formula

$$S = \frac{(1+S)}{2}.$$

Formula for Cosine Similarity:

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Prediction Computation: The second stage involves executing a recommendation system. It uses the items (already rated by the user) that are most similar to the missing item to generate rating. We hence try to generate predictions based on the ratings of similar products. We compute this using a formula which computes rating for a particular item using weighted sum of the ratings of the other similar products.

After finding the similarity between the items, we select top-k similar items to the target item, then we can make prediction $P_{u,i}$ for target item i. $P_{u,i}$ is the prediction on item i for the target user u.

$$P_{u,i} = \frac{\sum_{n \in S_k} r_{u,n} \times TotalSim(i,n)}{\sum_{n \in S_k} TotalSim(i,n)}$$

S_k denotes top-k similar items to the target item i. $r_{u,n}$ denotes user u's rating on item n.

Prediction Computation

$$rating(U, I_i) = \frac{\sum_j rating(U, I_j) * s_{ij}}{\sum_j s_{ij}}$$

Example:1

Let us consider one example. Given below is a set table that contains some items and the user who have rated those items. The rating is explicit and is on a scale of 1 to 5. Each entry in the table denotes the rating given by a i^{th} User to a j^{th} Item. In most cases majority of cells are empty as a user rates only for

few items. **Here, we have taken 5 users and 4 items**. We need to find the missing ratings for the respective user.

TABLE 1 Rating Matrix of User and Items

Users	I1	I2	I3	I4
U1	4	5		4
U2	5	4		3
U3		2	3	3
U4	1		5	
U5		2	4	

Step 1

TABLE 2 Computing Similarity between Items using Adjusted Cosine Similarity

	I2	I3	I4
I1	0.70	0.29	0.66
I2	1.0	0.25	0.80
I3	0.25	1	0.37
I4	0.80	0.37	1.0

The predictions are calculated as.

$P_{1,3}$=2.23, , $P_{2,3}$=1.81, $P_{3,1}$=1.19, $P_{4,2}$=0.90, $P_{4,4}$=0.70, $P_{5,1}$=0.85, $P_{5,4}$=0.83. Where P_{ij} denotes the prediction on the item j for user i.

Example:2

Let us consider one example. Given below is a set table that contains some items and the user who have rated those items. The rating is explicit and is on a scale of 1 to 5. Each entry in the table denotes the rating given by a i^{th} User to a j^{th} Item. In most cases majority of cells are empty as a user rates only for few items. Here, we have taken 4 users and 3 items. We need to find the missing ratings for the respective user.

User/Item	Item_1	Item_2	Item_3
User_1	2	–	3
User_2	5	2	–
User_3	3	3	1
User_4	–	2	2

Step 1: Finding similarities of all the item pairs.

Form the item pairs. For example in this example the item pairs are (Item_1, Item_2), (Item_1, Item_3), and (Item_2, Item_3). Select each item to pair one by one. After this, we find all the users who have rated for both the items in the item pair. Form a vector for each item and calculate the similarity between the two items using the cosine formula stated above.

```
Sim (Item1, Item2)

In the table, we can see only User_1 and User_2 have rated for both
items 1 and 2.

Thus, let I1 be vector for Item_1 and I2 be for Item_2. Then,

I1 = 5U2 + 3U3 and,

I2 = 2U2 + 3U3

Sim (Item2, Item3)

In the table we can see only User_3 and User_4 have rated for both
the items 1 and 2.

Thus, let I2 be vector for Item_2 and I3 be for Item_3. Then,

I2 = 3U3 + 2U4 and,
```

I3 = 1U3 + 2U4

Sim (Item1, Item3)

In the table we can see only User_1 and User_3 have rated for both the items 1 and 2.

Thus, let I1 be vector for Item_1 and I3 be for Item_3. Then,

I1 = 2U1 + 3U3 and,

I3 = 3U1 + 1U3

$$Similarity(I1, I2) = \frac{(5*2)+(3*3)}{\sqrt{5^2+3^2}\sqrt{2^2+3^2}} = 0.90$$

$$Similarity(I2, I3) = \frac{(3*1)+(2*2)}{\sqrt{3^2+2^2}\sqrt{1^2+2^2}} = 0.869$$

$$Similarity(I1, I3) = \frac{(2*3)+(3*1)}{\sqrt{2^2+3^2}\sqrt{3^2+1^2}} = 0.789$$

Step 2: Generating the missing ratings in the table

Now, in this step we calculate the ratings that are missing in the table.

```
Rating of Item_2 for User_1
```

$$r(U_1, I_2) = \frac{r(U_1,I_1)*s_{I_1I_2}+r(U_1,I_3)*s_{I_3I_2}}{s_{I_1I_2}+s_{I_3I_2}} = \frac{(2*0.9)+(3*0.869)}{(0.9+0.869)} = 2.49$$

```
Rating of Item_3 for User_2
```

$$r(U_2, I_3) = \frac{r(U_2,I_1)*s_{I_1I_3}+r(U_2,I_2)*s_{I_2I_3}}{s_{I_1I_3}+s_{I_2I_3}} = \frac{(5*0.789)+(2*0.869)}{(0.789+0.869)} = 3.43$$

```
Rating of Item_1 for User_4
```

$$r(U_4, I_1) = \frac{r(U_4,I_2)*s_{I_1I_2}+r(U_4,I_3)*s_{I_1I_3}}{s_{I_1I_2}+s_{I_1I_3}} = \frac{(2*0.9)+(2*0.789)}{(0.9+0.789)} = 2.0$$

Apriori Algorithm

Last Updated: 04-04-2020
Prerequisite – Frequent Item set in Data set (Association Rule Mining)
Apriori algorithm is given by R. Agrawal and R. Srikant in 1994 for finding frequent itemsets in a dataset for boolean association rule. Name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties. We apply an iterative approach or level-wise search where k-frequent itemsets are used to find k+1 itemsets.
To improve the efficiency of level-wise generation of frequent itemsets, an important property is used called *Apriori property* which helps by reducing the search space.
Apriori Property –
All non-empty subset of frequent itemset must be frequent. The key concept of Apriori algorithm is its anti-monotonicity of support measure. Apriori assumes that
All subsets of a frequent itemset must be frequent(Apriori propertry).
If an itemset is infrequent, all its supersets will be infrequent.

