Machine Learning

Chapter 3 – Hands on Data Analytics for Everyone

November 10, 2022

北京师范大学-香港浸会大学联合国际学院
United International College

Contents

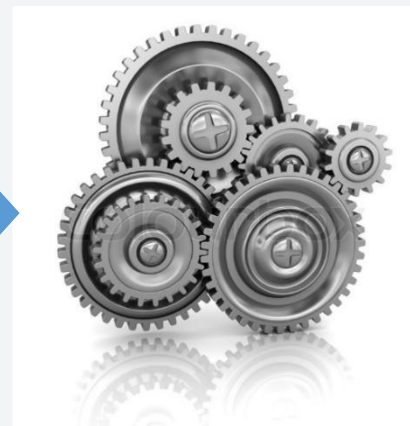
What is a Model (Learning Algorithm)?

A model or learning algorithm is simply a specification of a mathematical (or probabilistic) **relationship** that exists between different variables.

- Input features
- Input attributes
- Independent variables

$\boldsymbol{X} = (x_1, x_2, \dots, xn)$

Model



y

- Class
- Label
- Target
- Output feature/attribute
- Dependent variable

$$y = f(\boldsymbol{\beta}, \boldsymbol{X}) \quad \text{with } \boldsymbol{\beta} = [\beta_1, \beta_2, \dots, \beta_m]$$

A learning algorithm adjusts (learns) the model **parameters $\boldsymbol{\beta}$** throughout a number of iterations to maximize/minimize a likelihood/error function on y.

What Is Machine Learning?

Learning = Improving with **experience** at some **task**

Arthur Samuel (1959)

- Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.

Tom Mitchell (1998)

- Well-posed Learning Problem: A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

What Is Machine Learning?

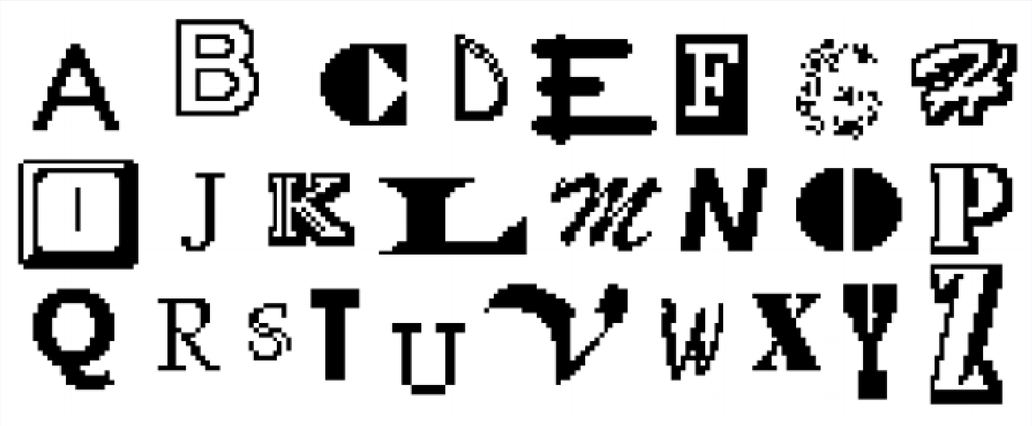
Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam.

What is the task T, the experience E and the measure P in this setting?

1. Classifying emails as spam or not spam.
2. Watching you label emails as spam or not spam.
3. The number (or fraction) of emails correctly classified as spam/not spam.

Character recognition

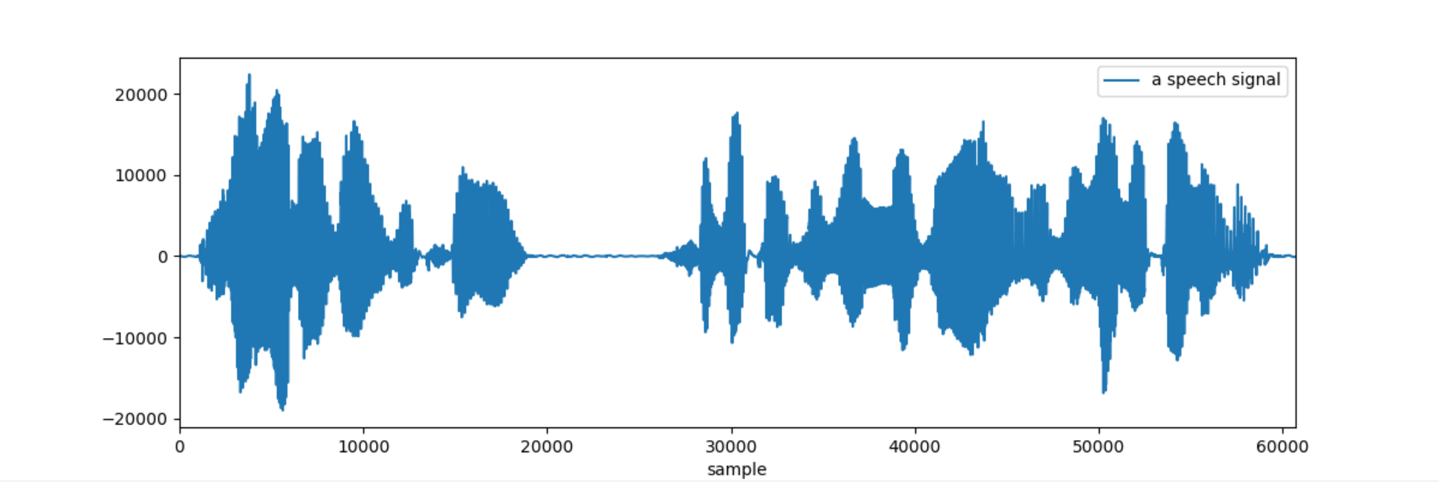
- Raw data: image



➢ Classification: numerals, English (Chinese, etc.) characters

Speech recognition

- Raw data: speech signal



➢ Classification: spoken words

Document Classification

- Raw data: (web) document

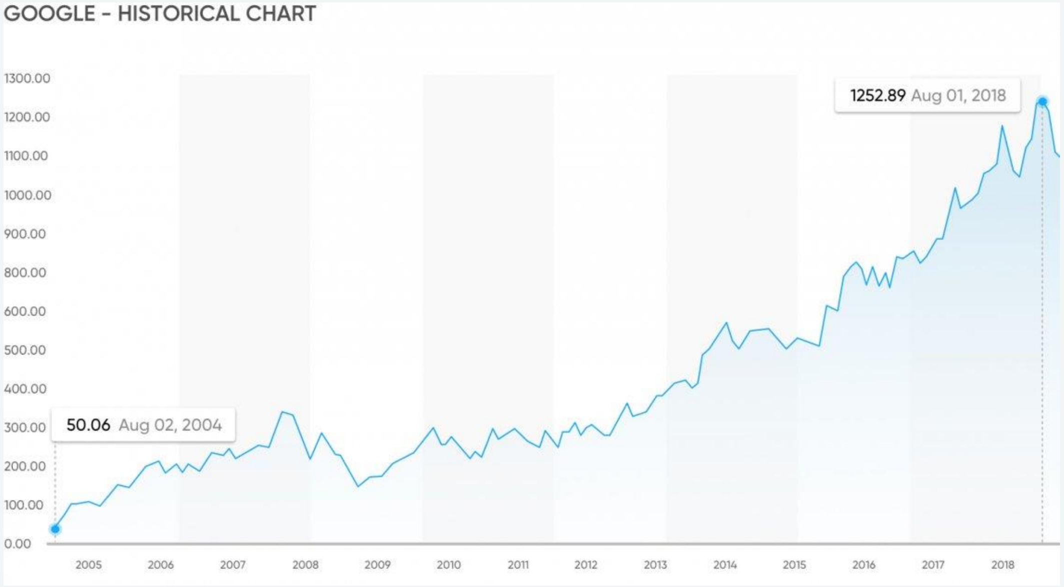
As the movie year winds down, I would like to express my gratitude to Martin Scorsese. Not only for making "The Irishman," his best movie in a long time and one of the best of 2019 (see below), but also for reminding the world of the value of cinema.
The art form is in one of its periodic identity crises. A big chunk of our collective attention — we don't yet know how big, or with what consequences — is migrating to streaming platforms whose offerings include a lot of the stand-alone single-episode narratives that we used to see mainly in theaters.

➢ Classification: semantic categories (movie, art, money,...)

Financial Engineering

- Raw data: financial time series (e.g., stock prices)

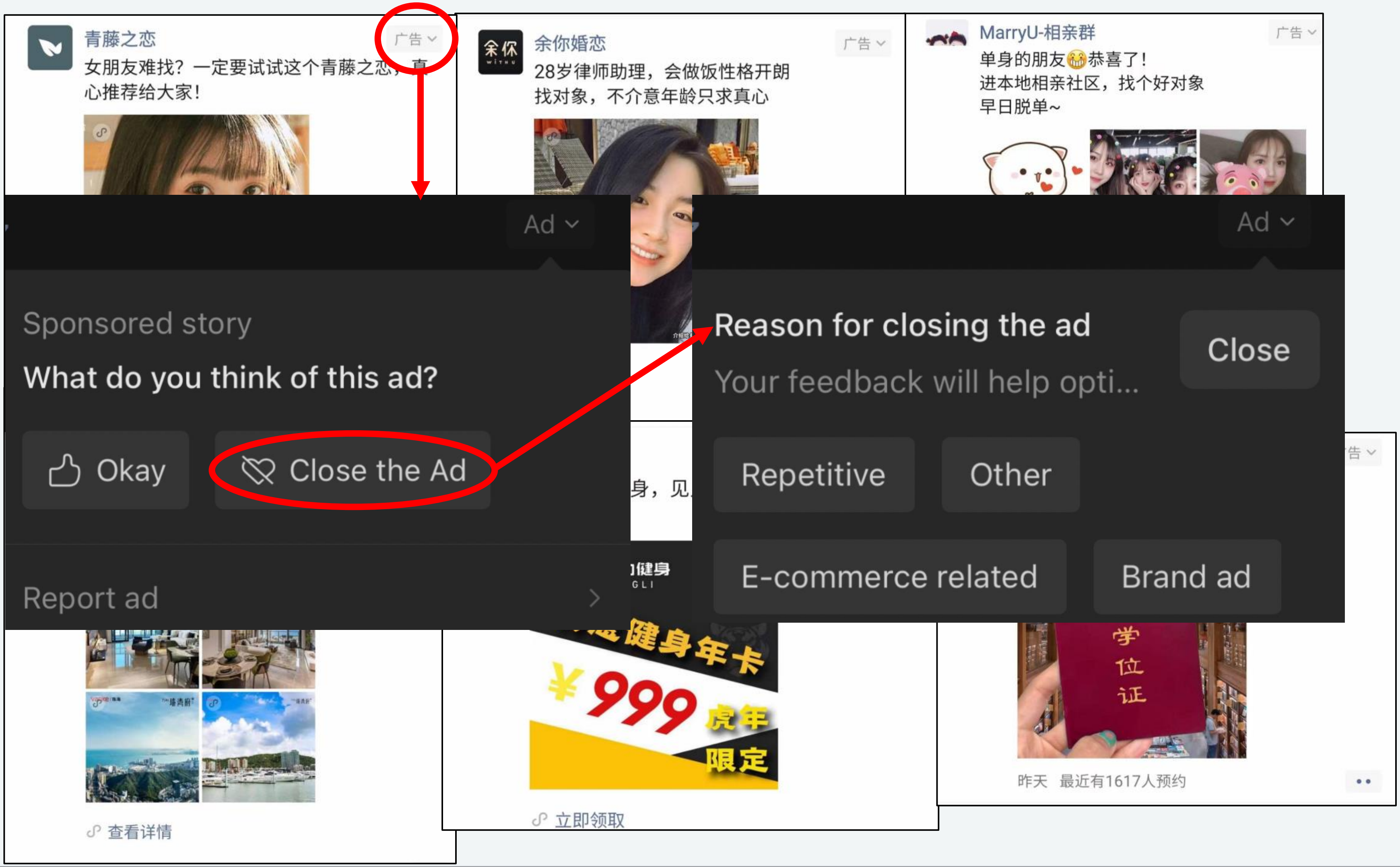
➢Classification: financially healthy / unhealthy company, stock prediction, etc.



Many other examples

- Facebook: photo tagging, ranking articles to your news feed
- Amazon: e-commerce fraud detection, forecasting demand, pricing
- NASA: identifying stars, supernovae, clusters, galaxies, quasars, exoplanets, etc.
- Google Spreadsheets: uses machine learning to fill in missing values
- E-commerce: predict whether an ad will be clicked by users
- …

Machine Learning Is Everywhere



Contents

Learning Paradigm

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Reinforcement Learning
- Transfer Learning
- Federated Learning
- ……

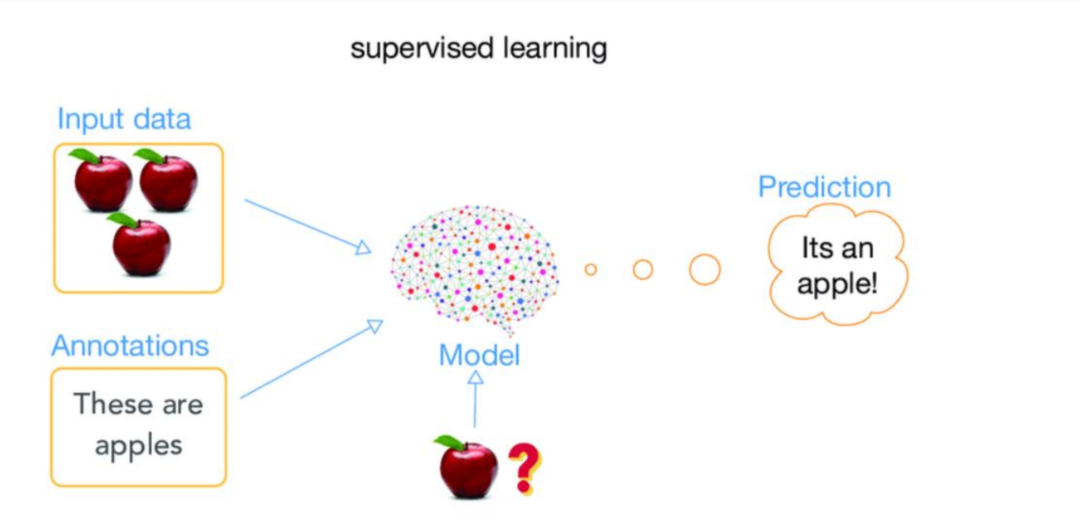
Supervised Learning

The learner is provided with a set of data inputs together with the corresponding desired outputs

- ➢ Data act as a "teacher"

Example:

- teach kids to recognize different animals
- grade examinations with correct answer provided



Supervised Learning Example

Example: Breast cancer (malignant/benign classification)

- Input: Tumor size samples
- Output: Whether the tumor is malignant or benign
- Task: Learn a classifier from the provided input and output, that can predict the label (category)for new tumor size inputs.

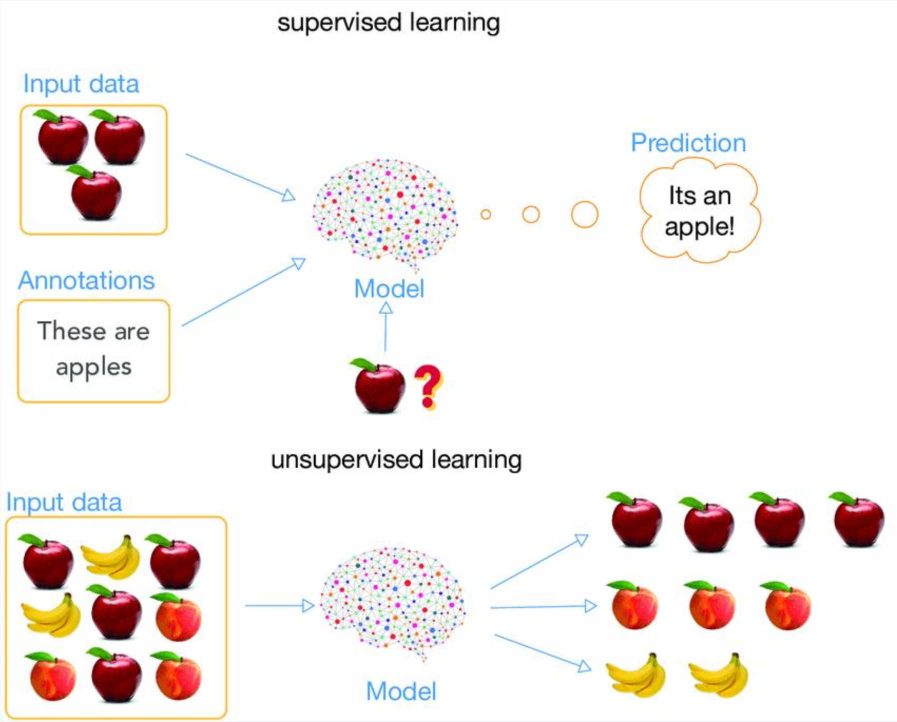
Example: House price prediction (Regression)

- Input: House size samples
- Output: House prices
- Task: Learn a model from the provided input and output, that can predict the house prices (quantity) for new house size inputs.

