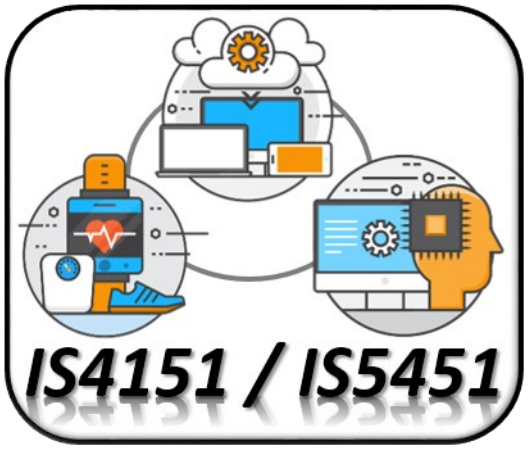


Lecture 11

Machine Learning for IoT Data (II)

IS4151/IS5451 – AIoT Solutions and Development
AY 2024/25 Semester 2

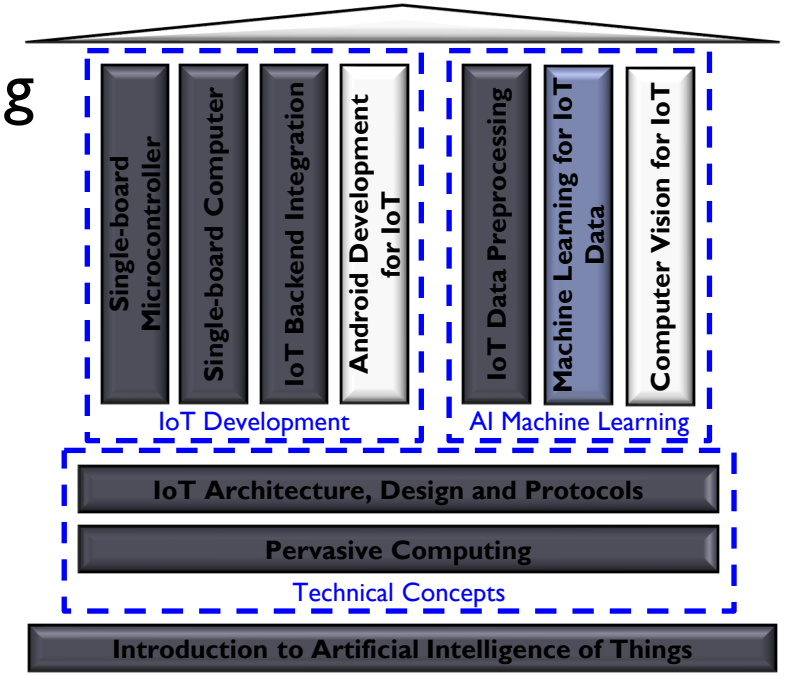
Lecturer: A/P TAN Wee Kek

Email: tanwk@comp.nus.edu.sg :: **Tel**: 6516 6731 :: **Office**: COM3-02-35

Consultation: Tuesday, 2 pm to 4 pm. Additional consultations by appointment are welcome.

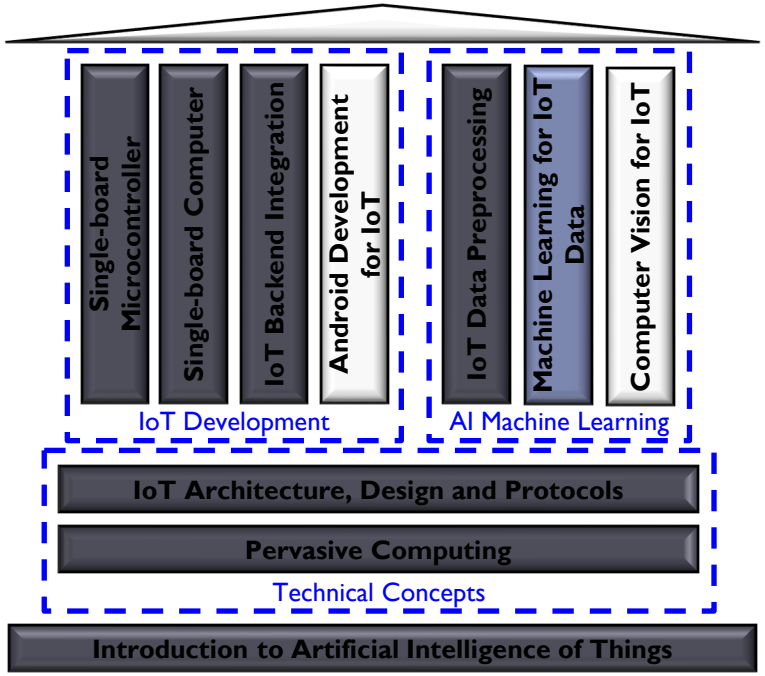
Quick Recap…

- In the previous lecture, we learnt:
  - How to predict a continuous numerical dependent variable with regression analysis.
  - How to predict a categorical dependent variable with classification.
- This lecture continues our learning journey to explore other useful machine learning techniques.



Learning Objectives

- How to perform prediction with probabilistic classification.
- How to perform prediction with advanced classifiers.
- How to perform segmentation with clustering.



Readings

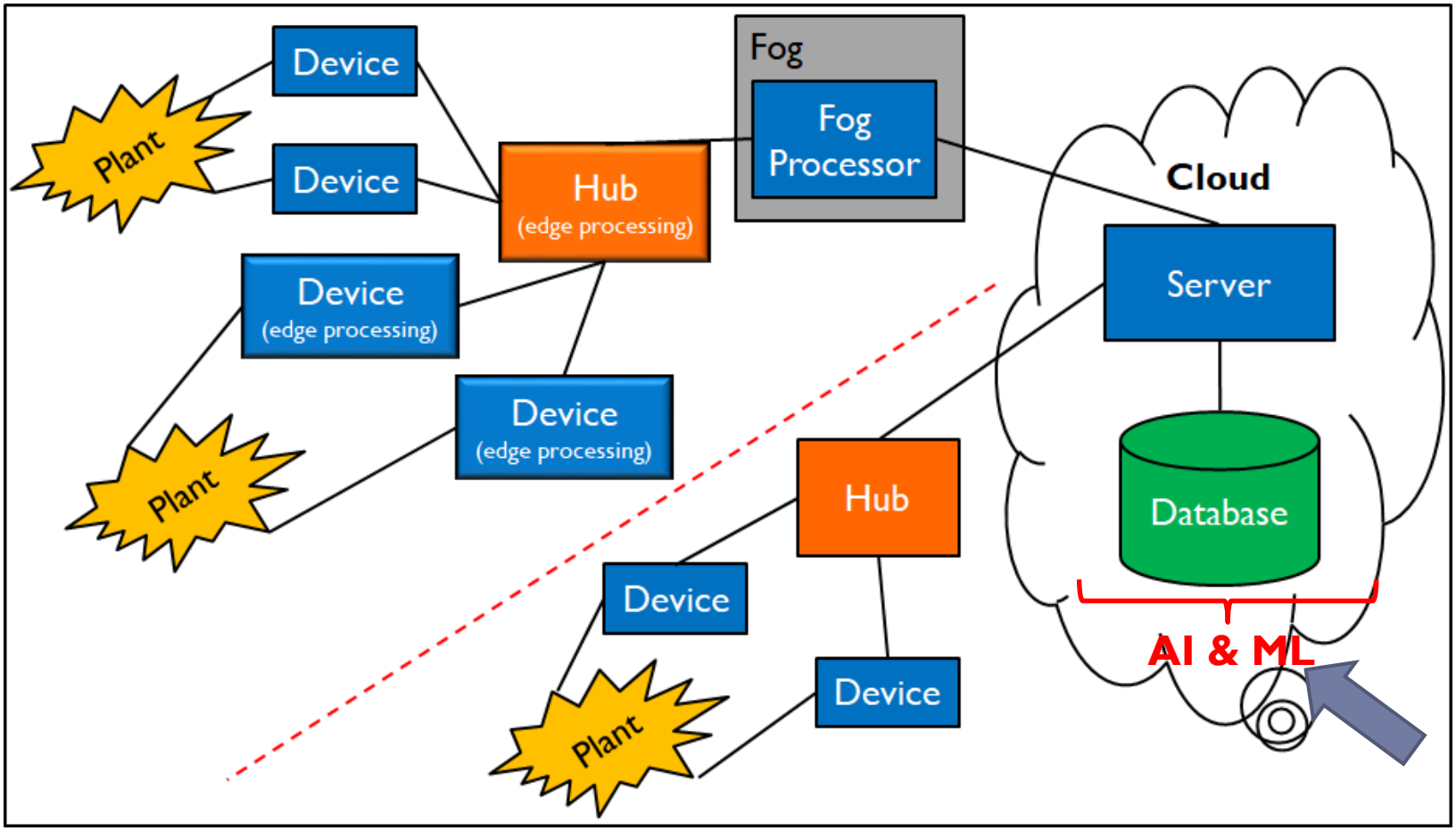
- Required readings:
  - None.
- Suggested readings:
  - None.

