Unit III

Instance based Learning

Prepared by Dr Y.UshaRani

Instance based Learning

- Instance-based learning methods such as nearest neighbor and locally weighted regression are conceptually straightforward approaches to approximating real-valued or discrete-valued target functions.

- store all training examples where x describes the attributes of each instance and f(x) denotes its class (or value).

Use a **Nearest Neighbor** method:

Given query instance ,

first locate nearest (most similar) training example ,

then estimate

Instance Based Learning

- Learning in these algorithms consists of simply storing the presented training data. When a new query instance is encountered, a set of similar related instances is retrieved from memory and used to classify the new query instance.
- One disadvantage of instance-based approaches is that the cost of classifying new instances can be high.

Instance Based Learning

- Instance Based Learning Classify based on local similarity
- Ranges from simple nearest neighbor to case-based and analogical reasoning
- Use local information near the current query instance to decide the classification of that instance
- As such can represent quite complex decision surfaces in a simple manner

KNN Model Representation

The most basic instance-based method is the k-NEAREST NEIGHBOR algorithm

- The model representation for KNN is the entire training dataset.
- It is as simple as that.
- KNN has no model other than storing the entire dataset, so there is no learning required.
- Efficient implementations can store the data using complex data structures

k-nearest neighbors (KNN)

•The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.

•The KNN algorithm assumes that similar things exist in proximity. In other words, similar things are near to each other.

•KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics.

•Every case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function.

KNN Algorithm

The KNN Algorithm

1. Load the data
2. Initialize K to your chosen number of neighbors
3. For each example in the data
    3.1 Calculate the distance between the query example and the current example from the data.
    3.2 Add the distance and the index of the example to an ordered collection
4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
5. Pick the first K entries from the sorted collection
6. Get the labels of the selected K entries

How Calculate the distance between the query example and the current example from the data.

- Euclidean distance method is one of the common method to find the position in defined dataset.
- To calculate the Euclidean distance we use the following formula

$$ED = [(X_{ov} - X_{cx})^2 + (Y_{ov} - Y_{cy})^2]^{1/2}$$

where

X_{ov}: observed value of X Attribute

X_{cx}: x value of centriod

Y_{ov}: observed value of Y Attribute

Y_{cy}: Y value of centriod

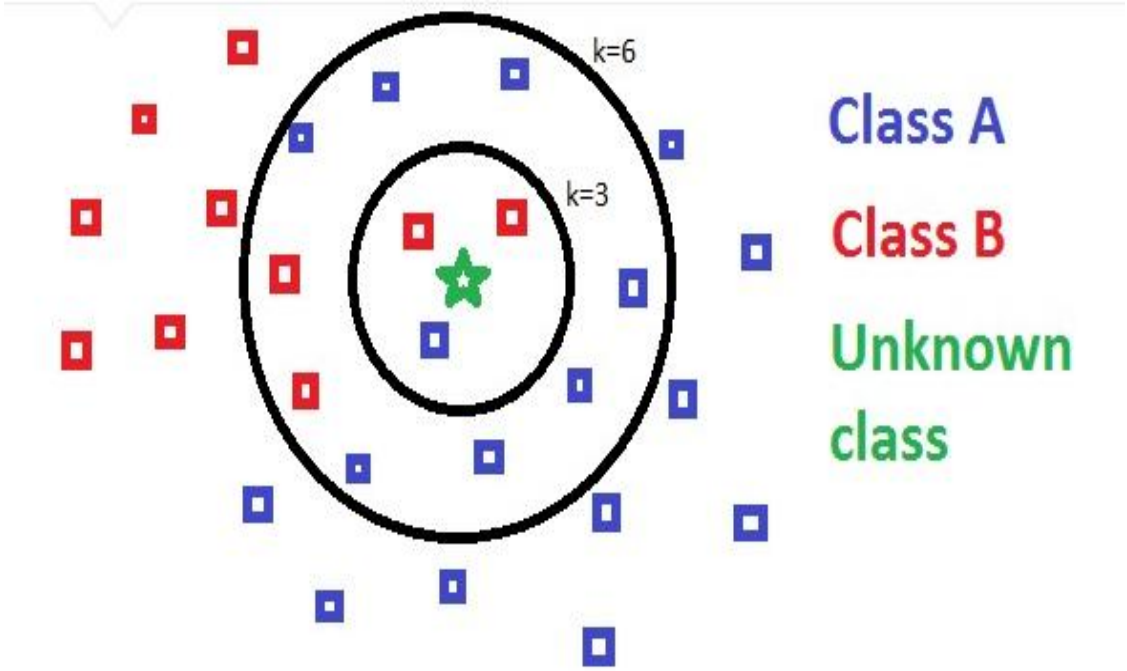
KNN can be summarized as below:

1.Computes the distance between the new data point with every training example.

2.For computing the distance measures such as Euclidean distance, Hamming distance or Manhattan distance will be used.

3.Model picks K entries in the database which are closest to the new data point.

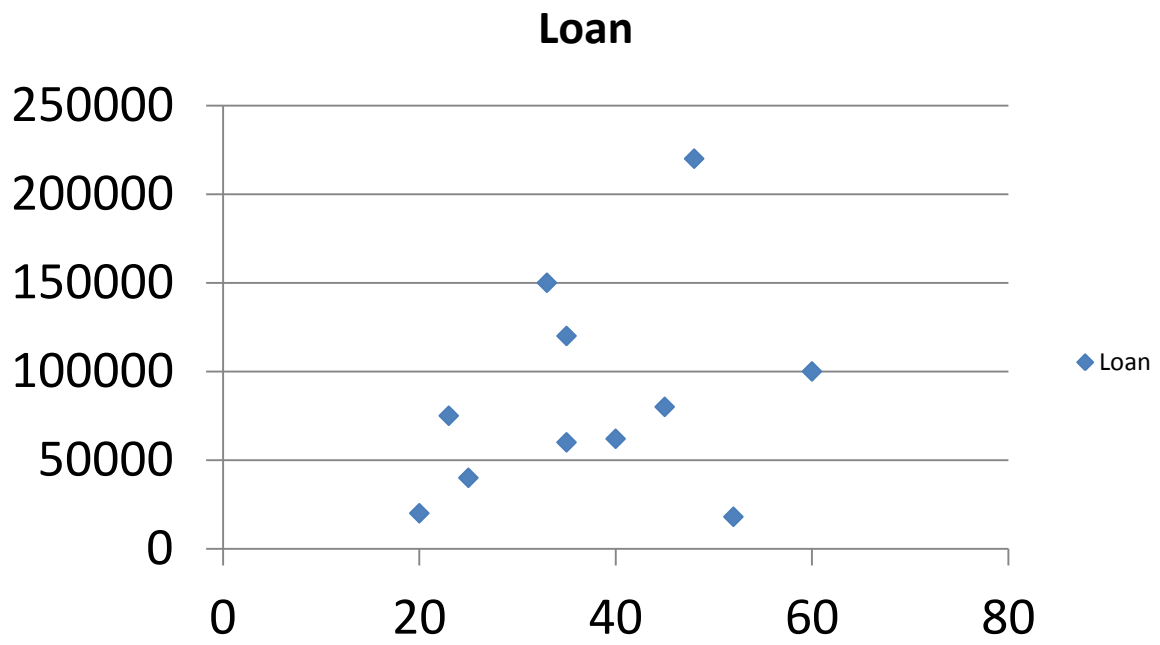
4.Then it does the majority vote i.e the most common class/label among those K entries will be the class of the new data point.



With K=3, Class B will be assigned,
with K=6 Class A will be assigned

Example

Consider the following data concerning credit default. Age and Loan are two numerical variables (predictors) and Default is the target.



classify an unknown case (Age=48 and Loan=$142,000) using Euclidean distance IF K=3

$$D = \text{Sqrt}[(48-33)^2 + (142000-150000)^2] = 8000.01$$

KNN Example

$$ED = [(X_{ov} - X_{cx})^2 + (Y_{ov} - Y_{cy})^2]^{1/2}$$

Maths	Physics	Chemistry	Eligibility	Distance
90	90	90	Y	Sqrt(59)
89	90	90	Y	Sqrt(50)
90	89	89	Y	Sqrt(45)
88	88	89	Y	Sqrt(26)
86	88	88	Y	Sqrt(11)
86	86	88	Y	Sqrt(11)
85	87	86	N	Sqrt(1)
85	86	87	N	Sqrt(5)
84	85	85	N	Sqrt(1)
84	86	87	Y	Sqrt(9)
84	87	88	N	Sqrt(10)
85	87	85	?	
86	86	86	?	

First case K= 3

First case K= 5

KNN Example

Consider the dataset T shirt size

Suppose we have height, weight and T-shirt size of some customers and we need to predict the T-shirt size of a new customer given only height and weight information we have.

Step 1 : Calculate Similarity based on distance function

There are many distance functions but Euclidean is the most commonly used measure. It is mainly used when data is continuous. Manhattan distance is also very common for continuous variables.

Euclidean :

$$d(x,y) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2}$$

Manhattan / city - block :

$$d(x,y) = \sum_{i=1}^{m}|x_i - y_i|$$

The idea to use distance measure is to find the distance (similarity) between new sample and training cases and then finds the k-closest customers to new customer in terms of height and weight.

New customer named "Akhil" has height 161cm and weight 61kg.

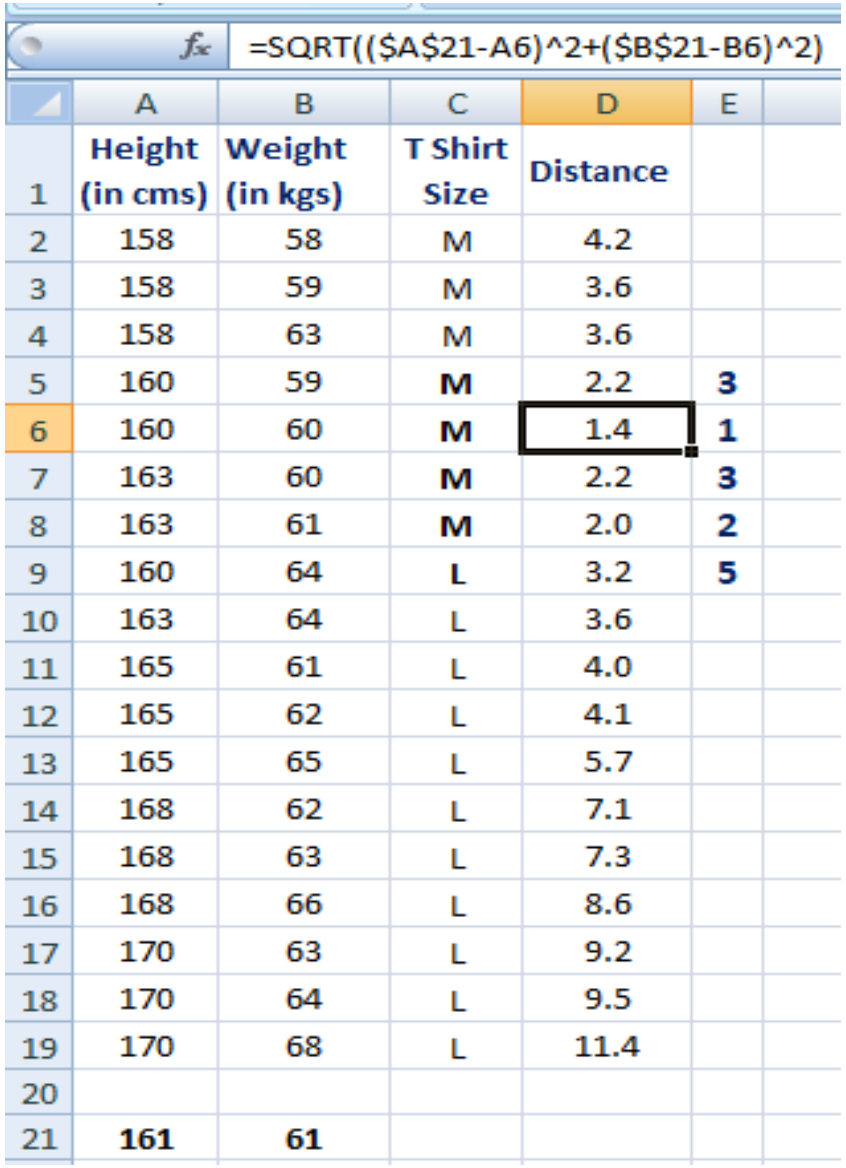
Euclidean distance between first observation and new observation (Akhil) is as follows -

=SQRT((161-158)^2+(61-58)^2)

Similarly, we will calculate distance of all the training cases with new case and calculates the rank in terms of distance. The smallest distance value will be ranked 1 and considered as nearest neighbor.