Unsupervised Learning

Training examples as input patterns, with no associated output

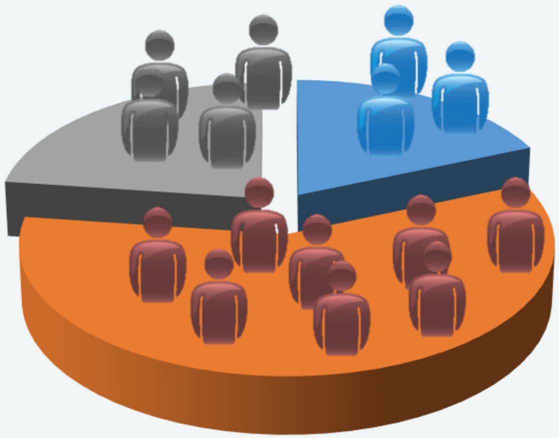
- no “teacher”
- similarity measure exists to detect groupings/ clusterings



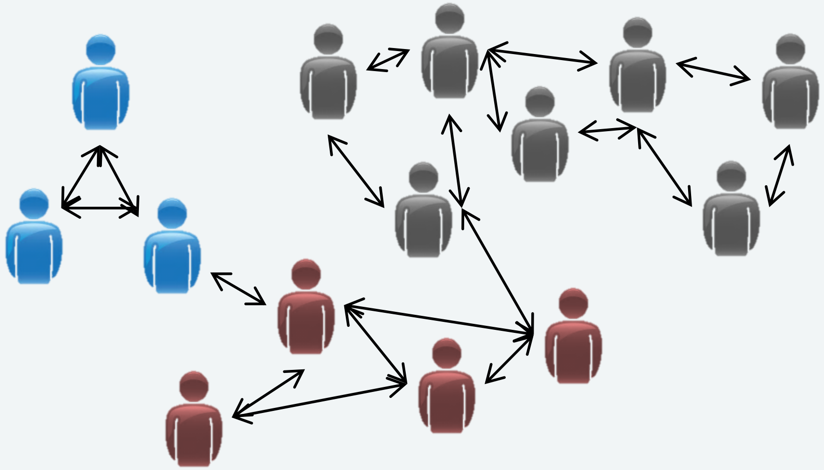
大学道
The Great Learning Way

Clustering

In the early stages of an investigation, it may be helpful to perform exploratory data analysis to gain some insight into the nature or structure of the data



Market segmentation



Social network analysis

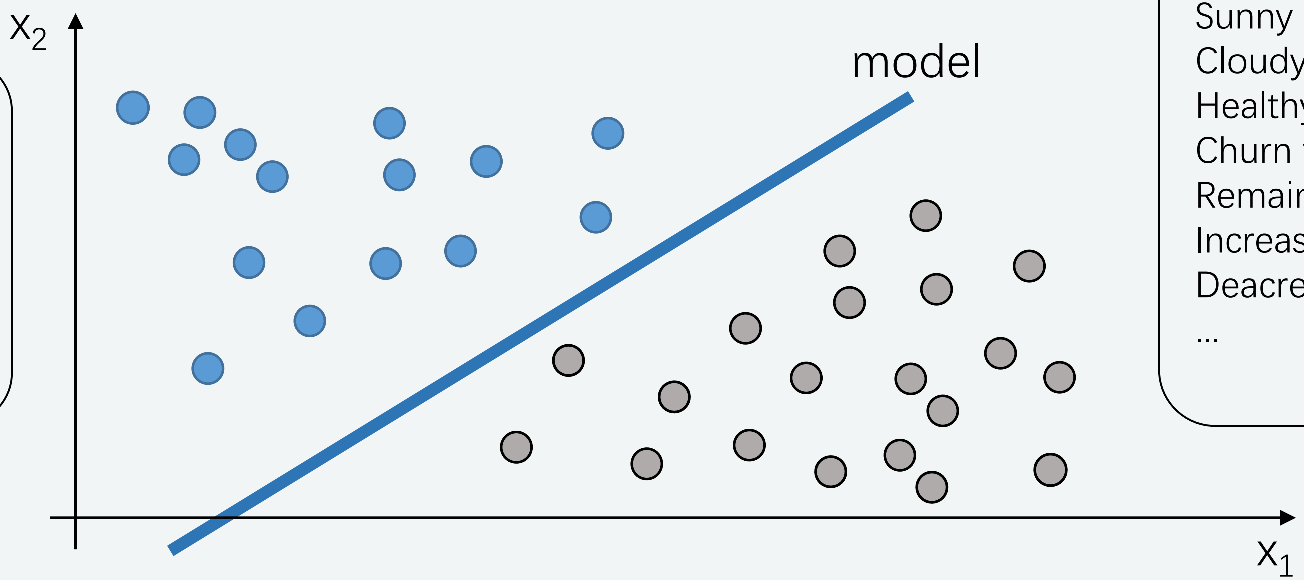
Supervised Learning Data Representation

$\boldsymbol{X} = (x_1, x_2)$ and $y = \{yellow, gray\}$

- A training set with many examples of $(\boldsymbol{X}, y)$
- The model learns on the examples of the training set to produce the right value of y for an input vector $\boldsymbol{X}$

$\boldsymbol{X}$

Age
Money
Temperature
Speed
Number of taxi
…



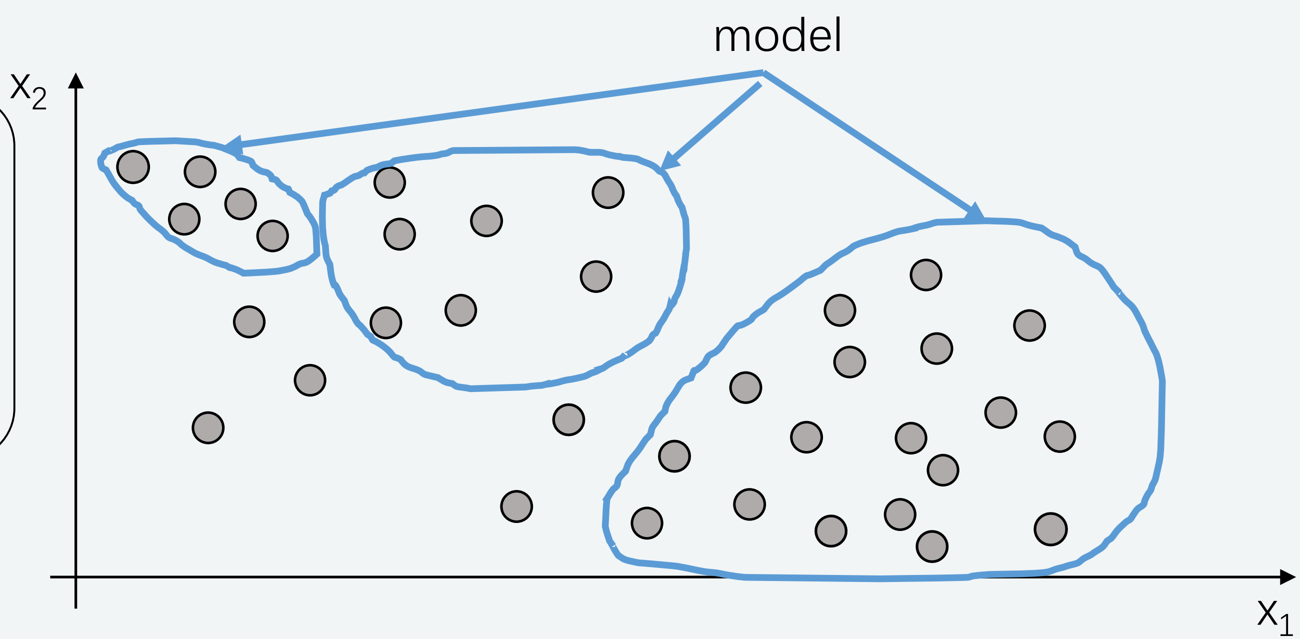
y

Sunny vs. Cloudy
Healthy vs. Sick
Churn vs. Remain
Increase vs. Deacrease
…

Unsupervised Learning Data Representation

- $\boldsymbol{X} = (x_1, x_2)$ and $\cancel{y = \{yellow, gray\}}$
- A training set with many examples of $(\boldsymbol{X}, \cancel{y})$
- The model learns to group the examples $\boldsymbol{X}$ of the training set based on similarity (closeness) or probability

X
Age
Money
Temperature
Speed
Number of taxi
…

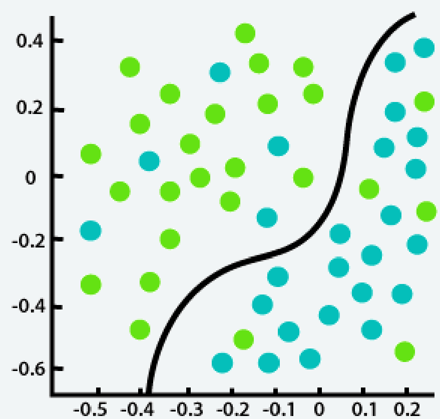


Contents

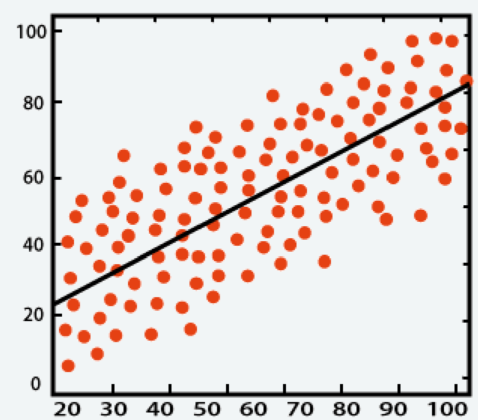
Regression vs Classification (Supervised Learning)

Supervised learning: Given the “right answer” for each example in the data.

- When the target variable that we’re trying to predict is continuous, we call the learning problem a regression problem.
- When the target variable can take on only a small number of discrete values, we call it a classification problem.



Classification



Regression

Classification (Supervised Learning)

Given a collection of records: each record contains a set of *attributes*, one of the attributes is the *class*.

- Find a *model* (train a classifier) for class attribute as a function of the values of other attributes.

Previously unseen records should be assigned a class as accurately as possible.

- A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

Classification Example

You are a bank loan officer and need to know which loan applicants are "safe" and which are "risky" for the bank

- Assign labels "safe" or "risky" to a loan applicant

You are a marketing manager at an electronics consumer shop and want to guess whether a customer will buy a new computer

- Predict the category of the customer, "buys" or "doesn't buy"

Classification is the task of assigning objects to several predefined categories/labels

Classification Example

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Tid	Refund	Marital Status	Taxable Income	Cheat
11	No	Married	80K	?
12	Yes	Single	100K	?

Regression Example

Suppose we have a dataset giving the **living areas** and **prices** of 47 houses from Portland, Oregon:

Size in feet^2 (x)	**Price ($) in 1000's (y)**
2104	460
1416	232
1534	315
852	178
…	…

Supervised Learning Example

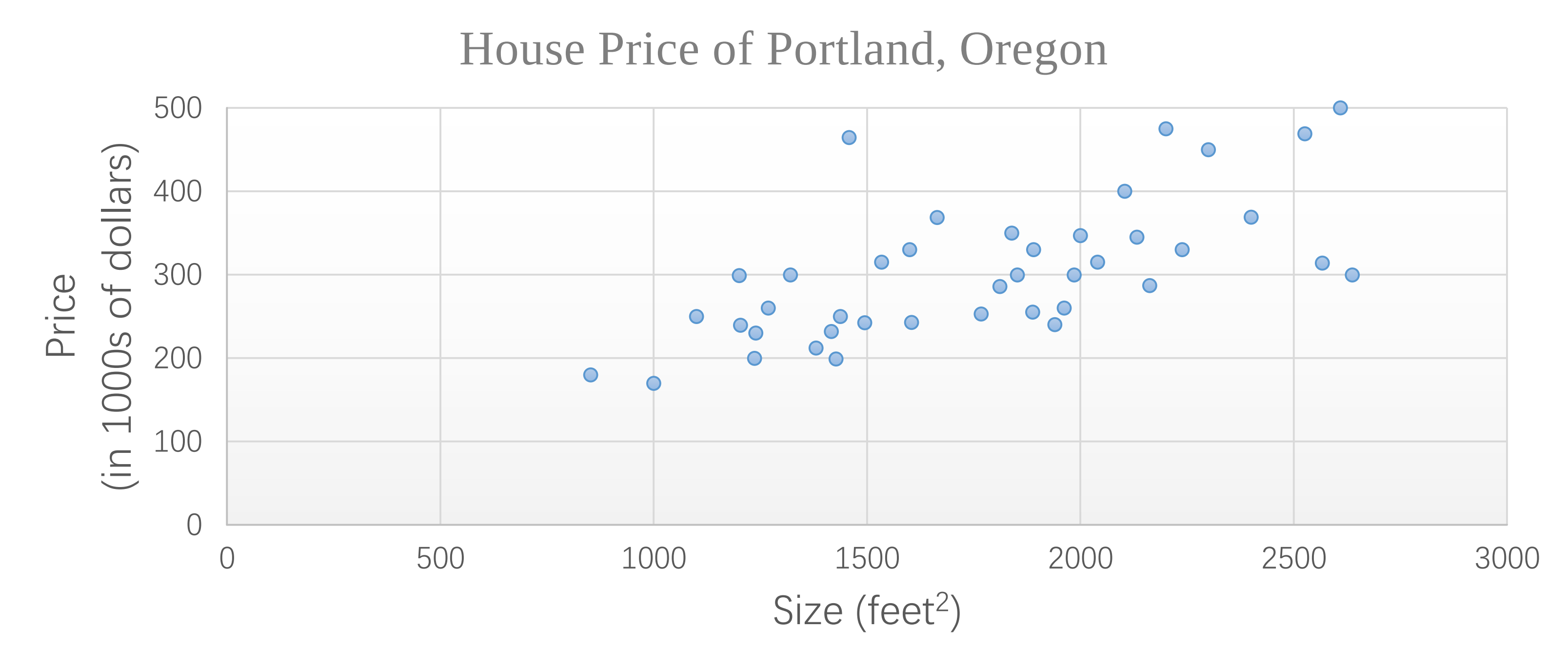
You're running a company, and you want to develop learning algorithms to address each of two problems.

- Problem 1: You have a large inventory of identical items. You want to predict how many of these **items will sell** over the next 3 months.
- Problem 2: You'd like software to examine individual customer accounts, and for each account decide if it has been **hacked/compromised**.

Should you treat these as classification or as regression problems?

Contents

Regression Example



Given data like this, how can we learn to predict the prices of other houses in Portland, as a function of the size of their living areas?

Linear Regression – Problem Description

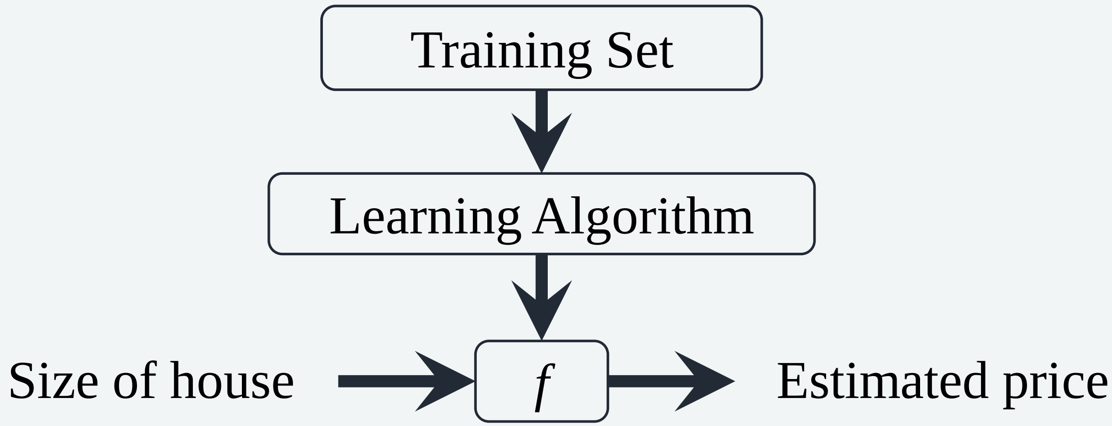
Notation:

- Input variable/feature: x
- Output/target variable: y
- Training example: $(x^{(i)}, y^{(i)})$
- Training set: $\{(x^{(i)}, y^{(i)}); i = 1,2, \dots, m\}$ (a list of m training examples)
- Space of input values: X ; space of output values: Y

To describe the problem slightly more formally, our goal is:

Given a training set, to learn a function (hypothesis/model) $f: X \longmapsto Y$, so that $f(x)$ is a "good" predictor for the corresponding value of y.

Linear Regression – Model Representation



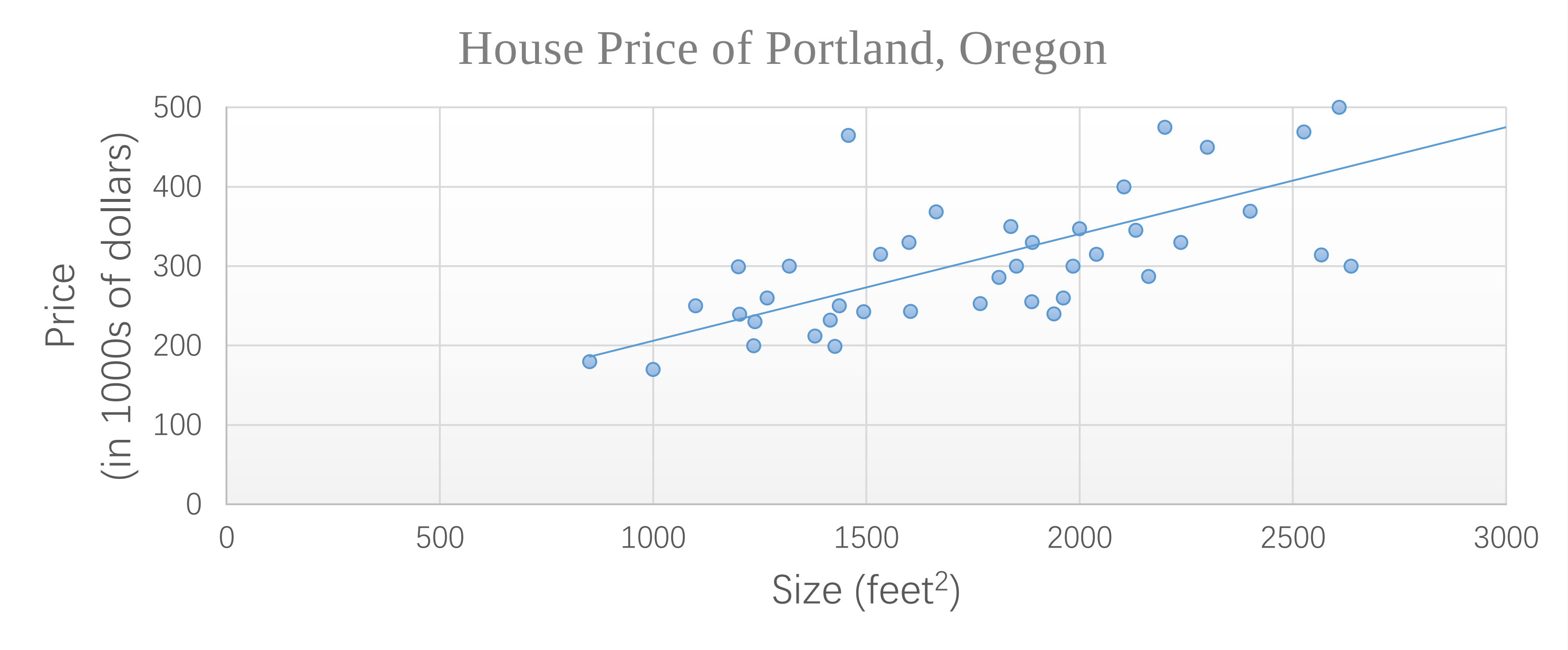
How do we represent *f*?

Linear Regression: $f(x) = \theta_0 + \theta_1 x$

- The model is in linear in terms of parameters θ_0 and θ_1
- Linear regression with one variable (univariate linear regression).

We will predict that y is a linear function of x (straight line)

Regression Example



We can predict that the house price is a linear function of house size

Regression Model Evaluation

Given dataset $D = \{(x^{(i)}, y^{(i)}); i = 1,2, \ldots, m\}$ and a regression model f, evaluate the performance of the model using following metrics.

Error Metric	Formula	Notes
Mean absolute error (MAE)	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i - f(x_i)\rvert$	Average of the absolute difference between the actual and predicted values.
Mean squared error (MSE)	$\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$	Average of the squared difference between the actual and predicted values.
Root mean squared error (RMSE)	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2}$	Square root of Mean Squared error.
R-squared	$1 - \frac{\sum_{i=1}^{n}(y_i - f(x_i))^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$	Proportion of the variance for a dependent variable that's explained by the regression model. Normally ranges from 0 to 1, the closer to 1 the better performance

Linear Regression Model Training

Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

Model: $f_\theta(x) = \theta_0 + \theta_1 x$

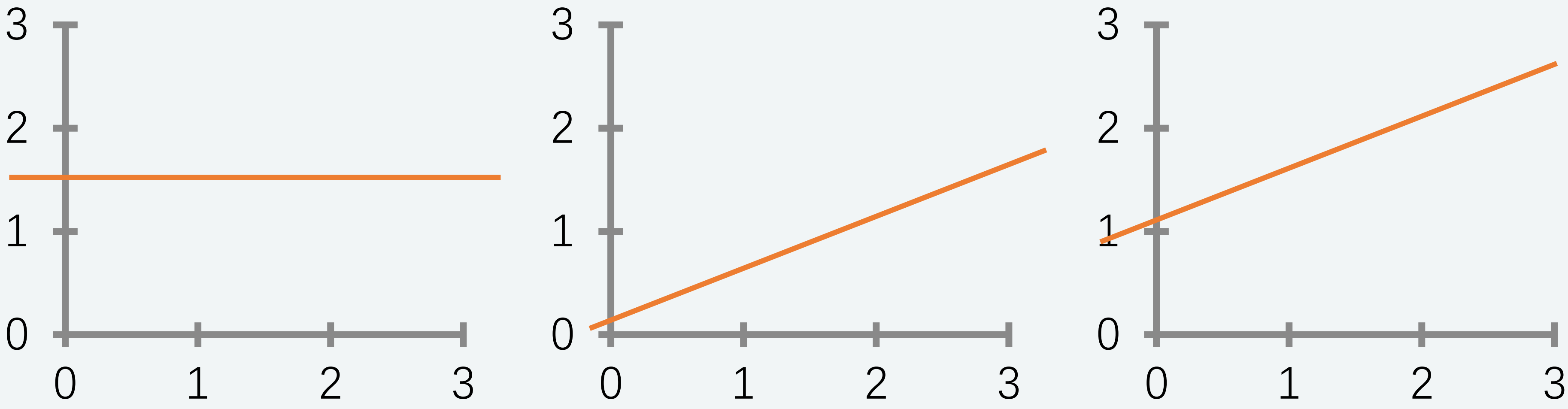
- $\theta_i{}'s$ are the parameters (weights)

Different setting of parameters result in different models, how to choose $\theta_i{}'s$?