Technical Roadmap for IS4151/IS5451

Single-board Microcontroller
Android Wear

Single-board Computer
Android App

IoT Backend Integration



Logistic Regression

Posterior Probability

- According to **Bayes' theorem**, the posterior probability $P(Y|X)$ refers to the probability of observing the target class Y given the instance X:
  - For a two-class binary classification problem, we have $P(Y = 0|X)$ and $P(Y = 1|X)$.
  - A classifier can be built to determine the posterior probabilities of any instance X.
  - An instance X is classified as either class 0 or 1 depending on which corresponding posterior probability is higher.
- **Naive Bayes** classifier:
  - A generative classifier based on the principle of conditional probability as defined by the Bayes' theorem.

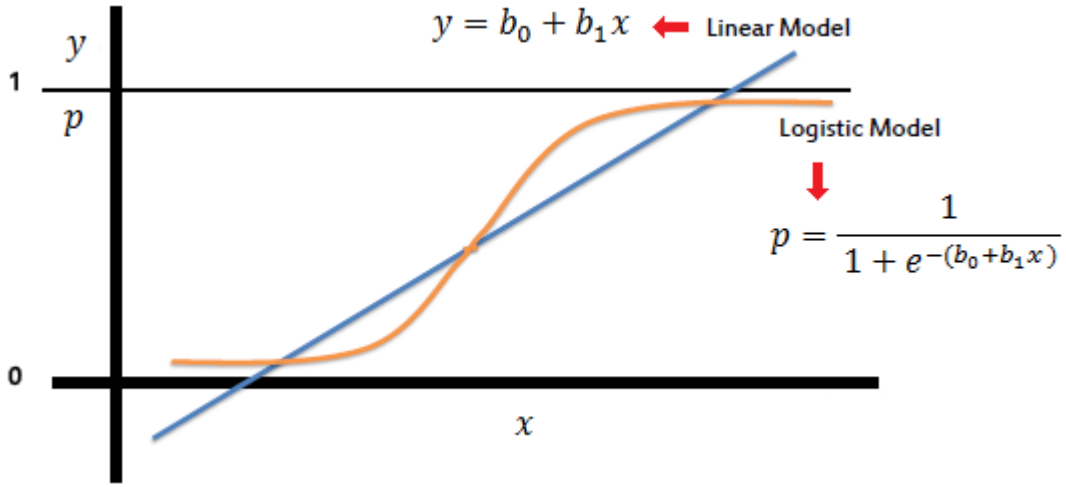
Posterior Probability (cont.)

- Posterior probability is computed from prior probability and class-conditional probability.

- **Logistic regression**:
  - A discriminative classifier that is based on the logistic model.
  - Models the event (i.e., posterior probability) by having the **log-odds** for the event being a linear combination of one or more independent variables.
  - Essentially estimating the parameters of a **logistic model** using a linear regression.

Posterior Probability (cont.)

- Problem of predicting a two-class output (i.e., 0 or 1) with a linear regression:



  - Recall that an exponential relationship $Y = e^{b+wX}$ can be linearized through a logarithmic transformation $Z = log(Y)$ into a linear relationship $Z = b + wX$.

Logistic Regression

- **Logistic regression** is a technique for converting binary classification problems into linear regression.
- Values of response variables are assumed to be 0 or 1.
- Using logistic regression, the posterior probability $P(Y|X)$ of the target variable Y conditioned to the input $X = (X_1, X_2, \ldots, X_n)$ is modeled according to the logistic function (where $e = 2.718281828 \ldots$):

$$\boldsymbol{P(Y=1 \mid X_1, X_2, \ldots, X_n)} = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\ldots+\beta_n X_n)}} = \frac{e^{\beta_0+\beta_1 X_1+\ldots+\beta_n X_n}}{1+e^{\beta_0+\beta_1 X_1+\ldots+\beta_n X_n}}$$

$$\boldsymbol{P(Y=0 \mid X_1, X_2, \ldots, X_n)}$$

$$= 1 - P(Y=1 \mid X_1, X_2, \ldots, X_n) = 1 - \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\ldots+\beta_n X_n)}} = \frac{1}{1+e^{\beta_0+\beta_1 X_1+\ldots+\beta_n X_n}}$$

Logistic Regression (cont.)

- Graph of the logistic function $f(x) = \frac{1}{1+e^{-x}}$:

- Hence, $0 \le P(Y=1 \mid X_1, X_2, ..., X_n) \le 1$ and
  $0 \le P(Y=0 \mid X_1, X_2, ..., X_n) \le 1$
- The above is known as the **sigmoid function**.

Logistic Regression (cont.)

- The ratio of the two conditional probabilities is:

$$\frac{P(Y=1|X_1,X_2,\ldots,X_n)}{P(Y=0|X_1,X_2,\ldots,X_n)} = e^{\beta_0+\beta_1X_1+\ldots+\beta_nX_n}$$

the odds of it being class 1

This is the **odds** in favor of $y=1$

  - E.g., If $P(Y=1|X_1,X_2,\ldots,X_n)=0.5$ and $P(Y=0|X_1,X_2,\ldots,X_n)=0.5$, the odds would be 1.
  - E.g., If $P(Y=1|X_1,X_2,\ldots,X_n)=0.67$ and $P(Y=0|X_1,X_2,\ldots,X_n)=0.33$, the odds would be ≈ 2.

so lets say 2/3 and 1/3 then the odds of y=1 would be 2 (two times)

- And its logarithm:

This is the **logit** function, or the logarithm of the **odds** known simply as the **log-odds**

$$\ln\left(\frac{P(Y=1|X_1,X_2,\ldots,X_n)}{P(Y=0|X_1,X_2,\ldots,X_n)}\right) = \beta_0+\beta_1X_1+\ldots+\beta_nX_n$$

Logistic Regression (cont.)

- If X_1 is increased by 1:

so if increase by 1 -> bi increases also

$$\text{logit}\,|_{X_i+1} = \text{logit}\,|_{X_i} + \beta_i$$

$$\text{odds}\,|_{X_i+1} = \text{odds}\,|_{X_i} \times e^{\beta_i}$$

additive impact vs multiplicative increase

- e^{β_i} is the odds-ratio – The multiplicative increase in the odds when X_1 increases by one (other variables remaining constant):

x increase

if beta increase

$\beta_i > 0 \Rightarrow e^{\beta_i} > 1 \Rightarrow$ odds and probability increase with X_1

$\beta_i < 0 \Rightarrow e^{\beta_i} < 1 \Rightarrow$ odds and probability decrease with X_1

multiplicative -> mulitplier

Logistic Regression Example 1

- A system analyst studied the effect of computer programming experience on ability to complete a complex programming task within a specified time.
- They had varying amount of experience (measured in months).
- All persons were given the same programming task and their success in the task was recorded:
  - $Y = 1$ if task was completed successfully within the allotted time.
  - $Y = 0$ otherwise.

Person	Months-Experience	Success
1	14	0
2	29	0
…	…	…
24	22	1
25	8	1

Logistic Regression Example 1 (cont.)

- A standard logistic package was run on the data and the parameter values found are: $\beta_0 = -3.0595$ and $\beta_1 = 0.1615$.
- The estimated mean response for $i = 1$, where $X_1 = 14$ is:

$$a = \beta_0 + \beta_1 X_1 = -3.0595 + 0.1615(14) = -0.7985$$

negative -> person cannot complete the task

$$e^a = e^{-0.7985} = 0.4500$$

$$P(Y = 1 \mid X_1 = 14) = \frac{e^a}{1 + e^a} = \frac{0.4500}{1 + 0.4500} = 0.3103$$

- The estimated probability that a person with 14 months experience will successfully complete the programming task is 0.3103.
- The odds in favor of completing the task $= 0.3103/(1 - 0.3103)$
  $= 0.4499$

Logistic Regression Example 1 (cont.)

