MONASH University
Information Technology

MONASH University
Information Technology

FIT5047 – Intelligent Systems

Machine Learning
Chapters 18.1-4, 18.6.1, 18.8.1, 20.1-2

Learning Objectives (I)

- **Machine learning – definition**
  - Relationship with data mining
  - Issues about learning from data
- **Approaches to learning from data**
  - Verification driven
  - Discovery driven – inductive learning

    > supervised and unsupervised learning

Learning Objectives (II)

- **Supervised machine learning**
  - Decision trees
  - Naïve Bayes
  - k Nearest Neighbour (k-NN)
  - Regression
  - (Logistic regression)
- **Clustering (Unsupervised learning)**
  - The clustering problem
  - Similarity measures
  - The K-means algorithm

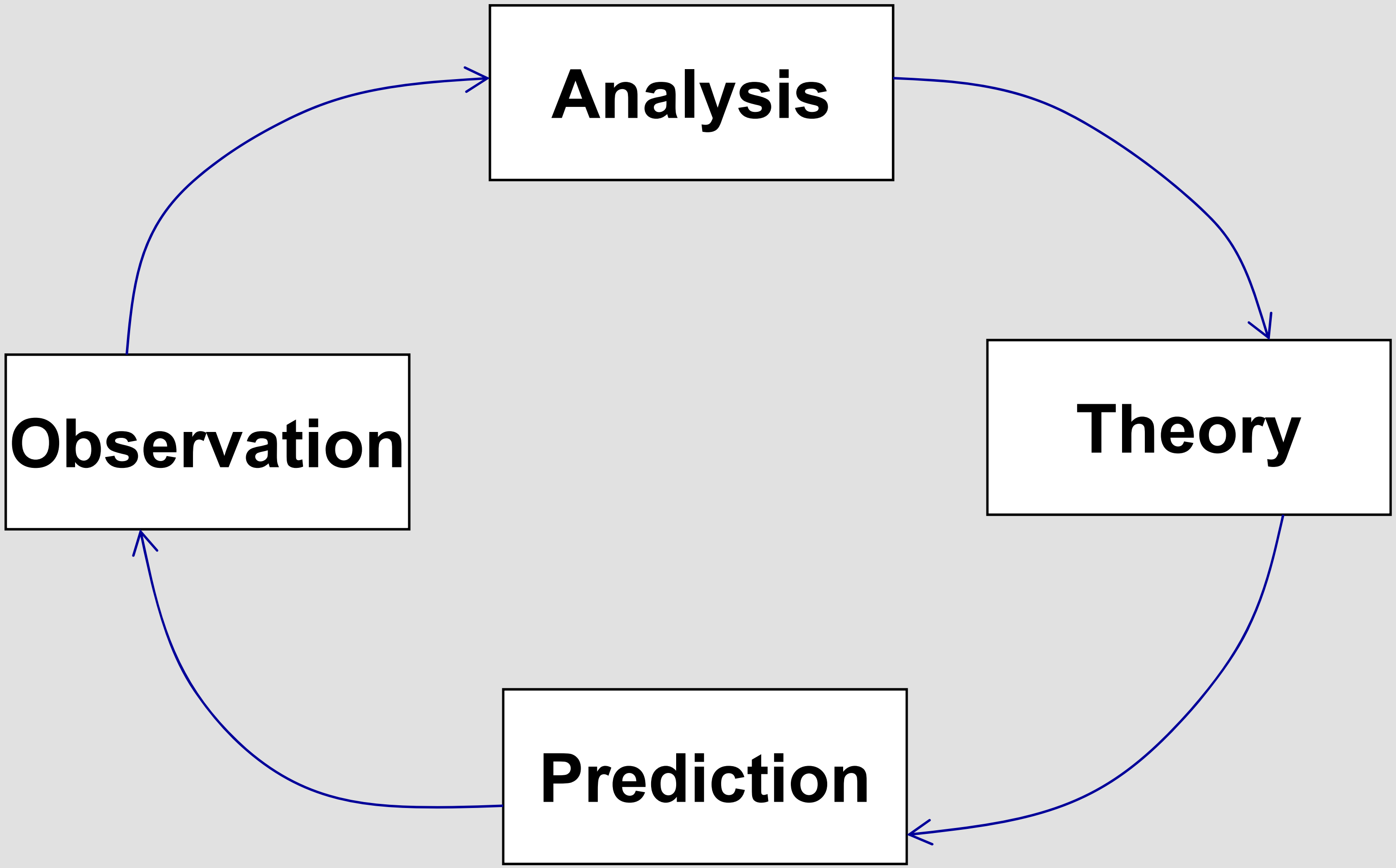
Why Learning?

- **Learning is essential for unknown environments**
  - e.g., when the designer lacks omniscience
- **Learning is necessary in dynamic environments**
  - An agent can adapt to changes in the environment not foreseen at design time
- **Learning is useful as a system construction method**
  - Expose the agent to reality rather than trying to approximate it through equations
- **Learning modifies the agent's decision mechanisms to improve performance**

Machine Learning

- ***Machine Learning*** **is an area of AI concerned with programs that learn from data**
  - Machine learning performs ***inductive learning***
- ***Inductive learning*** **is learning from examples**
- **A program makes predictions, then learns from feedback on the correctness of the predictions**
- **Machine Learning is used in many applications**
  - E.g., speech recognition, robot training, game playing, and classification of astronomical structures

The Empirical Cycle



Types of Learning Approaches

- **Verification Driven**
  - An expert formulates a hypothesis, which is validated by analyzing the data
- **Discovery Driven – Machine Learning**
  - Hypotheses are formulated bottom-up from the data
    - > Interesting patterns in the data are noticed and evaluated by the user/expert
  - Types of discovery-driven learning:
    - > Supervised learning, unsupervised learning, reinforcement learning

Learning Concepts with Machine Learning

- **Machine learning aims to reveal knowledge about data**
  - concepts, definitions, hypotheses
- **A learning algorithm considers many instances and infers useful regularities about them**
  - ➔It can ***infer (learn)*** a definition of a class
    - this definition may be viewed as ***patterns*** or ***structures*** within the data
- **The structure of the discovered patterns depends on the learning techniques**
  - E.g., rules, decision trees

Completeness and Consistency in Inductive Learning

- **A definition is**
  - ***complete*** if it recognizes all instances of a class
  - ***consistent*** if it does not classify any negative examples as instances of this class
- **An *incomplete* definition is too narrow**
  - it would not recognize some "in-class" instances as belonging to the class
- **An *inconsistent* definition is too broad**
  - it would classify some "out-of-class" instances as belonging to the class

Machine Learning as Search

- **Find the** best **hypothesis among many**
  - Need a measure of the quality of candidate hypotheses
- **The search space is very large**
- **We don’t know the optimal answer**

Hypothesis Characteristics

- **Performance**
  – how often the hypothesis is correct (different measures)
- **Significance**
  – must perform better than the naïve hypothesis (e.g., majority)
  – ***statistical significance*** – perform better than chance
- **Information content**
  – must take into account all relevant information
- **Transparency**
  – how easy it is to understand the generated hypotheses

Machine Learning as Compression

- **We have learnt something if we have created a description of the data that is shorter than the original data set**
- **Example – Input/Output**
  - **Input:** a table of instances
  - **Output:** a prediction of an instance given a set of features or a suggested clustering of the instances

Machine Learning and Data Mining

- **Machine learning (ML) and data mining (DM) overlap**
- **We can consider ML as a *scientific* discipline, and DM as an *engineering* discipline**

Machine Learning Principles

- ***"A general law can never be verified by a finite number of observations. It can, however, be falsified by only one observation."***

  Karl Popper

  – The patterns that machine learning algorithms find can never be definitive theories

- **Any results discovered** *must* **be tested for performance and statistical significance**

Types of Machine Learning

- ***Supervised learning*: correct answers are provided for each input**
  - E.g., Decision Trees, Naïve Bayes, K-Nearest Neighbour (k-NN), Regression, Neural Nets
- ***Unsupervised learning*: correct answers are *not* given, must discover patterns in input data**
  - E.g., K-Means, Snob (Minimum Message Length Principle)
- **(*Reinforcement learning*: occasional rewards (or punishments) are given**
  - E.g., Q-learning)

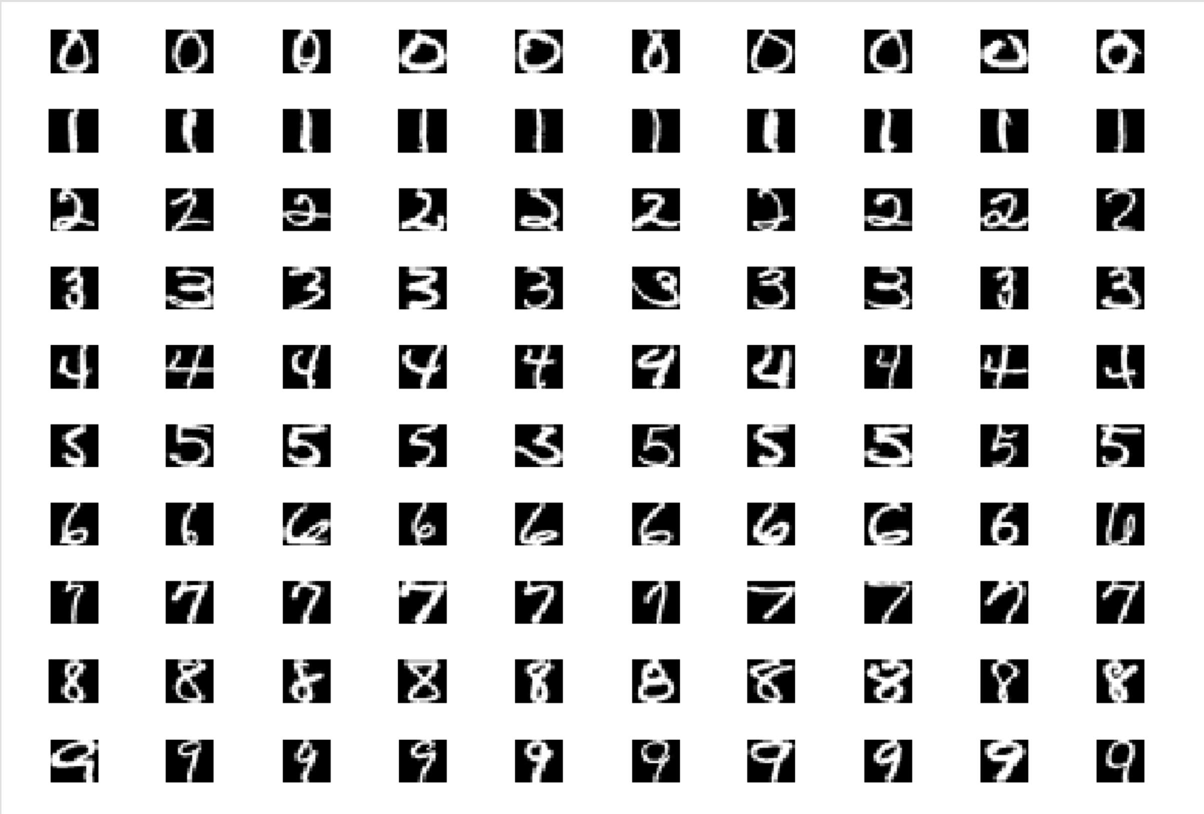
MONASH University
Information Technology

FIT5047 – Intelligent Systems

Mathematical Principles of Machine Learning

Supervised Learning Example: Handwritten Digit Recognition (I)

- **Classify images into one of the digits: 0, 1, …, 9**
  - Useful for the post office to automatically detect addresses



Supervised Learning Example: Handwritten Digit Recognition (II)

- **Learn a function f from examples**

Picked from a hypothesis space H

$h \approx f$

$x \longrightarrow y$

(images) (digits)

Training Data

(x_1,y_1)

(x_2,y_2)

…

(x_n,y_n)

image of digit

actual digit

Supervised Inductive Learning

- **Setup:**
  - f is the **unknown** target function
  - We are given some sample pairs from it (x, f(x))
- **Problem: learn a function hypothesis h**
  - Based on the set of ***training*** examples
  - Such that $h \approx f$ (h approximates f as best as possible)
    - > **h must generalize well on unseen examples**

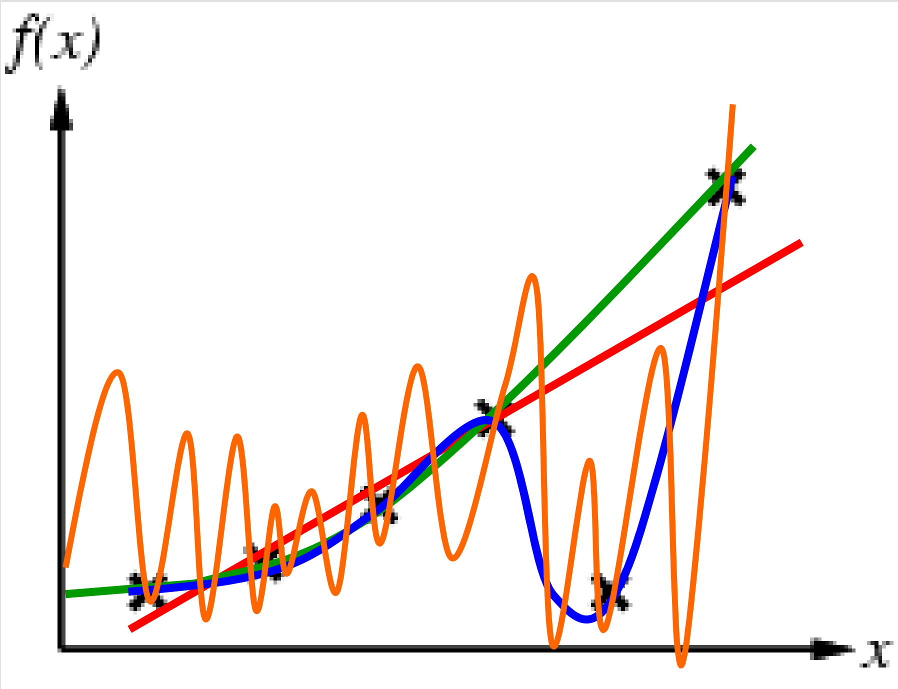
Picking the Best Hypothesis h

- **Big Idea 1:**
  Pick h from the space H which agrees with f on the training set
  - Complete and consistent

Inductive Learning – Example (I)

Curve fitting (regression):

h= Straight line? Quadratic? Cubic? Other?