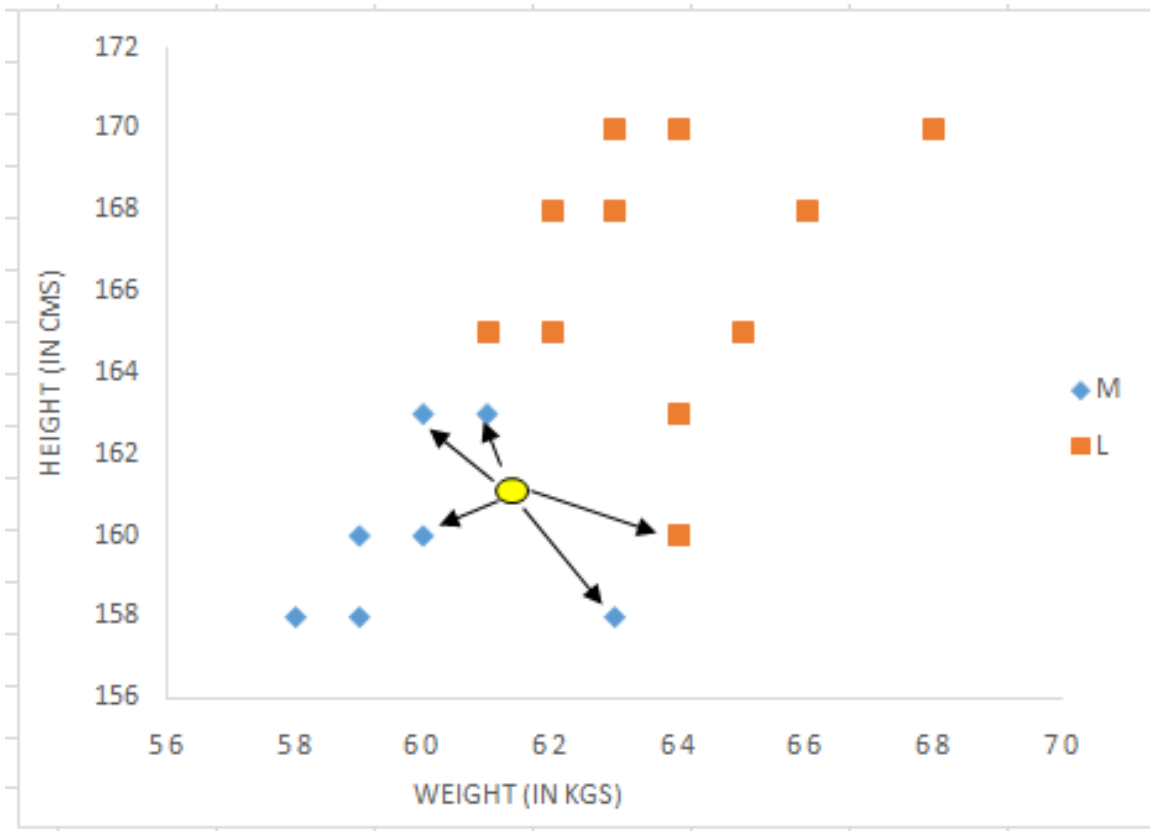
Step 2 : Find K–Nearest Neighbors

Let k be 5. Then the algorithm searches for the 5 customers closest to Akhil



fx =SQRT((A21-A6)^2+(B21-B6)^2)

	A	B	C	D	E
1	Height (in cms)	Weight (in kgs)	T Shirt Size	Distance	
2	158	58	M	4.2	
3	158	59	M	3.6	
4	158	63	M	3.6	
5	160	59	M	2.2	3
6	160	60	M	1.4	1
7	163	60	M	2.2	3
8	163	61	M	2.0	2
9	160	64	L	3.2	5
10	163	64	L	3.6	
11	165	61	L	4.0	
12	165	62	L	4.1	
13	165	65	L	5.7	
14	168	62	L	7.1	
15	168	63	L	7.3	
16	168	66	L	8.6	
17	170	63	L	9.2	
18	170	64	L	9.5	
19	170	68	L	11.4	
20					
21	161	61			



In the graph below, binary dependent variable (T-shirt size) is displayed in blue and orange color. 'Medium T-shirt size' is in blue color and 'Large T-shirt size' in orange color. New customer information is exhibited in yellow circle. Four blue highlighted data points and one orange highlighted data point are close to yellow circle. so the prediction for the new case is blue highlighted data point which is Medium T-shirt size.

Instance-based learning

Instance-based learning refers to a family of techniques for classification and regression, which produce a class label/predication based on the similarity of the query to its nearest neighbor(s) in the training set.

KNN Similarity based learning

- Needed: A feature space representation of the instance in the dataset and a measure of similarity between instances.
- Each instance in the training set is stored in a memory.
- Initial storing is standard however once all training examples are stored a second run needs to determine the distance between the different instances.
- The prediction is based on finding out what class the nearest instance belongs to.

Main Algorithm

Require: a set of training instances

Require: a query instance

1. Iterate across the instances in memory to find the nearest neighbour – this is the instance with the shortest distance across the feature space to the query instance.
2. Make a prediction for the query instance that is equal to the value of the target feature of the nearest neighbour.

Example

Table 5.2

The SPEED and AGILITY ratings for 20 college athletes and whether they were drafted by a professional team.

ID	SPEED	AGILITY	DRAFT	ID	SPEED	AGILITY	DRAFT
1	2.50	6.00	no	11	2.00	2.00	no
2	3.75	8.00	no	12	5.00	2.50	no
3	2.25	5.50	no	13	8.25	8.50	no
4	3.25	8.25	no	14	5.75	8.75	yes
5	2.75	7.50	no	15	4.75	6.25	yes
6	4.50	5.00	no	16	5.50	6.75	yes
7	3.50	5.25	no	17	5.25	9.50	yes
8	3.00	3.25	no	18	7.00	4.25	yes
9	4.00	4.00	no	19	7.50	8.00	yes
10	4.25	3.75	no	20	7.25	5.75	yes

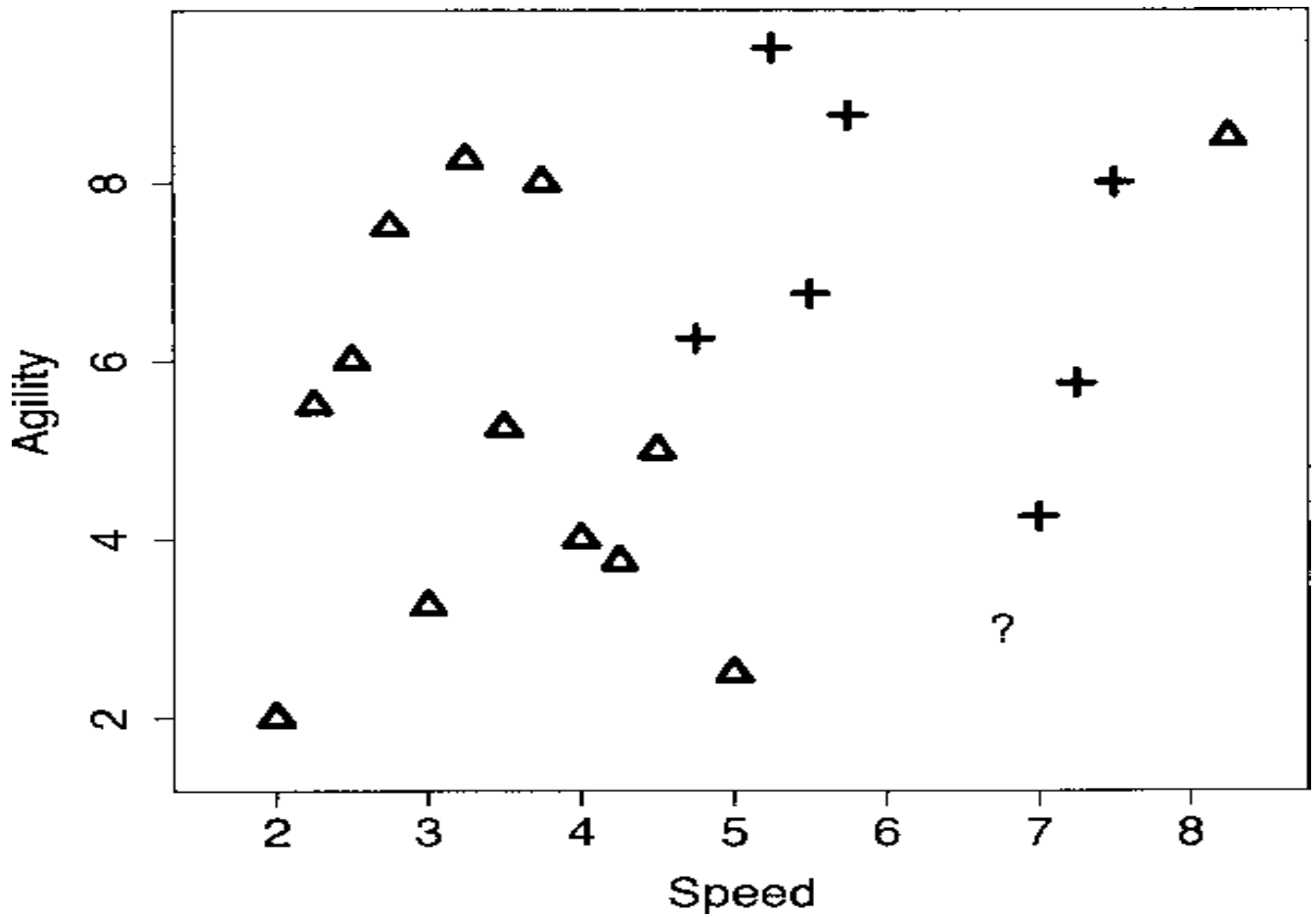
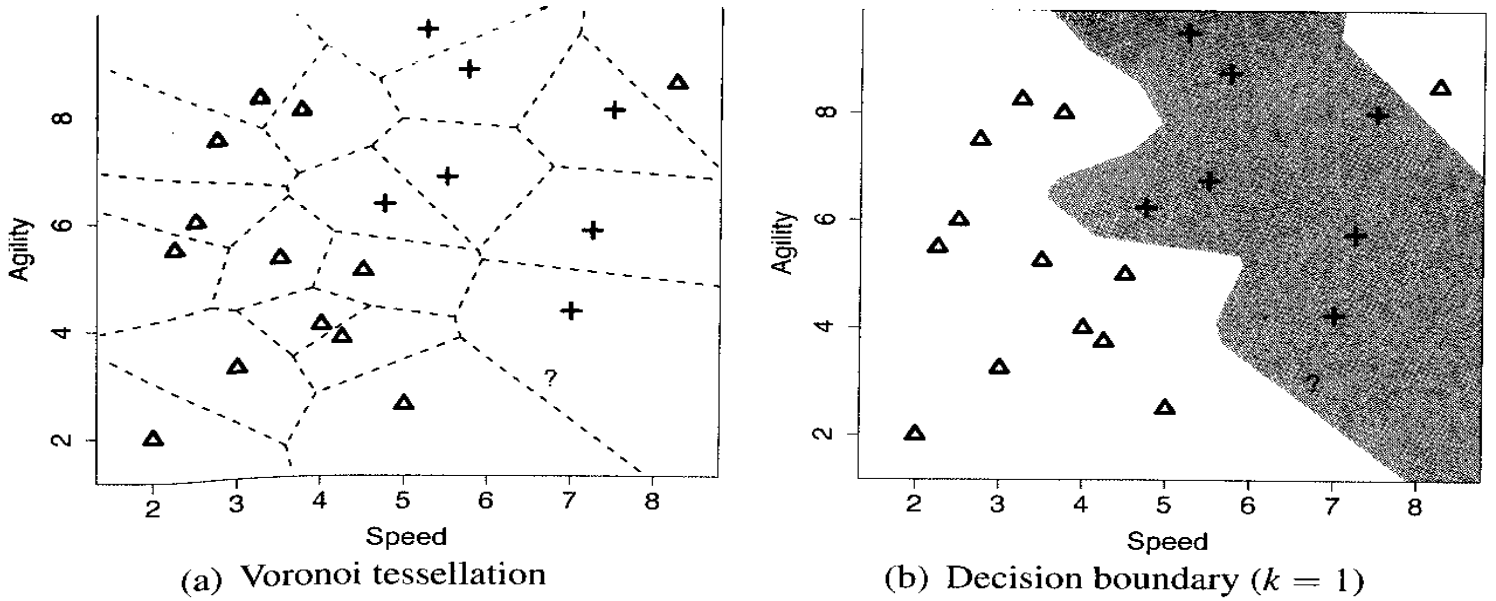

Agility
Speed
2
4
6
8
2
3
4
5
6
7
8
?