- Suppose there is another programmer with 15 months experience, i.e., $X_1 = 15$.
- Recall the parameter values are $\beta_0 = -3.0595$ and $\beta_1 = 0.1615$

$$b = \beta_0 + \beta_1 X_1 = -3.0595 + 0.1615(15) = -0.637$$ log odds

$$e^b = e^{-0.637} = 0.5289$$

$$P(Y = 1 \mid X_1 = 15) = \frac{e^b}{1 + e^b} = \frac{0.5289}{1 + 0.5289} = 0.3459$$

- The estimated probability that a person with 15 months experience will successfully complete the programming task is 0.3459.
- The odds in favor of completing the task $= 0.3459/(1 - 0.3459)$
  $= 0.5288$

Logistic Regression Example 1 (cont.)

- Comparing the two odds:

$$\frac{0.5288}{0.4499} = 1.1753 = e^{0.1615}$$

- The odds increase by 17.53% with each additional month of experience.

to break even
-> 100 / 17.53

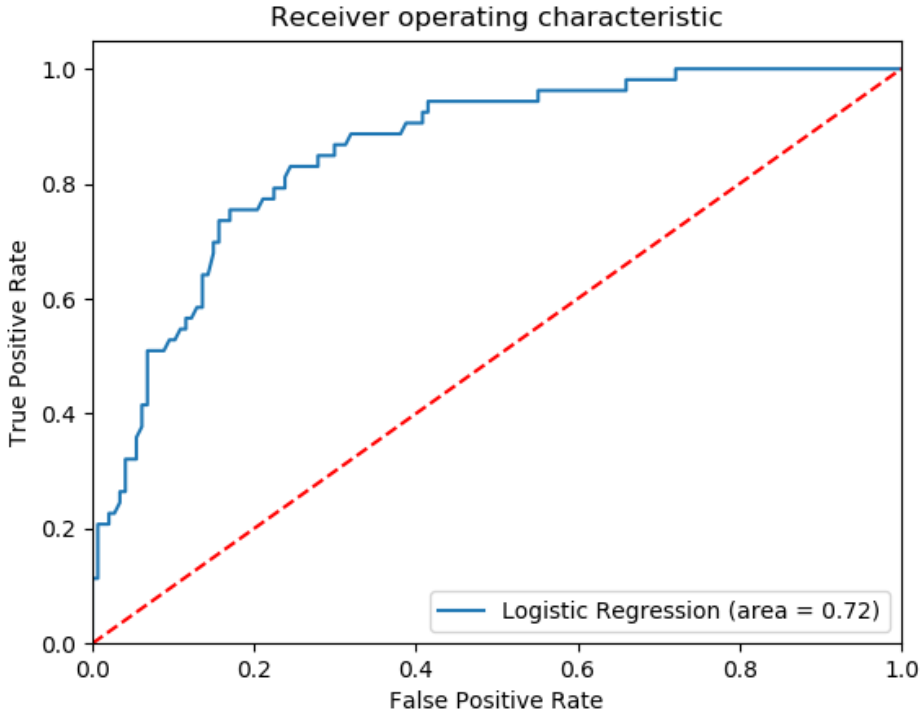
add the value
+ 14 => the
break even
value

Logistic Regression Example 2

- Predict the writing test score of 200 high school students:
  - Original dataset from UCLA.
  - The variables read, write, math, science and socst are the results of standardized tests on reading, writing, math, science and social studies (respectively).
  - The variable female is coded 1 if female and 0 if male.
  - The response variable is honcomp with two possible values:
    - High writing test score if the writing score is greater than or equal to 60 (honcomp = 1),
    - Low writing test score, otherwise (honcomp = 0).
- This is a two-class binary classification problem.

Logistic Regression Example 2 (cont.)

- The predictor variables used are gender (female), reading test score (read) and science test score (science).
- See the sample script file `src01`.



Logistic Regression Example 2 (cont.)

	precision	recall	f1-score	support
0	0.84	0.92	0.88	147
1	0.69	0.51	0.59	53
accuracy			0.81	200
macro avg	0.77	0.71	0.73	200
weighted avg	0.80	0.81	0.80	200

Logit Regression Results

Dep. Variable:	honcomp	**No. Observations:**	200
Model:	Logit	**Df Residuals:**	196
Method:	MLE	**Df Model:**	3
Date:	Thu, 30 Mar 2023	**Pseudo R-squ.:**	0.3072
Time:	16:07:41	**Log-Likelihood:**	-80.118
converged:	True	**LL-Null:**	-115.64
Covariance Type:	nonrobust	**LLR p-value:**	2.540e-15

	coef	std err	z	P>|z|	[0.025	0.975]
const	-12.7772	1.976	-6.467	0.000	-16.650	-8.905
female	1.4825	0.447	3.314	0.001	0.606	2.359
read	0.1035	0.026	4.018	0.000	0.053	0.154
science	0.0948	0.030	3.113	0.002	0.035	0.154

Logistic Regression Example 2 (cont.)

- Selected output from StatsModels:
  - Estimate for Intercept $\beta_0 = -12.7772$
  - Estimate for $\beta_1 = 1.4825$ (corresponds to variable female):
    - Odds ratio point estimate corresponds to variable female $= 4.404 = e^{\beta_1}$
    - The odds of a female student getting high writing test score is more than 4-fold higher than a male student (given the same reading and science test scores).
  - Estimate for $\beta_2 = 0.1035$ (corresponds to variable read)
  - Estimate for $\beta_3 = 0.0948$ (corresponds to variable science):
    - The estimated logistic regression coefficient for a one-unit change in science score, given the other variables in the model are held constant.
    - If a student were to increase his/her science score by one point, the difference in log-odds (logit response values) for high writing score is expected to increase by 0.0948 units, all other variables held constant.

Advanced Classifiers

Overview of Support Vector Machines

- **Support vector machines (SVMs)**: classification and regression sia
  - SVMs are a family of separation methods for classification and regression.
  - SVMs are developed in the context of statistical learning theory.
  - Among the best supervised learning algorithm:
    - Most theoretically motivated.
    - Practically most effective classification algorithms in modern machine learning.
  - SVMs are initially designed to fit a linear boundary between the samples of a binary problem, ensuring the maximum robustness in terms of tolerance to isotropic uncertainty.

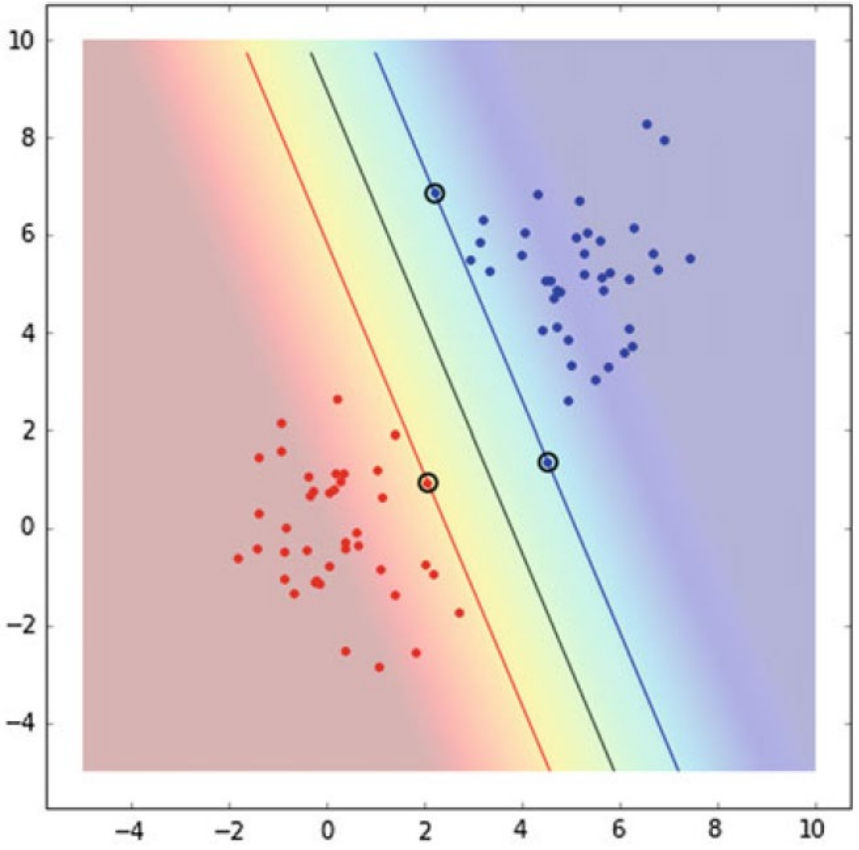
What is SVM? – A Video Introduction

How Does SVM Work?

- Given a set of labelled training examples for two categories:
  - An SVM training algorithm builds a model that assigns new examples to one category or the other.
  - This makes SVM a non-probabilistic binary linear classifier.
  - An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible.
  - New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.
  - The gist of SVM is to establish an optimal hyperplane for linearly separable patterns.

How Does SVM Work? (cont.)