Training Data
(x_1,y_1)

(x_2,y_2)
…
(x_n,y_n)

Ockham's Razor

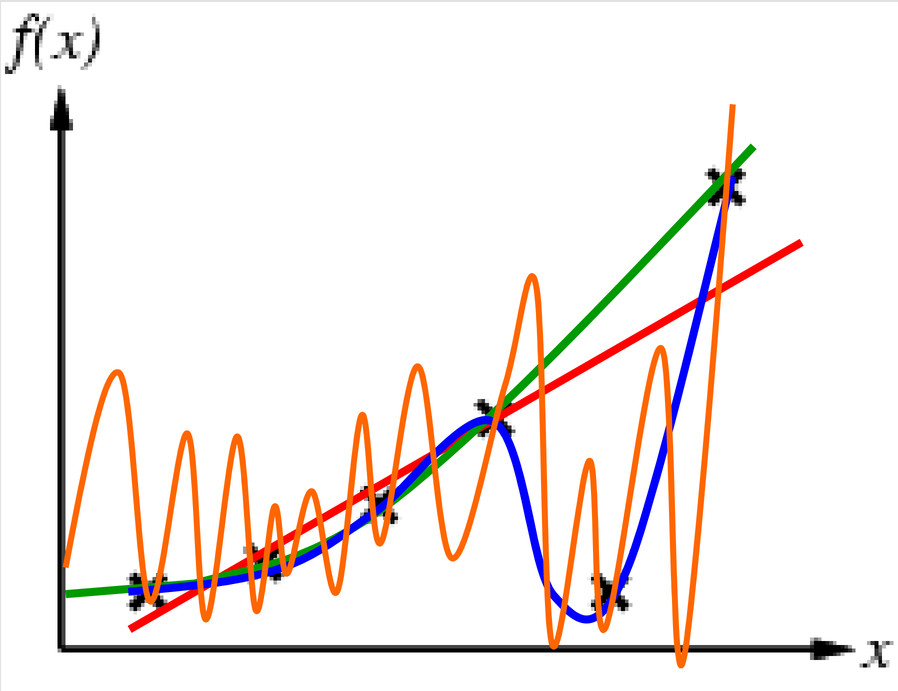
- ***"Entities must not be multiplied beyond necessity" ("Non sunt multiplicanda entia sine necessitate")***
  William of Ockham (1287–1347)

- **Big Idea 2: Prefer a simpler hypothesis to complex ones *provided both explain the data equally well***
  – Overly complex hypotheses tend to *overfit* the data

Inductive Learning – Example (II)

Curve fitting (regression):

h= Straight line? Quadratic? Cubic? Other?



Training Data

(x_1,y_1)

(x_2,y_2)

…

(x_n,y_n)

Mathematical Principles of Learning (I)

Idea 2: To improve generalizability and prevent overfitting

Bias

Choose the *simplest* hypothesis from H which is *complete and consistent* with the training data

Idea 1: Should be similar to the unknown true underlying function

Bias and Variance

- **Bias is the true error of the best classifier in the concept class**
  - Bias is high if the concept class cannot model the true data distribution well, e.g., it is too simple
  - High bias → both training and test error are high
- **Variance is the error of a trained classifier with respect to the best classifier in the concept class**
  - Variance decreases with more training data, and increases with more complicated classifiers
  - High variance → training error is low, and test error is high

Mathematical Principles of Learning (II)

Best learned hypothesis

Idea 2: penalize complexity – regularization

$$h^* = \operatorname{argmin}_{h \in H} \left\{ \left[\sum_{(x,y) \in D} error(h(x), y) \right] + \lambda Complexity(h) \right\}$$

Bias

Idea 1: Error on the training data

Learning is indeed Search/Optimization

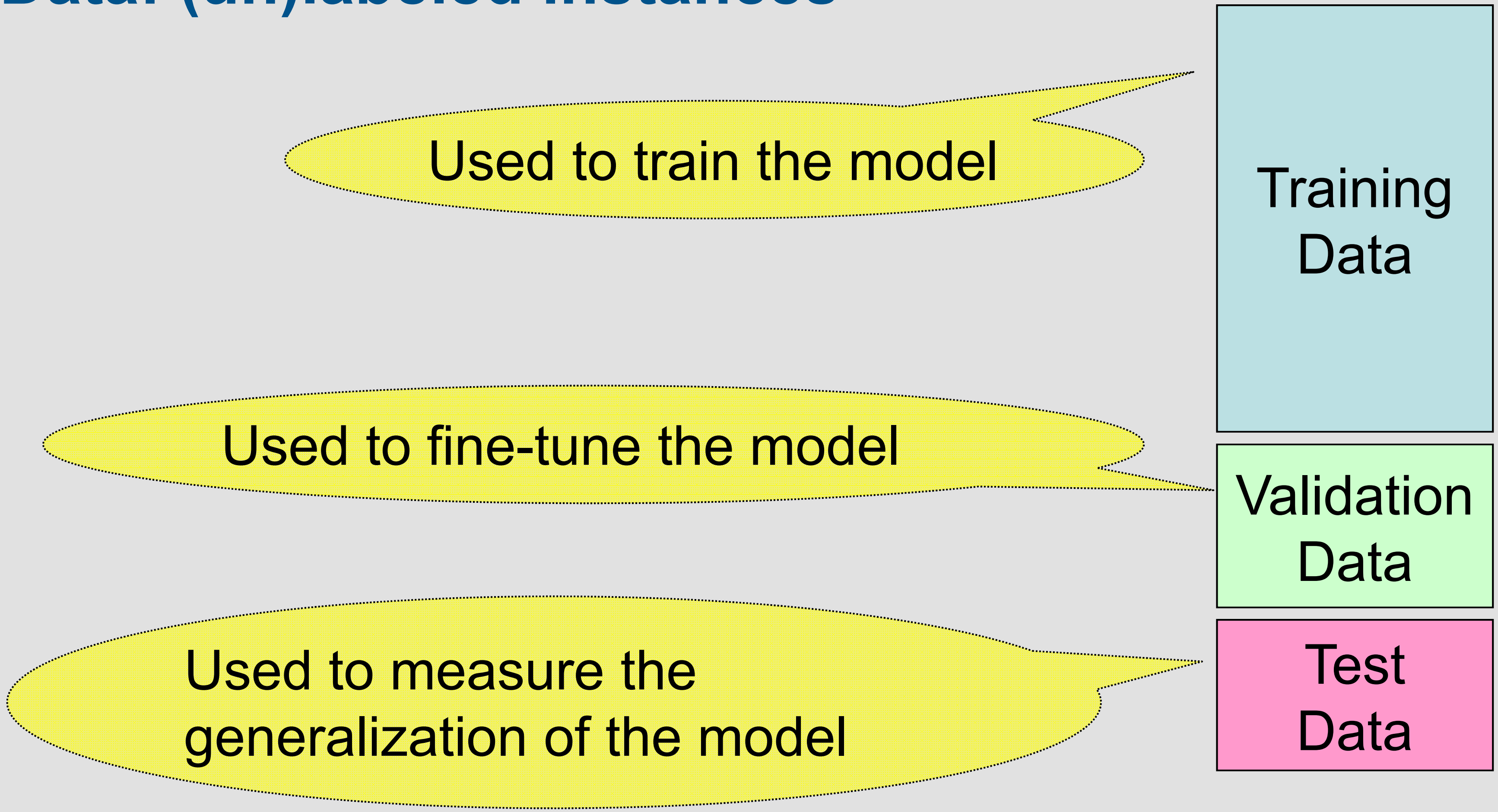
MONASH University
Information Technology

FIT5047 – Intelligent Systems

Concepts in Machine Learning

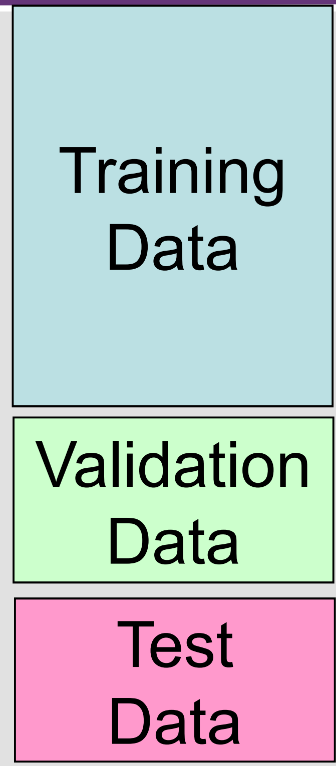
Types of Data

- **Data: (un)labeled instances**

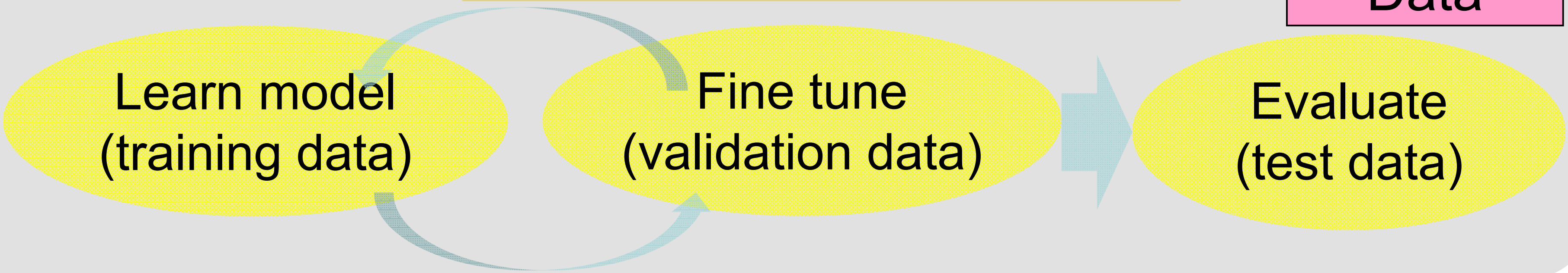


Inductive Learning Process

- **Features: attribute-value pairs which characterize each instance**
- **Experimentation cycle**
  - Learn model parameters on **Training Data**
  - Fine tune the model **on Validation (Held-out) Data**
  - Compute performance on **Test Data**



Very important: never "peek" at the test set!



Evaluation Metrics

- accuracy $\frac{correctly\ predicted}{predicted}$

If 80 predictions are correct out of 100 then accuracy = 80/100 =0.8

- recall $\frac{correctly\ predicted\ as\ class\ C}{instances\ in\ class\ C}$
- precision $\frac{correctly\ predicted\ as\ class\ C}{predicted\ as\ class\ C}$

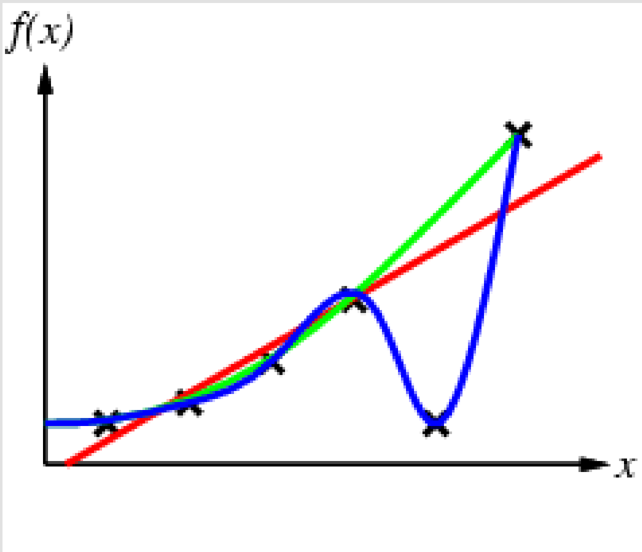
Training Data

Validation Data

Test Data

Overfitting and Generalization

- **We want a learned procedure that does well on *test* data**
  - Overfitting: fitting the training data very closely, but not generalizing well



Training Data

Validation Data

Test Data

MONASH University
Information Technology

FIT5047 – Intelligent Systems

Approaches to Learning from Data

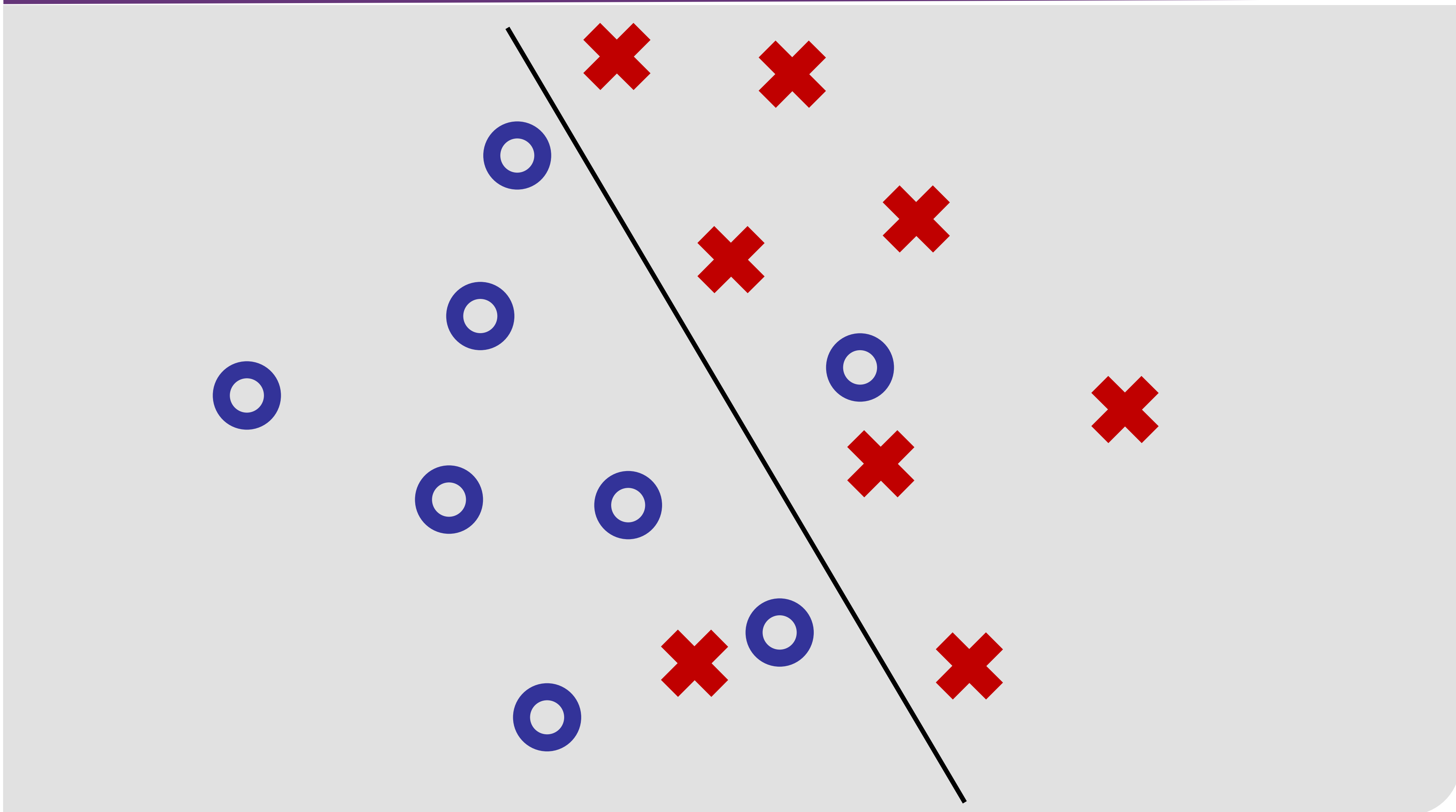
Discovery Driven – Supervised Learning

- **Build patterns by inferring an unknown attribute given the values of known attributes**
- **Principal Techniques: Classification and Regression**
- **Supervision:**
  – a set of observations (***training data***) is accompanied by labels indicating the class of each observation
  – new data are classified/predicted based on the model learned from the training data