Table 5.3

The distances (Dist.) between the query instance with SPEED = 6.75 and AGILITY = 3.00 and each instance in Table 5.2[182].

ID	SPEED	AGILITY	DRAFT	Dist.	ID	SPEED	AGILITY	DRAFT	Dist.
18	7.00	4.25	yes	1.27	11	2.00	2.00	no	4.85
12	5.00	2.50	no	1.82	19	7.50	8.00	yes	5.06
10	4.25	3.75	no	2.61	3	2.25	5.50	no	5.15
20	7.25	5.75	yes	2.80	1	2.50	6.00	no	5.20
9	4.00	4.00	no	2.93	13	8.25	8.50	no	5.70
6	4.50	5.00	no	3.01	2	3.75	8.00	no	5.83
8	3.00	3.25	no	3.76	14	5.75	8.75	yes	5.84
15	4.75	6.25	yes	3.82	5	2.75	7.50	no	6.02
7	3.50	5.25	no	3.95	4	3.25	8.25	no	6.31
16	5.50	6.75	yes	3.95	17	5.25	9.50	yes	6.67

- KNN creates local models (or neighbourhoods) across the feature space with each space defined by a subset of the training data.
- Implicitly a 'global' decision space is created with boundaries between the training data.

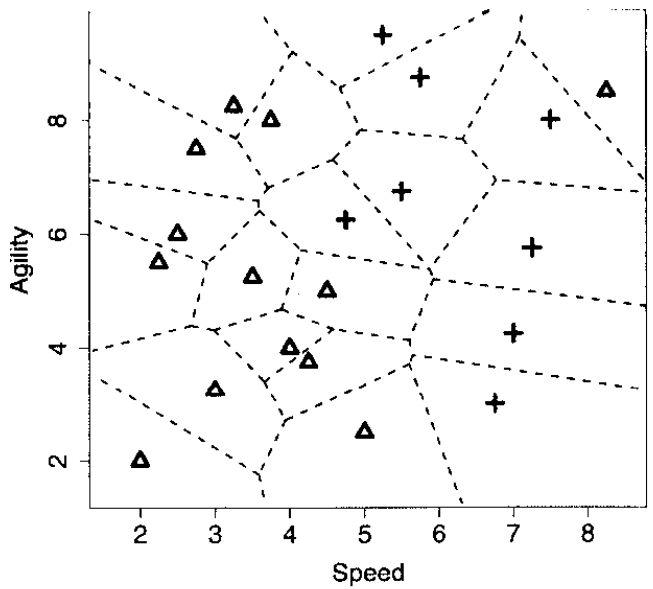


(a) Voronoi tessellation (b) Decision boundary ($k = 1$)

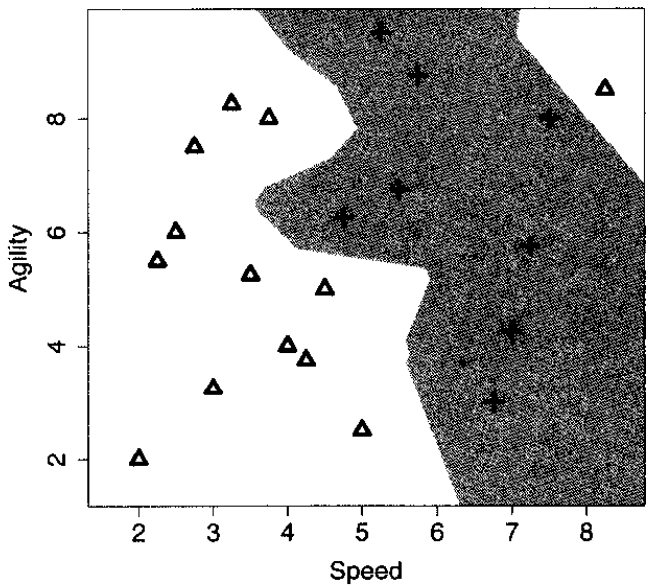
Figure 5.4

(a) The Voronoi tessellation of the feature space for the dataset in Table 5.2[182] with the position of the query represented by the ? marker; (b) the decision boundary created by aggregating the neighboring Voronoi regions that belong to the same target level.

- One advantage of KNN is that updating the decision space is easy.
- KNN is a nearest neighbour algorithm that creates an implicit global classification model by aggregating local models, or neighborhoods.



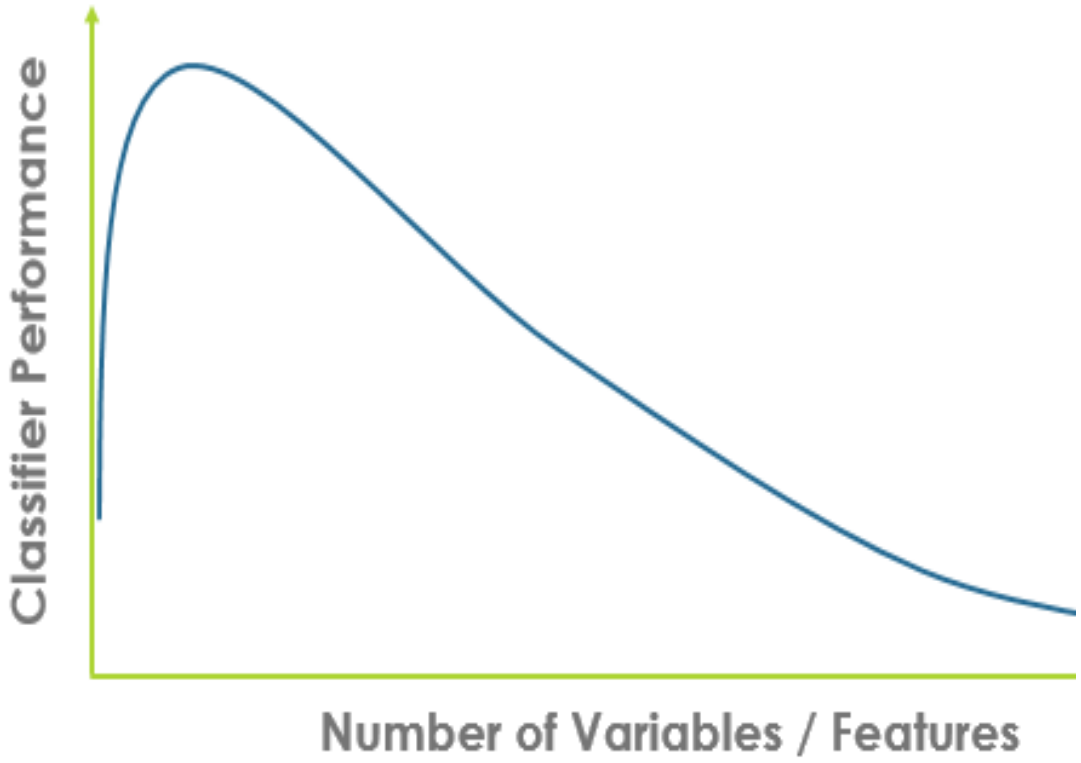
(a) Voronoi tessellation



(b) Decision boundary ($k = 1$)

The Curse of Dimensionality

- The Curse of Dimensionality is termed by mathematician R. Bellman in his book “Dynamic Programming” in 1957.
- The curse of dimensionality is the problem caused by the exponential increase in volume associated with adding extra dimensions to Euclidean space.
- The curse of dimensionality basically means that the error increases with the increase in the number of features.
- A higher number of dimensions theoretically allow more information to be stored, but practically it rarely helps due to the higher possibility of noise and redundancy in the real-world data.
- The curse of dimensionality is one of the most important problems in multivariate machine learning. It appears in many different forms, but all of them have the same net form and source: the fact that points in high-dimensional space are highly sparse.

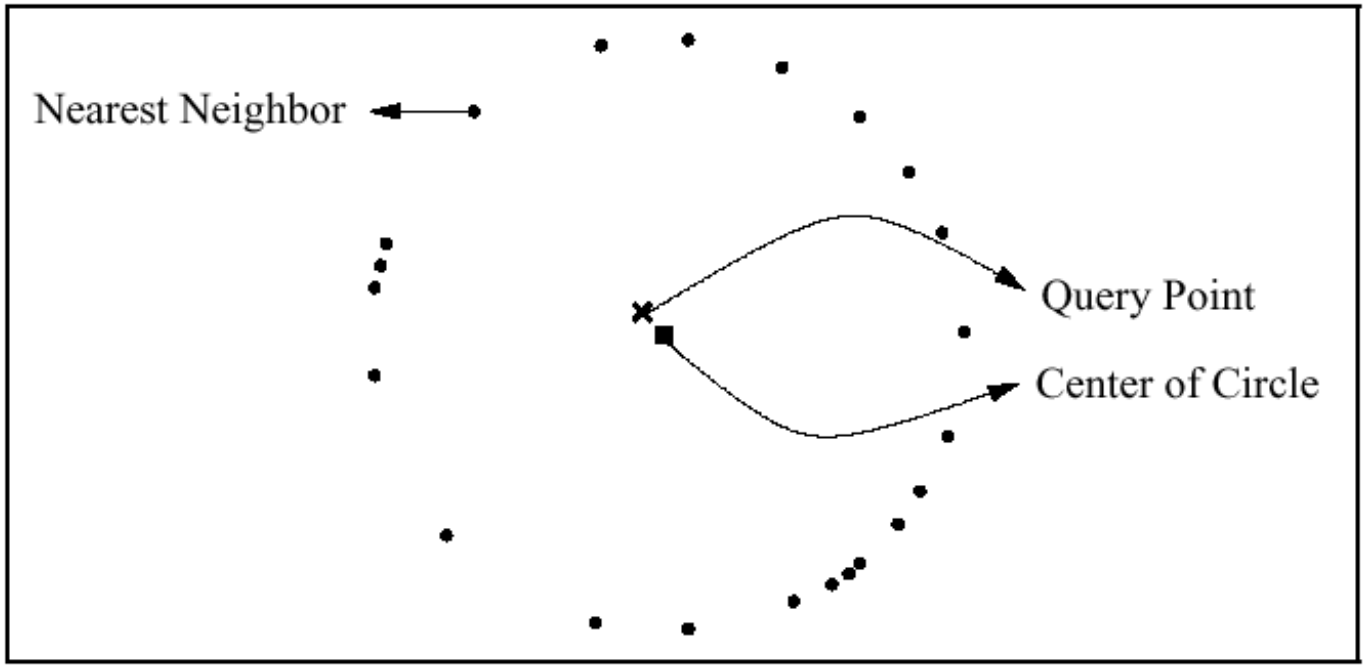


The Curse of Dimensionality

- In machine learning, to catch useful indicators and obtain a more accurate result, we tend to add as many features as possible at first. However, after a certain point, the performance of the model will decrease with the increasing number of elements. This phenomenon is often referred to as “The Curse of Dimensionality.”
- The curse of dimensionality occurs because the sample density decreases exponentially with the increase of the dimensionality. When we keep adding features without increasing the number of training samples as well, the dimensionality of the feature space grows and becomes sparser and sparser. Due to this sparsity, it becomes much easier to find a “perfect” solution for the machine learning model which highly likely leads to overfitting.
- Overfitting happens when the model corresponds too closely to a particular set of data and doesn’t generalize well. An overfitted model would work too well on the training dataset so that it fails on future data and makes the prediction unreliable.