Linear Regression Model Training

Different settings of θ_0 and θ_1 and the corresponding models for

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



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

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

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

House Price example with multiple features (variables)

Size in feet^2	Number of bedrooms	Number of floors	Age of home (years)	Price ($) in 1000's
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

Notation

- n: number of features; m: number of training examples
- $x^{(i)}$: input of i^{th} training example
- $x_j^{(i)}$: value of feature j in i^{th} training example
- $y^{(i)}$: output/target of i^{th} training example

Multivariate Linear Regression

House Price example with multiple features (variables)

Size in feet^2	Number of bedrooms	Number of floors	Age of home (years)	Price ($) in 1000's
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

- Univariate Linear Regression (previous): $f_\theta(x) = \theta_0 + \theta_1 x_1$
- Multivariate Linear Regression: $f_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$

To simplify our notation, we also introduce the convention of letting $x_0 = 1$ (this is the intercept term)

- $f_\theta(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$
- on the right-hand, we are viewing θ and x both as vectors

Contents

- Introduction to Machine Learning
  - Supervised and Unsupervised Learning
  - Classification and Regression
- Linear Regression (Supervised Learning)
  - Model
  - Performance Evaluation
- Classification (Supervised Learning)
  - How to Perform a Classification
  - Classification Tree Model
- Clustering Method (Unsupervised Learning)
  - Objective
  - Similarity Measures
  - (Optional) Method 1: Hierarchical Clustering
  - (Optional) Method 2: K-Means Method (Clustering by Partitioning)
- Lab (Demo): Unsupervised Learning
- Assignment 5: Supervised Learning
- Assignment 6: In-Class Quiz

Generally, the difference between the actual predicted output of the learner and the true output of the sample is called "error"

- Training error/ empirical error: the error of the learner/model on the training data
- Generalization error: the error on the new data

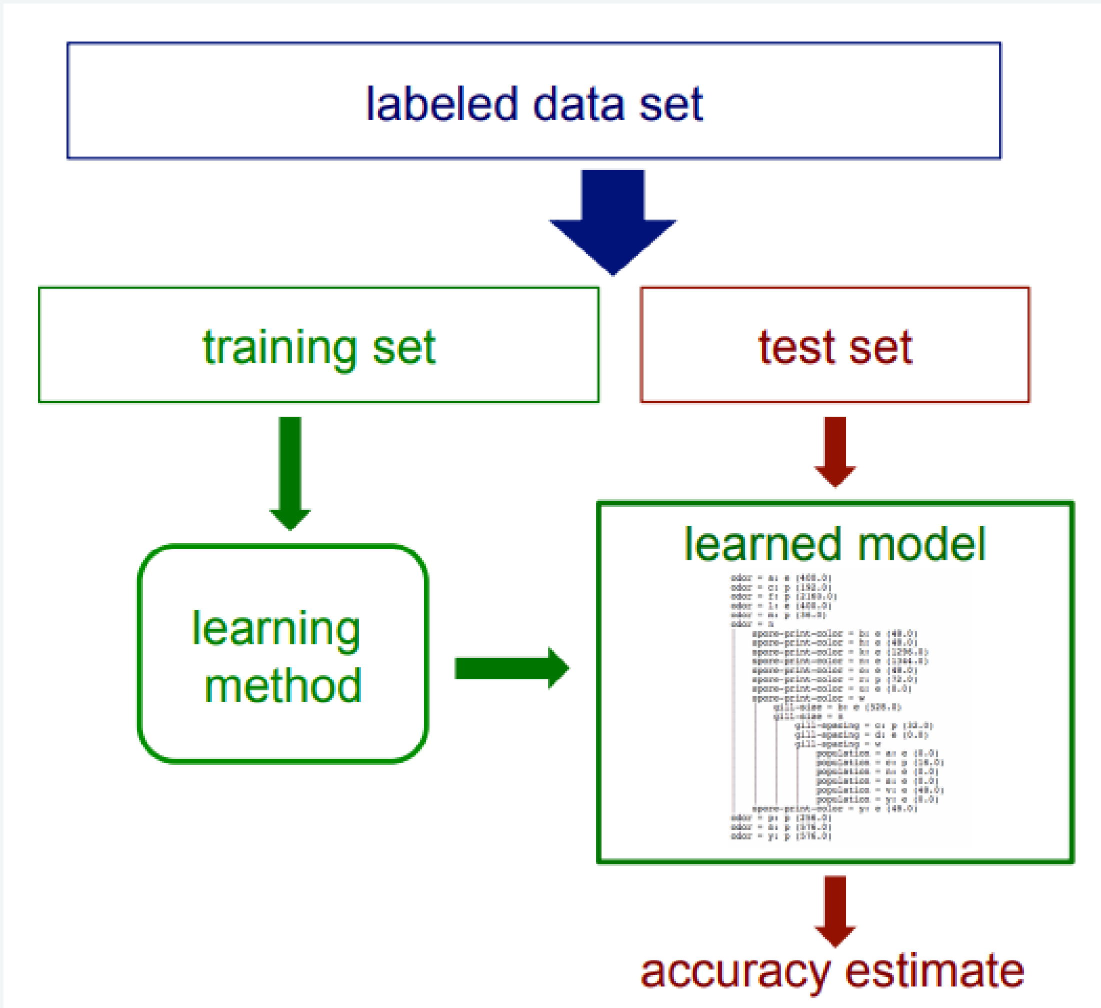
Model Evaluation and Selection

We want to get a learner with a small generalization error

However, we do not have the information for new data, instead we try to minimize the empirical error on the training data

- split data randomly into a training set and a test set (e.g., a 70%/30% split).
- train your model on the training set and see how it performs on the test set.
- use the "testing error" on the test set as an approximation of the generalization error

Model Evaluation and Selection



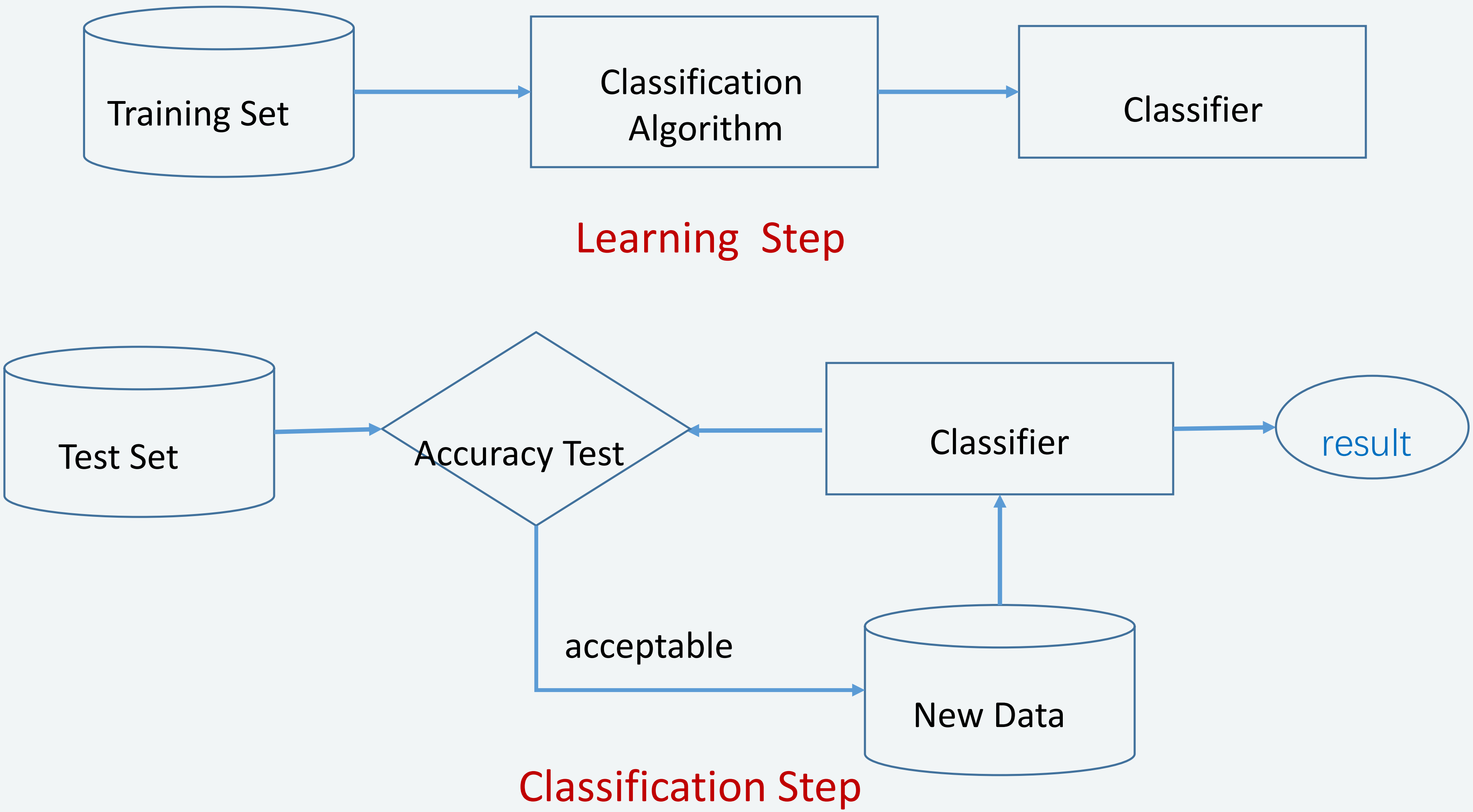
Strategies for generating training and testing/validation datasets

- **Hold-out**: just set aside some portion of the data for testing
- **Cross validation**: partition data into *k* disjoint datasets (called folds) of approximately equal size; iteratively take *k*-1 folds for training and validate on the remaining fold ; average the results
- **Boostrapping**: new datasets are generated by sampling with replacement (uniformly at random) from the original dataset; then train on the bootstrapped dataset and validate on the unselected data

No Need to Remember by Hard

Contents

How to Perform Classification?



Classification Accuracy

How to measure the accuracy of a classifier?

- Suppose we have already selected the training and test sets, and we have created a classifier based on the training set

We will measure the accuracy based on the test set

- Try to predict the class value of every tuple in the test set, and compare it against their actual ones (which are already stored in the test set)

Accuracy Measures

Classification accuracy

- The percentage of test set tuples that are correctly classified by the classifier
- A useful tool for analyzing how well the classifier can recognize tuples of different classes is the confusion matrix

c_{ij}: number of tuples from class i that are classied as class j by the classifier

Predicted class

accuracy of classifying "yes" tuples

Actual class

Classes	Buy="yes"	Buy="no"	Total	Accuracy
Buy="yes"	6,954	46	7000	99.34
Buy="no"	412	2,588	3000	86.27
Total	7,366	2,634	10,000	95.42

total accuracy

Accuracy Measures

Classification accuracy sometimes can be misleading

Let us focus on a two-class problem (e.g., “non cancer”/ “cancer" patients) where

- number of class C_1 tuples: 9,990
- number of class C_2 tuples: 10

- If the classifier predicts everything to be class C_1, then accuracy is 99.9%
- However, this is misleading because the classifier does not correctly predict any tuple from C_2

Accuracy Measures

Consider a two-class problem and the confusion matrix below

- The positives refers to the tuples of the main class of interest (C_1)

Predicted class

Actual class

Classes	C_1	C_2	Total
C_1	true positives (TP)	false negatives (FN)	positives
C_2	false positives (FP)	true negatives (TN)	negatives

Contents

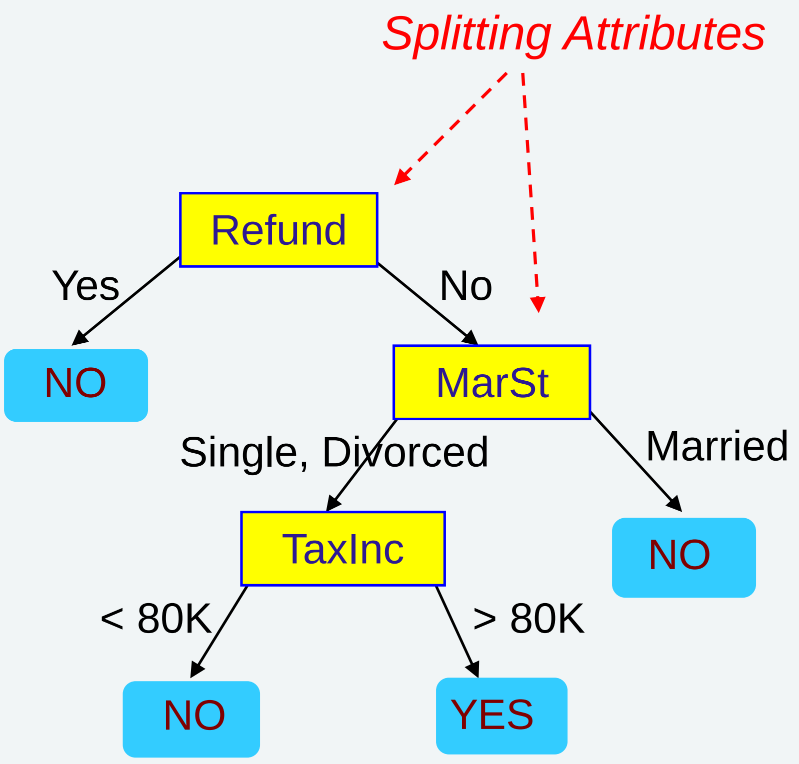
Decision Tree

- One way to solve the classification problem is by asking a series of questions about the attributes of the test record
- Each time we receive an answer, a follow-up question is asked until we reach a conclusion about the class label of the record
- The series of questions and their possible answers can be organized in the form of a decision tree, which is a hierarchical structure consisting of nodes and directed edges

Decision Tree

- Each internal node denotes a test on an attribute
- Each branch represents an outcome of the test
- Each leaf node holds a class label

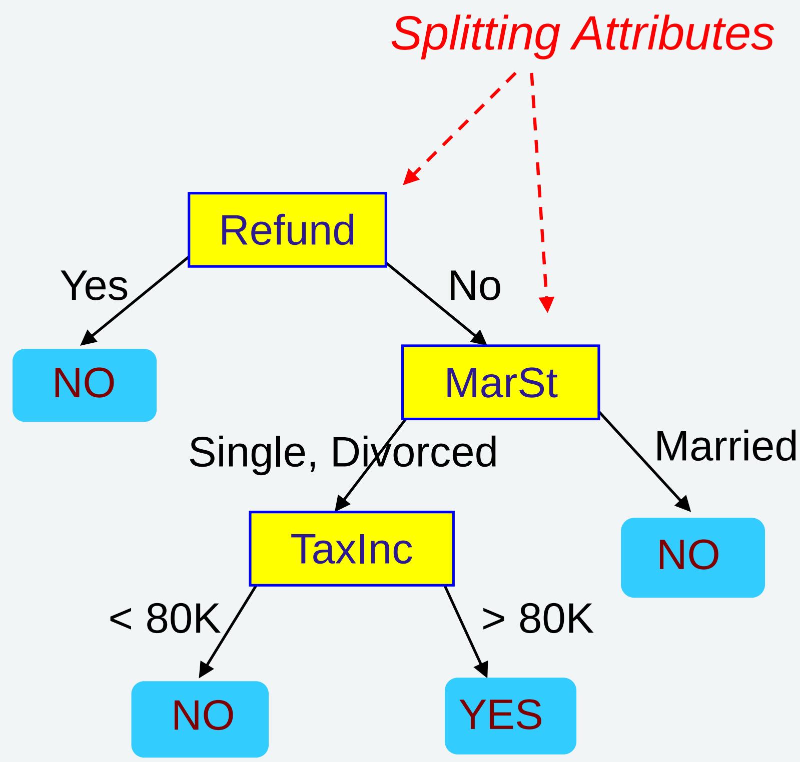
Model: Decision Tree



There could be more than one tree that fits the same data!

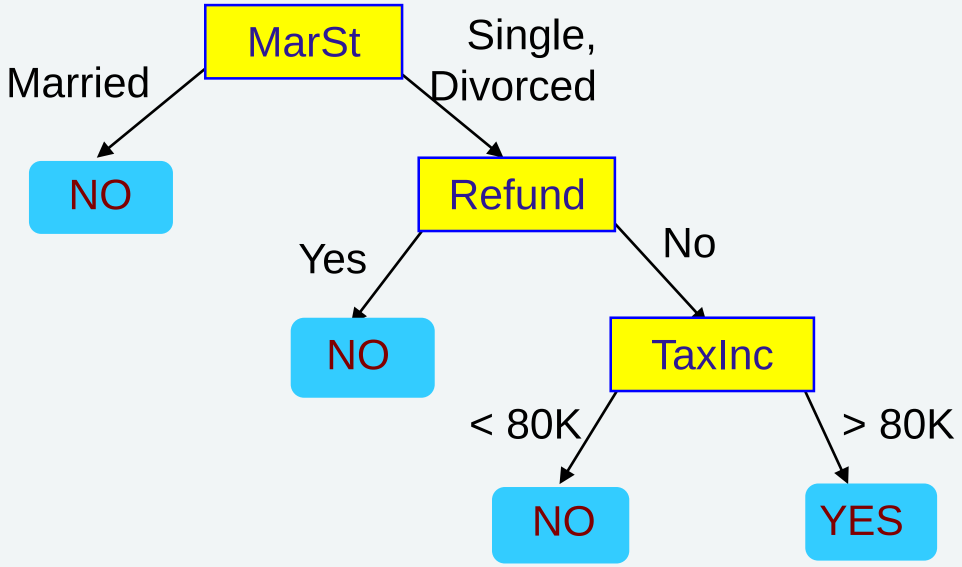
categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



There could be more than one tree that fits the same data!