Type of Inferred Value – Classification vs Regression

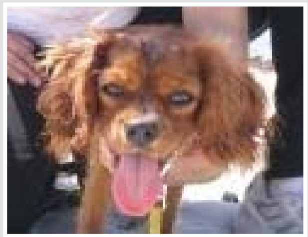
- **Classification**
  - Infers categorical or discrete values
- **Regression**
  - Infers continuous or ordered values

Classification

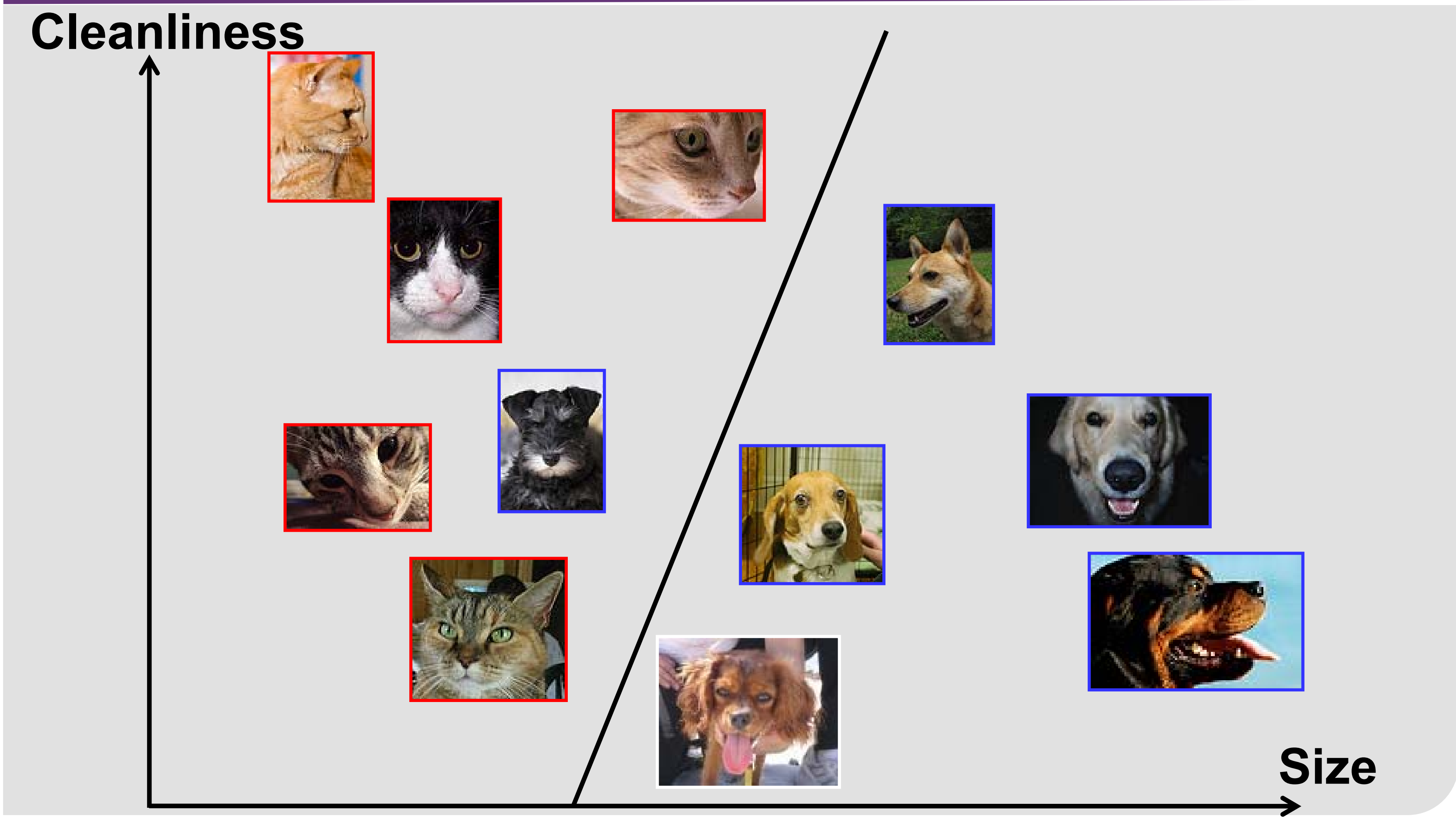


Classification – Example (I)

Cat **?** **Dog**



Classification – Example (II)

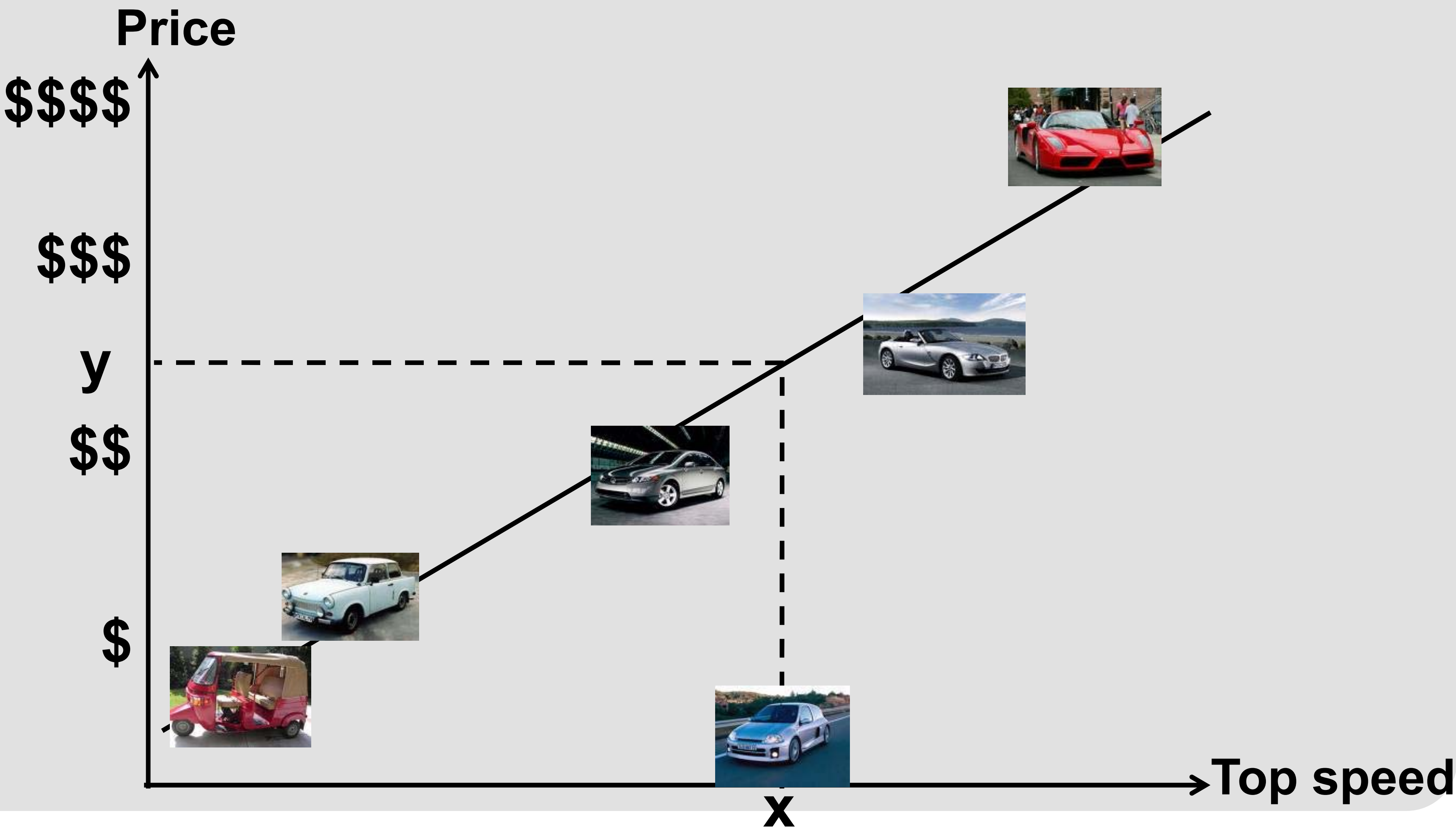


Regression – Example (I)

$ $$ $$$ $$$$

?

Regression – Example (II)



Time of Inferred Value – Classification vs Prediction

- **Classification considers the features of an object in order to assign it to a pre-defined class**
  - Examples:
    - > spotting fraudulent insurance claims
    - > identifying high-value customers
- **Prediction is the classification of future events**
  - Historical data is used to build a model that "explains" outputs for known inputs
  - The model is applied to new inputs to predict future outputs
    - > Examples:
      - predicting which customers won't default on a loan
      - predicting which footy team will win this weekend

Discovery Driven – Unsupervised Learning

- **Return "interesting" patterns in the data**
- **Principal Techniques: Clustering and Association Analysis**
- **Lack of supervision:**
  - given a set of observations (***training data***), infer classes or clusters in the data
  - training data is unlabelled – there are no pre-defined classes

Clustering vs Association Analysis

- **Clustering –**
  - Groups records of similar items
    - → The user must attach meaning to the clusters formed
  - Example application
    - > Identify different types of customers
- **Association analysis –**
  - Discovers relations hidden in the data
    - > represented in the form of ***association rules*** or ***sets of frequent items***
  - Example application
    - > Market basket analysis, e.g., *diapers* → *beer*

MONASH University
Information Technology

FIT5047 –Intelligent Systems

Supervised Machine Learning – Classification

Classification Process

1. **Model construction**
   – builds a model using a ***Training Set*** **–** a set of tuples, each comprising a list of features and a class
2. **Model evaluation – estimates the *performance* of the model**
   – uses a ***Test Set***, which is different from the training set
   – the known label of the items in the test set are compared with the class selected by the model
   – ***Performance*** can be measured by accuracy, recall, precision
3. **Model usage – classifies or predicts the class of unseen items**

Model Construction

Training Data

Classification Algorithm

NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Classifier (Model)

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Model Evaluation and Usage

Evaluation

Usage

Classifier

Test Data

Unseen Data

(Jeff, Professor, 4)

Tenured?

NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Features of Classification Methods (I)

- **Hypothesis features**
  - **Performance**
    > Different measures, e.g., accuracy, recall, precision
  - **Significance**
    > Must perform better than the naïve hypothesis
  - **Model size** (related to information content)
    > E.g., decision tree size, compactness of classification rules
  - **Transparency** (interpretability)
    > Ease of understanding the model

Features of Classification Methods (II)

- **Practicalities of the model**
  - **Robustness**
    - > Ability to handle noise and missing values
  - **Scalability**
    - > Speed – time to build the model, and time to use the model
    - > Size – effect of the size of the dataset on the model

Evaluating Classifiers

- **Performance**
  - depends on the representativeness of the training data
  - determined using a ***test set***
- **Need to perform *cross-validation*, Why?**
  - ***X-validation*** – repeated experiments on different test sets
    > separate the dataset into training and test sets
    > build the model from the training set, and compute performance on the test set
    > usually 10-fold or 5-fold X-validation
    > common set ups: 90/10%; 80/20%; leave-one-out
  - ***Stratified X-validation*** – the test sets are proportional to the classes in the data
    > e.g., 20% +ive, 80% -ive

MONASH University
Information Technology

FIT5047 –Intelligent Systems

Decision Trees

Decision Trees

- **Classify objects based on the values of their *explanatory attributes***
  - the target classes (***dependent attributes***) are pre-defined
- **Classification is based on a tree structure**
  - each non-leaf node is a ***decision node***
  - each leaf node represents a ***class***
- **To classify an object**
  - each decision node (starting from the root) compares an attribute of the object with a specific ***attribute value*** (or range)
  - a path from the root to a leaf node gives the class of the object

Decision Tree Example: Attributes

Day	Outlook	Temperature	Humidity	Wind	Play ball
D1	Sunny	Hot	High	Weak	No

$\underbrace{\text{Outlook, Temperature, Humidity, Wind}}$

Input x (vector of values for ***explanatory attributes***)

$\underbrace{\text{Play ball}}$

Output y (***target / dependent attribute***)