Problems with High Dimensional Data

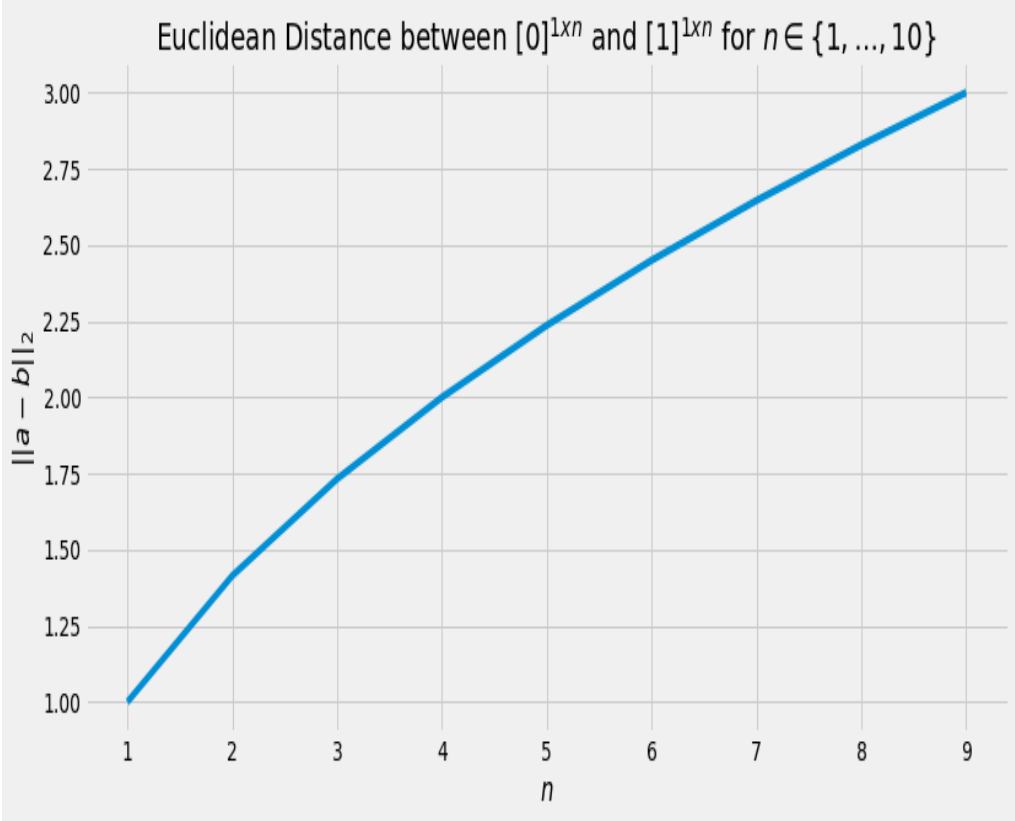
- A point's nearest neighbor (NN) loses meaning



Source: [2]

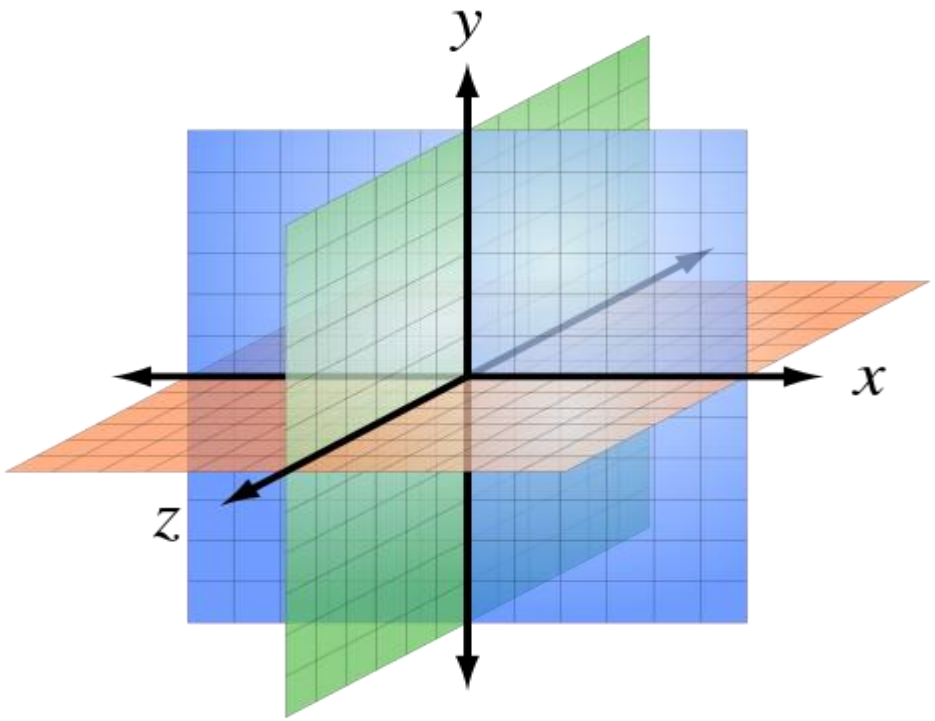
The Curse of Dimensionality

- We Can say Curse of Dimensionality refers to non-intuitive properties of data observed when working in high-dimensional space*, specifically related to usability and interpretation of distances and volumes.
- The simplest possible illustrative example is in just one to two dimensions. Suppose we have two points on a line, 0 and 1. These two points are unit distance away from one another. Suppose we introduce a second axis of "data", again distributed a unit distance away. Now we have two points, (0,0)(0,0) and (1,1)(1,1). But the distance between the points has grown to 2–√2!
- In three dimensions the two points will be 3–√3 away, and so on. By the time we reach 10 dimensions, the two points are now a distance of 3 away from one another: thrice as far as they were when the data was 1-dimensional:



The Curse of Dimensionality

- The curse of dimensionality occurs because the sample density decreases exponentially with the increase of the dimensionality. When we keep adding features without increasing the number of training samples as well, the dimensionality of the feature space grows and becomes sparser and sparser.



Dimensionality reduction

- Dimensionality reduction is the process of reducing the dimensionality of the feature space with consideration by obtaining a set of principal features. Dimensionality reduction can be further broken into feature selection and feature extraction.
- Feature selection tries to select a subset of the original features for use in the machine learning model. In this way, we could remove redundant and irrelevant features without incurring much loss of information.
- Feature extraction is also called feature projection. Whereas feature selection returns a subset of the original features, feature extraction creates new features by projecting the data in the high-dimensional space to a space of fewer dimensions. This approach can also derive informative and non-redundant features.
- In addition to avoiding overfitting and redundancy, dimensionality reduction also leads to better human interpretations and less computational cost with simplification of models.

Technique for reducing Dimensionality

Techniques for data-dimensionality reduction, including:

1. Ratio of missing values
2. Low variance in the column values
3. High correlation between two columns
4. Principle component analysis (PCA)
5. Candidates and split columns in a random forest
6. Backward feature elimination
7. Forward feature construction
8. Linear discriminant analysis (LDA)
9. Neural autoencoder
10. t-distributed stochastic neighbor embedding (t-SNE)

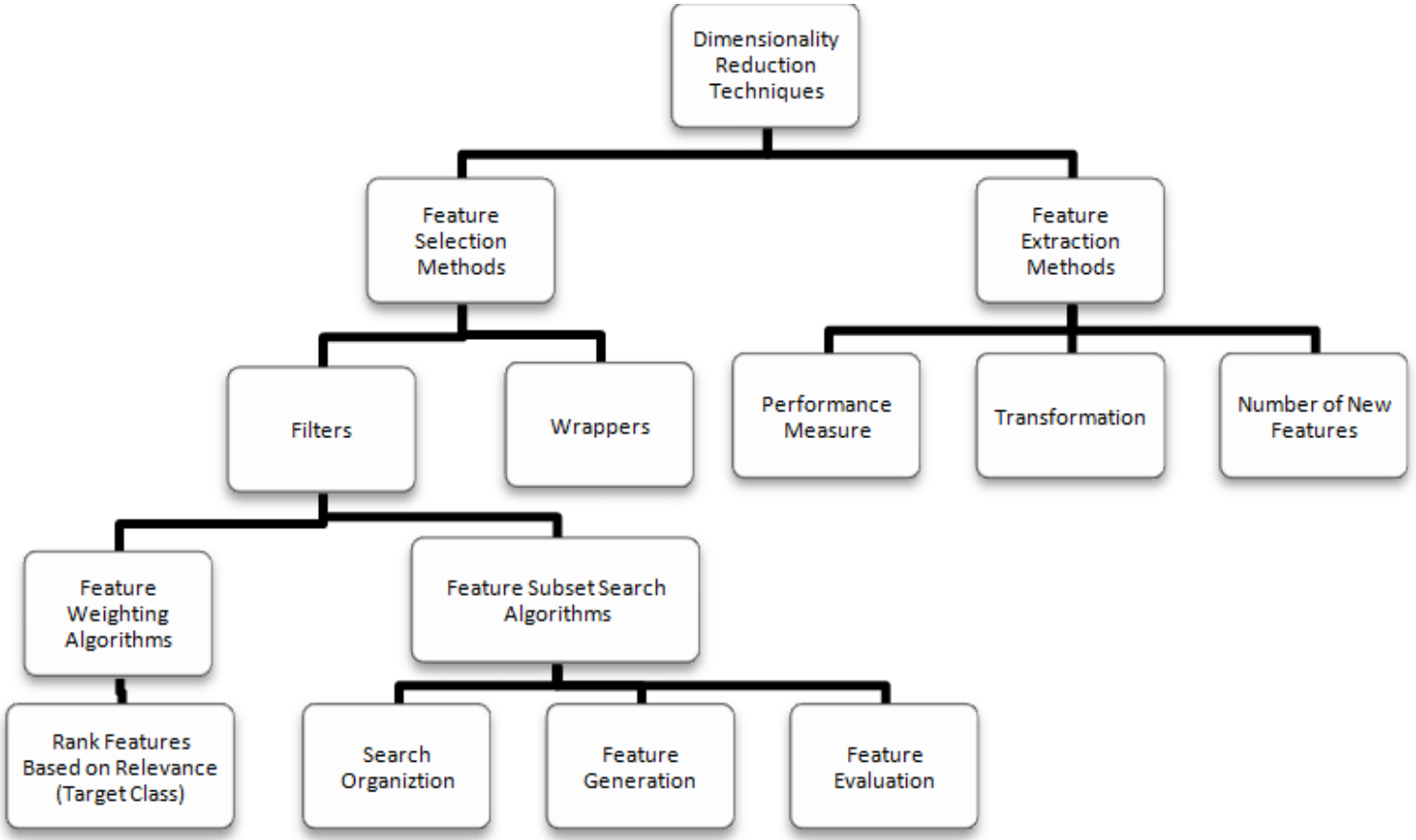
Feature selection & Feature extraction

- Dimensionality reduction as a preprocessing step to machine learning is effective in removing irrelevant and redundant data, increasing learning accuracy, and improving result comprehensibility.
- increase of dimensionality of data poses a severe challenge to many existing feature selection and feature extraction methods with respect to efficiency and effectiveness.
- In the field of machine learning and pattern recognition, dimensionality reduction is important area, where many approaches have been proposed.

Feature selection & Feature extraction

- One of the simplest ways to reduce dimensionality is by Feature Selection; one selects only those input dimensions that contain the relevant information for solving the particular problem.
- Feature Extraction is a more general method in which one tries to develop a transformation of the input space onto the low-dimensional subspace that preserves most of the relevant information.
- Feature extraction and selection methods are used isolated or in combination with the aim to improve performance such as estimated accuracy, visualization and comprehensibility of learned knowledge

Feature selection & Feature extraction



- In feature selection process a subset from available features data are selected for the process of learning algorithm. The best subset is the one with least number of dimensions that most contribute to learning accuracy .

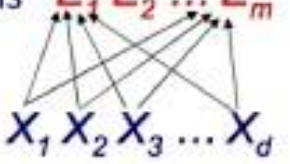
Feature selection & Feature extraction

- Feature extraction is generally used to mean the construction of linear combinations αTx of continuous features which have good discriminatory power between classes“.