Tid categorical	Refund categorical	Marital Status continuous	Taxable Income class	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Decision Tree: Prediction

Given a new (unseen) tuple: the associated class is unknown

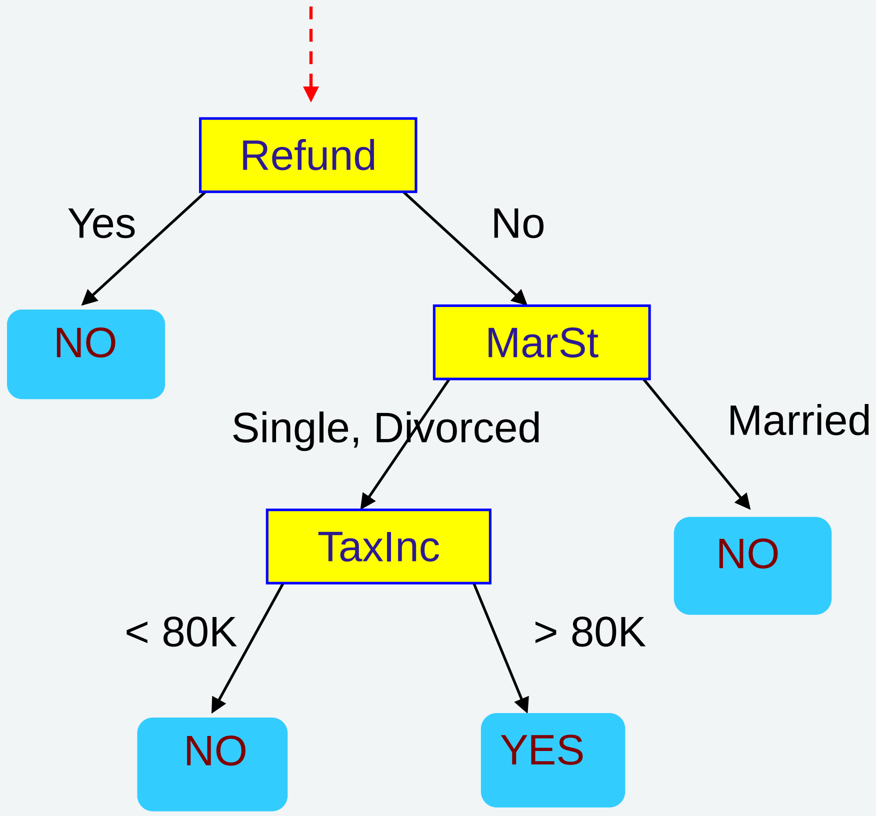
- Test the attribute values of the tuple against the decision tree

  Start from the root and trace a path to a leaf node (top-down), based on the attribute values of the tuple

- The class value included in this leaf is assigned to the tuple

Decision Tree: Prediction

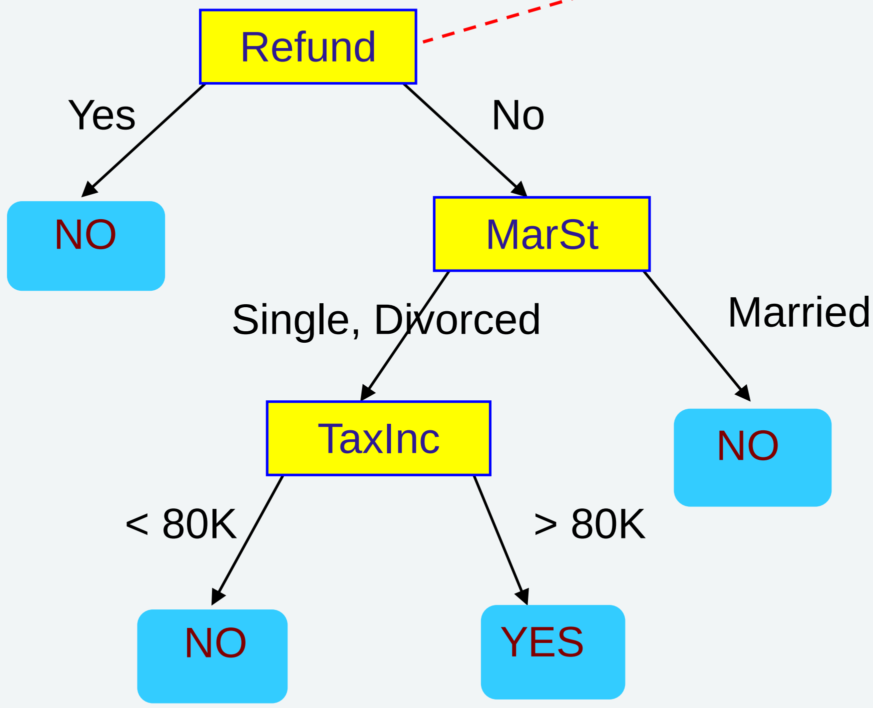
Start from the root of tree



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

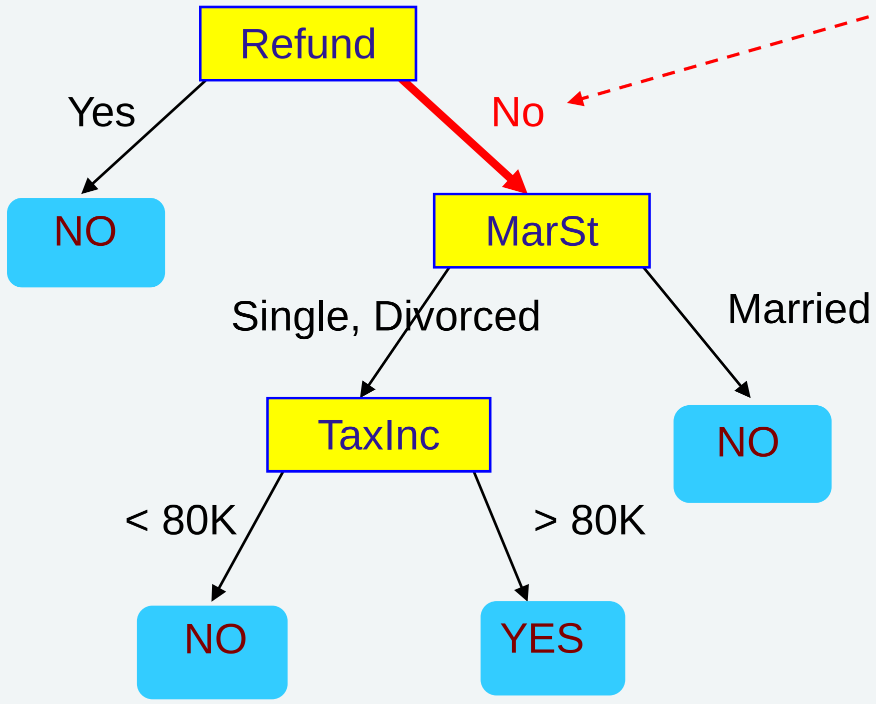
Decision Tree: Prediction



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Decision Tree: Prediction



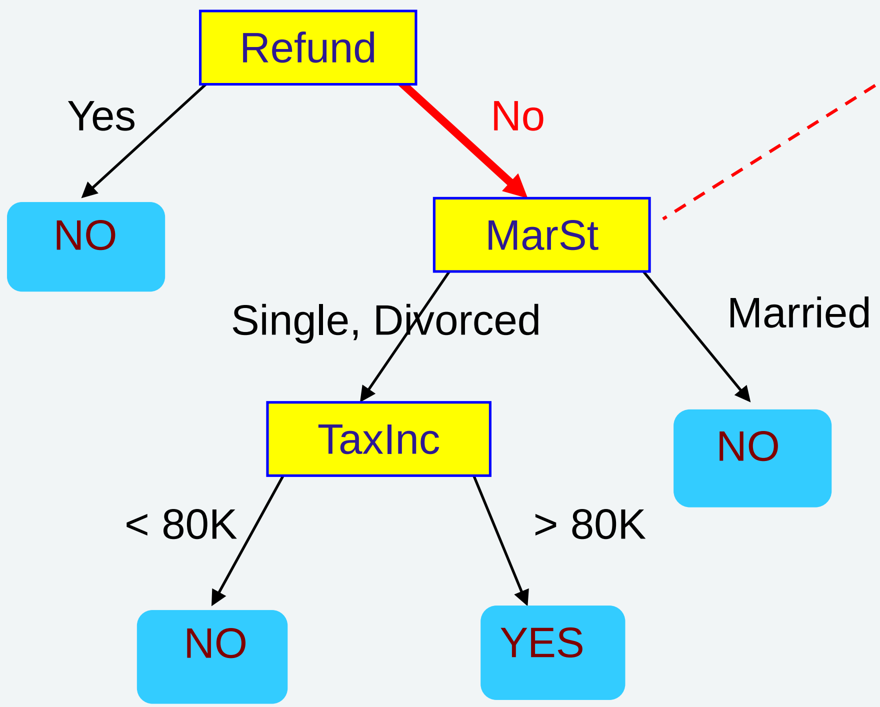
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Decision Tree: Prediction

Test Data

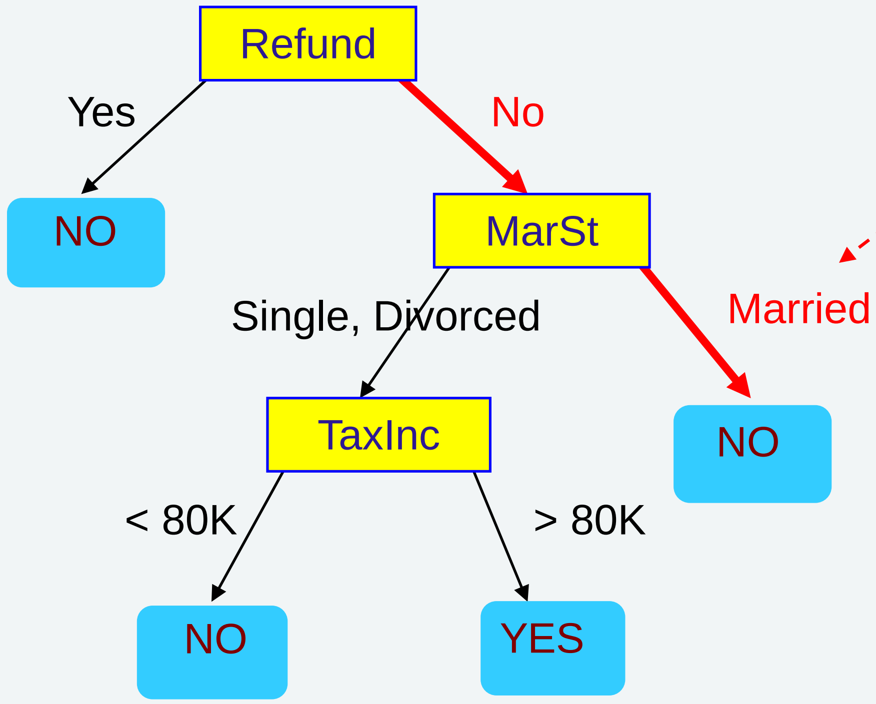
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



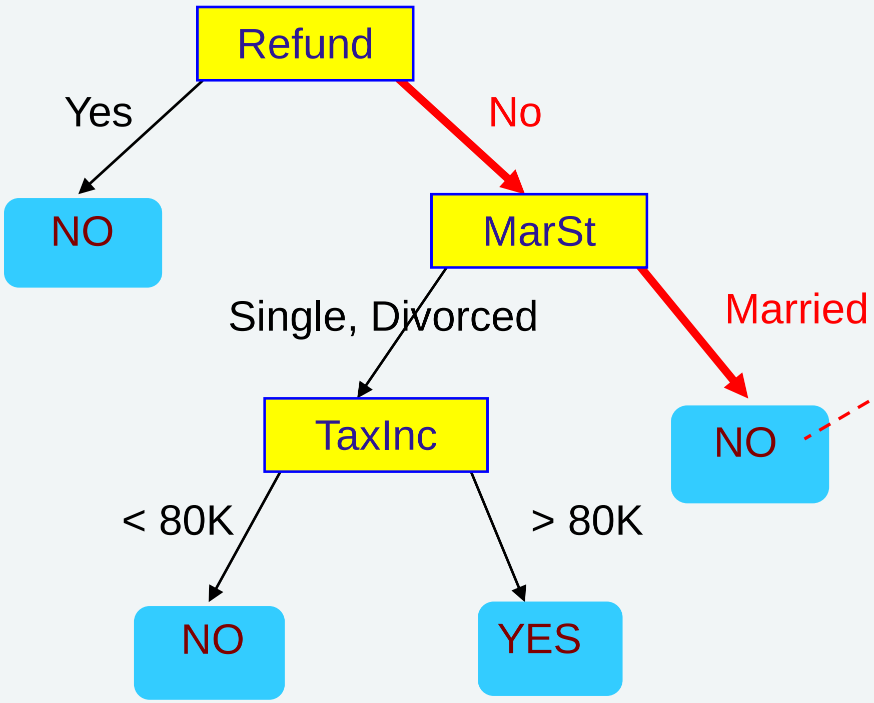
Decision Tree: Prediction

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Decision Tree: Prediction



Test Data

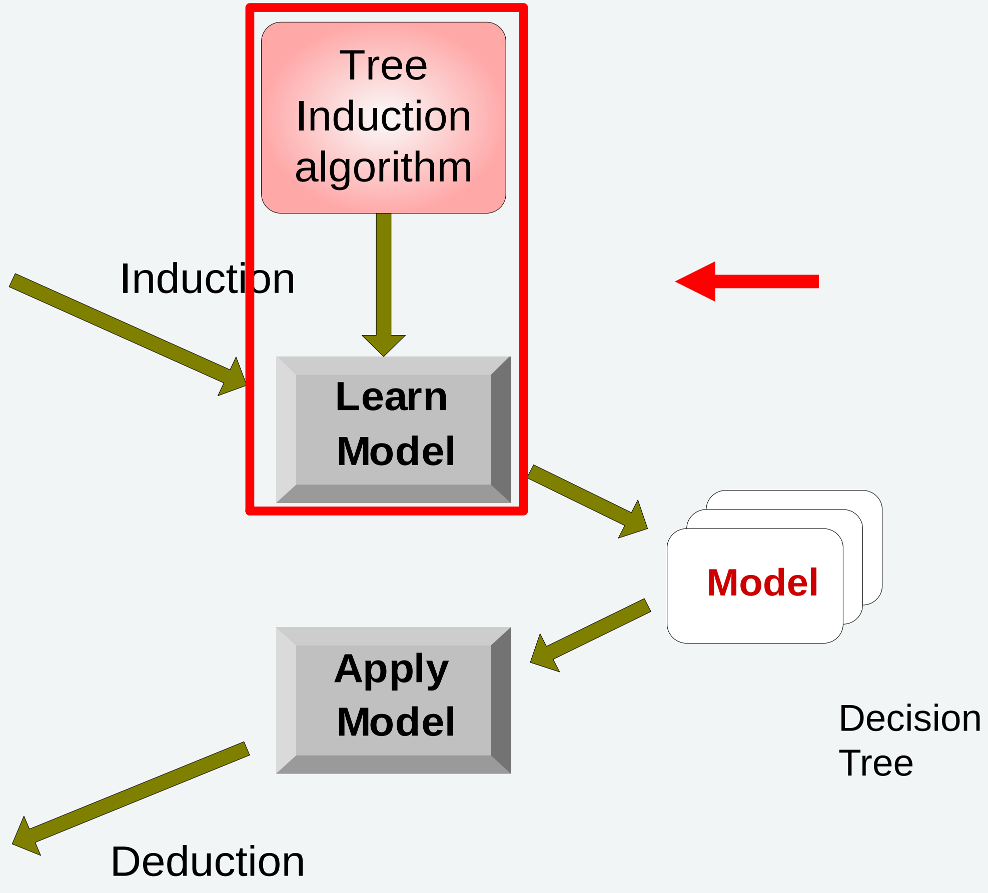
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Assign Cheat to "No"

Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

How to Build A Decision Tree

- There are exponentially many decision trees that can be constructed from a given set of attributes. While some of the tree are more accurate than others
- Efficient algorithms have been developed to induce a reasonably accurate decision tree in a reasonable amount of time
- These algorithms usually employ a greedy strategy that grow a decision tree by making a series of locally optimum decisions about which attribute to use for partitioning the data

Basic algorithm (a greedy algorithm)

- A top-down recursive divide-and-conquer manner

Input:

- *Node N*, the first time the algorithm is called, root of the tree
- *Dataset D* of training examples
  - ✓ Initially, entire training set
- *Attribute list*, holds a set of attributes
  - ✓ Initially the attributes that remained after data preprocessing
- Attribute selection method: a heuristic process for selecting the attribute that “best” discriminates the given tuples according to class

No Need to Remember by Hard

Decision Tree Induction

- Step1: Associate node N with dataset D

  Trivial; meant to emphasize that each decision tree node represents a subset of the original (entire) training set

- Step2: End this process if one of the terminating conditions is satisfied (will be discussed soon)

- Step3: Call attribute selection method
  - Use a splitting criterion to select an attribute to test at node N
    - try to "best" partition D into subsets, such that each subset is as "pure" as possible
    - a subset of D is pure , if it contains tuples belonging to the same class
  - This test will result in creating new nodes (as children of N), each of which representing a subset of D

No Need to Remember by Hard

Decision Tree Induction

- Step 4: Create a branch for each of the outcomes of the splitting criterion
  - A new node N_i is created for each branch i
  - If the number of new nodes is m, D is partitioned accordingly into m subsets $D_1, D_2, \dots, D_m$
  - Each D_i contains the tuples that satisfy the splitting criterion outcome of branch i

- Step 5: Remove the splitting attribute A (or the splitting attribute value) from attribute list
  - Call *Decision Tree Induction(N_i, D_i , attribute list, attribute selection method)* recursively for every newly created (N_i, D_i) pair

No Need to Remember by Hard

Decision Tree Induction

- Greedy strategy
  - ➢ Split the records based on an attribute test that optimizes a certain criterion

- Design Issues
  - ➢ Determine when to terminate splitting
  - ➢ Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?

No Need to Remember by Hard

Terminating Conditions

- All of the tuples in partition D belong to the same class
  - N becomes a leaf and is labeled with that class

- There are no remaining attributes in attribute list to help partitioning the tuples of D further
  - N becomes a leaf and is labeled with the majority class in D

- D is empty
  - N becomes a leaf and is labeled with the majority class of its parent's dataset

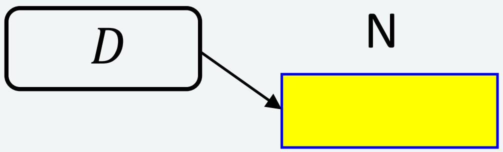
No Need to Remember by Hard

Example

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

Step1: The algorithm is called with the initial single root node N, the entire training set as D

Step2: No terminating condition is satisfied



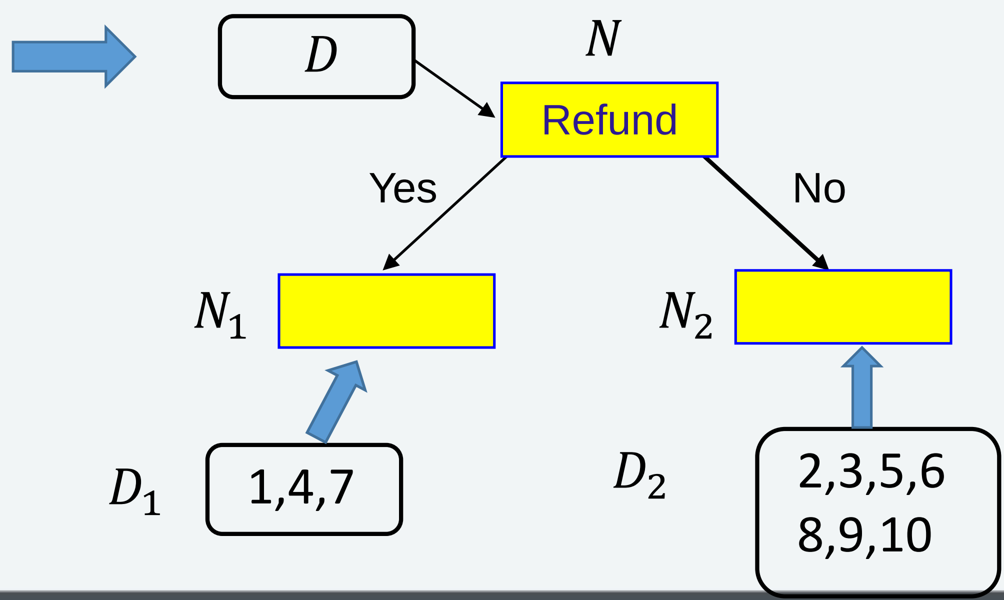
Example

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Step3: Suppose that the "Refund" is selected as the splitting attribute by the *attribute_selection_method*

Step4: Two branches and children are created for N, also D is partitioned into two datasets based on the "Refund" attribute value

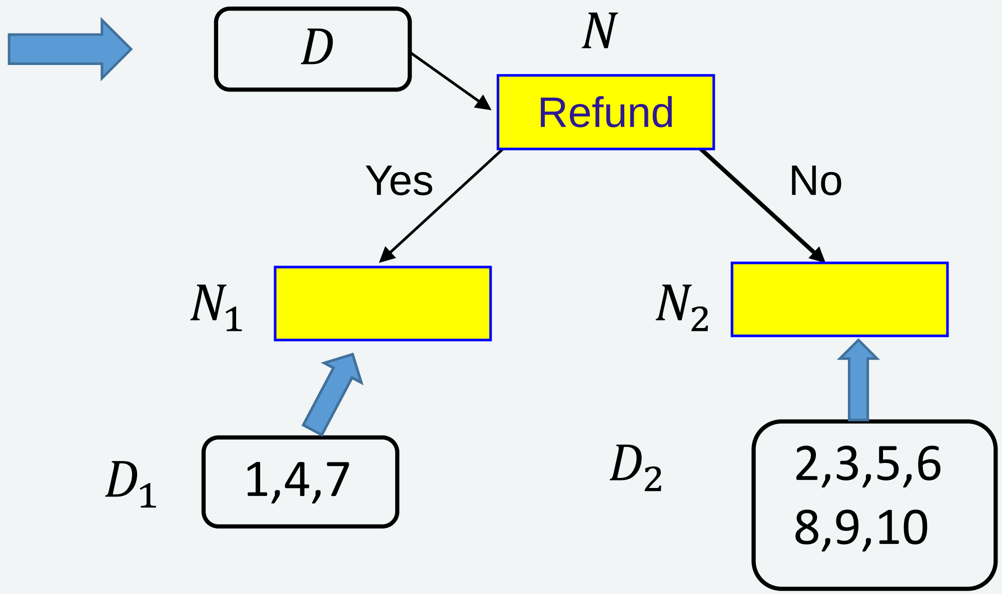


Example

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Step5: Remove the splitting attribute A from attribute list and call Decision Tree recursively for every newly created (Ni, Di) pair