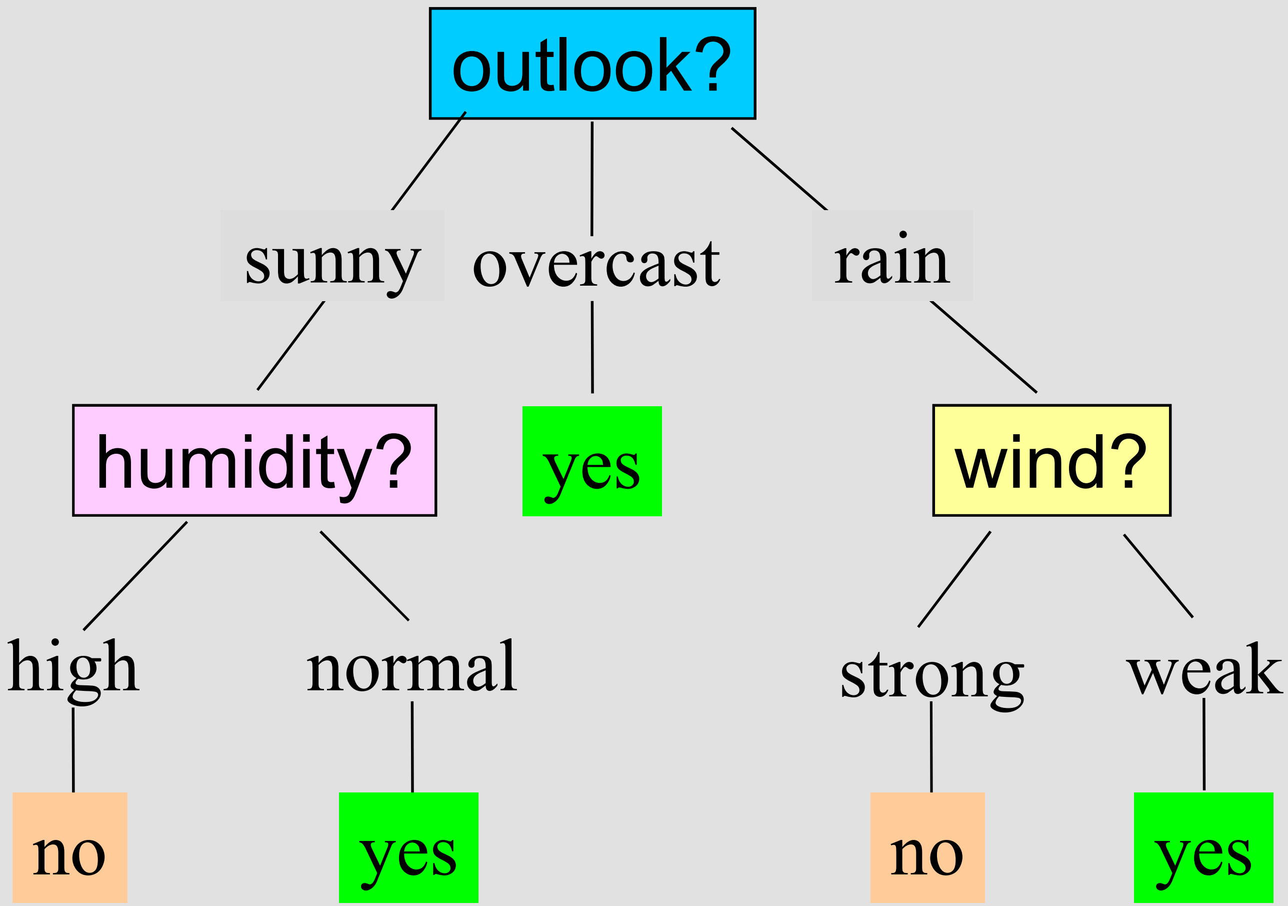
Classification by DT Induction

- **Training data – records of items that have *explanatory attributes* and a *dependent attribute***
  - **Input x:** Represented by a vector of attribute values
  - **Output y:** The corresponding target value
- **Tree construction algorithm – builds a decision tree**
  - Top-down, recursive divide-and-conquer approach
  - Based on a greedy algorithm

DT Example – Training Dataset

Day	Outlook	Temperature	Humidity	Wind	Play ball
D1	Sunny	Hot	High	Weak	**No**
D2	Sunny	Hot	High	Strong	**No**
D3	Overcast	Hot	High	Weak	**Yes**
D4	Rain	Mild	High	Weak	**Yes**
D5	Rain	Cool	Normal	Weak	**Yes**
D6	Rain	Cool	Normal	Strong	**No**
D7	Overcast	Cool	Normal	Strong	**Yes**
D8	Sunny	Mild	High	Weak	**No**
D9	Sunny	Cool	Normal	Weak	**Yes**
D10	Rain	Mild	Normal	Weak	**Yes**
D11	Sunny	Mild	Normal	Strong	**Yes**
D12	Overcast	Mild	High	Strong	**Yes**
D13	Overcast	Hot	Normal	Weak	**Yes**
D14	Rain	Mild	High	Strong	**No**

DT Example – Learned Model



Decision Tree Learning Algorithm (I)

1. **Start with all training examples at the root**
2. **Partition examples recursively based on selected attributes**
   - attributes are categorical
     > if continuous-valued, they are broken up into ranges
   - attributes are selected using heuristics or a statistical measure
     > e.g., ***information gain***
3. **Stop partitioning when there is no further gain in partitioning**

Employ majority voting for classifying the leafs

Choosing an Attribute for Splitting

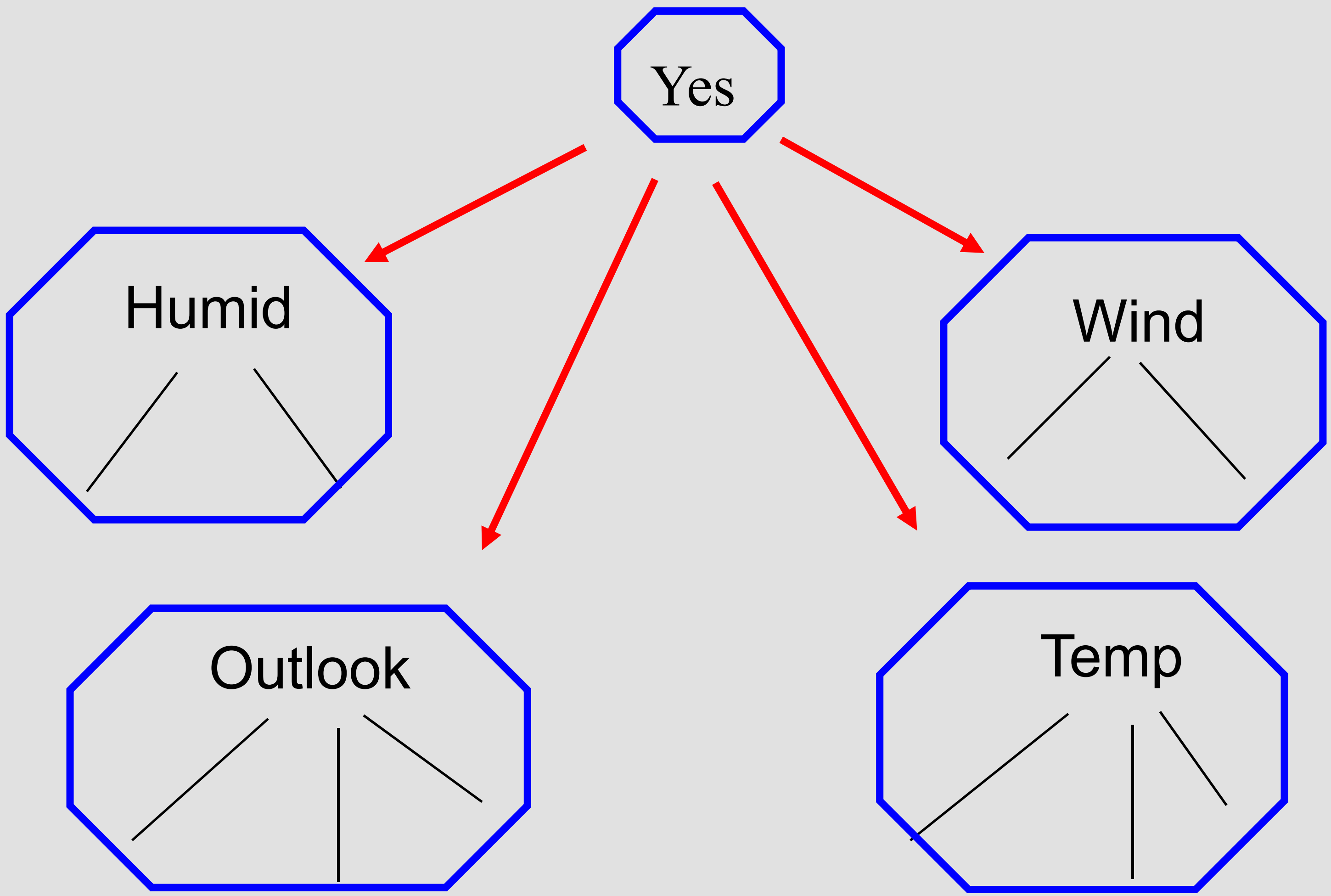
- **Approach – choose the attribute that best separates training examples into targeted classes**
- **Examples of proposed techniques**
  - ***Information Gain*** [Quinlan, 1975]
  - ***Message Length*** [Wallace and Patrick, 1993]

What is the “simplest” Tree? – Example

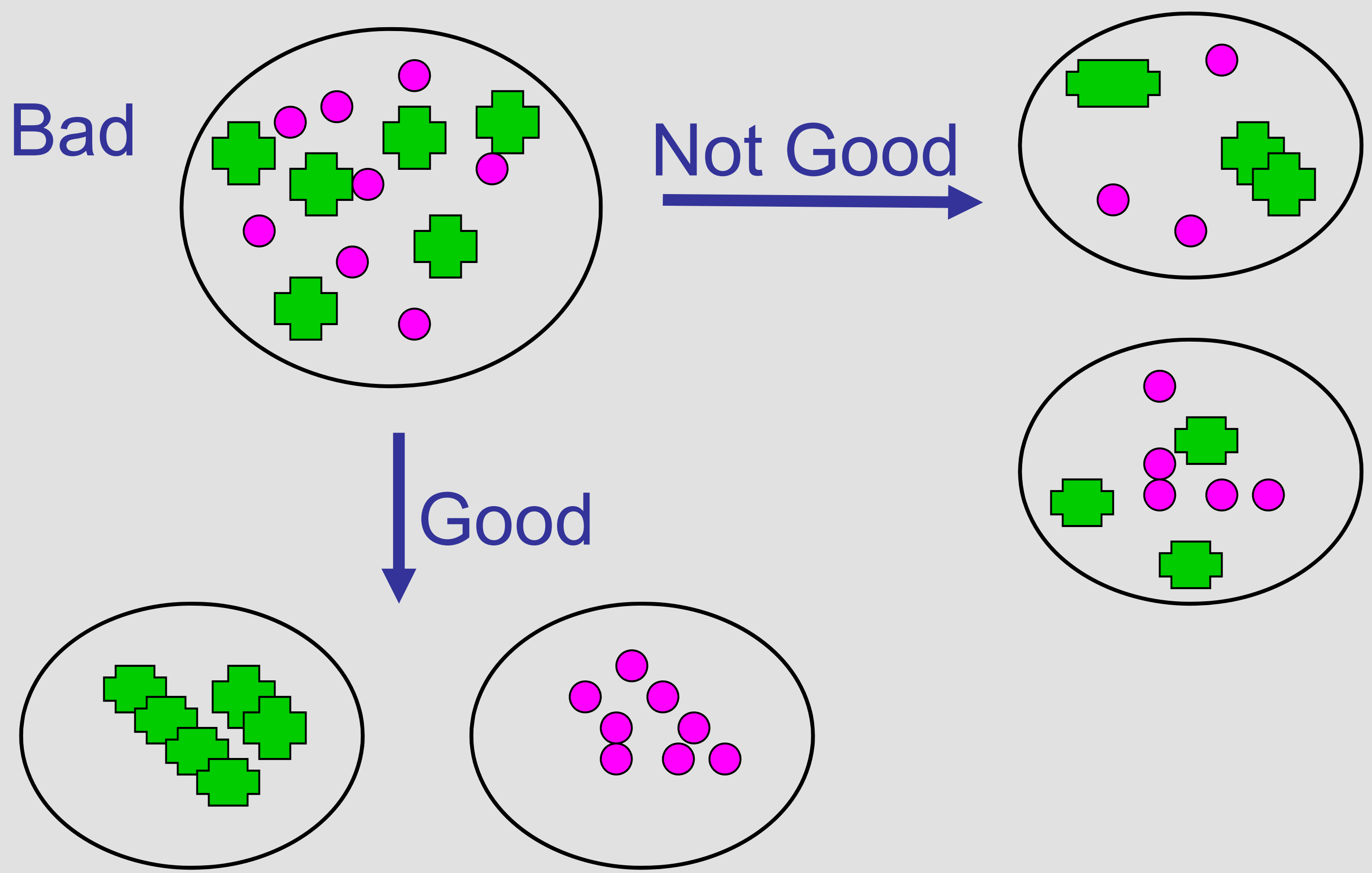
- **Always predict “yes”**
  - A tree with one node
- **How good it is?**
  - Correct on 9 examples
  - Incorrect on 5 examples
  - Notation: [9+,5-]

Day	Outlook	Temperature	Humidity	Wind	Play ball
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

On which Attribute do we Split?



Disorder is Bad, Homogeneity is Good



Using Information Theory to Quantify Uncertainty – Entropy

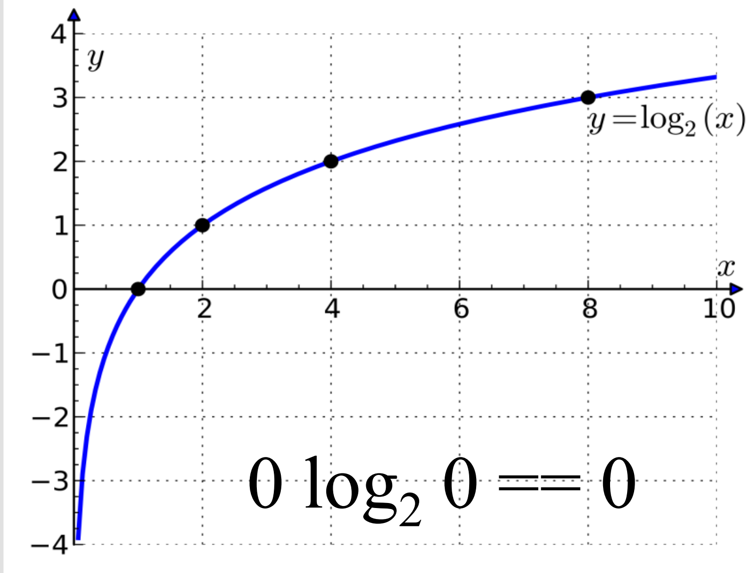
- ***Entropy*** **measures the amount of uncertainty in a probability distribution**
- **Given a discrete random variable on a finite set $X=\{ x_1, \ldots, x_n \}$, with probability distribution function $\Pr(x)=\Pr(X=x)$, the *entropy* $H(X)$ of X is defined as**