Before we start understanding the algorithm, go through some definitions which are explained in my previous post.
Consider the following dataset and we will find frequent itemsets and generate association rules for them.

TID	items
T1	I1, I2 , I5
T2	I2,I4
T3	I2,I3
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

minimum support count is 2
minimum confidence is 60%

Step-1: K=1
(I) Create a table containing support count of each item present in dataset – Called **C1(candidate set)**

Itemset	sup_count
I1	6
I2	7
I3	6
I4	2
I5	2

(II) compare candidate set item's support count with minimum support count(here min_support=2 if support_count of candidate set items is less than min_support then remove those items). This gives us itemset L1.

Itemset	sup_count
I1	6
I2	7
I3	6
I4	2
I5	2

Step-2: K=2

- Generate candidate set C2 using L1 (this is called join step). Condition of joining L_{k-1} and L_{k-1} is that it should have (K-2) elements in common.
- Check all subsets of an itemset are frequent or not and if not frequent remove that itemset.(Example subset of{I1, I2} are {I1}, {I2} they are frequent.Check for each itemset)
- Now find support count of these itemsets by searching in dataset.

Itemset	sup_count
I1,I2	4
I1,I3	4
I1,I4	1
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I3,I4	0
I3,I5	1
I4,I5	0

(II) compare candidate (C2) support count with minimum support count(here min_support=2 if support_count of candidate set item is less than min_support then remove those items) this gives us itemset L2.

Itemset	sup_count
I1,I2	4
I1,I3	4
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I2,I5	2

Step-3:

- Generate candidate set C3 using L2 (join step). Condition of joining L_{k-1} and L_{k-1} is that it should have (K-2) elements in common. So here, for L2, first element should match.
  So itemset generated by joining L2 is {I1, I2, I3}{I1, I2, I5}{I1, I3, i5}{I2, I3, I4}{I2, I4, I5}{I2, I3, I5}
- Check if all subsets of these itemsets are frequent or not and if not, then remove that itemset.(Here subset of {I1, I2, I3} are {I1, I2},{I2, I3},{I1, I3} which are frequent. For {I2, I3, I4}, subset {I3, I4} is not frequent so remove it. Similarly check for every itemset)
- find support count of these remaining itemset by searching in dataset.

Itemset	sup_count
I1,I2,I3	2
I1,I2,I5	2

(II) Compare candidate (C3) support count with minimum support count(here min_support=2 if support_count of candidate set item is less than min_support then remove those items) this gives us itemset L3.

Itemset	sup_count
I1,I2,I3	2
I1,I2,I5	2

Step-4:

- Generate candidate set C4 using L3 (join step). Condition of joining L_{k-1} and L_{k-1} (K=4) is that, they should have (K-2) elements in common. So here, for L3, first 2 elements (items) should match.
- Check all subsets of these itemsets are frequent or not (Here itemset formed by joining L3 is {I1, I2, I3, I5} so its subset contains {I1, I3, I5}, which is not frequent). So no itemset in C4
- We stop here because no frequent itemsets are found further

---

Thus, we have discovered all the frequent item-sets. Now generation of strong association rule comes into picture. For that we need to calculate confidence of each rule.

Confidence –
A confidence of 60% means that 60% of the customers, who purchased milk and bread also bought butter.

```
Confidence(A->B)=Support_count(AUB)/Support_count(A)
```

So here, by taking an example of any frequent itemset, we will show the rule generation.
Itemset {I1, I2, I3} //from L3
SO rules can be
[I1^I2]=>[I3] //confidence = sup(I1^I2^I3)/sup(I1^I2) = 2/4*100=50%
[I1^I3]=>[I2] //confidence = sup(I1^I2^I3)/sup(I1^I3) = 2/4*100=50%
[I2^I3]=>[I1] //confidence = sup(I1^I2^I3)/sup(I2^I3) = 2/4*100=50%
[I1]=>[I2^I3] //confidence = sup(I1^I2^I3)/sup(I1) = 2/6*100=33%
[I2]=>[I1^I3] //confidence = sup(I1^I2^I3)/sup(I2) = 2/7*100=28%
[I3]=>[I1^I2] //confidence = sup(I1^I2^I3)/sup(I3) = 2/6*100=33%

So if minimum confidence is 50%, then first 3 rules can be considered as strong association rules.

Limitations of Apriori Algorithm
Apriori Algorithm can be slow. The main limitation is time required to hold a vast number of candidate sets with much frequent itemsets, low minimum support or large itemsets i.e. it is not an efficient approach for large number of datasets. For example, if there are 10^4 from frequent 1-itemsets, it need to generate more than 10^7 candidates into 2-length which in turn they will be tested and accumulate. Furthermore, to detect frequent pattern in size 100 i.e. v1, v2… v100, it have to generate 2^100 candidate itemsets that yield on costly and wasting of time of candidate generation. So, it will check for many sets from candidate itemsets, also it will scan database many times repeatedly for finding candidate itemsets. Apriori will be very low and inefficiency when memory capacity is limited with large number of transactions. [Source
: https://arxiv.org/pdf/1403.3948.pdf]