- The figure below depicts the SVM decision boundary and the support vectors:
  - The boundary shown has the the largest distance to the closest point of both classes.
  - Any other separating boundary will have a point of a class closer to it than this one.
  - The figure also shows the closest points of the classes to the boundary.
  - These special points are called **support vectors**.

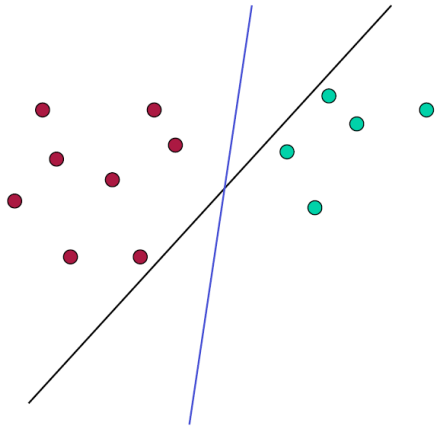


How Does SVM Work? (cont.)

- In fact, the boundary only depends on the support vectors:
  - If we remove any other point from the dataset, the boundary remains intact.
  - But in general, if any of these support vectors is removed, the boundary will change.
- In addition to performing linear classification, SVMs can also perform non-linear classification efficiently:
  - SVMs can be extended to patterns that are not linearly separable by transformations of original data to map into new space using **Kernel functions**.
  - This approach is known as the kernel trick – transforming data into another dimension that has a clear dividing margin between classes of data.

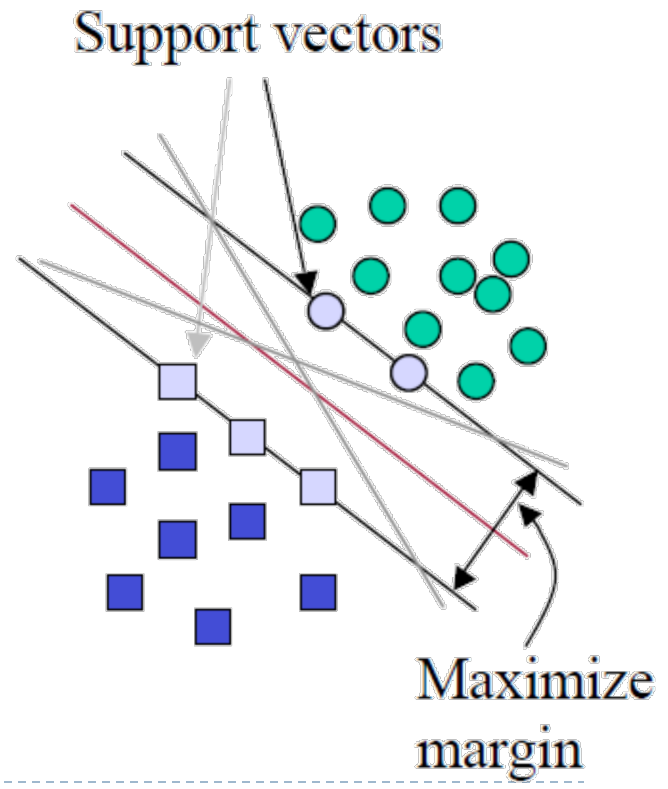
Support Vectors

- **Support vectors** are the data points that lie closest to the decision surface (or hyperplane):
  - They are the data points that are the most difficult to classify.
  - They have direct bearing on the optimum location of the decision surface.
- Which separating hyperplane should we use?
  - In general, there are many possible solutions.
  - SVM finds an optimal solution.



Support Vector Machine

- SVMs maximize the margin around the separating hyperplane:
  - This is known as the "street".
- This is known as the **maximum-margin hyperplane**.
- The decision function is fully specified by a (usually very small) subset of training samples, i.e., the support vectors.
- This is essentially a quadratic programming problem that can be solved by standard methods.



Support Vector Machine (cont.)

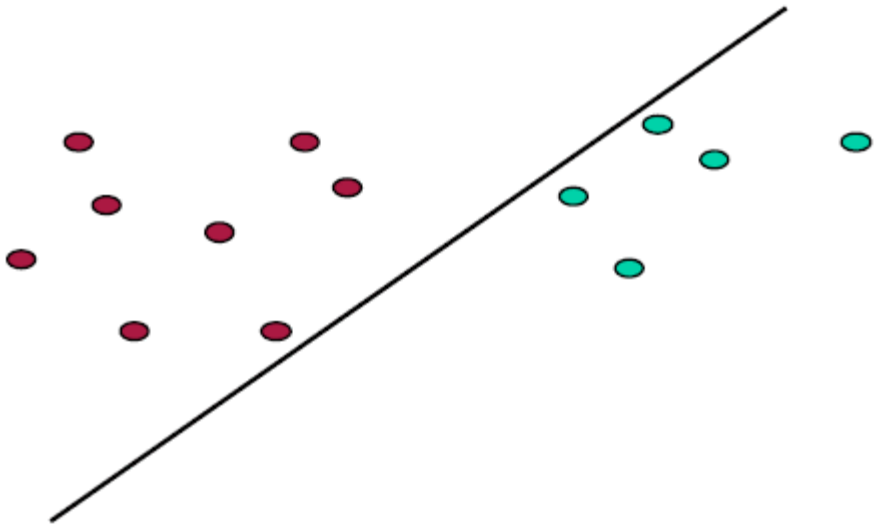
- Separation by hyperplanes:
  - We will assume linear separability for now and will relax this assumption later.
  - In two dimensions, we can separate by a line.
  - In higher dimensions, we need hyperplanes.
- General input/output for SVMs:
  - Similar to neural nets but there is one important addition.
  - Input:
    - Set of (input, output) training pair samples.
    - The input are the sample features $x_1, x_2, \ldots, x_n$.
    - The output is the result y.
    - Typically, there can be lots of input features x_i.

Support Vector Machine (cont.)

- Output:
  - A set of weights w_i, one for each feature.
  - The linear combination of the weights predicts the value of y.
  - Thus far, this is similar to neural nets.
- Important difference:
  - We use the optimization of maximizing the margin ("street width") to reduce the number of weights that are nonzero to just a few that correspond to the important features that 'matter' in deciding the separating line (hyperplane).
  - These nonzero weights correspond to the support vectors (because they 'support' the separating hyperplane).

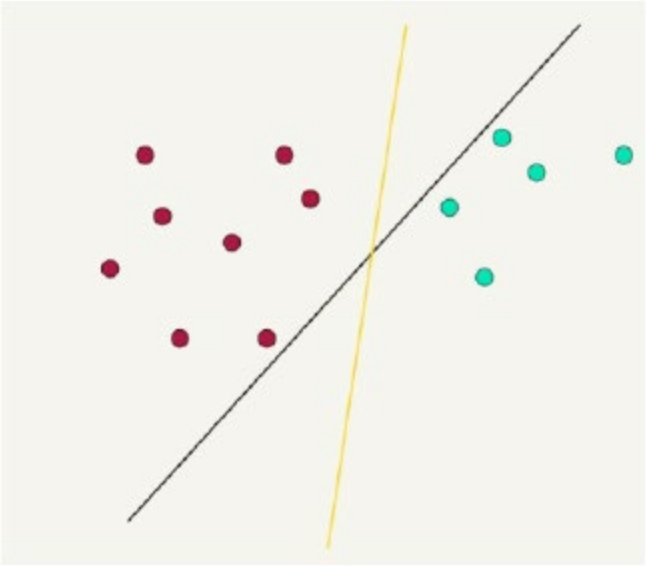
Support Vector Machine (cont.)

- Two-dimensional case:
  - Find a, b and c such that
  - $ax + by \geq c$ for the red points.
  - $ax + by \leq (\text{or} <)c$ for the green points.



Support Vector Machine (cont.)