$$H(X) = -\sum_{i=1}^{n} \Pr(x_i) \log_2 \Pr(x_i)$$

where $0 \leq H(X)$

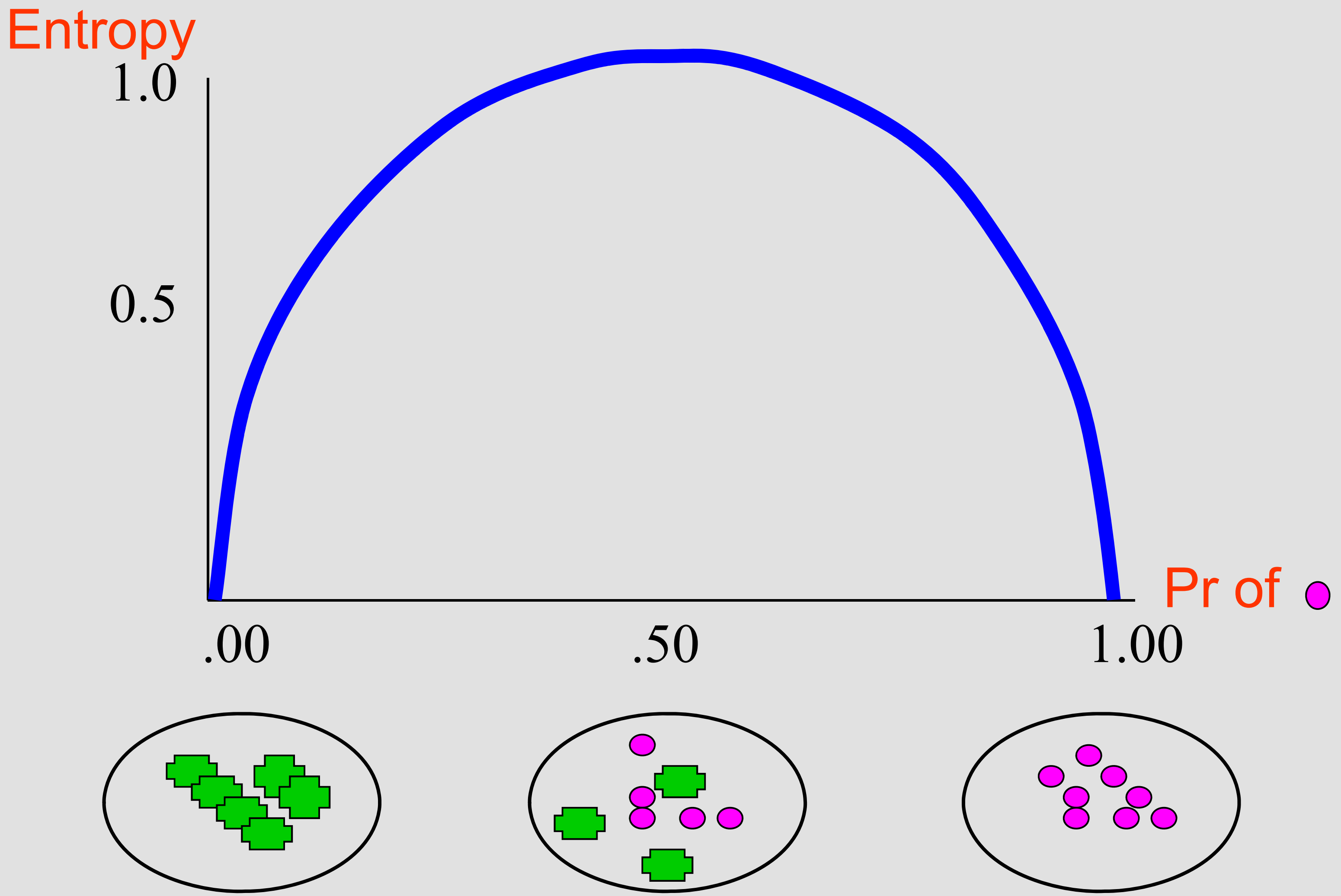
Entropy – Example

$$H(X) = -\sum_{i=1}^{n} \Pr(x_i) \log_2 \Pr(x_i)$$



- **Suppose there is a random variable *S* that has value *a* or *b***
  – Let Pr(a) = 1 and Pr(b) = 0
  – Let Pr(a) = 0.9 and Pr(b) = 0.1
  – Let Pr(a) = 0.5 and Pr(b) = 0.5
  – Which probability assignment <u>maximizes</u> ***H(S)*** ?

Entropy



Entropy is Bad, Homogeneity is Good

- **Let S be a set of examples**
  - Labeled positive or negative
- **Entropy(S) = -P $\log_2$(P) - N $\log_2$(N)**
  - **P** is the proportion of positive examples
  - **N** is the proportion of negative examples
  - 0 log 0 == 0
- **Example: S has 9 pos and 5 neg**
  - Entropy([9+, 5-])

$$= -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$$

Information Gain (I)

- **Information gain is a measure of the expected reduction in entropy resulting from splitting along attribute** A

$$IG(S, A) = H(S) - \underbrace{\sum_{v \in Values(A)} \frac{S_v}{S} H(S_v)}_{\text{Expected value of H from splitting on A}}$$

where

– v is a value for A (we sum over all values)

– S_v is a subset of S for which attribute A has value v

Using Information Gain to Learn a DT

- **The same attributes must describe each sample**
- **Attributes are assumed to be categorical (for now)**
- **Select the attribute with the highest Information Gain → largest reduction in entropy**

Information Gain of Splitting on Wind

Values(wind)=weak, strong

S = [9+, 5-]

S_{weak} = [$\boxed{6+, 2-}$]

S_{strong} = [3+, 3-]

$$IG(S, \text{wind}) = H(S) - \sum_{v \in \{\text{weak,strong}\}} \frac{|s_v|}{|s|} H(s_v)$$

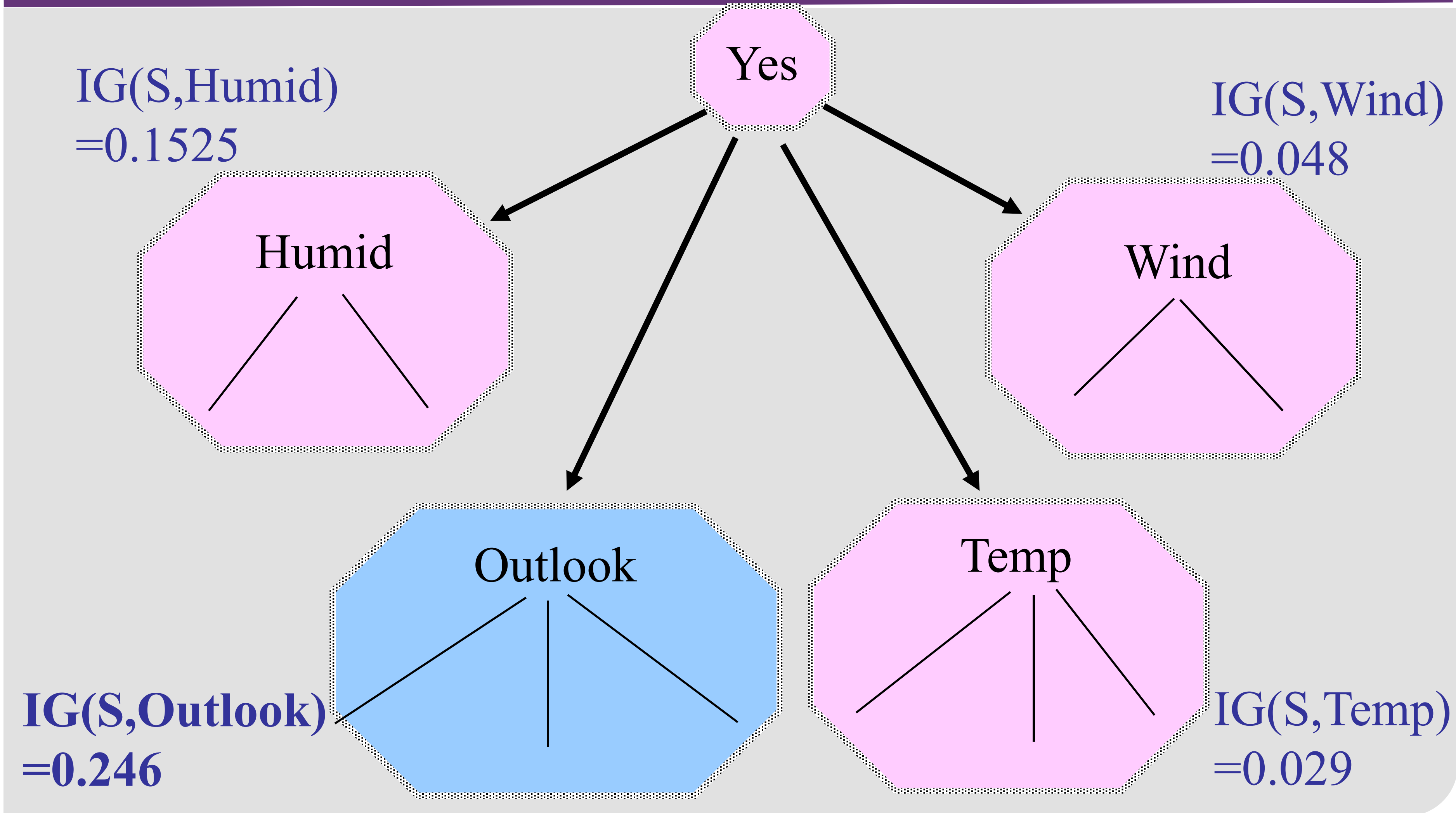
= H(S) - 8/14 H(S_{weak}) - 6/14 H(S_{strong})

= 0.94 - (8/14) 0.811 - (6/14) 1.00

= 0.048

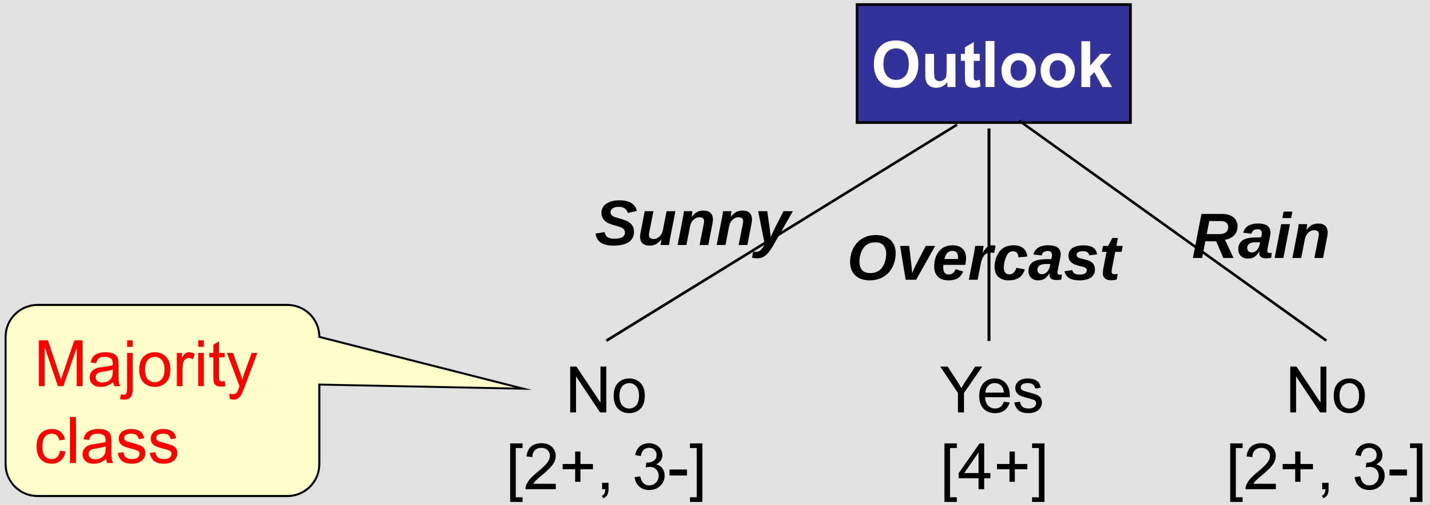
Day	Wind	Play ball?
d1	weak	no
d2	strong	no
d3	weak	yes
d4	weak	yes
d5	weak	yes
d6	strong	no
d7	strong	yes
d8	weak	no
d9	weak	yes
d10	weak	yes
d11	strong	yes
d12	strong	yes
d13	weak	yes
d14	strong	no

Evaluating Attributes

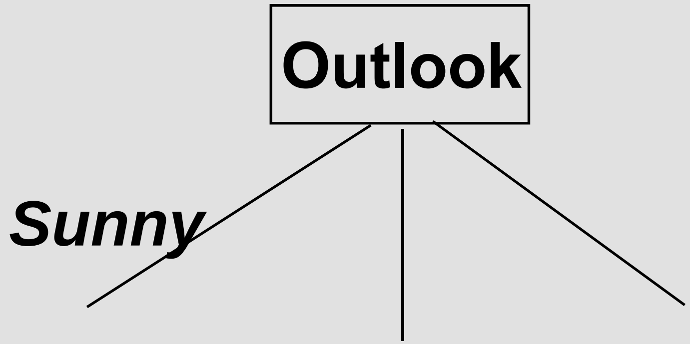


Building a Decision Tree (I)

Good day for playing ball?