Dimensionality reduction

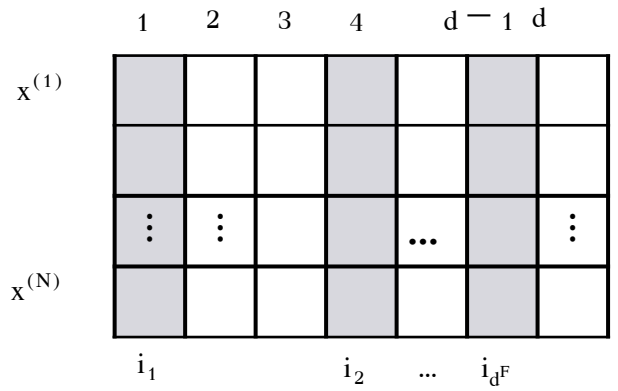
- Goal: represent instances with fewer variables
  - try to preserve as much structure in the data as possible
  - discriminative: only structure that affects class separability
- Feature selection
  - pick a subset of the original dimensions $X_1\ X_2\ X_3 \ldots X_{d-1}\ X_d$
  - discriminative: pick good class “predictors” (e.g. gain)
- Feature extraction
  - construct a new set of dimensions $E_1\ E_2 \ldots E_m$

    $E_i = f(X_1 \ldots X_d)$
  - (linear) combinations of original $X_1\ X_2\ X_3 \ldots X_d$



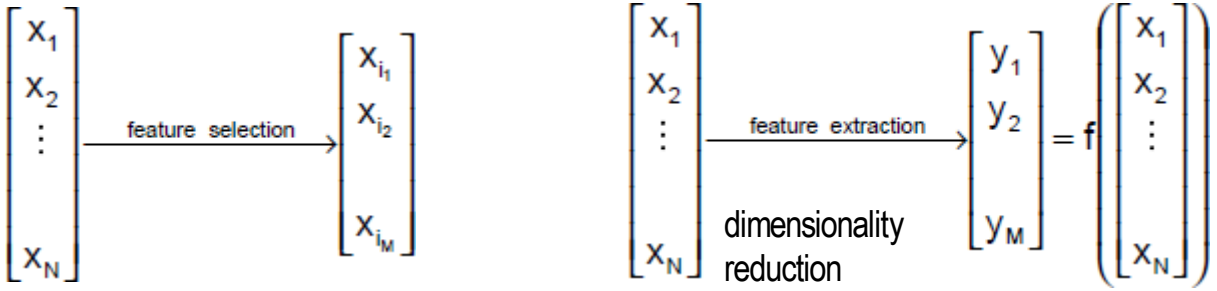
Feature Selection

- Data may contain many irrelevant and redundant variables and often comparably few training examples
- Application for which datasets with tens or hundreds or thousands of variables are available
  - text or document processing, gene expression array analysis



Feature Selection

- Given a set of **n** features, the goal of **feature selection** is to select a subset of **d** features (**d** < **n**) in order to minimize the classification error.

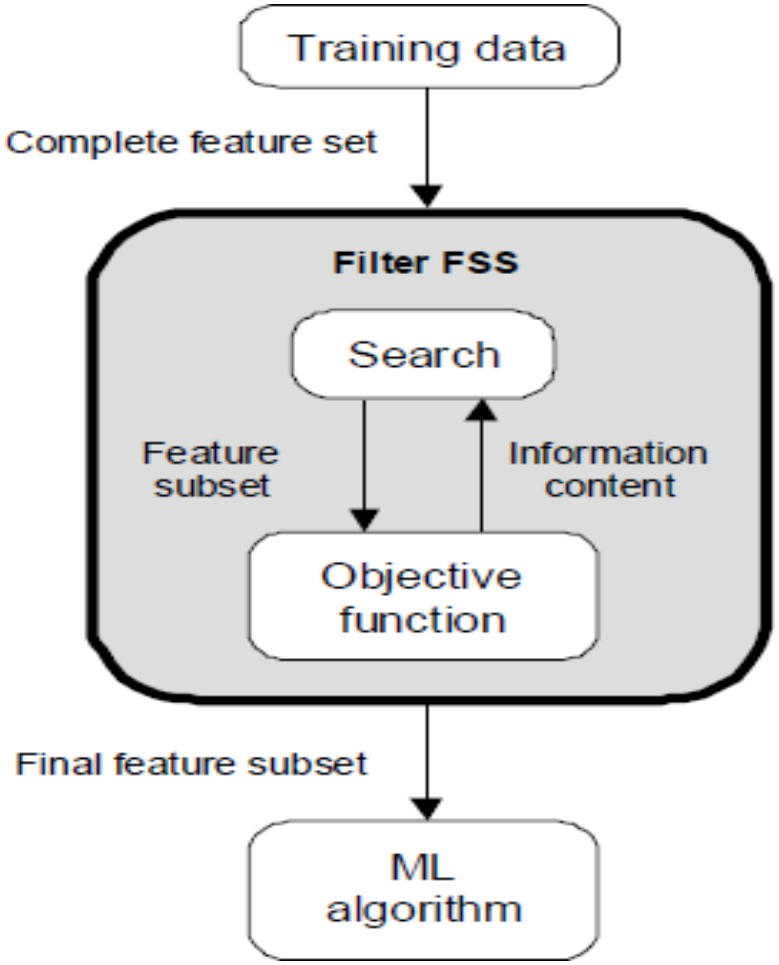


- Why perform feature selection?
  - Data interpretation\knowledge discovery (insights into which factors which are most **representative** of your problem)
  - Curse of dimensionality (amount of data grows **exponentially** with # of features $O(2^n)$)

- Fundamentally different from dimensionality reduction (we will discuss next time) based on feature combinations (i.e., **feature extraction**).

Feature Selection Steps

- Feature selection is an **optimization** problem.
  - Step 1: Search the space of possible feature subsets.
  - Step 2: Pick the subset that is optimal or near-optimal with respect to some objective function.



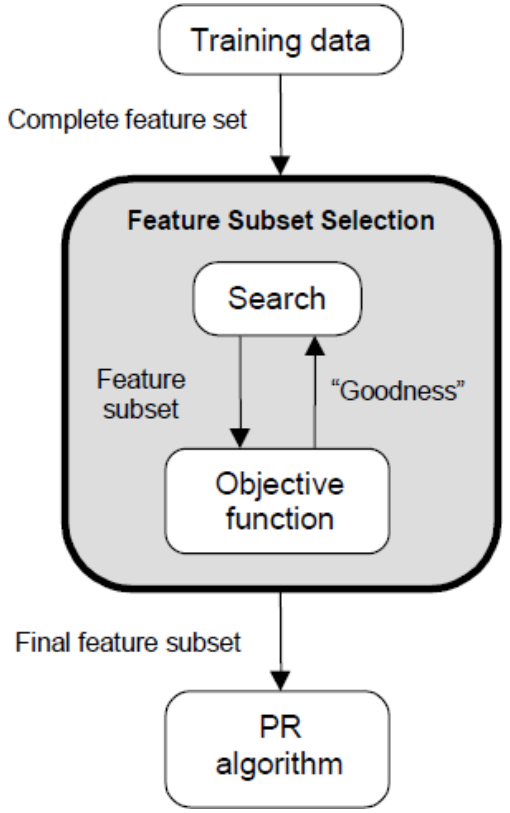
Feature Selection Steps (cont'd)

Search strategies

- Optimal
- Heuristic

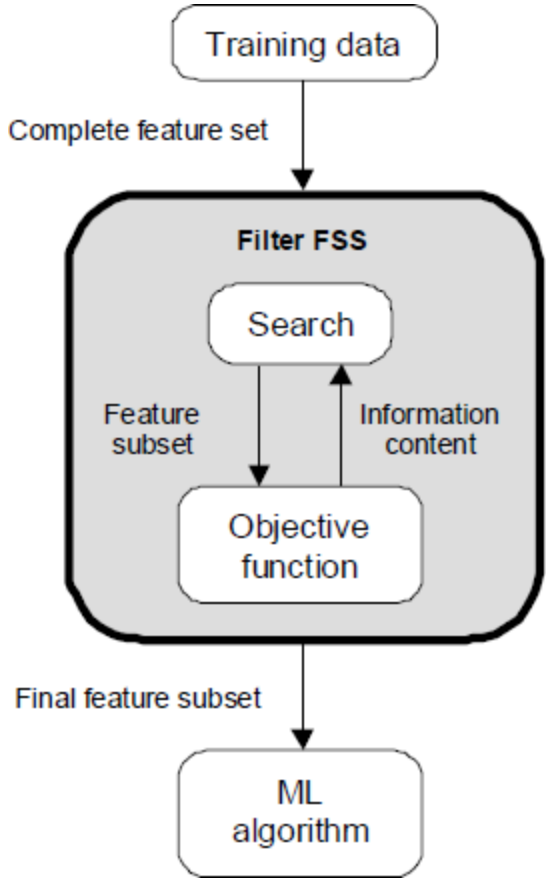
Evaluation strategies

- Filter methods
- Wrapper methods



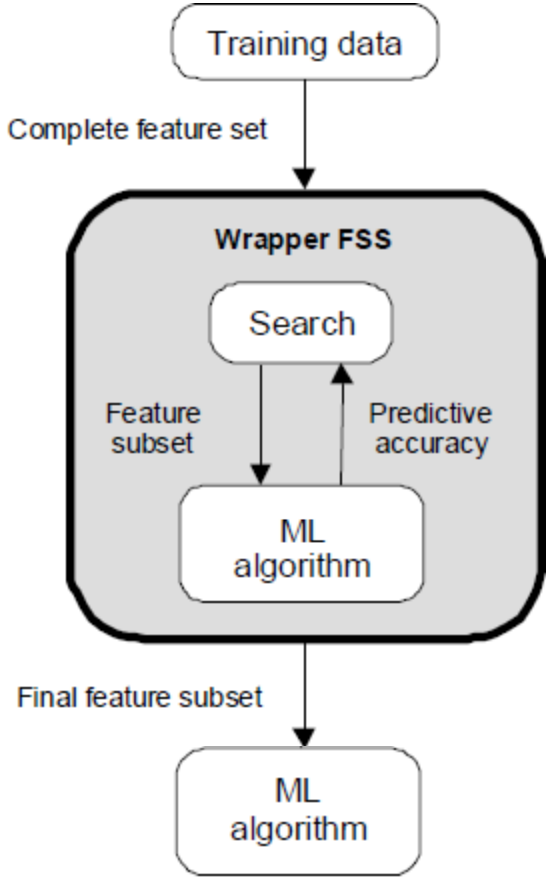
Evaluation Strategies

- Filter Methods
  - Evaluation is **independent** of the classification algorithm.
  - The objective function evaluates feature subsets by their information content, typically interclass distance, statistical dependence or information-theoretic measures.



Evaluation Strategies

- Wrapper Methods
  - Evaluation uses criteria **related** to the classification algorithm.
  - The objective function is a pattern classifier, which evaluates feature subsets by their predictive accuracy (recognition rate on test data) by statistical resampling or cross-validation.



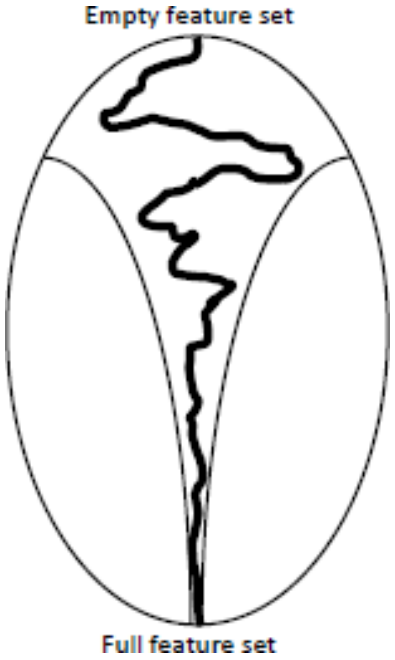
Filter vs. Wrapper Approaches

- **Filters**
  - Advantages
    - **Fast execution**: Filters generally involve a non-iterative computation on the dataset, which can execute much faster than a classifier training session
    - **Generality**: Since filters evaluate the intrinsic properties of the data, rather than their interactions with a particular classifier, their results exhibit more generality: the solution will be "good" for a larger family of classifiers
  - Disadvantages
    - **Tendency to select large subsets**: Since the filter objective functions are generally monotonic, the filter tends to select the full feature set as the optimal solution. This forces the user to select an arbitrary cutoff on the number of features to be selected

Sequential forward selection (SFS)

(heuristic search)

- First, the best **single** feature is selected (i.e., using some criterion function).
- Then, **pairs** of features are formed using one of the remaining features and this best feature, and the best pair is selected.
- Next, **triplets** of features are formed using one of the remaining features and these two best features, and the best triplet is selected.
- This procedure continues until a predefined number of features are selected.