- Which hyperplane to choose?
  - There are lots of possible solutions for a, b and c.
  - Some methods find a separating hyperplane, but not the optimal one (e.g., neural net).
  - But the important question is which points should influence optimality?
    - All points?
      - Linear regression
      - Neural nets
    - Or only "difficult points" close to decision boundary?
      - Support vector machines

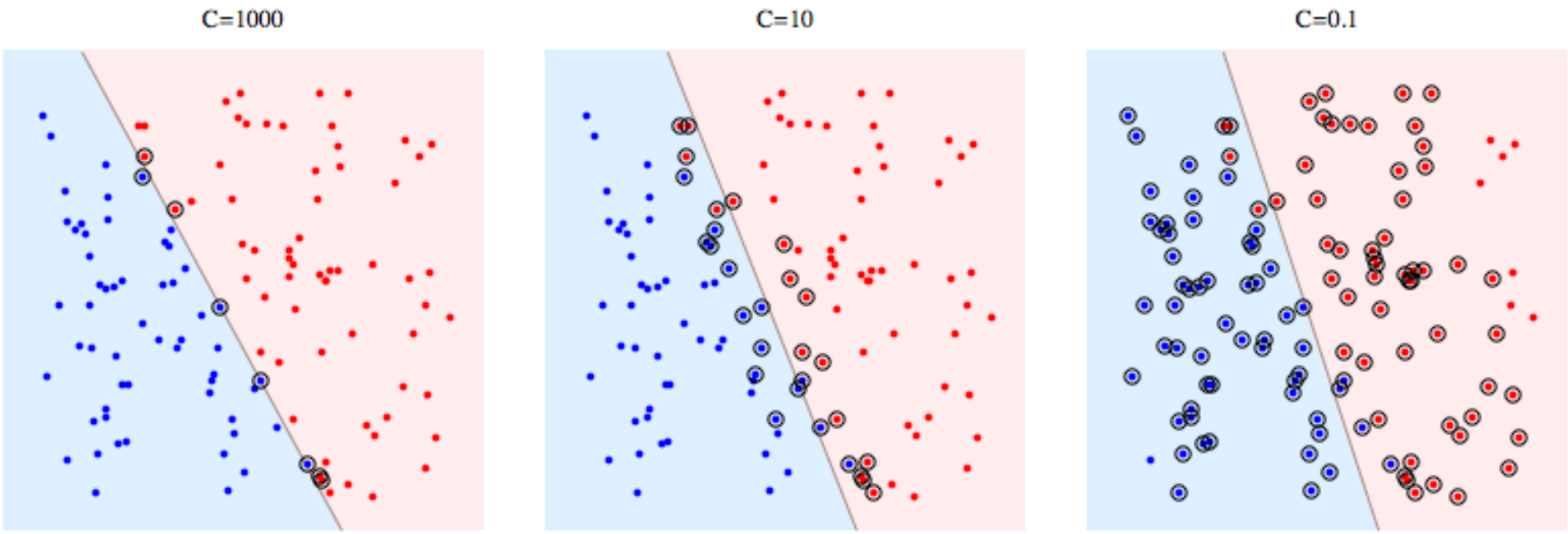


Hyperparameter Tuning for SVM

- To improve the model accuracy, there are several parameters that need to be tuned.
- The three major parameters include:

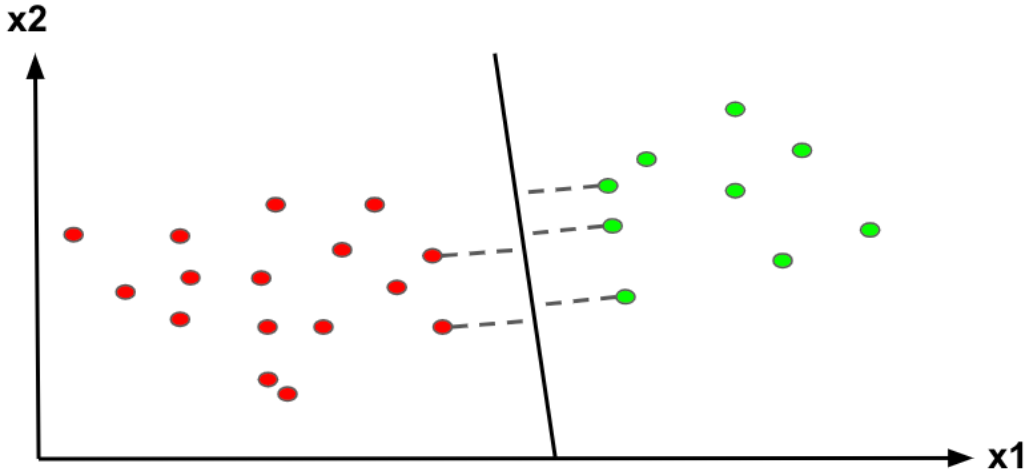
Parameter	Description
Kernel	Kernel takes low dimensional input space and transform it into a higher-dimensional space. It is mostly useful in non-linear separation problem.
C (Regularisation)	C is the penalty parameter, which represents misclassification or error term. C tells the SVM optimisation how much error is bearable. Control the trade-off between decision boundary and misclassification term. When C is high, it will classify all the data points correctly but there is a chance to overfit; smaller C gives larger margin and more misclassification.
Gamma	Defines how far influences the calculation of plausible line of separation. When gamma is higher, nearby points will have high influence; low gamma means far away points also be considered to get the decision boundary.

Hyperparameter Tuning for SVM (cont.)



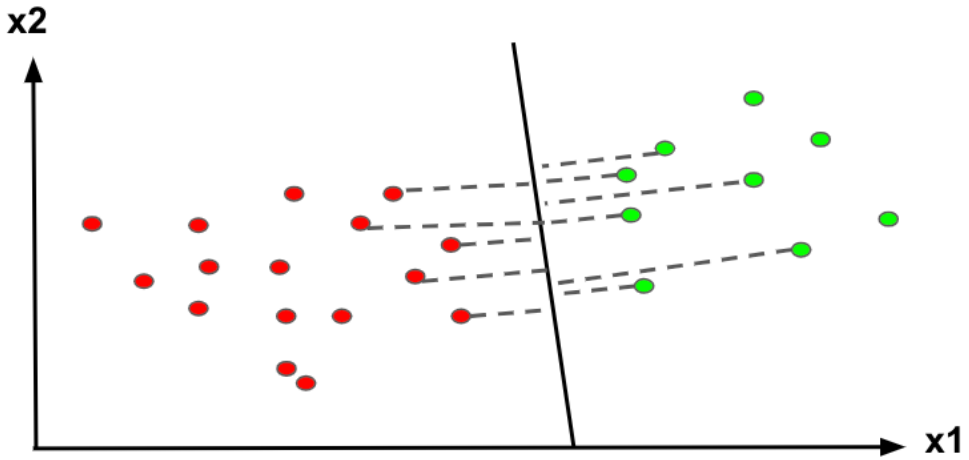
C (Regularisation)

Hyperparameter Tuning for SVM (cont.)



High Gamma

- only near points are considered.



Low Gamma

- far away points are also considered

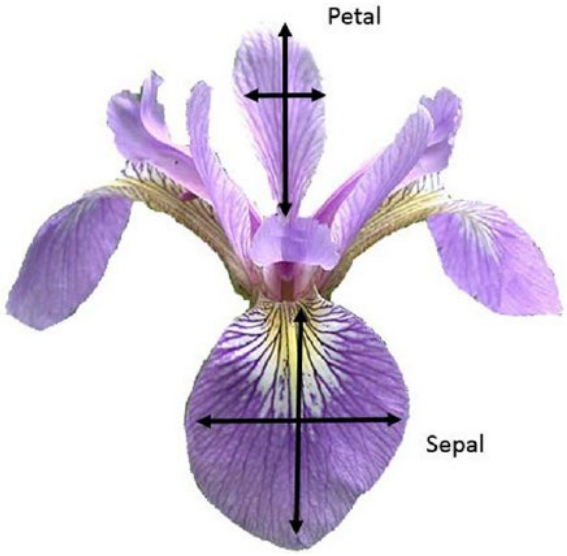
Gamma

Hyperparameter Tuning for SVM (cont.)

- Hyperparameters are parameters that are not directly learnt within estimators:
  - They are passed as arguments to the algorithm.
  - Grid search is commonly used as an approach to hyperparameter tuning that will methodically build and evaluate a model for each combination of algorithm parameters specified in a grid.

Let's Revisit the Iris Flower Dataset

- We will classify the three different species of iris with SVM:
  - Without hyperparameter tuning – Sample code `src02`.
  - With hyperparameter tuning – Sample code `src03`.
- Can you observe any difference in the results?