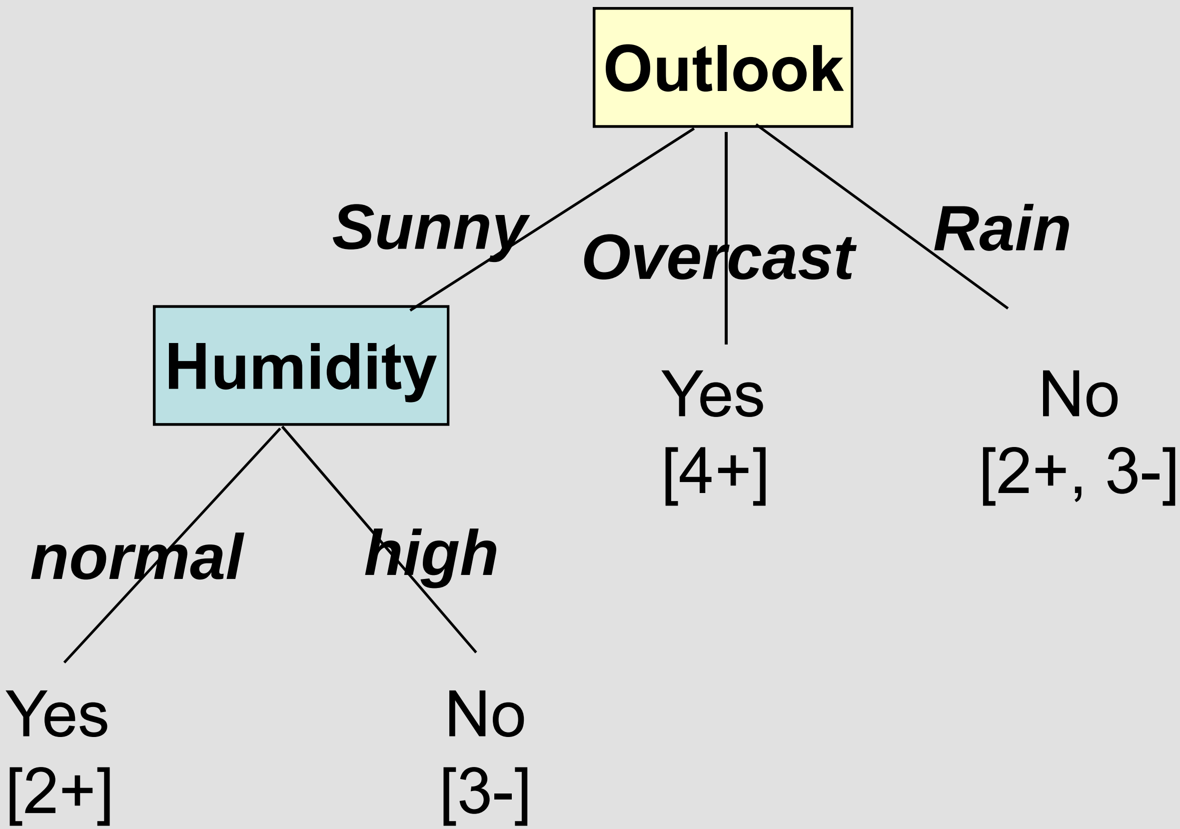


Building a Decision Tree (II)



Day	Temp	Humid	Wind	Play ball?
d1	H	H	weak	no
d2	H	H	strong	no
d8	M	H	weak	no
d9	C	N	weak	yes
d11	M	N	strong	yes

Building a Decision Tree (III)



Decision Tree Learning Algorithm (II)

BuildTree(TrainingData)
- Split(TrainingData)

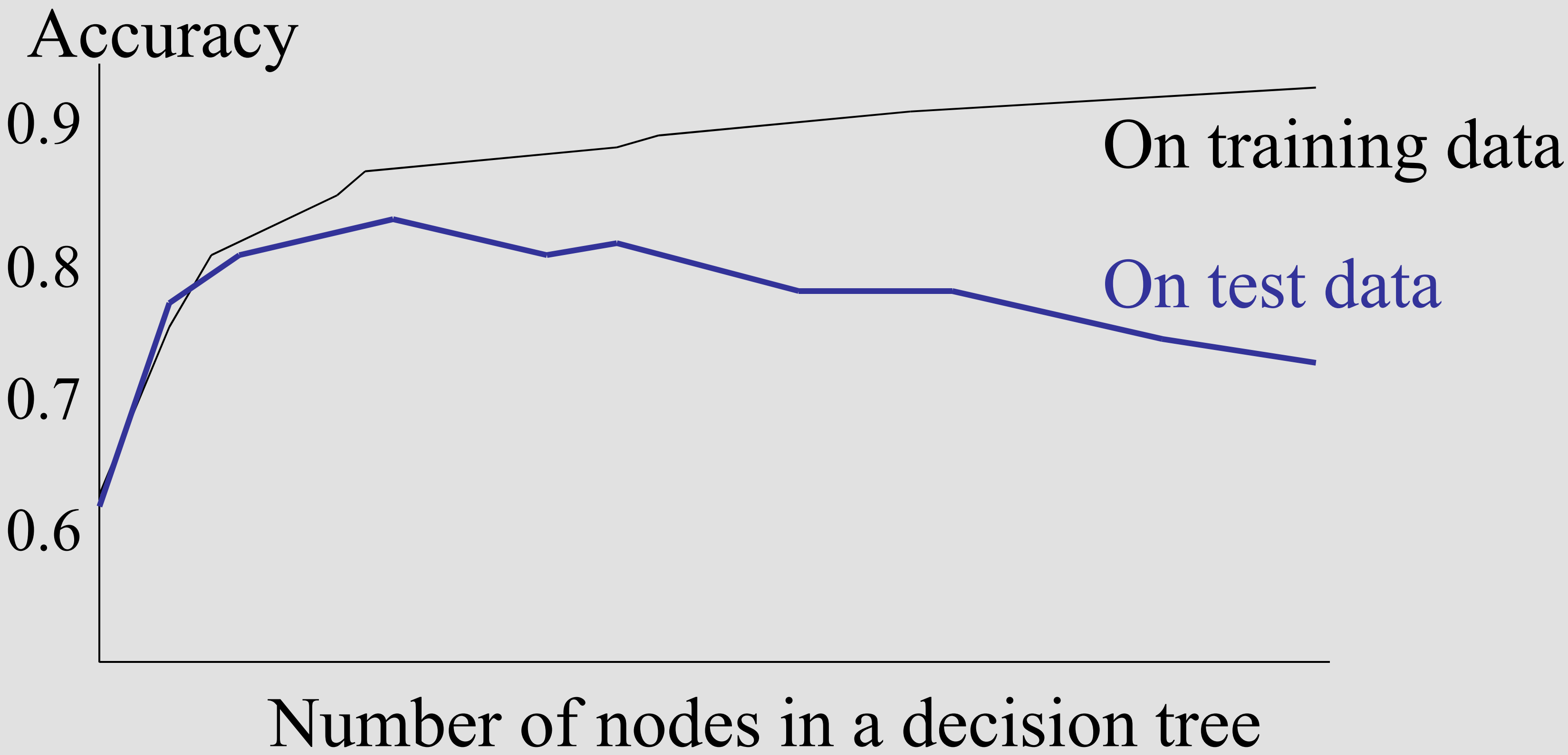
Split(D)
1. **If** all points in D are of the same class
   Then Return
2. **For each** attribute A
   a. Evaluate splits on attribute A
3. Use best split to partition D into D1, D2, …
   a. Split (D1)
   b. Split (D2)
   . . .

recursive call

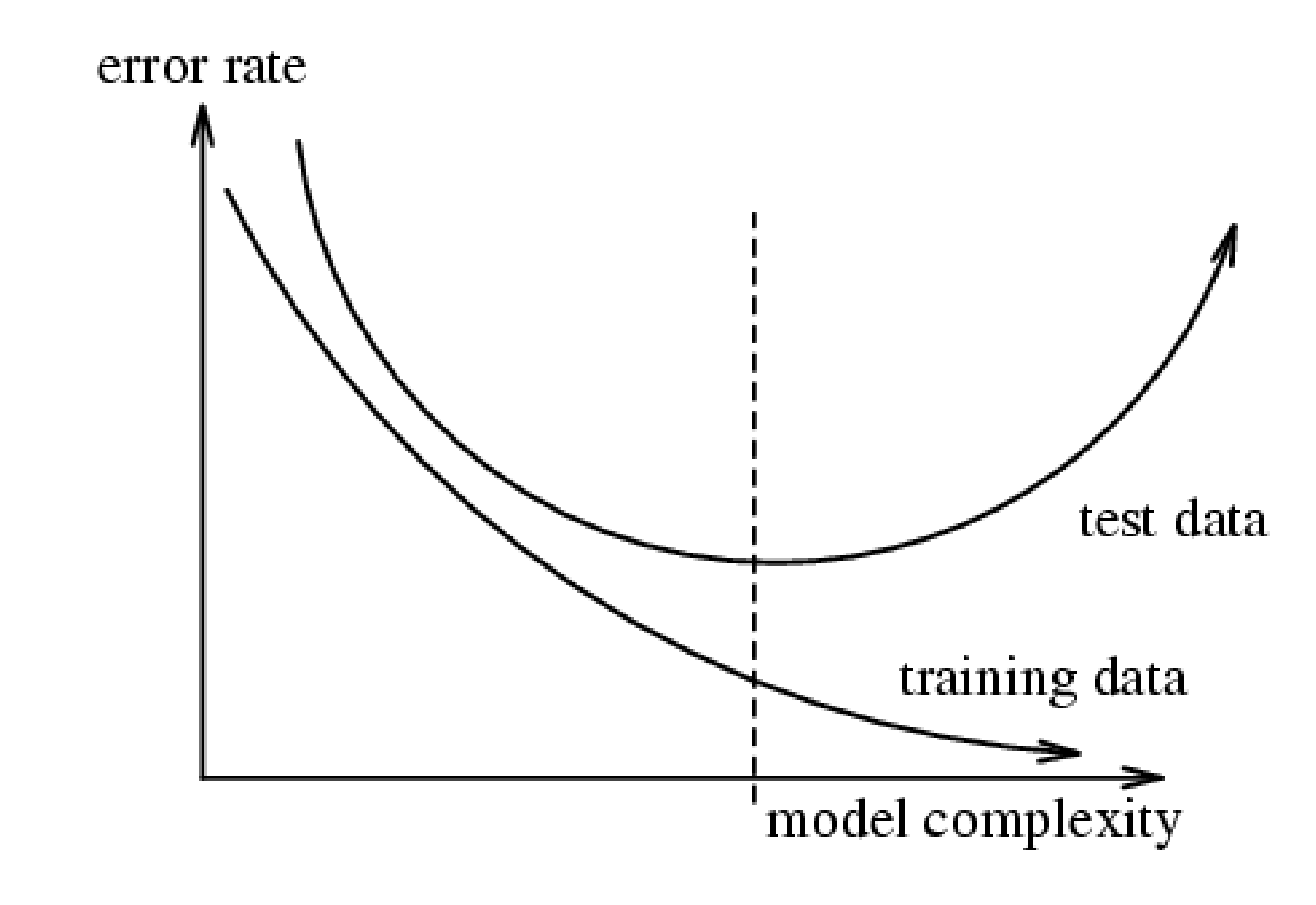
Overfitting (I)

The generated tree may *overfit* the training data – try to fit noise or outliers

→ reduced accuracy for unseen samples



Overfitting (II)



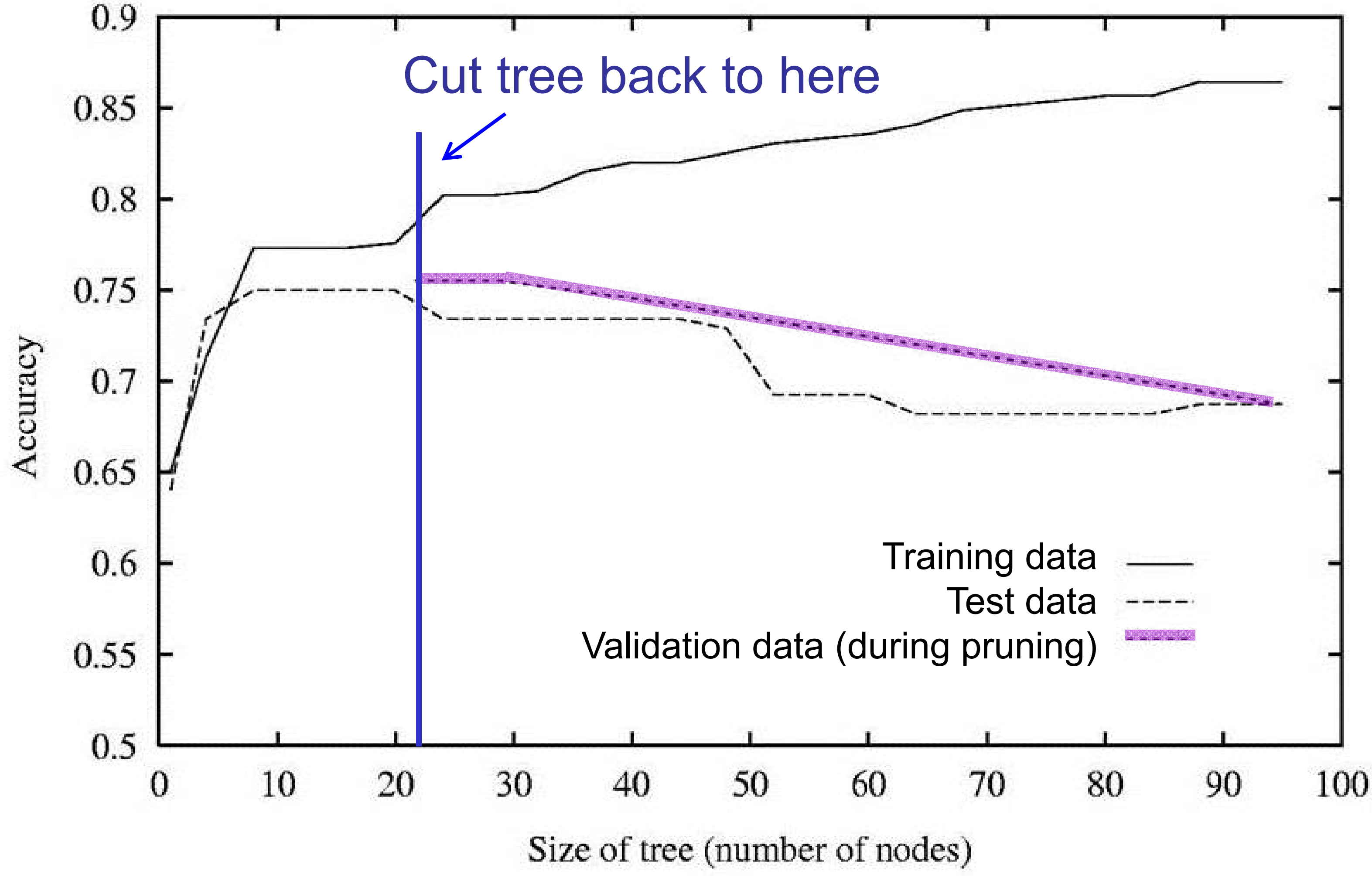
Overfitting (III)

- **DT is overfitting if there exists another DT' such that**
  - DT has a smaller error than DT' on training examples
  - DT has larger error than DT' on test examples
- **Causes of overfitting**
  - Noisy data
  - Training set is too small

Avoiding Overfitting

- **How to prevent overfitting:**
  - **Pre-pruning:** Stop growing the tree if the goodness measure falls below a threshold
  - **Post-pruning:** Grow the full tree, then prune
  - **Regularization:** Add complexity penalty to the performance measure
    - E.g., Complexity = Number of nodes in the tree
- **How to select the best tree?**
  - Measure performance on training data
  - Measure performance on a separate validation set

Effect of Pruning after Growing the DT



Enhancements to DT Induction

- **Handle continuous-valued attributes**
  - **Input:** Use threshold to split
  - **Output:** Estimate a linear function at each leaf (e.g., mean)
- **Handle missing attribute values**
  - **Categorical**
    - > use the most common value of the attribute
    - > probabilistic selection according to the distribution of values
  - **Continuous**
    - > use the mean of the attribute values
- **Reduce overfitting**
  - E.g., by post-pruning
- **Attribute construction**
  - create new attributes based on existing ones that are sparsely represented → reduces sparseness

done manually

Continuous-valued Attributes

Partition the continuous attribute value into intervals

1. **Fit a distribution to the values for attribute A**
   - commonly Normal
2. **Search for a point to split on**
   - perform binary search
   - can split on the same attribute again (lower in the tree)
3. **Calculate the Information Gain obtained from splitting attribute A at that point**