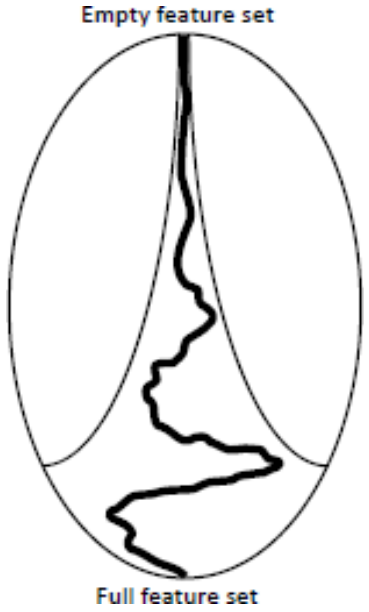
1. Start with the empty set $Y_0 = \{\emptyset\}$
2. Select the next best feature $x^+ = \arg\max_{x \notin Y_k} J(Y_k + x)$
3. Update $Y_{k+1} = Y_k + x^+;\ k = k + 1$
4. Go to 2

SFS performs best when the optimal subset is small.

Sequential backward selection (SBS)

(heuristic search)

- First, the criterion function is computed for all **n** features.
- Then, each feature is deleted one at a time, the criterion function is computed for all subsets with **n-1** features, and the worst feature is discarded.
- Next, each feature among the remaining **n-1** is deleted one at a time, and the worst feature is discarded to form a subset with **n-2** features.
- This procedure continues until a predefined number of features are left.



1. Start with the full set $Y_0 = X$
2. Remove the worst feature $x^- = \arg\max_{x \in Y_k} J(Y_k - x)$
3. Update $Y_{k+1} = Y_k - x^-;\ k = k + 1$
4. Go to 2

SBS performs best when the optimal subset is large.

Bidirectional Search (BDS)

- BDS applies SFS and SBS simultaneously:
  - SFS is performed from the empty set.
  - SBS is performed from the full set.
- To guarantee that SFS and SBS converge to the same solution:
  - Features already selected by SFS are not removed by SBS.
  - Features already removed by SBS are not added by SFS.

1. Start SFS with $Y_F = \{\emptyset\}$
2. Start SBS with $Y_B = X$
3. Select the best feature

$$x^+ = \underset{\substack{x \notin Y_{F_k} \\ x \in F_{B_k}}}{\arg\max} J(Y_{F_k} + x)$$

$$Y_{F_{k+1}} = Y_{F_k} + x^+$$

4. Remove the worst feature

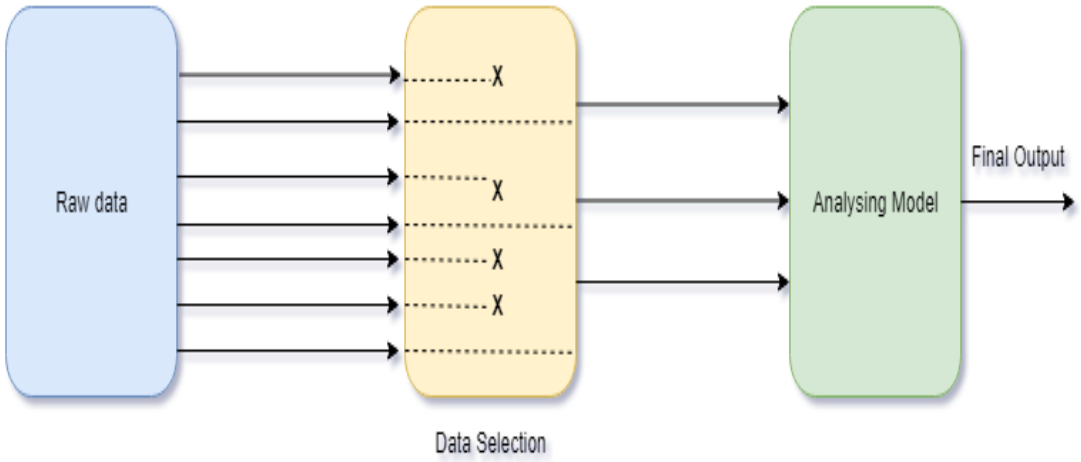
$$x^- = \underset{\substack{x \in Y_{B_k} \\ x \notin Y_{F_{k+1}}}}{\arg\max} J(Y_{B_k} - x)$$

$$Y_{B_{k+1}} = Y_{B_k} - x^-;\ k = k + 1$$

5. Go to 2

Feature Selection Techniques

- As a stand-alone task, feature selection can be unsupervised (e.g. Variance Thresholds) or supervised (e.g. Genetic Algorithms). You can also combine multiple methods if needed.



Feature Selection Techniques

Variance Thresholds:

This technique looks for the variance from one observation to another of a given feature and then if the variance is not different in each observation according to the given threshold, Features that don’t change much don’t add much effective information. Using variance thresholds is an easy and relatively safe way to reduce dimensionality at the start of your modeling process.

But this alone will not be sufficient if you want to reduce the dimensions as it’s highly subjective and you need to tune the variance threshold manually.

Feature Selection Techniques

Correlation Thresholds:

the features are considered and checked whether those features are correlated to each other closely. If they are, the overall effect to the final output of both features would be similar even to the result we get when we used one of those features.

Which one should you remove?

Feature Selection Techniques

Feature Selection using Genetic Algorithms:

In machine learning, one of the uses of genetic algorithms is to pick up the right number of variables in order to create a predictive model.

To pick up the right subset of variables is a problem of combinatory and optimization. The advantage of this technique over others is, it allows the best solution to emerge from the best of prior solutions. An evolutionary algorithm which improves the selection over time.

The idea of GA is to combine the different solutions generation after generation to extract the best genes (variables) from each one. That way it creates new and more fitted individuals.

Feature Selection Techniques

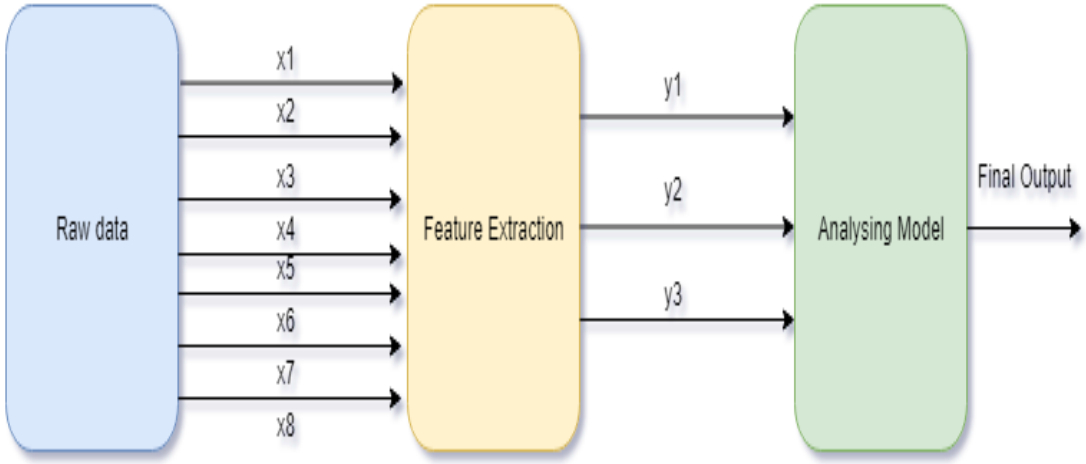
Feature Selection using Stepwise Regression:

In statistics, stepwise regression is a method of fitting regression models in which the choice of predictive variables is carried out by an automatic procedure. In each step, a variable is considered for addition to or subtraction from the set of explanatory variables based on some prespecified criterion.

This has two flavors: forward and backward. For forward stepwise search, you start without any features. Then, you'd train a 1-feature model using each of your candidate features and keep the version with the best performance. You'd continue adding features, one at a time, until your performance improvements stall. Backward stepwise search is the same process, just reversed: start with all features in your model and then remove one at a time until performance starts to drop substantially.

Feature Extraction Techniques

- Feature extraction is for creating a new, smaller set of features that still captures most of the useful information. This can come as supervised(e.g. LDA) and unsupervised(e.g. PCA) methods.



Dimensionality Reduction

- Dimensionality reduction is the process of reducing the number of random variables or attributes under consideration. High-dimensionality data reduction, as part of a data pre-processing-step, is extremely important in many real-world applications.
- High-dimensionality reduction has emerged as one of the significant tasks in data mining applications.
- Consider one example, you may have a dataset with hundreds of features (columns in your database). Then dimensionality reduction is that you reduce those features of attributes of data by combining or merging them in such a way that it will not loose much of the significant characteristics of the original dataset.

Components of Dimensionality Reduction

There are two components of dimensionality reduction:

a. Feature selection

- In this, we need to find a subset of the original set of variables. Also, need a subset which we use to model the problem. It usually involves three ways:
  - Filter
  - Wrapper
  - Embedded

b. Feature Extraction

- We use this, to reduces the data in a high dimensional space to a lower dimension space, i.e. a space with lesser no. of dimensions.

Dimensionality Reduction Methods

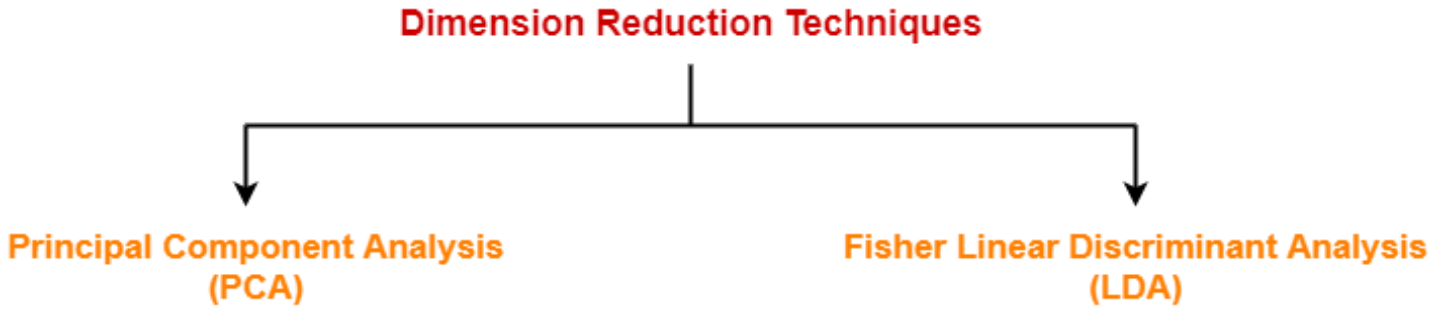
The various methods used for dimensionality reduction include:

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Generalized Discriminant Analysis (GDA)

Dimensionality reduction may be both linear or non-linear, depending upon the method used.