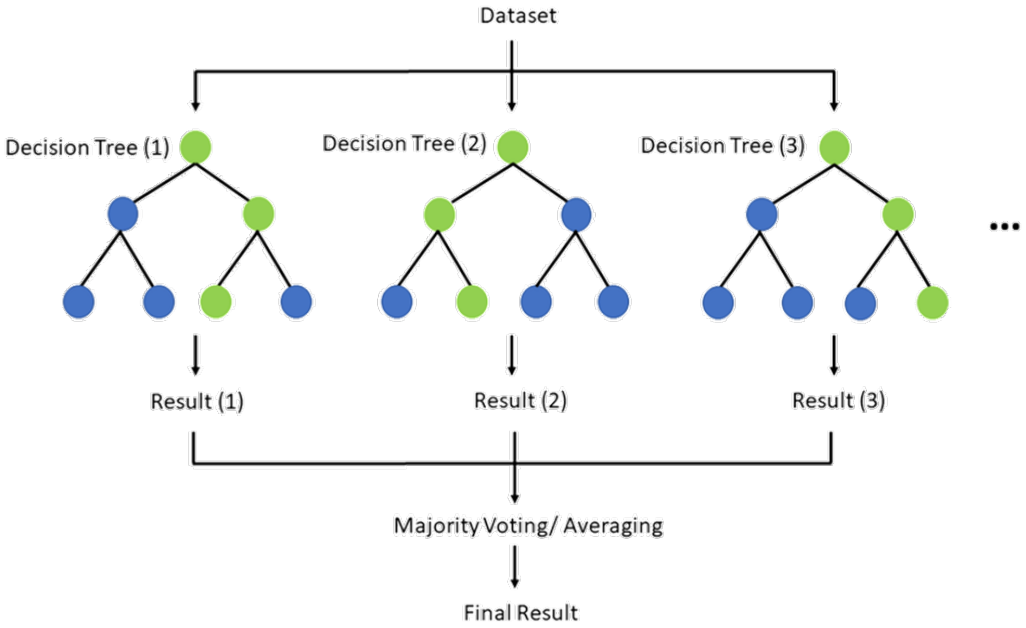


Random Forest

- **Random forest** is a popular and versatile machine learning method for classification and regression.
- **Ensemble Learning**:
  - Random forest aggregates the predictions of multiple decision trees to improve accuracy and reduce overfitting.
- **Bagging (Bootstrap Aggregating)**:
  - Each decision tree is trained on a random subset of the training data (with replacement).
  - This approach ensures diversity among the trees.

Random Forest (cont.)

- **Random Feature Selection**:
  - At each split in a tree, a random subset of features is considered to determine the best split.
  - This approach helps reduce correlation between trees and further improves the ensemble's performance.



How Does Random Forest Work?

- Create Bootstrapped Datasets:
  - Randomly sample the training data with replacement to create multiple subsets (bootstrapped datasets).
- Build Decision Trees:
  - For each subset, construct a decision tree.
  - At each node in the tree, randomly select a subset of features and choose the best split among them.
  - Allow the trees to grow to their full depth (often without pruning).
- Make Predictions:
  - For classification:
    - Each tree votes for a class, and the final prediction is the majority vote.

How Does Random Forest Work? (cont.)

- For regression:
  - The final prediction is the average of all tree predictions.

Advantages and Disadvantages of Random Forest

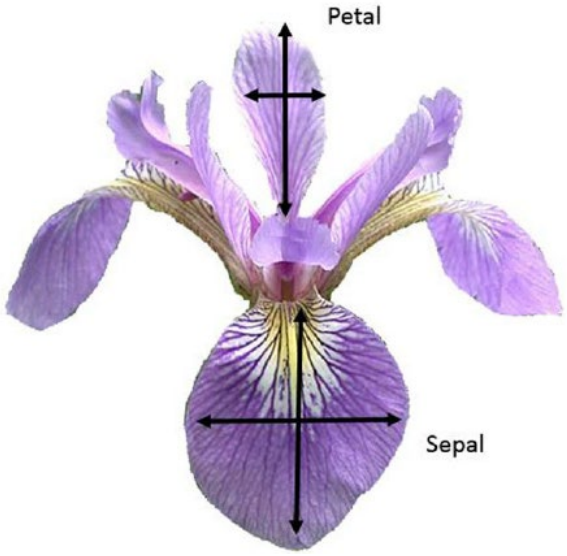
- **Advantages**:
  - Improved Accuracy – Combining predictions reduces the risk of overfitting and increases generalization.
  - Robust to Noise – The algorithm is less sensitive to outliers and noisy data.
  - Handles High-Dimensional Data – It can effectively handle datasets with many features.
- **Disadvantages**:
  - Complexity – Random forests are more computationally intensive than single decision trees.

Advantages and Disadvantages of Random Forest (cont.)

- Interpretability – While decision trees are easy to interpret, random forests lose this simplicity due to the aggregation of multiple trees.
- Overfitting – Although less prone to overfitting, random forests can still overfit with too many trees or insufficient diversity among them.

Let's Revisit the Iris Flower Dataset

- We will classify the three different species of iris with Random Forest Classifier:
  - Use 100 trees.
  - Sample code `src04`.
- Does the classification of the Iris flower dataset benefit from the use of Random Forest?



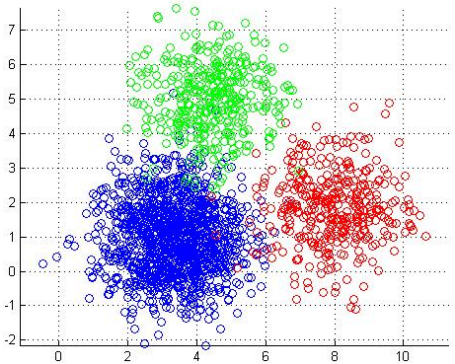
Segmentation with Clustering

Overview of Clustering

- **Clusters** are homogeneous groups of observations.
- To measure similarity between pairs of observations, a distance metric must be defined.
- **Clustering** is an unsupervised learning process.
- Focus of our discussions will be on:
  - Features of clustering models.
  - A partition method: ***K*-means**.
  - Quality indicators for clustering methods.

Clustering Methods

- **Aim** – To subdivide the records of a dataset into homogeneous groups of observations called clusters.
- Observations in a cluster are similar to one another and are dissimilar from observations in other clusters.



- Purpose of clustering:
  - As a tool which could provide meaningful interpretation of the phenomenon of interest:
    - Example – Grouping consumers based on their purchase behavior may reveal the existence of a market niche.

Clustering Methods (cont.)

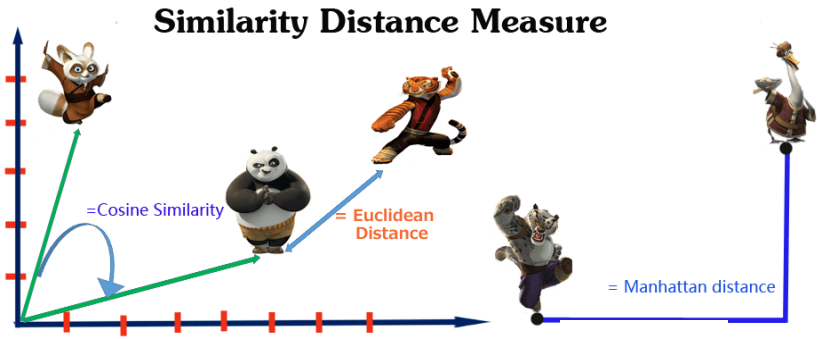
- As a preliminary phase of a data mining project that will be followed by other methodologies within each cluster:
  - Example:
    - Clustering is done before classification.
    - In retention analysis, distinct classification models may be developed for various clusters to improve the accuracy in spotting customers with high probability of churning.
- As a way to highlight outliers and identify an observation that might represent its own cluster.

Taxonomy of Clustering Methods

- Based on the logic used for deriving the clusters.
- **Partition methods**:
  - Develop a subdivision of the given dataset into a predetermined number *K* of non-empty subsets.
  - They are usually applied to small or medium sized data sets.
- **Hierarchical methods**:
  - Carry out multiple subdivisions into subsets.
  - Based on a tree structure and characterized by different homogeneity thresholds within each cluster and inhomogeneity threshold between distinct clusters.
  - No predetermined number of clusters is required.

Affinity Measures

- Clustering models are typically based on a measure of similarity between observations.
- The measure can typically be obtained by defining an appropriate notion of distance between each pair of observations:
  - Shorter distance between a pair of observations indicates greater similarity.
- There are many popular metrics depending on the type of variables being analyzed.



Affinity Measures (cont.)

- Given a dataset $\mathbb{D}$ having m observations $\mathrm{X}_1, \mathrm{X}_2, \mathrm{X}_3, \ldots \mathrm{X}_m$ each described by *n*-dimensional variables, we compute the **distance** matrix D:

$$\mathrm{D} = \left[d_{ik} \right] = \begin{bmatrix} 0 & d_{12} & \cdots & d_{1,m-1} & d_{1m} \\ & 0 & \cdots & d_{2,m-1} & d_{2m} \\ & & \cdots & \vdots & \vdots \\ & & & 0 & d_{m-1,m} \\ & & & & 0 \end{bmatrix}$$

where d_{ik} is the distance between observations X_i and X_k.

$$d_{ik} = \mathrm{dist}(\mathrm{X}_i, \mathrm{X}_k) = \mathrm{dist}(\mathrm{X}_k, \mathrm{X}_i) \quad \text{for } i, k = 1,2,\ldots,m$$

D is a symmetric $m \times m$ matrix with zero diagonal.