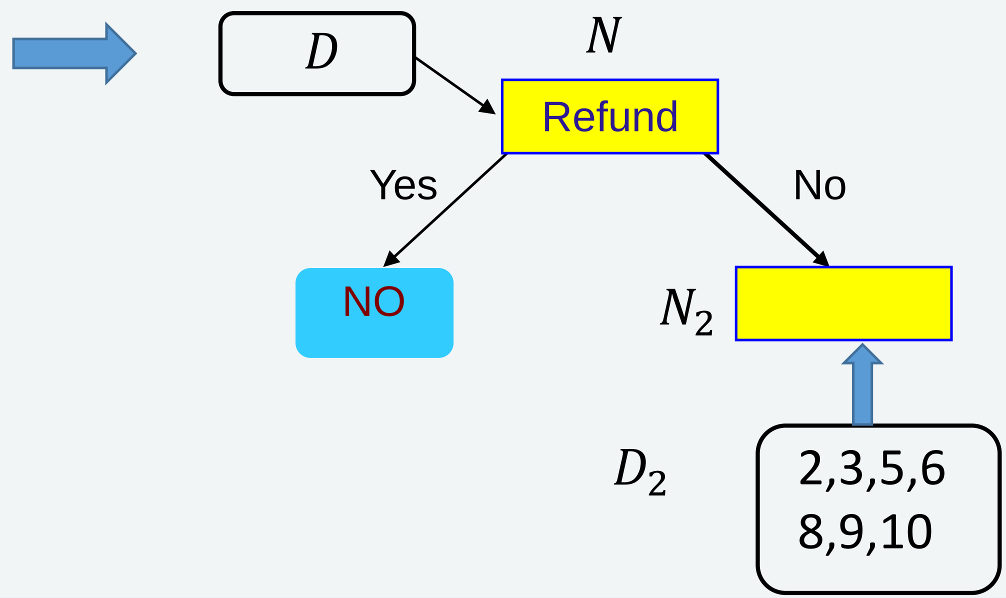


Example

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Step5: Remove the splitting attribute A from attribute list and call Decision Tree recursively for every newly created (Ni, Di) pair

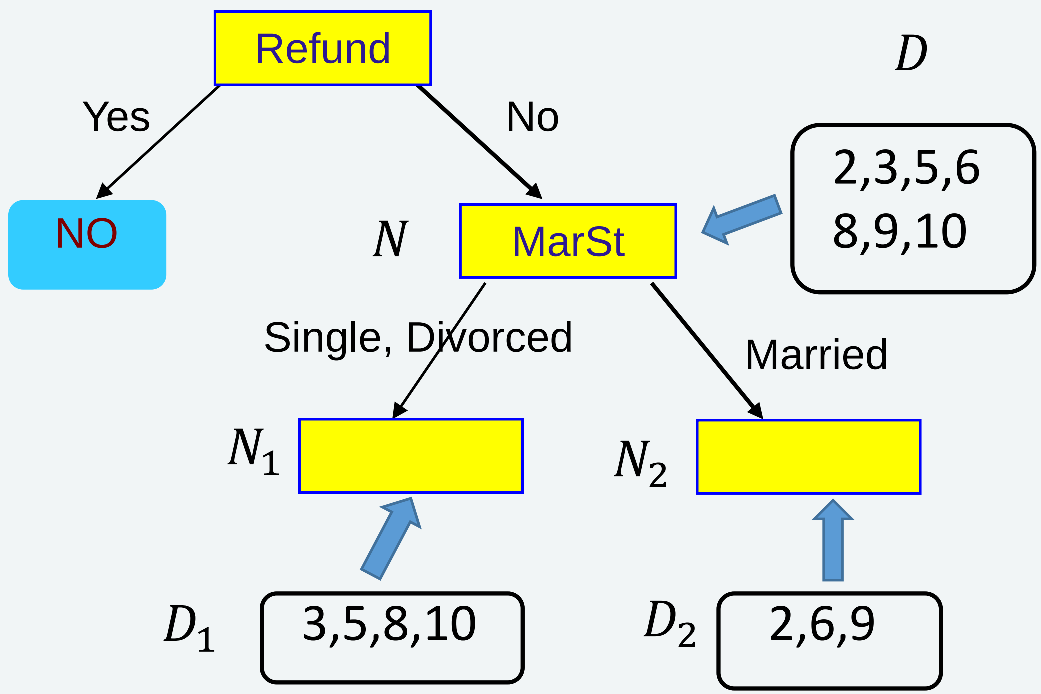


Example

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

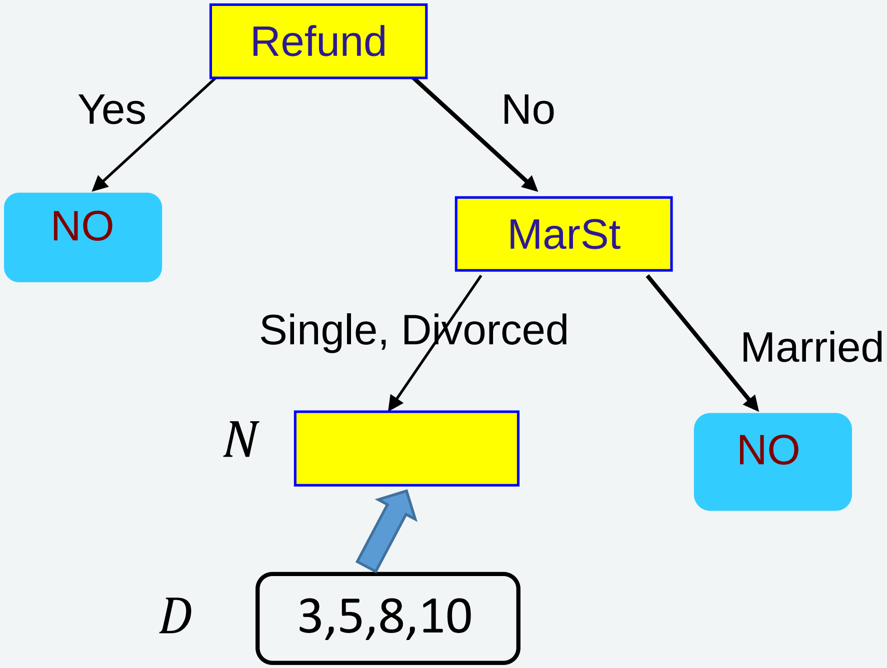
Step5: Remove the splitting attribute A from attribute list and call Decision Tree recursively for every newly created (Ni, Di) pair



Example

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Step5: Remove the splitting attribute A from attribute list and call Decision Tree recursively for every newly created (Ni, Di) pair

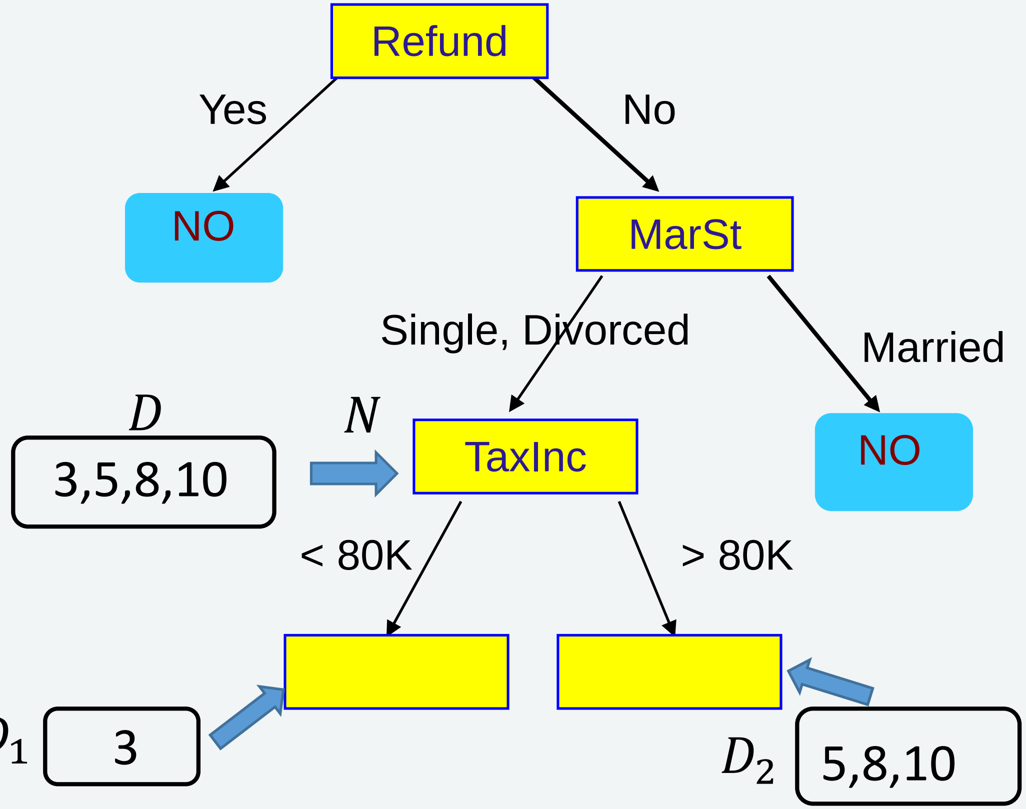


Example

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Step5: Remove the splitting attribute A from attribute list and call Decision Tree recursively for every newly created (Ni, Di) pair

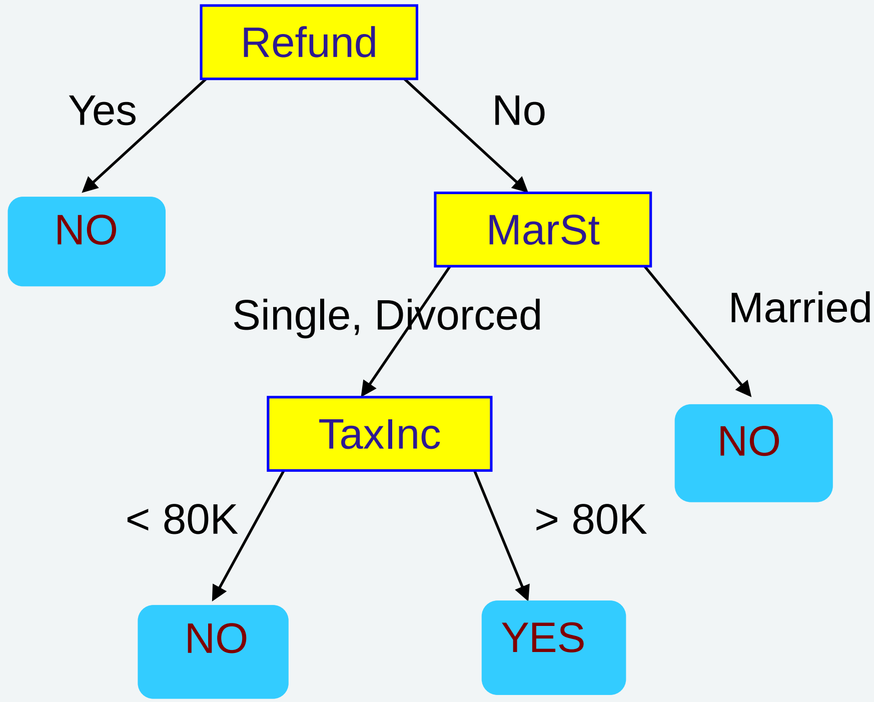


Example

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Step5: Remove the splitting attribute A from attribute list and call Decision Tree recursively for every newly created (Ni, Di) pair



Attribute Selection Measures

- Ideally, the best splitting criterion is the one that decomposes D into subsets having only tuples of a single class (these subsets are called pure)
- Since it may not be always possible to select a splitting criterion that derives only pure subsets, the attribute selection measure provides a ranking for each attribute
  - The attribute with the best score is selected as the splitting attribute

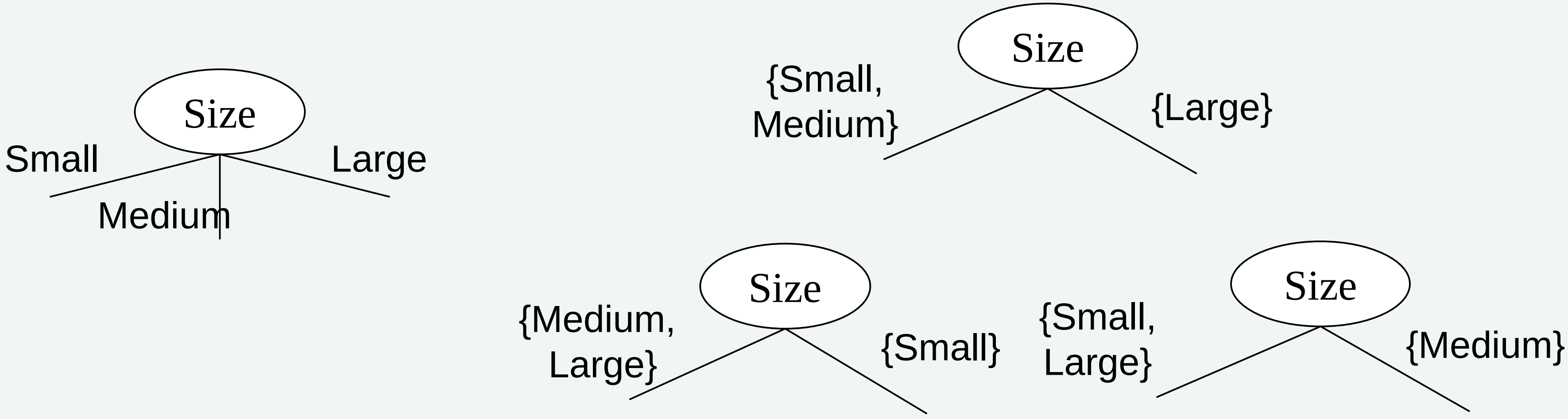
- Following are three common attribute selection measures for splitting
  1. Information Gain (used in ID3)
  2. Gain Ratio (used in C4.5)
  3. Gini Index (used in CART)

Not part of the course program

Partition Categorical Attributes

When a predictor is categorical we can decide to split it to create either

- only two child nodes (*binary* split) or
- one child node per class (*multiway* splits)



Contents

- Introduction to Machine Learning
  - Supervised and Unsupervised Learning
  - Classification and Regression
- Linear Regression (Supervised Learning)
  - Model
  - Performance Evaluation
- Classification (Supervised Learning)
  - How to Perform a Classification
  - Classification Tree Model
- **Clustering Method (Unsupervised Learning)**
  - **Objective**
  - Similarity Measures
  - (Optional) Method 1: Hierarchical Clustering
  - (Optional) Method 2: K-Means Method (Clustering by Partitioning)
- Lab (Demo): Unsupervised Learning
- Assignment 5: Supervised Learning
- Assignment 6: In-Class Quiz

Goal of Cluster Analysis

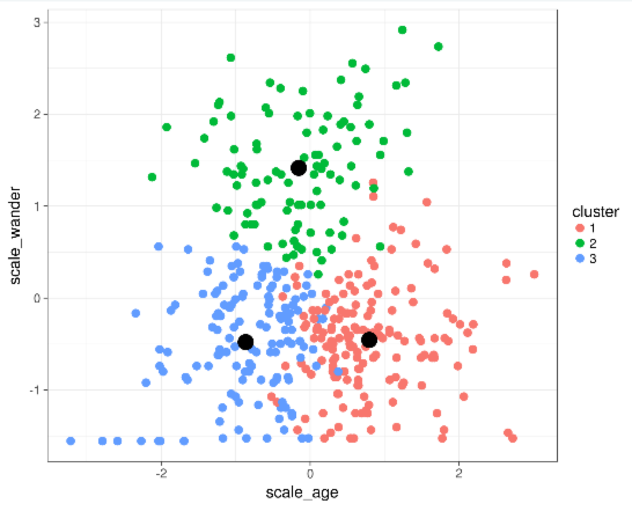
Discover hidden structures in unlabeled data (unsupervised)

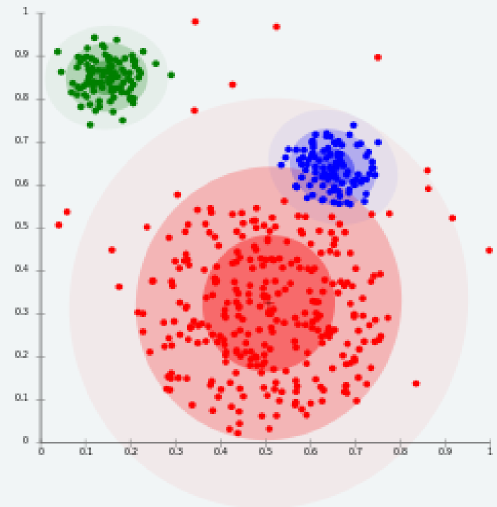
Clustering identifies a finite set of groups (*clusters*) $C_1, C_2 \cdots, C_k$ in the dataset such that:

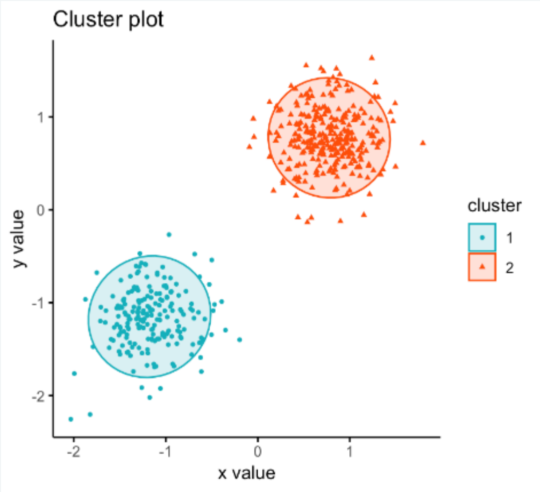
- Objects within the *same* cluster C_i shall be as similar as possible
- Objects of *different* clusters C_i, C_j $(i \neq j)$ shall be as dissimilar as possible

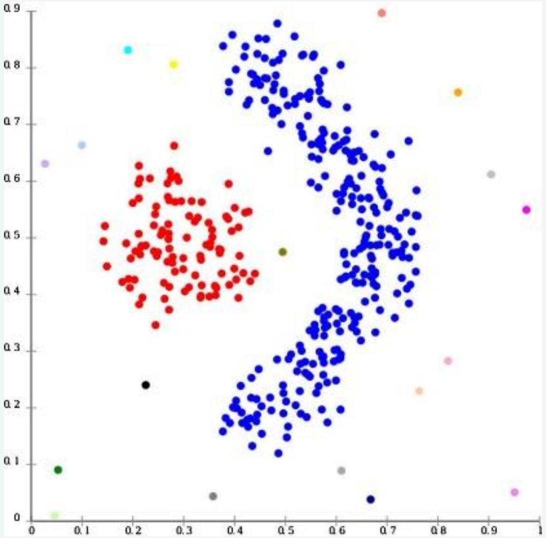
Cluster Properties

- Clusters may have different sizes, shapes, densities
- Clusters may form a hierarchy
- Clusters may be overlapping or disjoint







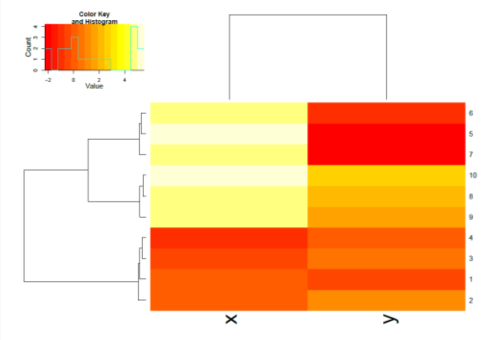


Application Examples

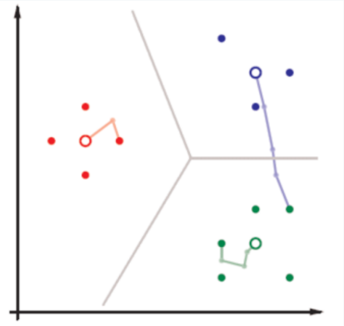
- **Customer segmentation**
  - Find groups of customers with similar behaviour; find customers with unusual behavior
- **Molecule search**
  - Find molecules with similar structure to already working ones
- **Anomaly detection**
  - Find unusual patterns in data from sensors monitoring mechanical engines
- **Determining user groups on the WWW**
  - *Clustering of activities in web-logs*
  - *Find groups of social media users with similar attitude.*
- **Structuring large sets of text documents**
  - *hierarchical clustering of the text documents*
- **Generating thematic maps from satellite images**
  - *clustering sets of raster images of the same area (feature vectors)*