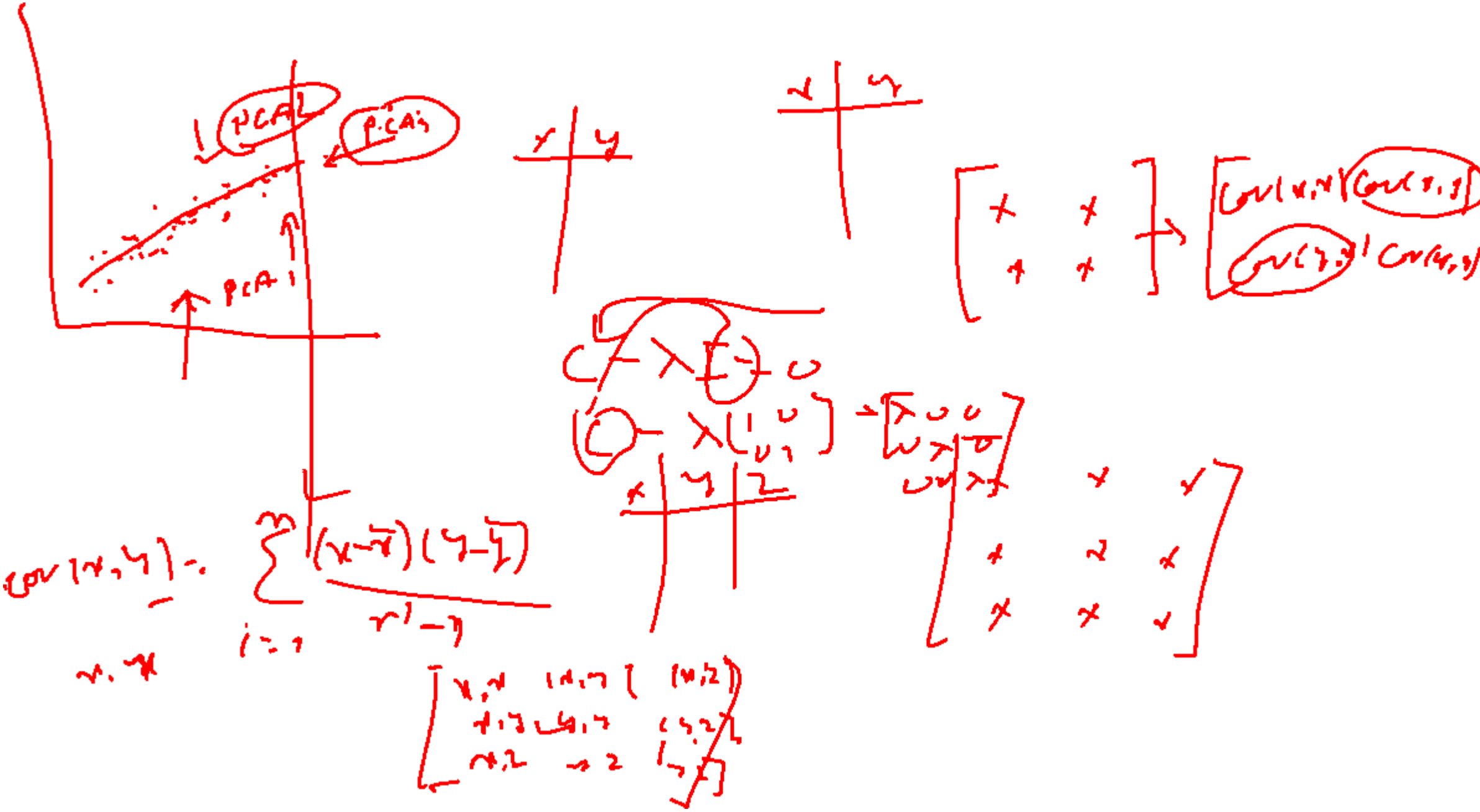
Dimensionality Reduction Methods

- The two popular and well-known dimension reduction techniques are-



Principal Component Analysis (PCA)

- Principal Component Analysis is a well-known dimension reduction technique.
- It transforms the variables into a new set of variables called as principal components.
- These principal components are linear combination of original variables and are orthogonal.
- The first principal component accounts for most of the possible variation of original data.
- The second principal component does its best to capture the variance in the data.
- There can be only two principal components for a two-dimensional data set.


PCA2
PCA1
PCA1

Principal Component Analysis (PCA)

PCA is a dimensionality reduction that identifies important relationships in our data, transforms the existing data based on these relationships, and then quantifies the importance of these relationships so we can keep the most important relationships. To remember this definition, we can break it down into four steps:

1. We identify the relationship among features through a Covariance Matrix.
2. Through the linear transformation or eigendecomposition of the Covariance Matrix, we get eigenvectors and eigenvalues.
3. Then we transform our data using Eigenvectors into principal components.
4. Lastly, we quantify the importance of these relationships using Eigenvalues and keep the important principal components.

Principal Component Analysis (PCA)

The steps involved in PCA Algorithm are as follows-

Step-01: Get data.

Step-02: Compute the mean vector (μ).

Step-03: Subtract mean from the given data.

Step-04: Calculate the covariance matrix.

Step-05: Calculate the eigen vectors and eigen values of the covariance matrix.

Step-06: Choosing components and forming a feature vector.

Step-07: Deriving the new data set.

Example

Consider the two-dimensional patterns (2, 1), (3, 5), (4, 3), (5, 6), (6, 7), (7, 8). Compute the principal component using PCA Algorithm.

Or

Compute the principal component of following data-

CLASS 1

X = 2 , 3 , 4

Y = 1 , 5 , 3

CLASS 2

X = 5 , 6 , 7

Y = 6 , 7 , 8

Solution

Consider the two-dimensional patterns

(2, 1), (3, 5), (4, 3), (5, 6), (6, 7), (7, 8).

Compute the principal component using PCA Algorithm.

Or

Compute the principal component of following data-

CLASS 1

X = 2 , 3 , 4

Y = 1 , 5 , 3

CLASS 2

X = 5 , 6 , 7

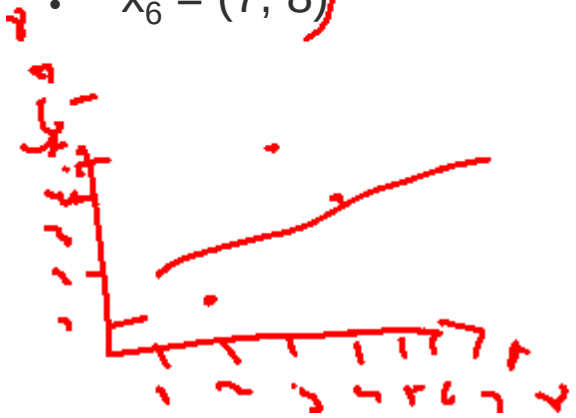
Y = 6 , 7 , 8

Step-01:

Get data.

The given feature vectors are-

- $x_1 = (2, 1)$
- $x_2 = (3, 5)$
- $x_3 = (4, 3)$
- $x_4 = (5, 6)$
- $x_5 = (6, 7)$
- $x_6 = (7, 8)$



Step-02:

Calculate the mean vector (μ).

Mean vector (μ)
= ((2 + 3 + 4 + 5 + 6 + 7) / 6,
(1 + 5 + 3 + 6 + 7 + 8) / 6)
= (4.5, 5)

Thus,

$$\text{Mean vector } (\mu) = \begin{bmatrix} 4.5 \\ 5 \end{bmatrix}$$

Solution

Step-01:

Get data.

Thegiven feature vectors are-

- $x_1 = (2, 1)$
- $x_2 = (3, 5)$
- $x_3 = (4, 3)$
- $x_4 = (5, 6)$
- $x_5 = (6, 7)$
- $x_6 = (7, 8)$

Step-02:

Calculate the mean vector (μ).

Mean vector (μ)
= ((2 + 3 + 4 + 5 + 6 + 7) / 6,
(1 + 5 + 3 + 6 + 7 + 8) / 6)
= (4.5, 5)

Thus,

$$\text{Mean vector } (\mu) = \begin{bmatrix} 4.5 \\ 5 \end{bmatrix}$$

Step-03:

Subtract mean vector (μ) from the given feature vectors.

- $x_1 - \mu = (2 - 4.5, 1 - 5) = (-2.5, -4)$
- $x_2 - \mu = (3 - 4.5, 5 - 5) = (-1.5, 0)$
- $x_3 - \mu = (4 - 4.5, 3 - 5) = (-0.5, -2)$
- $x_4 - \mu = (5 - 4.5, 6 - 5) = (0.5, 1)$
- $x_5 - \mu = (6 - 4.5, 7 - 5) = (1.5, 2)$
- $x_6 - \mu = (7 - 4.5, 8 - 5) = (2.5, 3)$

$$\begin{bmatrix} -2.5 \\ -4 \end{bmatrix} \begin{bmatrix} -1.5 \\ 0 \end{bmatrix} \begin{bmatrix} -0.5 \\ -2 \end{bmatrix} \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \begin{bmatrix} 1.5 \\ 2 \end{bmatrix} \begin{bmatrix} 2.5 \\ 3 \end{bmatrix}$$

Solution

Step-03:

Subtract mean vector (μ) from the given feature vectors.

- $x_1 - \mu = (2 - 4.5, 1 - 5) = (-2.5, -4)$
- $x_2 - \mu = (3 - 4.5, 5 - 5) = (-1.5, 0)$
- $x_3 - \mu = (4 - 4.5, 3 - 5) = (-0.5, -2)$
- $x_4 - \mu = (5 - 4.5, 6 - 5) = (0.5, 1)$
- $x_5 - \mu = (6 - 4.5, 7 - 5) = (1.5, 2)$
- $x_6 - \mu = (7 - 4.5, 8 - 5) = (2.5, 3)$

$$\begin{bmatrix} -2.5 \\ -4 \end{bmatrix} \begin{bmatrix} -1.5 \\ 0 \end{bmatrix} \begin{bmatrix} -0.5 \\ -2 \end{bmatrix} \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \begin{bmatrix} 1.5 \\ 2 \end{bmatrix} \begin{bmatrix} 2.5 \\ 3 \end{bmatrix}$$

Step-04:

Calculate the covariance matrix

$$\text{Covariance Matrix} = \frac{\sum (x_i - \mu)(x_i - \mu)^t}{n}$$

$$\text{Covariance Matrix} = \frac{1}{6} \begin{bmatrix} 17.5 & 22 \\ 22 & 34 \end{bmatrix}$$

$$\text{Covariance Matrix} = \begin{bmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{bmatrix}$$

Step-05:

Calculate the eigen values and eigen vectors of the covariance matrix.

λ is an eigen value for a matrix M if it is a solution of the characteristic equation $|M - \lambda I| = 0$.

So, we have-

$$\begin{vmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{vmatrix} - \begin{vmatrix} \lambda & 0 \\ 0 & \lambda \end{vmatrix} = 0$$

$$\begin{vmatrix} 2.92 - \lambda & 3.67 \\ 3.67 & 5.67 - \lambda \end{vmatrix} = 0$$

Solution

Step-05:

Calculate the eigen values and eigen vectors of the covariance matrix.

λ is an eigen value for a matrix M if it is a solution of the characteristic equation $|M - \lambda I| = 0$.

So, we have-

$$\begin{vmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{vmatrix} - \begin{vmatrix} \lambda & 0 \\ 0 & \lambda \end{vmatrix} = 0$$

$$\begin{vmatrix} 2.92 - \lambda & 3.67 \\ 3.67 & 5.67 - \lambda \end{vmatrix} = 0$$

From here,

$$(2.92 - \lambda)(5.67 - \lambda) - (3.67 \times 3.67) = 0$$

$$16.56 - 2.92\lambda - 5.67\lambda + \lambda^2 - 13.47 = 0$$

$$\lambda^2 - 8.59\lambda + 3.09 = 0$$

Solving this quadratic equation, we get

$$\lambda = 8.22, 0.38$$

Thus, two eigen values are $\lambda_1 = 8.22$ and $\lambda_2 = 0.38$.

We use the following equation to find the eigen vector-

$$MX = \lambda X$$

where-

- M = Covariance Matrix
- X = Eigen vector
- λ = Eigen value

Solution

Step-05:

Calculate the eigen values and eigen vectors of the covariance matrix.

λ is an eigen value for a matrix M if it is a solution of the characteristic equation $|M – \lambda I| = 0$.

So, we have-

$$\begin{vmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{vmatrix} - \begin{vmatrix} \lambda & 0 \\ 0 & \lambda \end{vmatrix} = 0$$

$$\begin{vmatrix} 2.92 - \lambda & 3.67 \\ 3.67 & 5.67 - \lambda \end{vmatrix} = 0$$

From here,

$(2.92 – \lambda)(5.67 – \lambda) – (3.67 \times 3.67) = 0$

$16.56 – 2.92\lambda – 5.67\lambda + \lambda^2 – 13.47 = 0$

$\lambda^2 – 8.59\lambda + 3.09 = 0$

Solving this quadratic equation, we get

$\lambda = 8.22, 0.38$

Thus, two eigen values are $\lambda_1 = 8.22$ and $\lambda_2 = 0.38$.