Continuous-valued Attributes – Example

- **Humidity has a Normal distribution with mean $\mu=81.643$ and standard deviation $\sigma=10.285$**
- **When we split on 75 (after outlook=sunny), we obtain**

normal humidity high humidity

$$IG(S,h\le 75)=H(S)\quad -\frac{2}{5}H(h\le 75)-\frac{3}{5}H(h>75)$$

Extracting Classification Rules from DTs

- **Represent the inferred structure in the form of IF-THEN rules**
- **One rule is created for each path from the root to a leaf**
- **Each attribute-value pair along a path forms a conjunction**
- **The leaf node holds the class prediction**

Extracting Rules from DTs – Example

- **IF *outlook = "sunny" and humidity="high"* THEN *play-ball = "NO"***
- **IF *outlook = "sunny" and humidity="normal"* THEN *play-ball = "YES"***
- **IF *outlook = "overcast"* THEN *play-ball = "YES"***
- **IF *outlook = "rain" and wind="strong"* THEN *play-ball = "NO"***
- **IF *outlook = "rain" and wind="weak"* THEN *play-ball = "YES"***

MONASH University
Information Technology

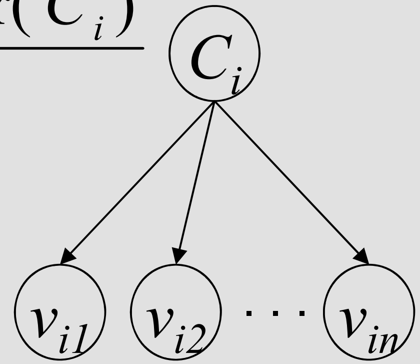
FIT5047 –Intelligent Systems

Naïve Bayes Classifier

Naïve Bayes Classifier (I)

- **Based on Bayes rule** $\Pr(C_i \mid V_i) = \dfrac{\Pr(V_i \mid C_i)\Pr(C_i)}{\Pr(V_i)}$

  where
  - C_i is the class of item i
  - $V_i=\{v_{i1},\ldots,v_{in}\}$ are the values of a set of attributes for item i
  - v_{ij} is the value of attribute j for item i



$$\Pr(C_i = c \mid v_{i1},\ldots,v_{in}) = \frac{\Pr(v_{i1},\ldots,v_{in} \mid C_i = c)\Pr(C_i = c)}{\Pr(v_{i1},\ldots,v_{in})}$$

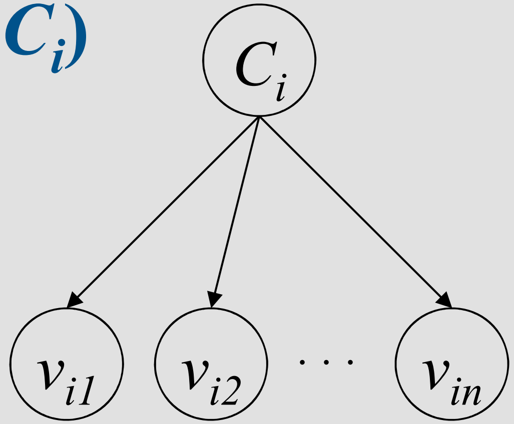
- **Assumes conditional independence of the attribute values for different classes**

$$\Pr(C_i = c \mid v_{i1},\ldots,v_{in}) = \alpha \prod_{k=1}^{n} \Pr(v_{ik} \mid C_i = c)\Pr(C_i = c)$$

  - where α is a normalizing constant

Naïve Bayes Classifier (II)

- **The parameters: $\Pr(C_i)$ and $\Pr(V_i \mid C_i)$**

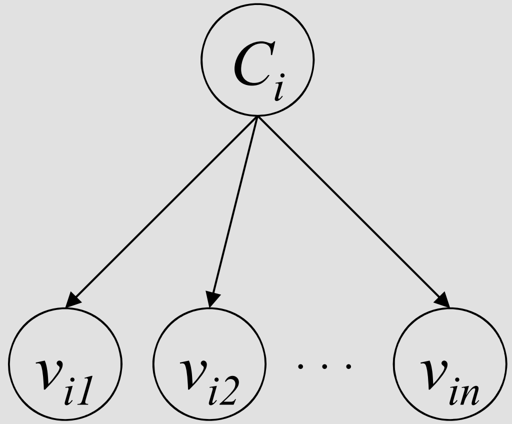


- **BIG questions:**
  - **Prediction/classification: Given that we know the parameters, how do we predict or classify an instance?**
  - **Learning: How do we learn the parameters?**

Naïve Bayes Classifier – Usage

- **The predicted class for a new $x_i=(v_{i1},\ldots,v_{in})$ is:**

$$c^* = \arg\max_c \Pr(c \mid v_{i1},\ldots,v_{in})$$
$$= \arg\max_c \prod_{k=1}^{n} \Pr(v_{ik} \mid c)\Pr(c)$$



Naïve Bayes Classifier – Example

- **4 attributes – outlook, temperature, humidity, wind**
- **2 target classes – YES/NO (play ball)**
- **Probability of a class:**

$c =$	YES	NO
$\Pr(C_i=c)$	9/14	5/14

obtained from the data

- **Calculating** $Pr(C_i=YES|v_{i1}=sunny,v_{i2}=hot,v_{i3}=high,v_{i4}=weak)$

$$\Pr(C_i = YES \mid v_{i1} = sunny, v_{i2} = hot, v_{i3} = high, v_{i4} = weak)$$
$$= \alpha \Pr(v_{i1} = sunny \mid C_i = YES) \times \Pr(v_{i2} = hot \mid C_i = YES) \times$$
$$\Pr(v_{i3} = high \mid C_i = YES) \times \Pr(v_{i4} = weak \mid C_i = YES) \times \Pr(C_i = YES)$$
$$= \alpha \frac{2}{9} \times \frac{2}{9} \times \frac{3}{9} \times \frac{6}{9} \times \frac{9}{14} = \alpha \times 0.007$$

- **This calculation is repeated for all values of** c

Learning the Parameters

Estimating the CPTs $\Pr(c)$ and $\Pr(v \mid c)$

- ***Empirically:* use training data**
  - Estimate the probability of a class c
  - For each attribute value w_a, estimate the probability of this value from all instances in a class
- ***Elicitation:* ask a person**
  - Usually need domain experts, and sophisticated ways of eliciting probabilities
  - Trouble calibrating

Estimating Parameters Empirically

- **Maximum Likelihood Estimator (MLE):**

count

$$\Pr_{ML}(var = w_a) = \frac{count(var = w_a)}{\#\,of\;\;samples} = \frac{|\,var = w_a\,|}{\sum_{i=1}^{m} |\,var = w_i\,|}$$

  – where m is the number of values for var
- **Example:** T H T T

  Assume the classes of interest are {H,T}
  – MLE over all samples
  $\Pr_{ML}(H) = {}^1/_4$, $\Pr_{ML}(T) = {}^3/_4$
  – MLE of an attribute over one class
  $\Pr_{ML}(green|T) = {}^2/_3$, $\Pr_{ML}(yellow|T) = {}^1/_3$, $\Pr_{ML}(red|T) = 0$

Maximum Likelihood Principle

- **Suppose you have data D, and a probabilistic model parameterized by Θ**
  - Need to learn parameter Θ from data D
- ***Likelihood*: The probability of data given the model**
- ***Maximum likelihood*: Choose Θ* which maximizes (the log of) the likelihood function:**

$$\Theta^* := \text{argmax}_{\Theta} \log \Pr_{\Theta}(D)$$

Likelihood

– Set the derivative to 0

$$\frac{\partial}{\partial\theta}[\log \Pr_\theta(D)] = 0 \Rightarrow \frac{count(var = w_a)}{\#\, of\, samples}$$

The Sparse Data Problem – Smoothing

- **Not all instances are found in the data set or in a particular class**
  - If value w_a is not found in class c, w_a =0 → MLE for Pr($w_a|c$)=0
  - If variable var is not found in the training set, MLE for Pr(w_a) is undefined (denominator is zero)
- ***Expected Likelihood Estimator (ELE)***

$$\Pr_{EL}(var = w_a) \cong \frac{|w_a| + \varepsilon}{\sum_{i=1}^{m} \{|w_i| + \varepsilon\}} = \frac{|w_a| + \varepsilon}{\sum_{i=1}^{m} |w_i| + m\varepsilon}$$

  - where m is the number of values for var
  - If a variable is not found in the dataset, ELE is $1/m$
  - ELE is conservative
- **Use Smoothing when estimating the parameters of Naïve Bayes, i.e., $\Pr(C)$ and $\Pr(v|C)$**

Expected Likelihood Estimation – Example

T H T T

$$\Pr_{ML}(c = H) \cong \frac{|H|}{\sum_{i=1}^{2} |C_i|} = \frac{1}{\{1+3\}} = \frac{1}{4}$$

$$\Pr_{ML}(C) = \begin{array}{c} \text{H T} \\ \left\langle \frac{1}{4}, \frac{3}{4} \right\rangle \end{array}$$

$$\Pr_{EL}(c = H) \cong \frac{|H| + \varepsilon}{\sum_{i=1}^{2} |C_i| + 2\varepsilon} = \frac{1+\varepsilon}{\{1+3\} + 2\varepsilon} = \frac{1+\varepsilon}{4+2\varepsilon}$$

for ε=1

$$\Pr_{EL}(C) = \begin{array}{c} \text{H T} \\ \left\langle \frac{1}{3}, \frac{2}{3} \right\rangle \end{array}$$

Enhancements to Naïve Bayes

- **Handle continuous-valued attributes**
- **Handle missing attribute values**
  - **Categorical**
    - > ignore missing values
    - > consider a missing value (?) to be an additional value
  - **Continuous**
    - > use the mean of the attribute values

Continuous-valued Attributes

1. **Fit a distribution to the values for attribute A**
   - commonly a Normal distribution
2. **Calculate the probability of the value x of attribute A for item i given the value of class C_i**
   - use a Normal density function

$$\Pr(v_{iA} = t \mid C_i = c) \approx \Pr(t - \frac{\varepsilon}{2} \le v_{iA} \le t + \frac{\varepsilon}{2} \mid C_i = c)$$

$$\approx \varepsilon \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t-\mu)^2}{2\sigma^2}}$$

Continuous-valued Attributes – Example

- **For class C_i=*YES*, v_{i2} (temperature) has a Normal distribution with mean $\mu=73$ and standard deviation $\sigma=5.23$**
- **The probability of temperature 85 F given class *YES* is**

$$\Pr(v_{i2} = 85 \mid C_i = YES) = \varepsilon \frac{1}{\sqrt{2\pi} \times 5.23} e^{-\frac{(85-73)^2}{2\times 5.23^2}} = 0.0055\varepsilon$$

Decision Trees versus Naïve Bayes

	Decision Trees	Naïve Bayes
Accuracy	depends on the features of the data	
Robustness	good	good
Generality	YES	assumes independent attributes
Model construction speed (scalability)	slower	faster
Model size	bigger	smaller
Interpretability	higher	lower

MONASH University
Information Technology

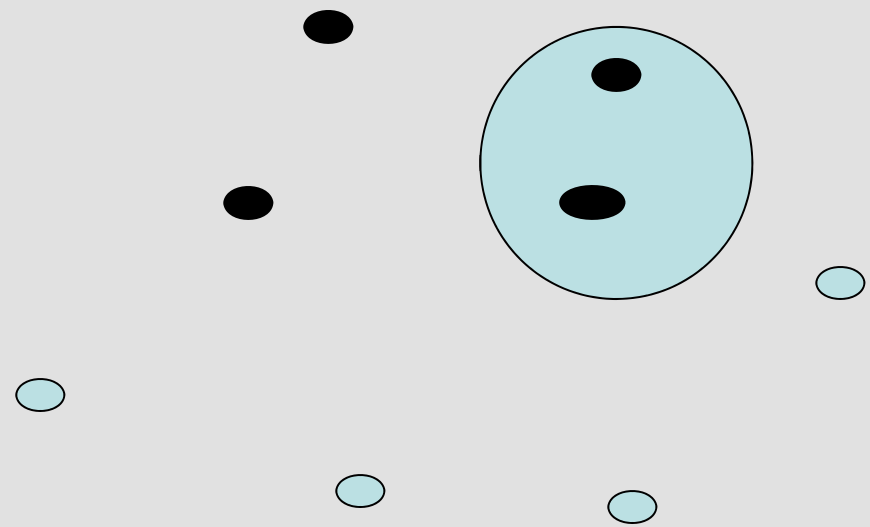
FIT5047 –Intelligent Systems

K Nearest Neighbour (k-NN)

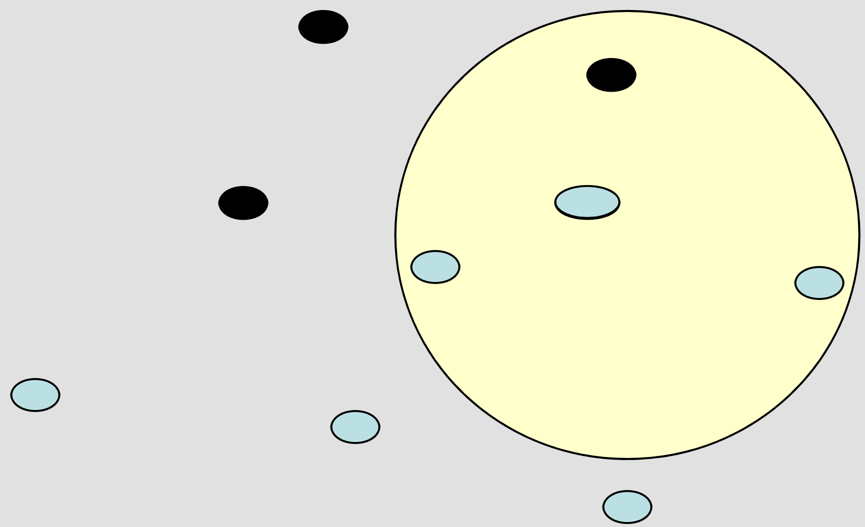
k-Nearest Neighbour (I)

- **All instances correspond to points in an n-dimensional space**
- **Classification is performed**
  - when a new instance arrives
  - by comparing features of the new instance with features of *k* training instances that are closest to it in the space (nearest neighbours)
- **The target function may be discrete or continuous**
  - **Discrete** – majority vote of the new instance's neighbours
  - **Continuous** – mean value of the k nearest training examples

1-Nearest Neighbour



3-Nearest Neighbour



k-Nearest Neighbour (II)

- **An instance x_i is represented by ($f_{i1}, f_{i2}, \ldots, f_{in}$)**
  - $f_{i,k}$ is the value of the kth feature for x_i
- **Distance measures between two instances x_i and x_j**
  - **Continuous features:** Euclidean distance

$$Ed(x_i, x_j) = \sqrt{\sum_{k=1}^{n} (f_{ik} - f_{jk})^2}$$

  - **Categorical features:** Jaccard coefficient

$$Jc(x_i, x_j) = \frac{|\{f_{i1}, \ldots, f_{in}\} \cap \{f_{j1}, \ldots, f_{jn}\}|}{|\{f_{i1}, \ldots, f_{in}\} \cup \{f_{j1}, \ldots, f_{jn}\}|}$$

Must be applied to all features

Computing Similarity – Example

- **Continuous features:**

  x_i={0.7, 30, 80, 10} and x_j={0.2, 32, 85, 40}

  $$Ed(x_i, x_j) = \sqrt{(0.7-0.2)^2 + (30-32)^2 + (80-85)^2 + (10-40)^2}$$

  Smaller is better!