Affinity Measures (cont.)

- **Similarity measure** can be obtained by letting:

$$s_{ik} = \frac{1}{1 + d_{ik}} \quad \text{or} \quad s_{ik} = \frac{d_{\max} - d_{ik}}{d_{\max}}$$

where $d_{\max} = \max_{i,k} d_{ik}$ is the max value of D .

Affinity Measures for Numerical Variables

- If all n variables of the observations $X_1, X_2, X_3, \ldots X_m$ are numerical, the distance between X_i and X_k can be computed in four ways.
- **Euclidean distance** (or 2 norm):

$$\text{dist}(X_i, X_k) = \sqrt{\sum_{j=1}^{n} (x_{ij} - x_{kj})^2} = \sqrt{(x_{i1} - x_{k1})^2 + (x_{i2} - x_{k2})^2 + \ldots + (x_{in} - x_{kn})^2}$$

- **Manhattan distance** (or 1 norm):

$$\text{dist}(X_i, X_k) = \sum_{j=1}^{n} |x_{ij} - x_{kj}| = |x_{i1} - x_{k1}| + |x_{i2} - x_{k2}| + \ldots + |x_{in} - x_{kn}|$$

  - Manhattan distance is preferred for high-dimensional data, i.e., as the number of variables increases.

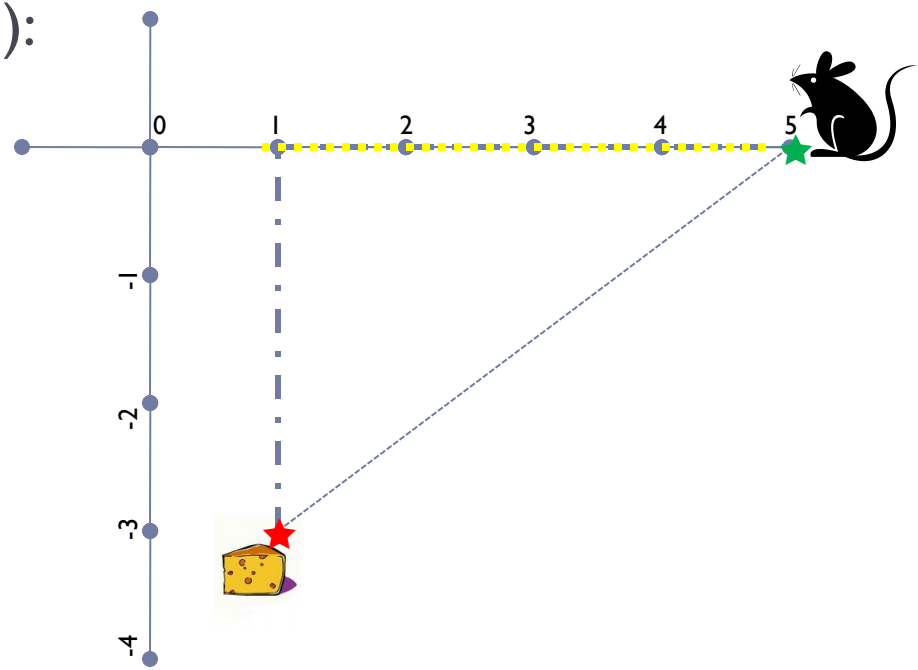
Affinity Measures for Numerical Variables (cont.)

- **Example:** $x_1 = (5,0)$ **and** $x_2 = (1,-3)$
  - Euclidean distance (or 2 norm):

$$\text{dist}(x_1, x_2) = \sqrt{(5-1)^2 + (0-(-3))^2}$$
$$= \sqrt{16+9} = 5$$

  - Manhattan distance (or 1 norm):

$$\text{dist}(x_1, x_2) = |5-1| + |0-(-3)|$$
$$= 4+3 = 7$$



Partition Methods

- Given a dataset $\mathbb{D}$, each represented by a vector in *n*-dimensional space, construct a collection of subsets $C = \{C_1, C_2, ..., C_K\}$ where $K \leq m$.
- K is the number of clusters and is generally predetermined.
- Clusters generated are usually exhaustive and mutually exclusive – Each observation belongs to only one cluster.
- Partition methods are iterative:
  - Assign m observations to the K clusters.
  - Then iteratively reallocate to improve overall quality of clusters.

Partition Methods (cont.)

- **Criteria for quality**:
  - Degree of homogeneity of observations in the same clusters.
  - Degree of heterogeneity with respect to observations in other clusters.
- The methods terminate when during the same iteration no reallocation occurs, i.e., clusters are stable.

K-means Algorithm

1. Initialize: choose K observations arbitrarily as the **centroids** of the clusters.
2. Assign each observation to a cluster with the nearest centroid.
3. If no observation is assigned to different cluster with respect to previous iteration, stop.
4. For each cluster, the new centroid is computed as the mean of the values belonging to that cluster. Go to Step 2.

K-means Algorithm (cont.)

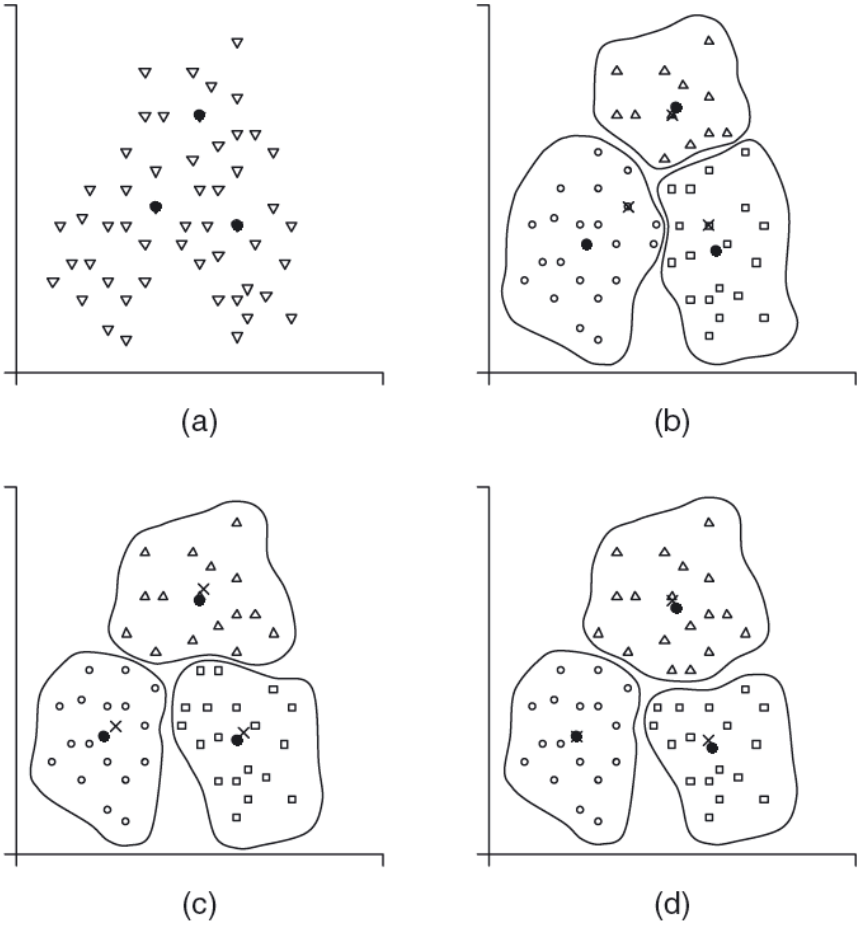


Figure 12.2 An example of application of the K-means algorithm

Source: Vercellis (2009), pp. 304

K-means Algorithm (cont.)

- Given a cluster C_h , $h = 1,2,...,K$, the **centroid** of the cluster is the point z_h having coordinates equal to the mean value of each variable in the observations belonging to that cluster:

$$z_{hj} = \frac{\sum_{X_i \in C_h} x_{ij}}{\text{card}\{C_h\}}$$

where $\text{card}\{C_h\}$ is the number of observations in cluster C_h .

K-means Algorithm (cont.)