We use the following equation to find the eigen vector-

$$MX = \lambda X$$

where-

- M = Covariance Matrix
- X = Eigen vector
- λ = Eigen value

$$\begin{bmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{bmatrix} \begin{bmatrix} X1 \\ X2 \end{bmatrix} = 8.22 \begin{bmatrix} X1 \\ X2 \end{bmatrix}$$

Solving these, we get-

$2.92X_1 + 3.67X_2 = 8.22X_1$

$3.67X_1 + 5.67X_2 = 8.22X_2$

On simplification, we get-

$5.3X_1 = 3.67X_2$ ……(1)

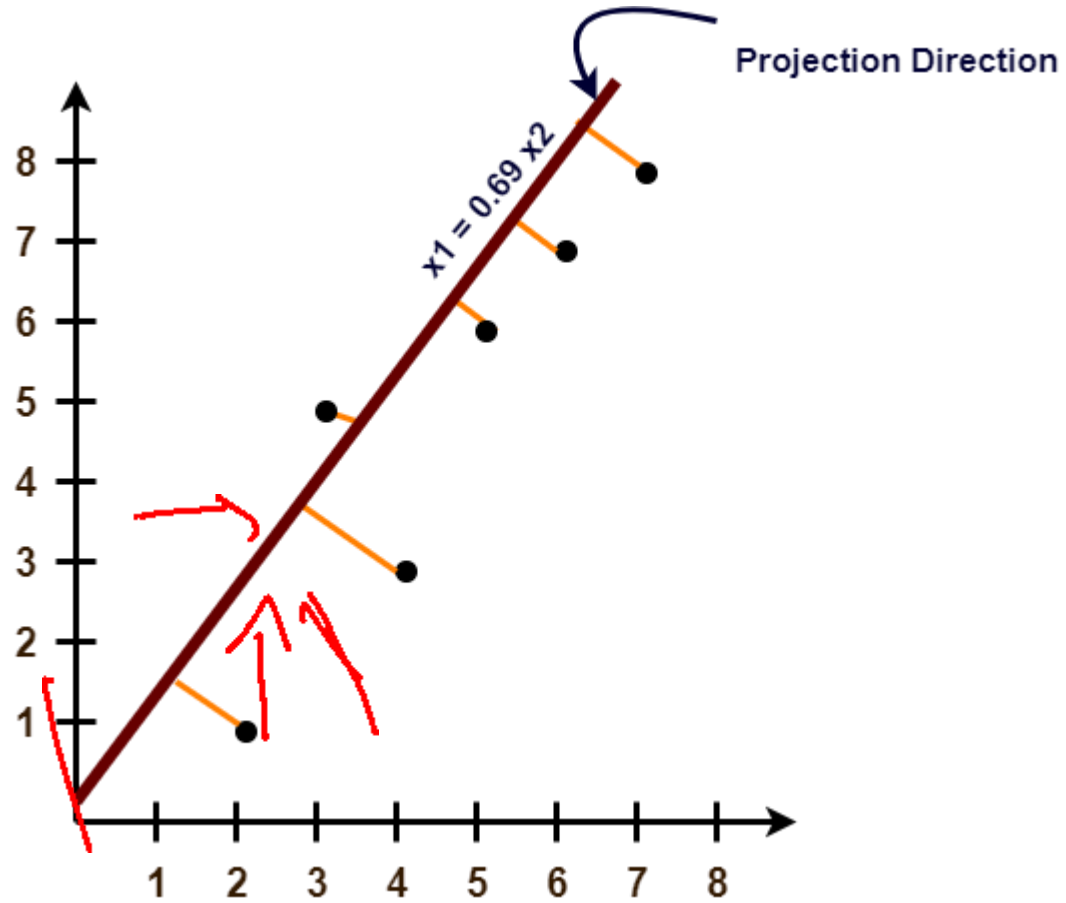
$3.67X_1 = 2.55X_2$ ……(2)

From (1) and (2), $\mathbf{X_1 = 0.69X_2}$

Eigen Vector : $\begin{bmatrix} X1 \\ X2 \end{bmatrix} = \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}$

principal component for the given data set is-

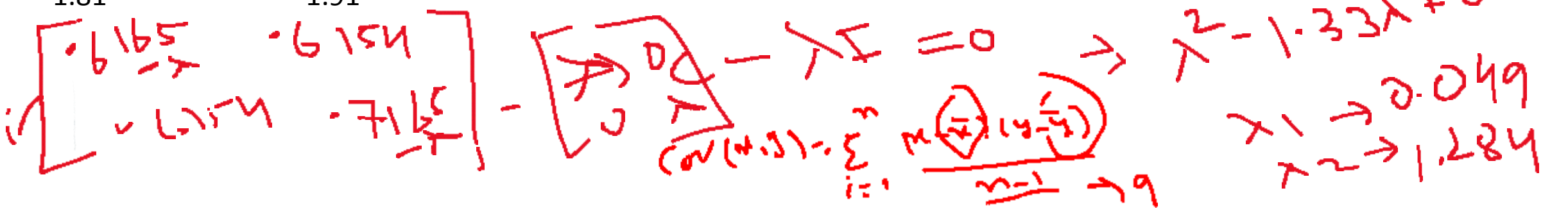
Principal Component : $\begin{bmatrix} X1 \\ X2 \end{bmatrix} = \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}$



X	Y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

x-x'	(x-x')*(x-x')	y-y'	(y-y')*(y-y'		(x-x')*(y-y')
0.69	0.4761	0.49	0.2401		0.3381
-1.31	1.7161	-1.21	1.4641		1.5851
0.39	0.1521	0.99	0.9801		0.3861
0.09	0.0081	0.29	0.0841		0.0261
1.29	1.6641	1.09	1.1881		1.4061
0.49	0.2401	0.79	0.6241		0.3871
0.19	0.0361	-0.31	0.0961		-0.0589
-0.81	0.6561	-0.81	0.6561		0.6561
-0.31	0.0961	-0.31	0.0961		0.0961
-0.71	0.5041	-1.01	1.0201		0.7171
0	0.61655556	0	0.7165556		0.615444

1.81 1.91



WITH THE HELPS PCA , REDUCE THE DIM. FROM MENTIONED TABLE

X	Y	Z	P
1	1	1	2
1	2	1	2
1	3	2	2
1	4	3	2

Linear Discriminant Analysis (LDA)

The main goal of dimensionality reduction techniques is to reduce the dimensions by removing the redundant and dependent features by transforming the features from higher dimensional space to a space with lower dimensions.

LDA uses the information from multiple features to create a new axis and projects the data on to the new axis in such a way as to minimizes the variance and maximizes the distance between the means of the classes.

LDA is a supervised method that can only be used with labeled data. It consists of statistical properties of your data, calculated for each class.

Linear Discriminant Analysis is a dimensionality reduction technique used as a preprocessing step in Machine Learning and pattern classification applications.

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis(LDA) is a very common technique used for supervised classification problems.

This category of dimensionality reduction is used in biometrics, bioinformatics and chemistry.

Linear Discriminant Analysis (LDA)

$$S_w = \sum_{i=1}^{g} (N_i - 1) S_i = \sum_{i=1}^{g} \sum_{j=1}^{N_i} (x_{i,j} - \bar{x}_i)(x_{i,j} - \bar{x}_i)^T$$

This can be achieved in three steps：

The first step is to calculate the separability between different classes(i.e the distance between the mean of different classes) also called as between-class variance

Second Step is to calculate the distance between the mean and sample of each class, which is called the within class variance

$$S_b = \sum_{i=1}^{g} N_i (\bar{x}_i - \bar{x})(\bar{x}_i - \bar{x})^T$$

Linear Discriminant Analysis (LDA)

This can be achieved in three steps：

The third step is to construct the lower dimensional space which maximizes the between class variance and minimizes the within class variance

$$P_{lda} = \arg\max_{P} \frac{\left| P^T S_b P \right|}{\left| P^T S_w P \right|}$$

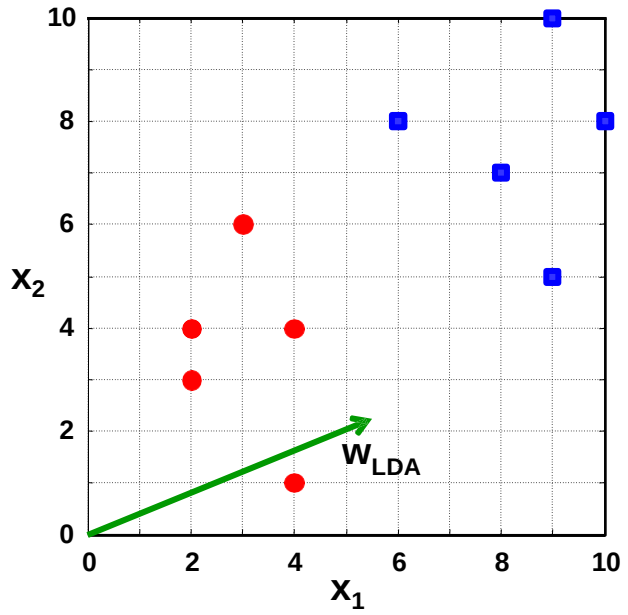
https://people.revoledu.com/kardi/tutorial/LDA/Numerical%20Example.html

LDA example

- **Compute the Linear Discriminant projection for the following two-dimensional dataset**
  - $X1=(x_1,x_2)=\{(4,1),(2,4),(2,3),(3,6),(4,4)\}$
  - $X2=(x_1,x_2)=\{(9,10),(6,8),(9,5),(8,7),(10,8)\}$
- **SOLUTION (by hand)**
  - The class statistics are:

$$S_1=\begin{bmatrix}0.80 & -0.40\\ -0.40 & 2.60\end{bmatrix};\ S_2=\begin{bmatrix}1.84 & -0.04\\ -0.04 & 2.64\end{bmatrix}$$

$$\mu_1=\begin{bmatrix}3.00 & 3.60\end{bmatrix}\qquad \mu_2=\begin{bmatrix}8.40 & 7.60\end{bmatrix}$$



  - The within- and between-class scatter are

$$S_B=\begin{bmatrix}29.16 & 21.60\\ 21.60 & 16.00\end{bmatrix};\ S_W=\begin{bmatrix}2.64 & -0.44\\ -0.44 & 5.28\end{bmatrix}$$

  - The LDA projection is then obtained as the solution of the generalized eigenvalue problem

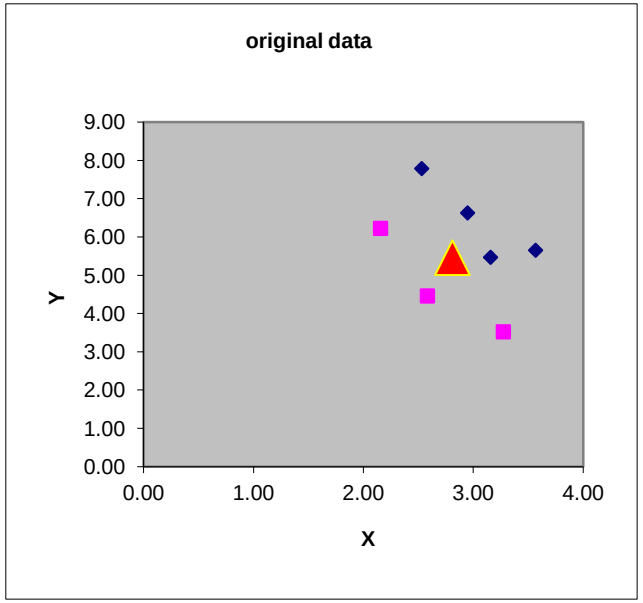
$$S_W^{-1}S_B\,v=\lambda v \Rightarrow \left|S_W^{-1}S_B-\lambda I\right|=0 \Rightarrow \begin{vmatrix}11.89-\lambda & 8.81\\ 5.08 & 3.76-\lambda\end{vmatrix}=0 \Rightarrow \lambda=15.65$$

$$\begin{bmatrix}11.89 & 8.81\\ 5.08 & 3.76\end{bmatrix}\begin{bmatrix}v_1\\ v_2\end{bmatrix}=15.65\begin{bmatrix}v_1\\ v_2\end{bmatrix}\Rightarrow\begin{bmatrix}v_1\\ v_2\end{bmatrix}=\begin{bmatrix}0.91\\ 0.39\end{bmatrix}$$

  - Or directly by

$$w^*=S_W^{-1}(\mu_1-\mu_2)=\begin{bmatrix}-0.91 & -0.39\end{bmatrix}^T$$

Training Data, D		
class	X1	X2
1	2.95	6.63
1	2.53	7.79
1	3.57	5.65
1	3.16	5.47
2	2.58	4.46
2	2.16	6.22
2	3.27	3.52



$$\mathbf{x} = \begin{bmatrix} 2.95 & 6.63 \\ 2.53 & 7.79 \\ 3.57 & 5.65 \\ 3.16 & 5.47 \\ 2.58 & 4.46 \\ 2.16 & 6.22 \\ 3.27 & 3.52 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 2 \end{bmatrix}$$