Types of Clustering Approach

Linkage Based
e.g. Hierarchical Clustering



Clustering by Partitioning
e.g. k-Means



We will use those two approaches only

Contents

(Dis-)similarity Functions for Numeric Attributes

For two objects $\boldsymbol{x} = (x_1, x_2, \cdots, x_d)$ and $\boldsymbol{y} = (y_1, y_2, \cdots, y_d)$:

- ***Minkowski-Distance*** **(*L_p-Metric*)**

$$d_p(\boldsymbol{x}, \boldsymbol{y}) = \sqrt[p]{\sum_{i=1}^{d} |x_i - y_i|^p}$$

- ***Euclidean Distance*** **($L_2 - p = 2$)**

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

- ***Manhattan-Distance*** **($L_1 - p = 1$)**

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

- *Tschebyschew-Distance* ($L_\infty - p = \infty$)

$$d_\infty(\boldsymbol{x}, \boldsymbol{y}) = \max_{1 \le i \le d} \{|x_i - y_i|\}$$

- *Cosine Distance*

$$d_C(\boldsymbol{x}, \boldsymbol{y}) = 1 - \frac{\boldsymbol{x}^T \boldsymbol{y}}{\|\boldsymbol{x}\| \|\boldsymbol{y}\|}$$

- *Tanimoto Distance*

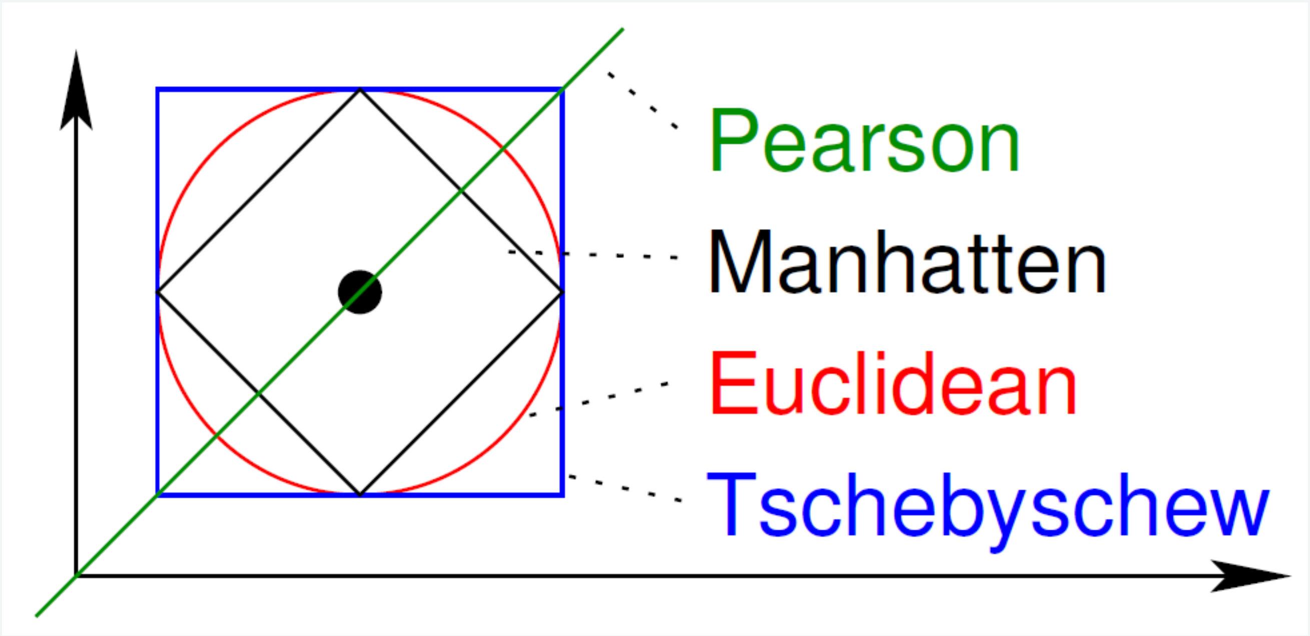
$$d_T(\boldsymbol{x}, \boldsymbol{y}) = 1 - \frac{\boldsymbol{x}^T \boldsymbol{y}}{\|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 - \boldsymbol{x}^T \boldsymbol{y}}$$

- *Pearson Distance*

Euclidean distance of z-score transformed $\boldsymbol{x}, \boldsymbol{y}$

(Dis-)similarity Functions for Numeric Attributes

Various choice of dis-similarity between two numerical vectors



Influence of Distance Function / Similarity

- Clustering vehicles:
  - red Ferrari
  - green Porsche
  - red Bobby car

A. Red Ferrari

B. Green Porche

C. Red Bobby car

- Distance function based on maximum speed (numeric distance function):
  - Cluster 1: Ferrari & Porsche
  - Cluster 2: Bobby car
- Distance function based on color (nominal attributes):
  - Cluster 1: Ferrari and Bobby car
  - Cluster 2: Porsche

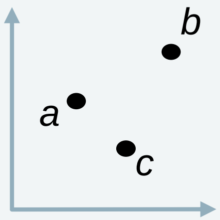
The distance function affects the shape of the clusters

Similarities, Dissimilarities, and Distances

Given data points, how can we summarize how different they are? (Rather than summarizing the similarity)

- Dissimilarity metric – ***distance***
- Distance matrix – pairwise differences of all data points
- Distance $d_{i,j}$ (between i and j) calculated as the Euclidean distance

id	X	Y
a	1	2
b	3	3
c	2	1



$[d_{i,j}]$	a	b	c
a	0.00	2.23	1.41
b	2.23	0.00	2.23
c	1.41	2.23	0.00

Contents

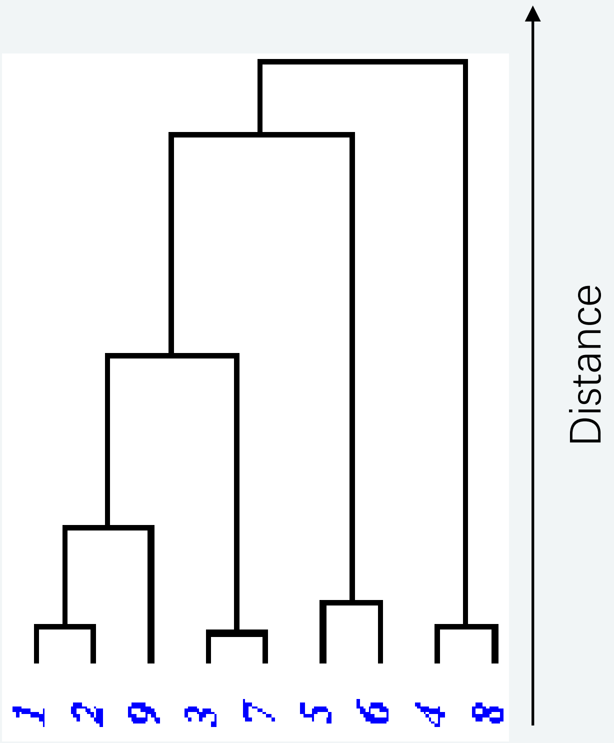
Linkage Hierarchies: Basics

Goal

- Construction of a hierarchy of clusters (*dendrogram*) by merging/separating clusters with minimum/maximum distance

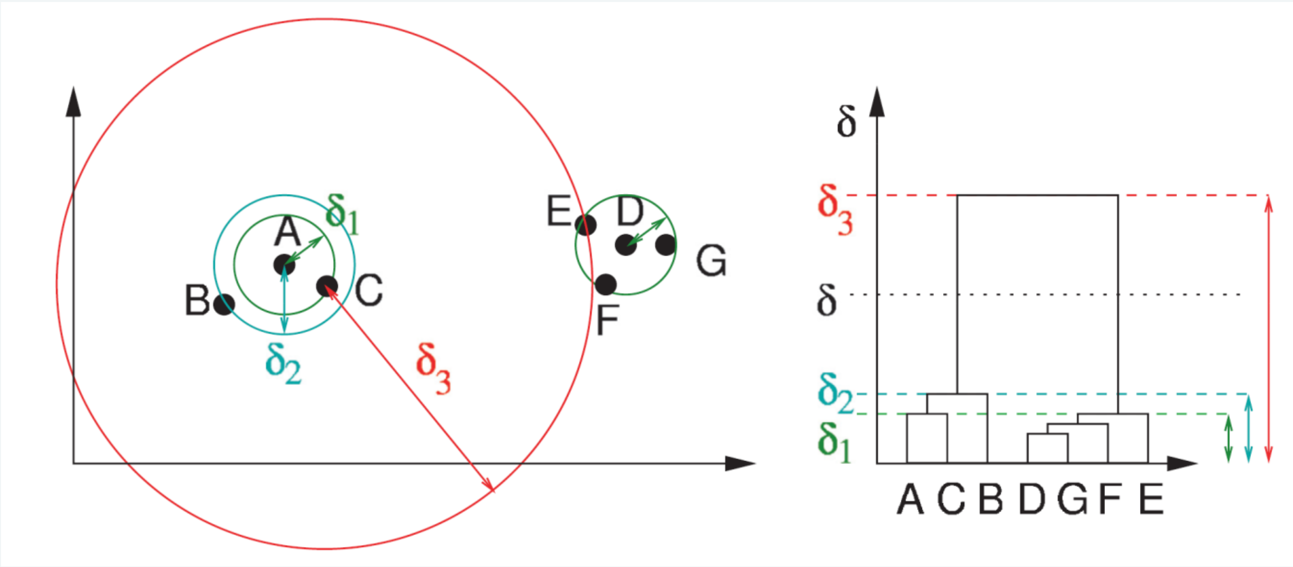
Dendrogram:

- A tree representing a hierarchy of clusters, with the following properties:
  - Root: single cluster with the whole data set.
  - Leaves: clusters containing a single object.
  - Branches: merges / separations between larger clusters and smaller clusters / objects



Hierarchy of Clusters

The hierarchy of the clustering is described by a ***dendrogram***



- At $\delta = 0$ ➔ 7 clusters of singletons
- At $\delta = \mathrm{d}(\mathrm{D}, \mathrm{G})$ ➔ D & G form a cluster, all others are singleton clusters
- At $\delta = \mathrm{d}(\mathrm{A}, \mathrm{C})$ ➔ A & C form a cluster, B remains a singleton, D, E, F, & G is another cluster
- And so on

Agglomerative Hierarchical Clustering

Data points with $distance < \delta$ ➔ belong to the same cluster

- This is known as ***agglomerative*** clustering
- Different values of δ ➔ different partitions

What should be the ideal δ?

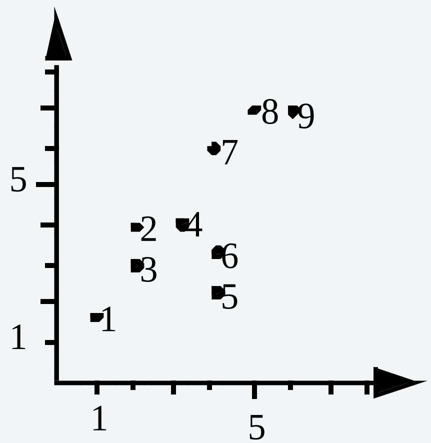
- ***Stable*** and ***robust*** clusters – small alterations should not produce completely different clusters

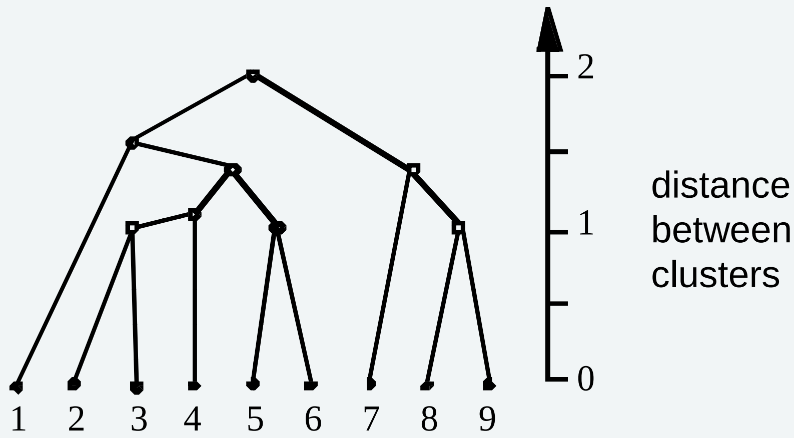
Some properties:

- Clusters found at δ_1 are contained in clusters found at δ_2 (for $\delta_1 < \delta_2$)
- Increasing δ results in a hierarchy of clusters

Linkage Hierarchies: Basics

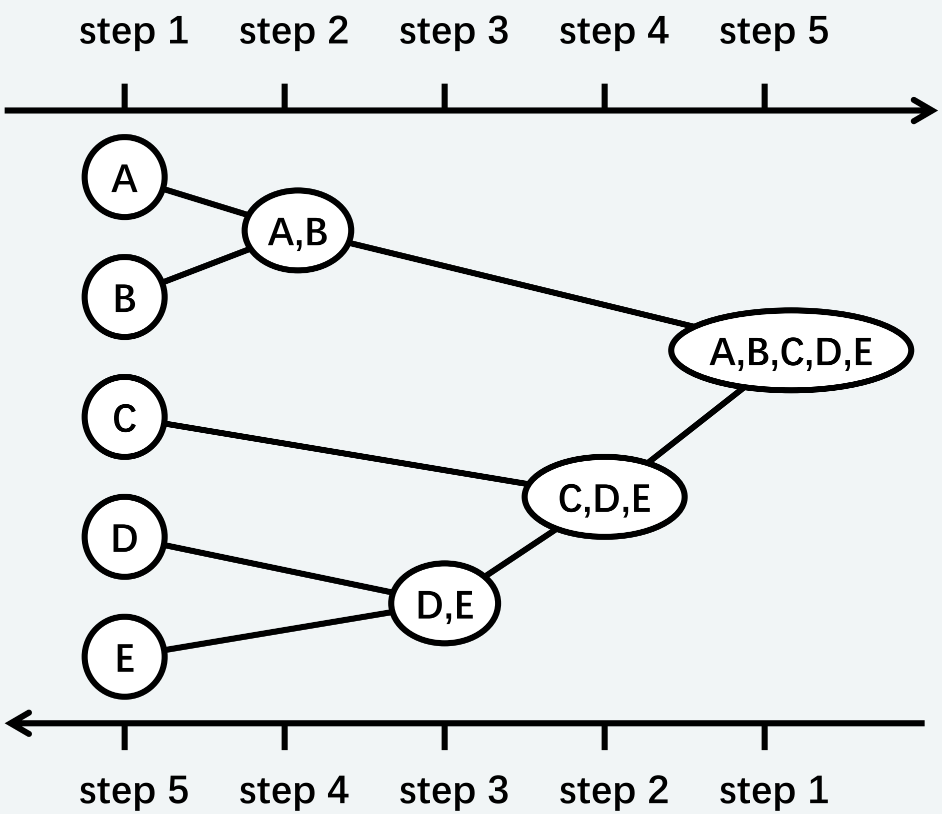
Example dendrogram





- Types of hierarchical methods
  1. Bottom-up construction of dendrogram (*agglomerative*)
  2. Top-down construction of dendrogram (*divisive*)

Agglomerative vs. Divisive Hierarchical Clustering



AGlomerative NESting (AGNES)

DIvisive ANAlysis (DIANA)

Clustering Algorithm

No Need to Remember by Hard

- Start at $\delta = 0$, with each data point as a cluster
- Calculate the distance matrix between all clusters
- Merge the two clusters with smallest distance
- Go back to re-calculate the distance matrix
- Repeat until there is only a single cluster

Algorithm HC($\mathcal{D}$) : $(\mathcal{P}_t)_{t=0..n-1}$, $(\delta_t)_{t=0..n-1}$

input: data set $\mathcal{D}$, $|\mathcal{D}| = n$
output: series of hierarchically nested partitions $(\mathcal{P}_t)_{t=0..n-1}$
series of hierarchy levels $(\delta_t)_{t=0..n-1}$

```
P_0 = {{x} | x ∈ D}
t = 0, δ_t = 0
while current partition P_t has more than one cluster
    find pair of clusters (C_1, C_2) with minimal distance d'(C_1, C_2)
    δ_{t+1} = d'(C_1, C_2)
    construct P_{t+1} from P_t by removing C_1 and C_2 and inserting C_1 ∪ C_2
    t = t + 1
end while
```

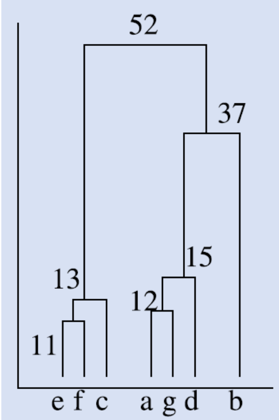
Clustering Algorithm

Example: Distance matrix at each iteration of cluster forming

No Need to Remember by Hard

	a	b	c	d	e	f	g
a	0	50	63	17	72	81	12
b		0	49	41	42	54	37
c			0	52	13	16	61
d				0	56	66	15
e					0	**11**	64
f						0	74
g							0

	a	b	c	d	ef	g
a	0	50	63	17	72	**12**
b		0	49	41	42	37
c			0	52	13	61
d				0	56	15
ef					0	74
g						0