- Example – Suppose we have 2-dimensional data with the variables $\{\text{Weight}, \text{Height}\}$:
  - In Cluster 1, the observations are: $\{65,168\}, \{69,172\}$.
  - In Cluster 2, the observations are: $\{50,165\}, \{58,158\}, \{54,157\}$.
  - The centroids are:
    - Cluster 1:

$$z_1 = \{z_{11}, z_{12}\} = \left\{\frac{65+69}{2}, \frac{168+172}{2}\right\} = \{67,170\}$$

    - Cluster 2:

$$z_2 = \{z_{21}, z_{22}\} = \left\{\frac{50+58+54}{3}, \frac{165+158+157}{3}\right\} = \{54,160\}$$

Clustering Example – *K*-means

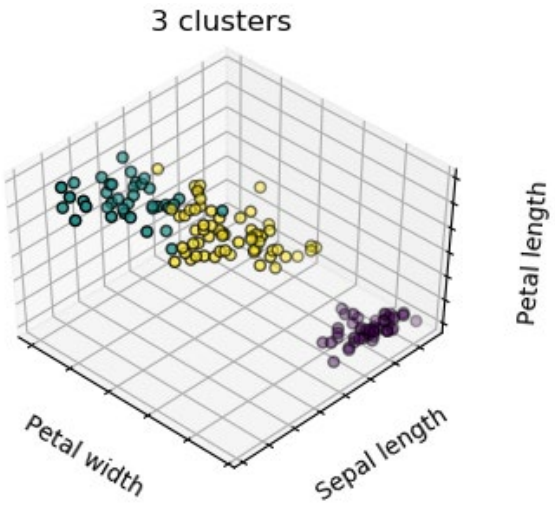
- Iris classification problem:
  - 3 classes – Setosa, Versicolor and Virginica.

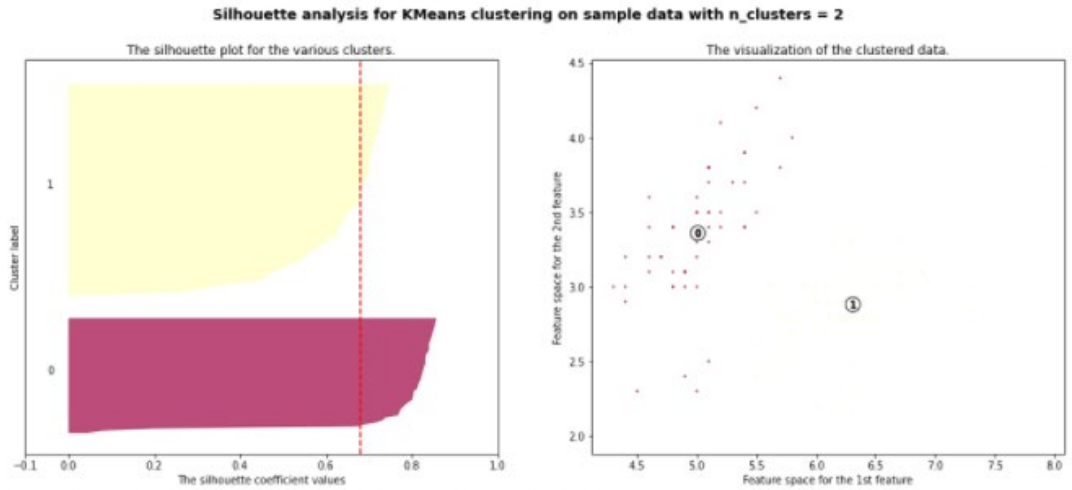


  - 4 variables – Sepal length, sepal width, petal length and petal width.
- We use *K*-means clustering with *K*=3:
  - Silhouette Score = 0.5526
  - Silhouette Score should be positive and closer to 1.0 is better.
- Refer to sample source file `src05` for the example.

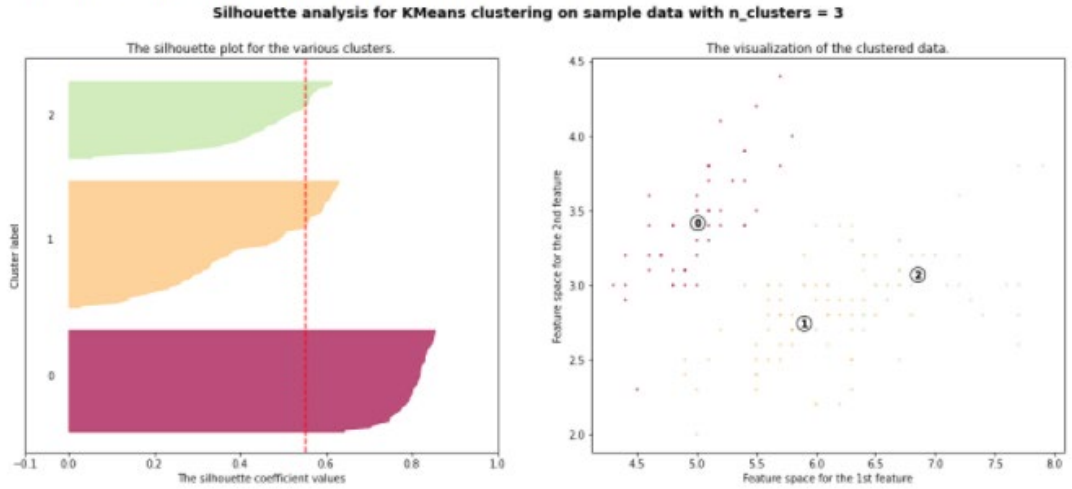
Clustering Example – *K*-means (cont.)

- We can generate the silhouette diagrams for *K*=2 and *K*=3 for comparison:
  - See the sample script `src06`.






```
For n_clusters = 3 The average silhouette_score is : 0.5525919445499757
```

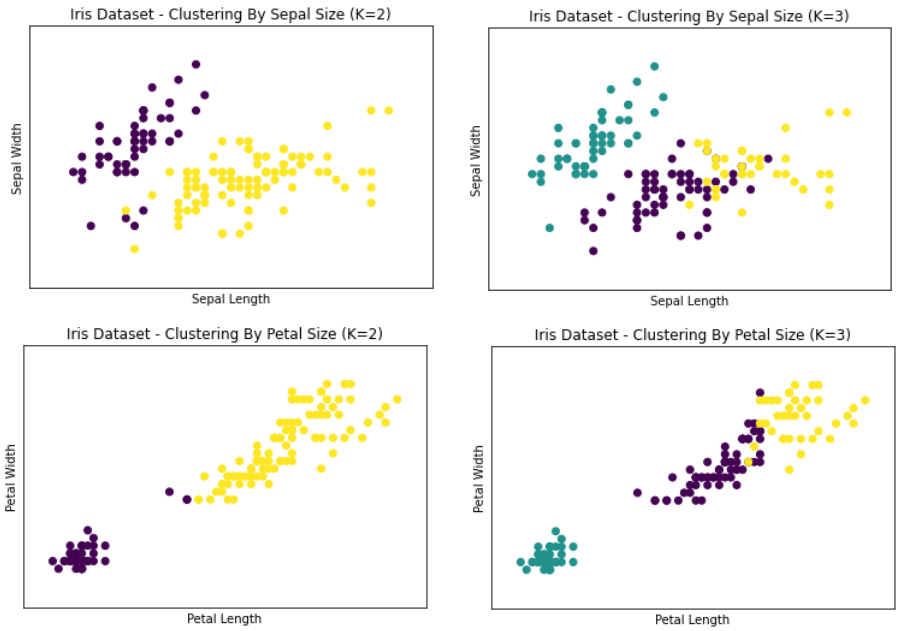


Clustering Example – *K*-means (cont.)

- To identify the distinguishing characteristics of observations in each cluster:
  - We can compute the within-cluster means and standard deviations of the independent variables.
  - Plot scatter plots of the observations using the required independent variables.
  - See the sample script src07.

Cluster	sepal_length	sepal_width	petal_length	petal_width
0	5.006 (0.343)	3.360 (0.440)	1.562 (0.440)	0.289 (0.212)
1	6.301 (0.634)	2.887 (0.327)	4.959 (0.780)	1.696 (0.416)

Cluster	sepal_length	sepal_width	petal_length	petal_width
0	5.902 (0.466)	2.748 (0.296)	4.394 (0.509)	1.434 (0.297)
1	5.006 (0.352)	3.418 (0.381)	1.464 (0.174)	0.244 (0.107)
2	6.850 (0.494)	3.074 (0.290)	5.742 (0.489)	2.071 (0.280)



Summary

- Probabilistic classifiers such as the logistics regression enables an AIoT system to automate decision making at a finer granularity.
- Advanced classification techniques such as hyperparameter tuning and ensemble methods can be applied to IoT sensor data.
- K-means can be used to perform clustering analysis.
- Clustering techniques such as k-means are useful when working with IoT sensor data that are unlabeled.

Q&A

Next Lecture…

- Learn about:
  - Introduction to Computer Vision.
  - Introduction to Object Detection.
  - Object Detection with YOLO.