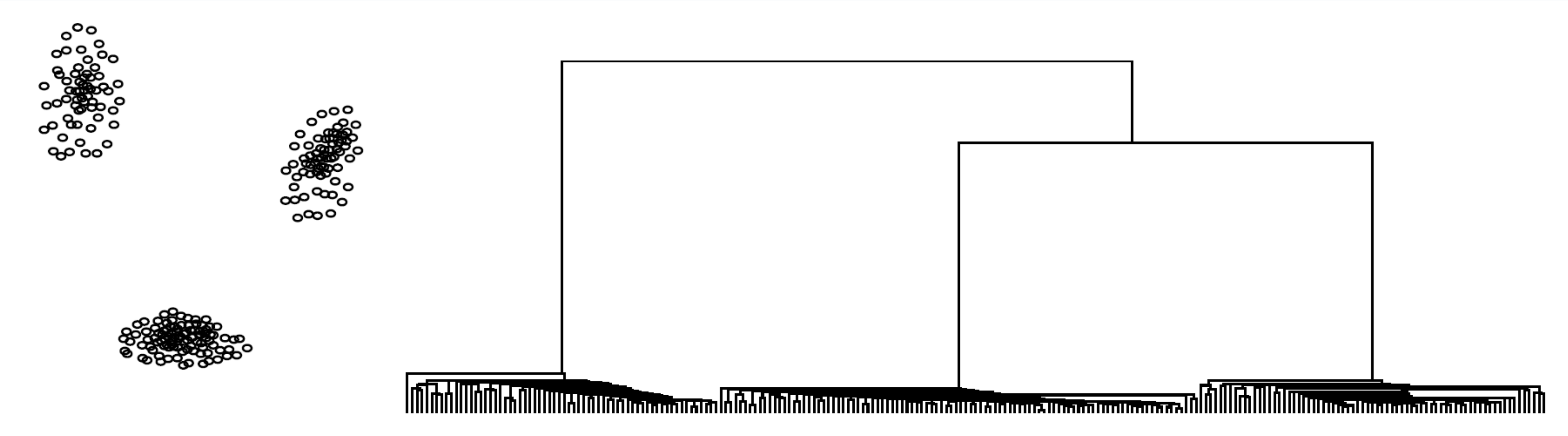


	ag	b	c	d	ef
ag	0	37	61	15	72
b		0	49	41	42
c			0	52	**13**
d				0	56
ef					0

	ag	b	d	efc
ag	0	37	**15**	61
b		0	41	42
d			0	52
efc				0

	agd	b	efc
agd	0	**37**	52
b		0	54
efc			0

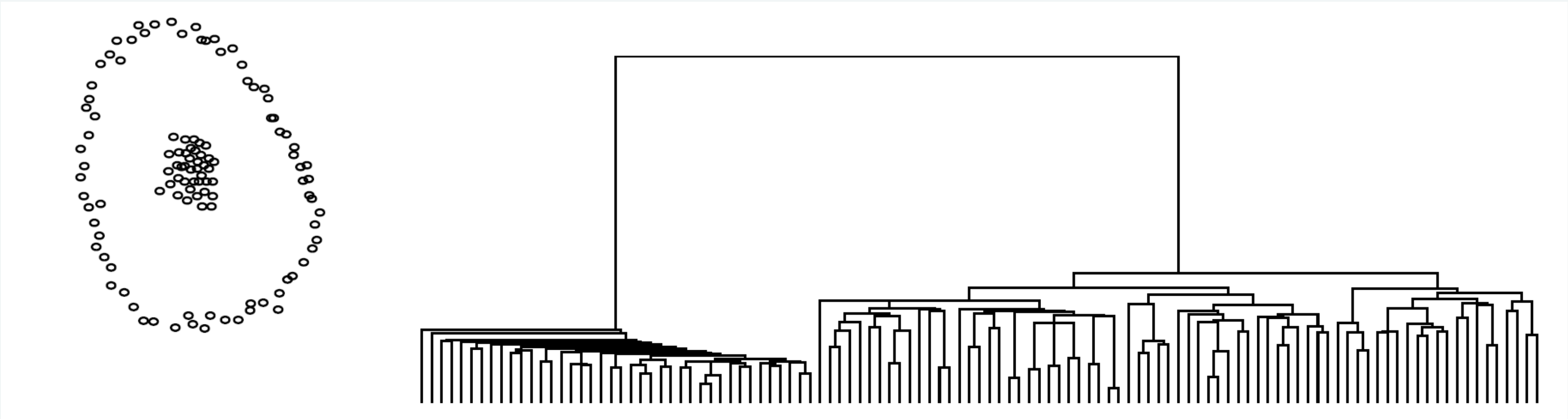
Hierarchy of Clusters (Examples)



In this example:

- Three well-separated clusters
- Formed with small δ
- Remains stable for a wide range of δ (until large δ)

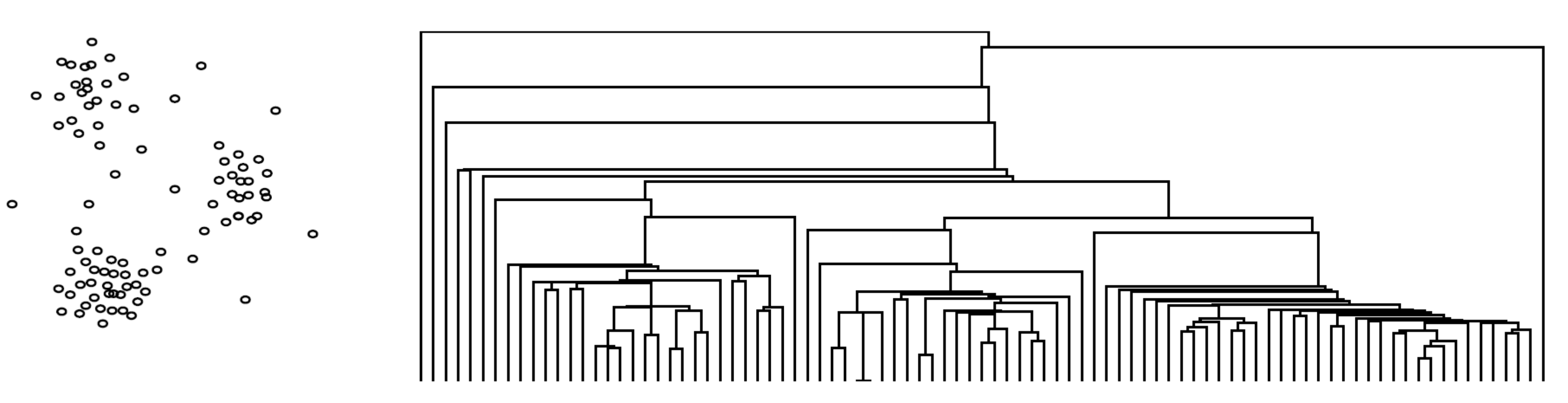
Hierarchy of Clusters (Examples)



In this example:

- Two well-separated clusters
- Clusters differ in sizes and shapes

Hierarchy of Clusters (Examples)



In this example:

- Small noises to clusters
- Isolated noise points are ***chained*** – forming clusters with small δ
- No clearly defined robust clusters
- Hierarchical clustering (esp. single-linkage) is susceptible to noises

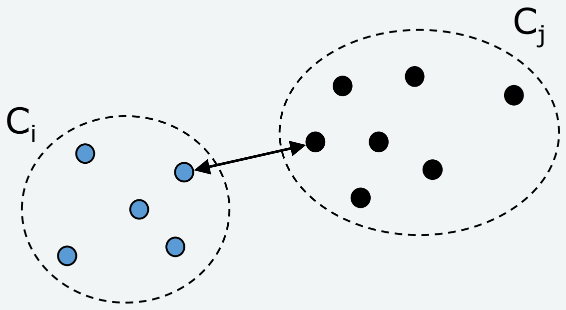
Single Linkage

- Distance between clusters ≡ distance between two closest points

$$d(C_i, C_j) = \min_{x,y}\{d(x,y) \mid x \in C_i, y \in C_j\}$$

Distance of the closest two points, one from each cluster

- Merge Step: Union of two subsets of data points



No Need to Remember by Hard

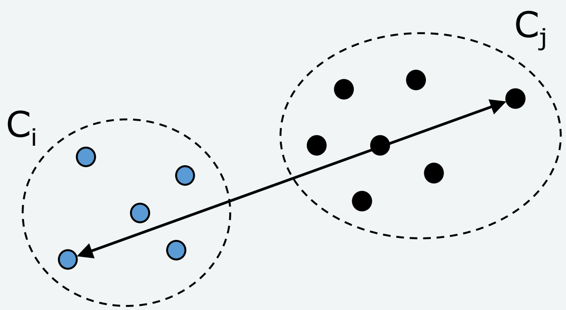
Complete Linkage

- Distance between clusters ≡ distance between two farthest points

$$d(C_i, C_j) = \max_{x,y}\{d(x,y) \mid x \in C_i, y \in C_j\}$$

Distance of the farthest two points, one from each cluster

- Merge Step: Union of two subsets of data points



No Need to
Remember by Hard

Average Linkage / Centroid Method

- Distance between clusters (nodes):

$$Dist_{avg}(C_1, C_2) = \frac{1}{|C_1| \cdot |C_2|} \sum_{p \in C_1} \sum_{p \in C_2} dist(p, q)$$

Average distance of all possible pairs of points between C_1 and C_2

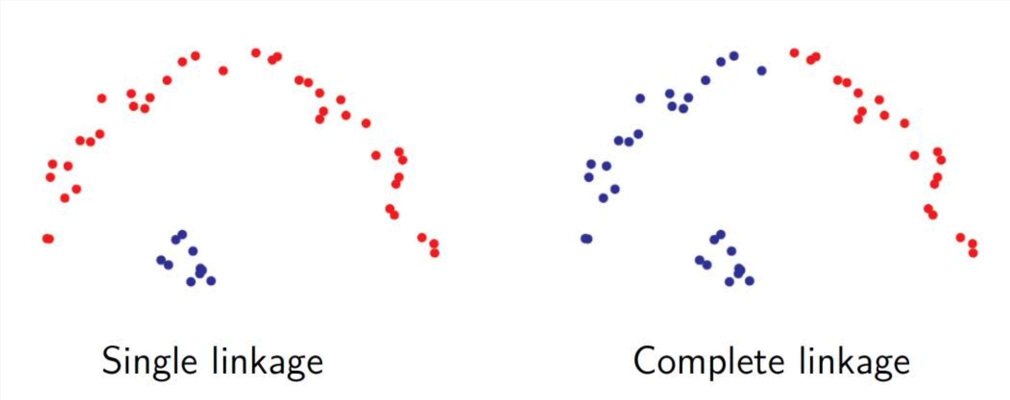
$$Dist_{mean}(C_1, C_2) = dist\big(mean(C_1), mean(C_2)\big)$$

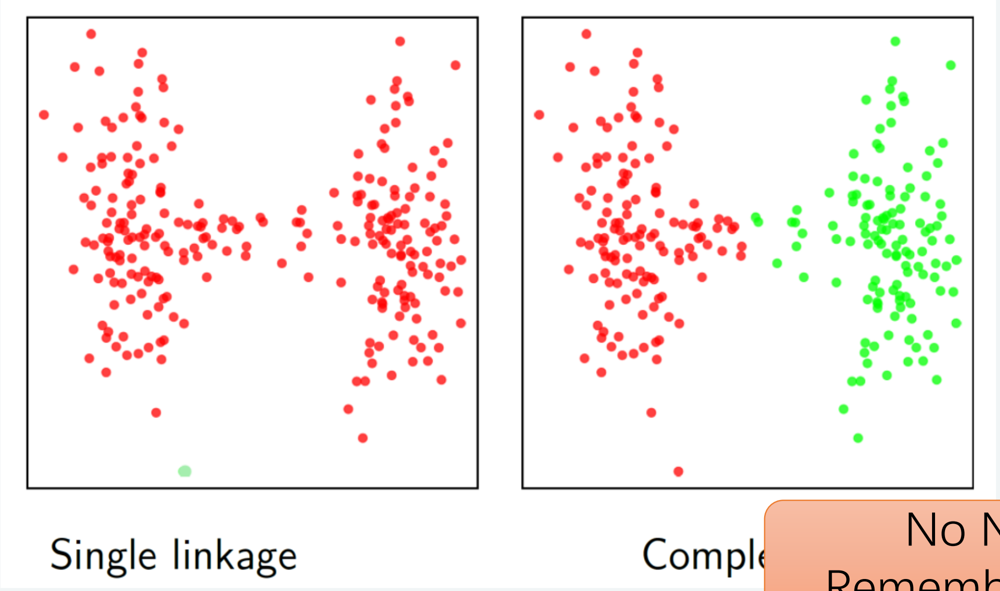
Distance between two centroids

- Merge Step:
  - union of two subsets of data points
  - construct the mean point of the two clusters

No Need to Remember by Hard

Single Linkage vs. Complete Linkage

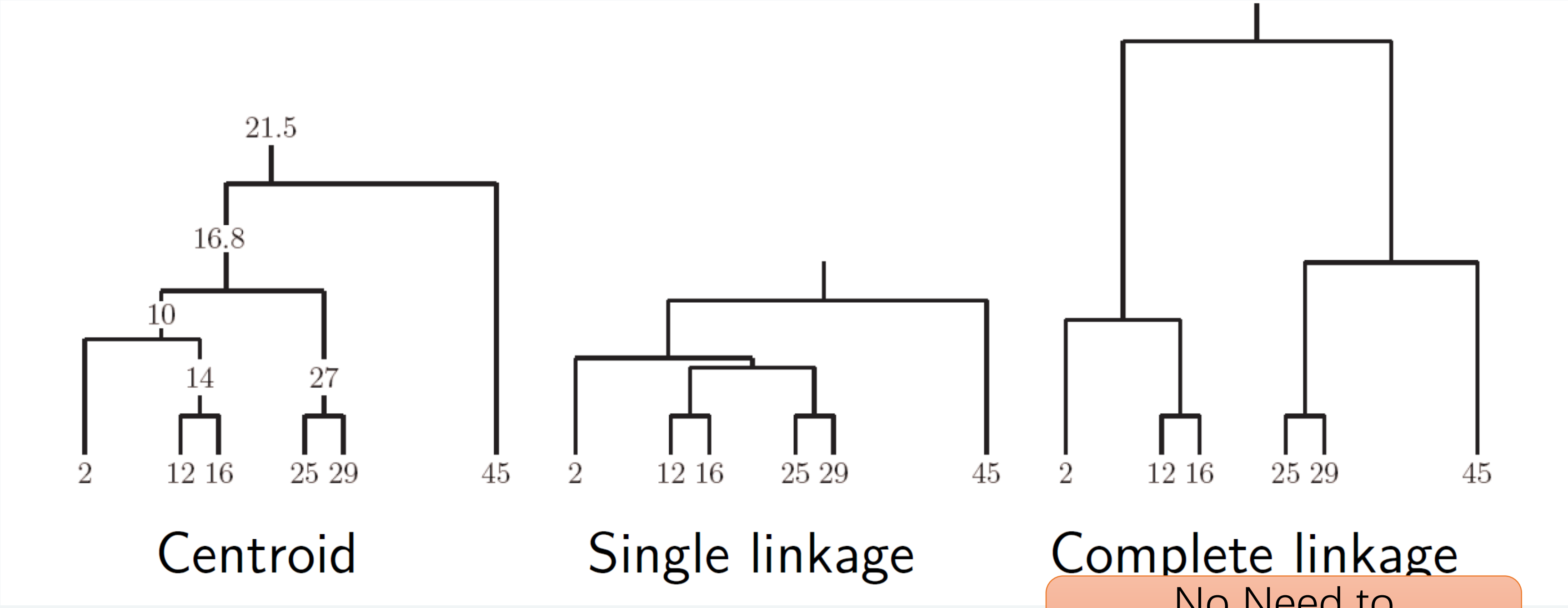




No Need to Remember by Hard

Single vs. Complete vs. Average Linkage

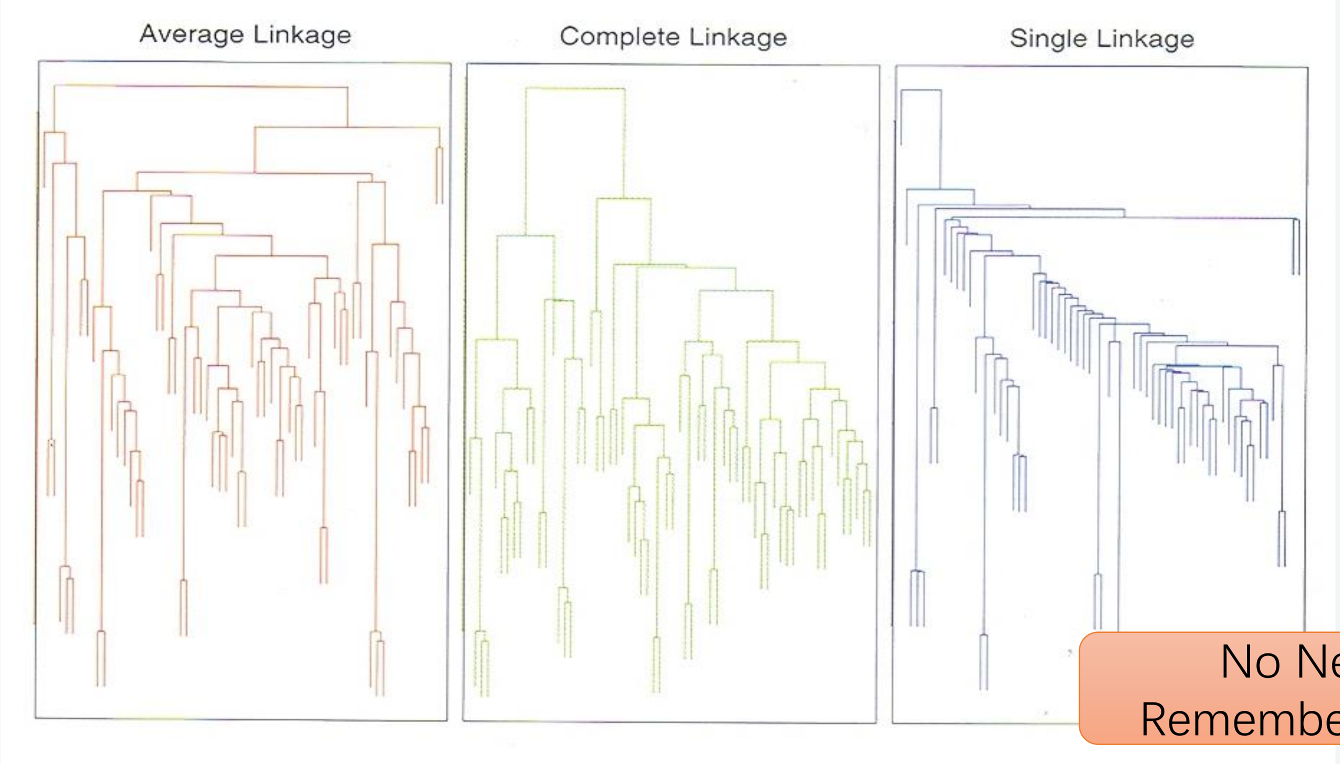
- Clustering of the 1-dimensional data set {2, 12, 16, 25, 29, 45}.
- All three approaches to measure the distance between clusters lead to different dendrograms.



No Need to Remember by Hard

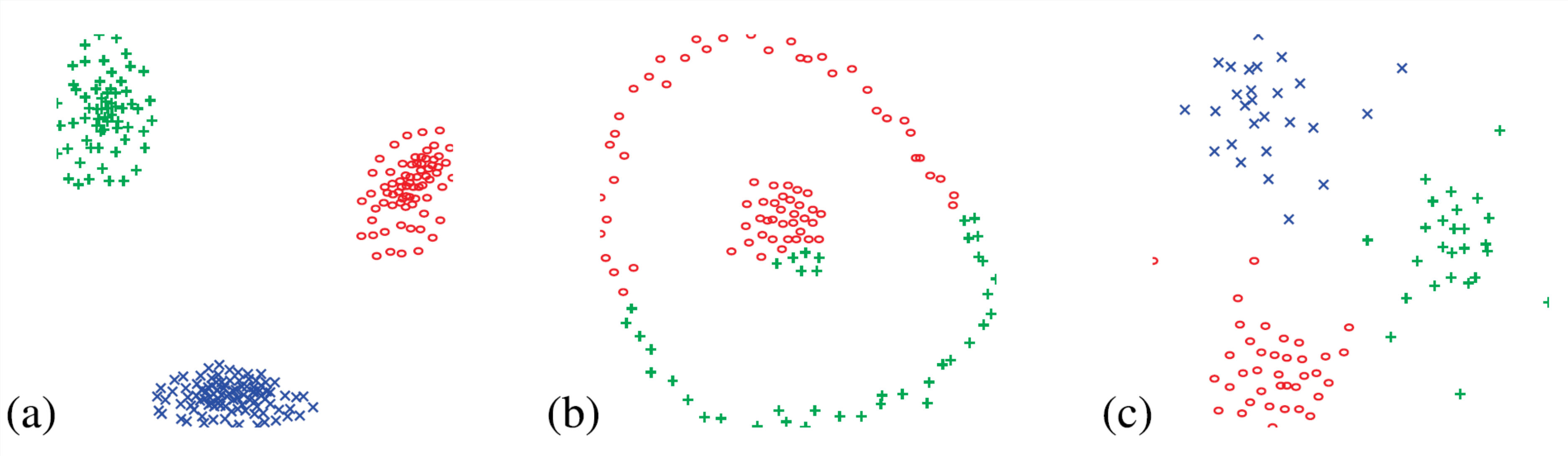
Linkage Based Clustering

- Single Linkage: Prefers well-separated clusters
- Complete Linkage: Prefers small, compact clusters
- Average Linkage: Prefers small, well-separated clusters…



No Need to Remember by Hard

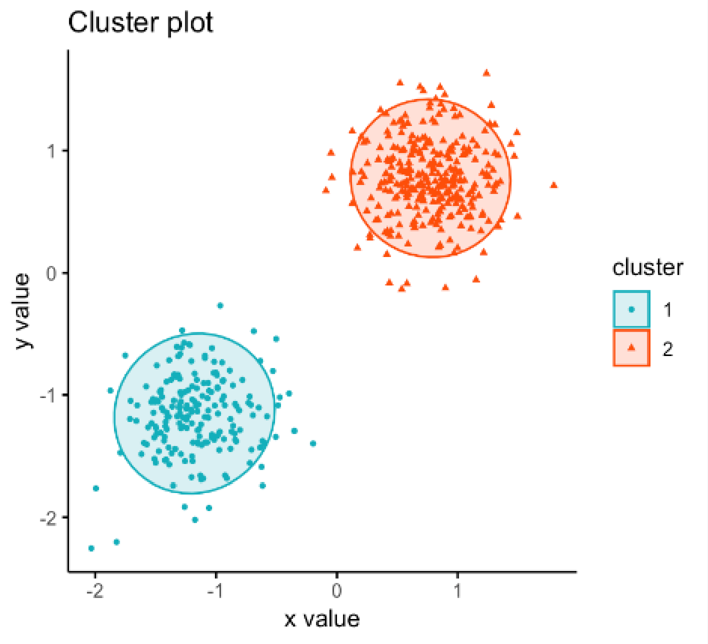
Contents



- Works well when the assumption is true (i.e. closest prototype represents a cluster)
- But does not work all cases – e.g., circular clusters

Partitioning

Goal: A (disjoint) partitioning into k clusters with minimal costs



Local optimization method:

- choose k initial cluster representatives
- optimize these representatives iteratively
- assign each object to its **most similar cluster representative**

Types of cluster representatives:

- Mean of a cluster (*construction of central points*)
- Median of a cluster (*selection of representative points*)
- Probability density function of a cluster (*expectation maximization*)

Construction

Cluster partition is described by a ***membership matrix*** $[p_{i|j}]$

- $p_{i|j}$: membership belongingness of data point *j* to cluster *i*.
- $p_{i|j}$ can be binary or real-valued

- The number of clusters k must be known beforehand
- Data points are assigned to the closest model or prototype
- Update the prototype based on the new cluster partition
- Repeat until the model / prototype stops moving

k-Means

Given k, the k-Means algorithm is implemented in four steps:

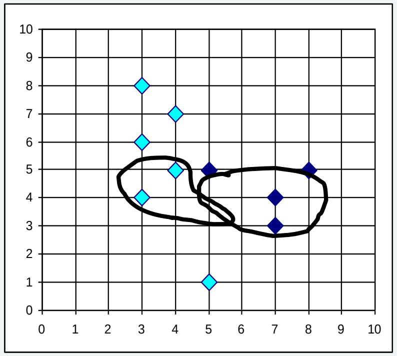
1. Partition objects into k non-empty subsets, calculate their **centroids** (i.e., **mean point**, of the cluster)
2. Assign each object to the cluster with the **nearest** centroid (Euclidean distance)
3. Compute the centroids from the current partition as $p_i = \frac{\sum_{j=1}^{n} p_{i|j}\ x_j}{\sum_{j=1}^{n} p_{i|j}}$
4. Go back to Step 2, repeat until the updated centroids stop moving significantly

Note: Each data point can only belong to a single cluster

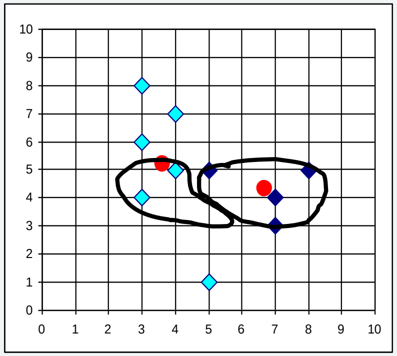
- $p_{i|j} = 1$ for the cluster with closest prototype, 0 otherwise

No Need to Remember by Hard

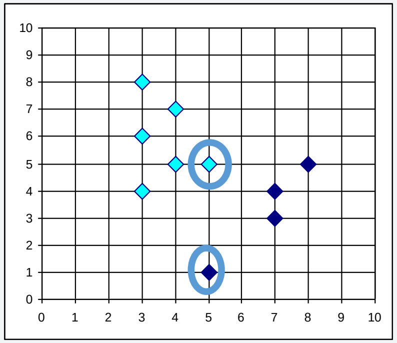
k-Means Algorithm



Calculation of new centroids →

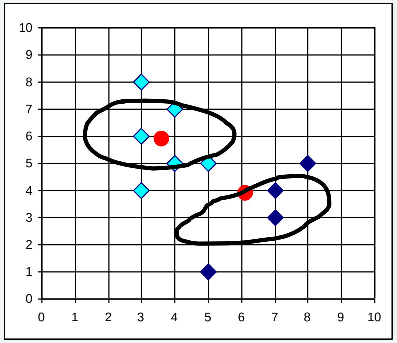


Cluster assignment ↓



No Need to Remember by Hard

← Calculation of new centroids



Comments of the k-Means Method

Advantages:

- Relatively efficient: $O(tkn)$ where n is #objects, k is #clusters, and t is #iterations; usually, $\mathrm{k}, t \ll n$ (t typically $5-10$)
- Simple implementation
- **k-means is the most popular partitioning clustering method!**

Weaknesses:

- Often terminates at a local optimum. A better local optimum may be found using techniques such as: deterministic annealing and genetic algorithms.
- Applicable only when mean is defined (what about categorical data?)
- Need to specify k, the number of clusters, in advance
- Unable to handle noisy data and outliers
- Not suitable to discover clusters with non-convex shapes

No Need to Remember by Hard

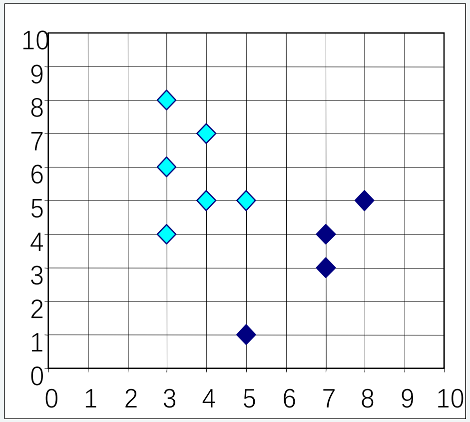
Outliers: k-Means vs k-Medoids

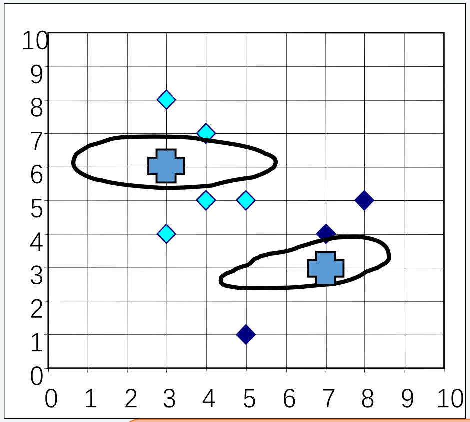
Problem with K-Means

An object with an extremely large value can substantially distort the distribution of the data.

- **One solution: K-Medoids**

Instead of taking the **mean** value of the objects in a cluster as a reference point, **medoids** can be used, which are the most centrally located objects in a cluster.





No Need to Remember by Hard

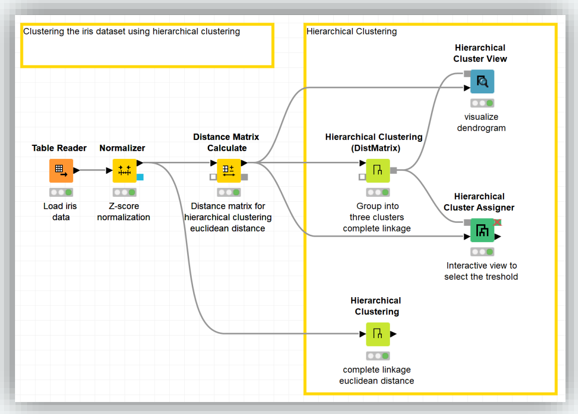
Contents

Datasets

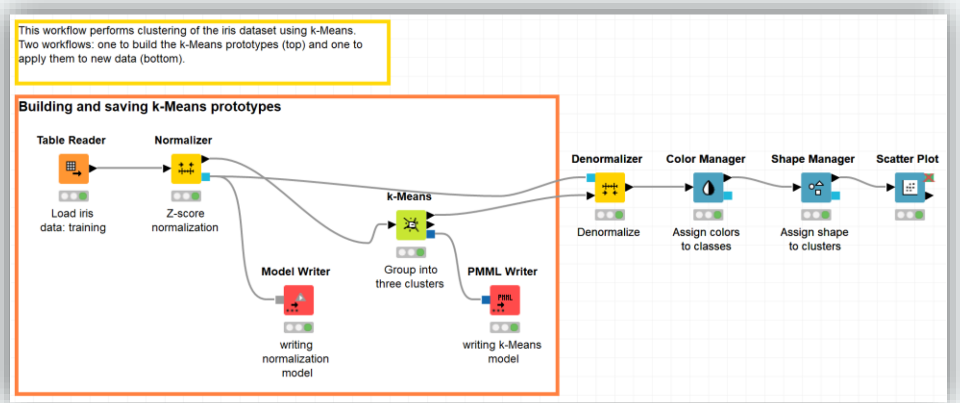
Dataset used : iris dataset

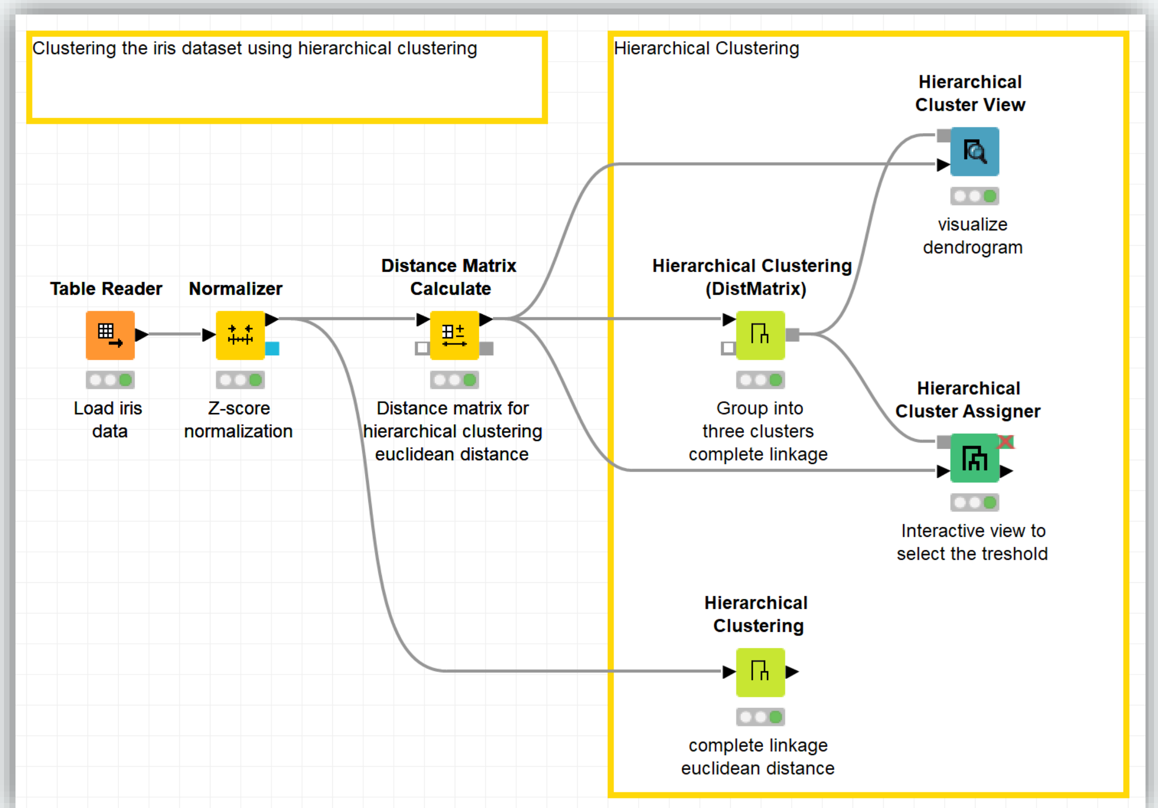
Workflows:

- Hiearchical Clustering“ https://kni.me/w/rlXFxYxQmbgNgSsM
    - Normalization
    - Distance calculation
    - Hiearachical clustering



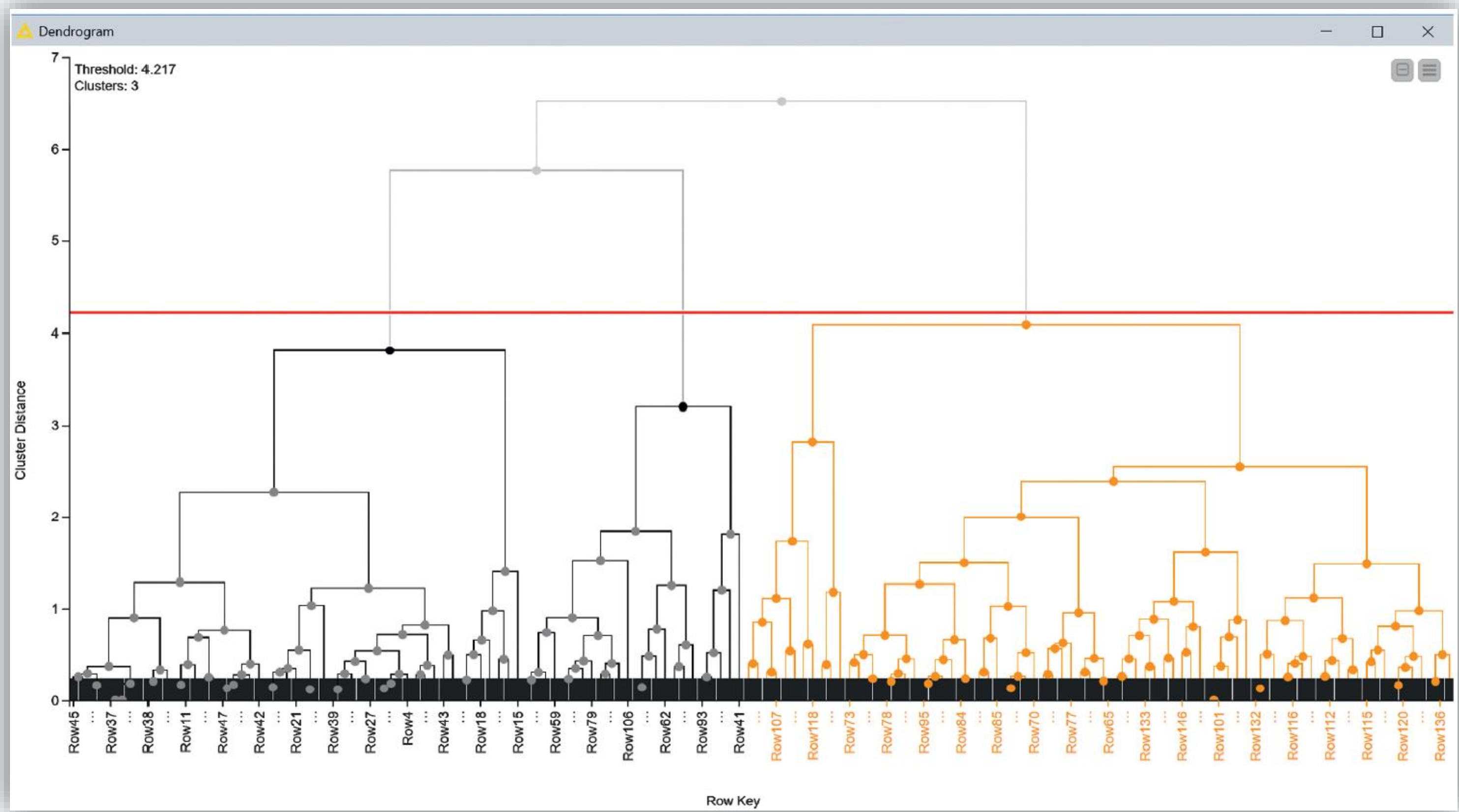
- K-means clustering“ https://kni.me/w/t8UVEQH1sTTkus_w
    - Partitioning
    - Numeric error measures





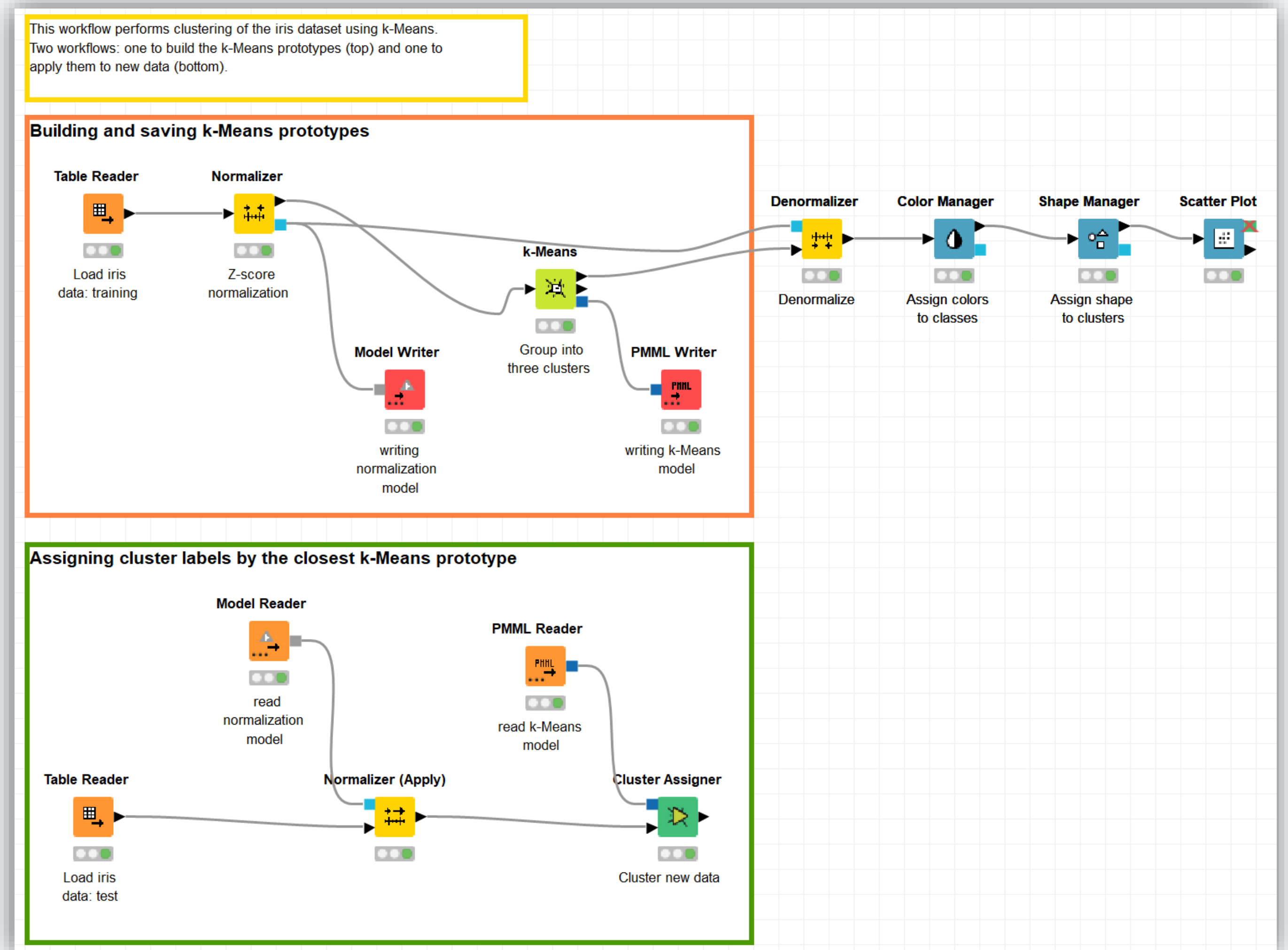
Workflow implementing hierarchical clustering with the simple Hierarchical Clustering node and with the more complex sequence of nodes, including the Distance Matrix Calculate node

Hierarchical Clustering



A dendrogram for the iris data obtained with Euclidean distance and complete linkage. Moving the threshold line changes the number of clusters and the assignment for the input rows.

K-Means Clustering



Building of k-Means prototypes (top) and cluster assignment (bottom)