- **Categorical features (Jaccard adaptation):**

  x_i={sunny, hot, high, weak} and x_j={rainy, hot, high, strong}

  $$Jc(x_i, x_j) = \frac{|\{hot, high\}|}{|\{sunny, rainy, hot, high, weak, strong\}|} = \frac{2}{6} = 0.33$$

  Larger is better!

- **Need to normalize features**

Parameter Learning

- **Given the training data and the distance function, there is no training**
  - The algorithm memorizes the training data
  - As data comes in, the model size grows
- **In some cases, the distance function is parameterized and its parameters are learnt**

Other Classification Methods

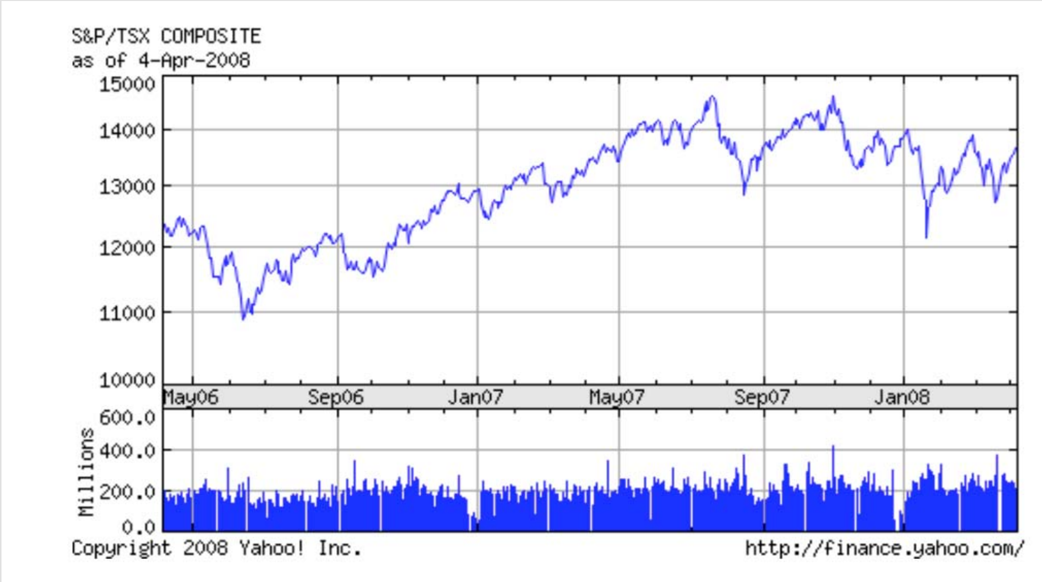
- **Perceptrons**
- **Support vector machines**
- **Logistic regression**

MONASH University
Information Technology

FIT5047 – Intelligent Systems

Regression

Regression



- **Given a training set $\{(\boldsymbol{x_1},t_1),\ldots,(\boldsymbol{x_m},t_m)\}$**
  - $\boldsymbol{x_i}$ and t_i are continuous → regression
  - assume $\boldsymbol{x_i} \in R^n, t_i \in R$
- **E.g., t_i is stock price, x_i contains company profit, debt, cash flow, …**

Error Function

- **Given a training set $\{(x_1,t_1),\ldots,(x_m,t_m)\}$**
  - assume $\boldsymbol{x_i} \in R^n, t_i \in R$

vector

offset

- **Linear regression model: $t = \boldsymbol{w} \cdot \boldsymbol{x} + w_0$**
  - we want to learn the parameters $\boldsymbol{w} \in R^n, w_0 \in R$
- **Error: Square of the difference between the true and predicted target value for x_i**

$$E(w) = \sum_{i=1}^{m} (\underbrace{t_i}_{\text{truth}} - \underbrace{(\boldsymbol{w} \cdot \boldsymbol{x_i} + w_0)}_{\text{prediction}})^2$$

Learning the Parameters

- **Find w^* that minimizes the error function**

$$\boldsymbol{w}^* = \operatorname{argmin}_w E(w)$$

  – where E(w) is a convex function, so w^* is unique

- **How to find w^*?**
  – Set the derivatives to zero: $\frac{\partial}{\partial w_k} E(w) = 0$
    > For a linear function $t = w_1 x + w_0$

$$w_0 = \frac{\sum t_i - w_1 \sum x_i}{N} \qquad w_1 = \frac{N \sum x_i t_i - \sum x_i \sum t_i}{N \sum x_i^2 - (\sum x_i)^2}$$

  – Use an iterative algorithm such as gradient descent

Linear Regression – Example (I)

$x_1,t_1 \quad x_2,t_2 \quad x_3,t_3$

- **Training data: {(1,3),(2.1,0.5),(-5,6.2)}**
- **Linear regression model:** $t = w_1 x + w_0$
- **Error function:**

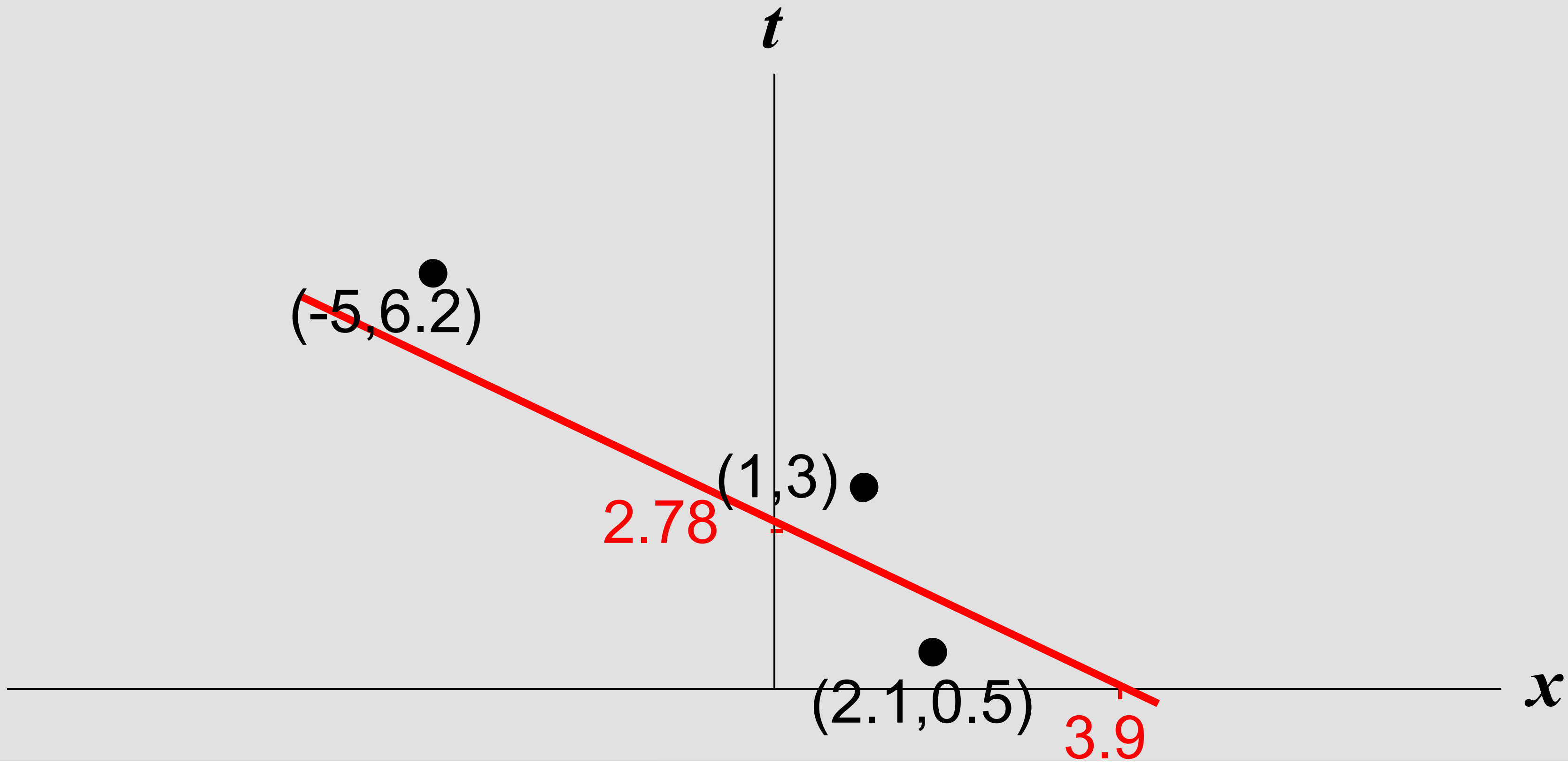
$$E(w) = (3 - 1w_1 - w_0)^2 + (0.5 - 2.1w_1 - w_0)^2 + (6.2 - (-5)w_1 - w_0)^2$$

$$w_0 = \frac{(3+0.5+6.2) - w_1(1+2.1-5)}{3} = \frac{9.7 + w_1 1.9}{3} = 2.78$$

$$w_1 = \frac{3(1\times 3 + 2.1\times 0.5 + (-5)\times 6.2) - (1+2.1-5)(3+0.5+6.2)}{3(1^2+2.1^2+(-5)^2) - (1+2.1-5)^2} = -0.712$$

Linear Regression – Example (II)

- **Training data: {(**1,3**),(**2.1,0.5**),(**-5,6.2**)}**
- **Function:** $t = -0.712x + 2.78$



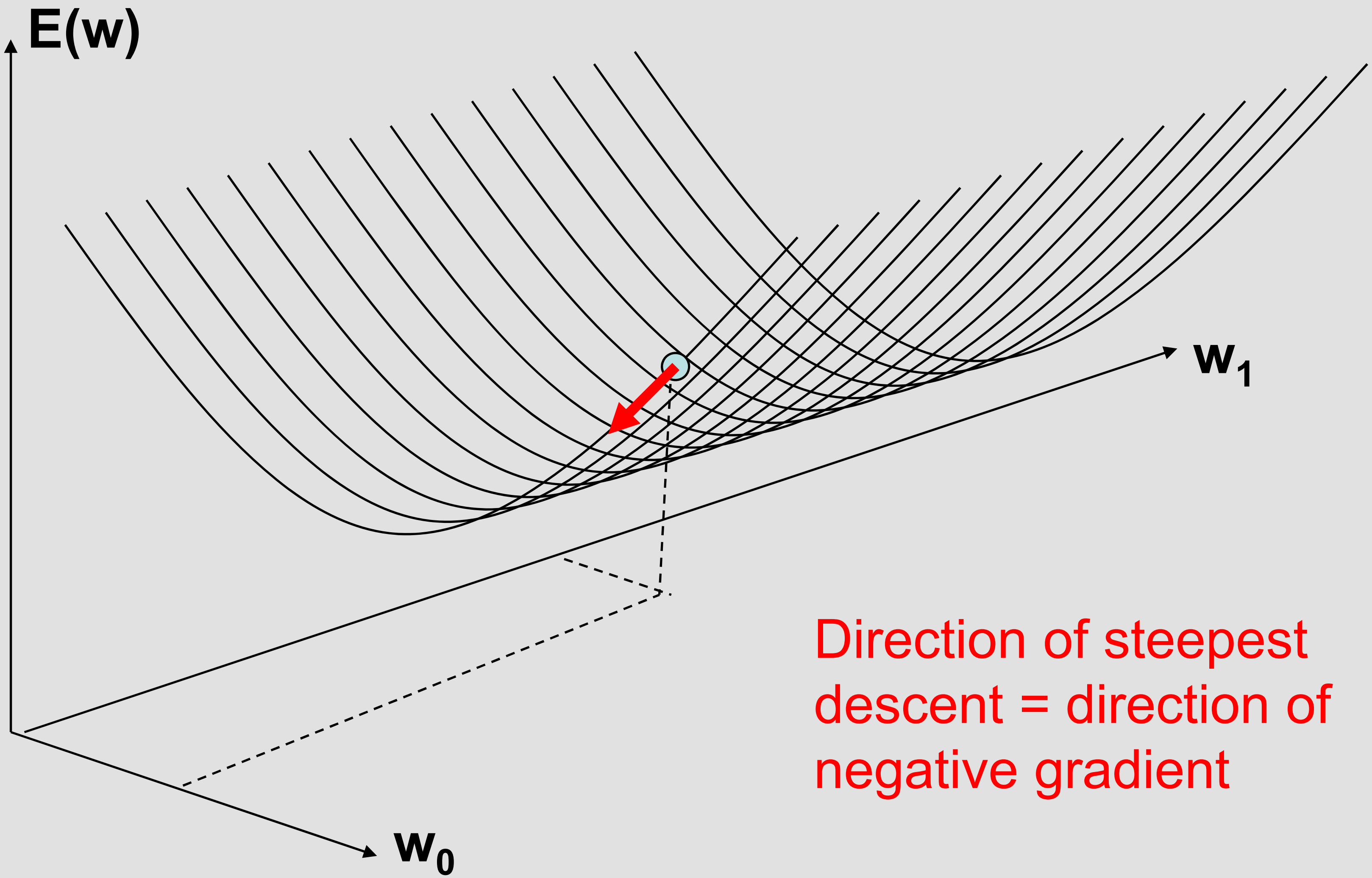
Gradient Descent Algorithm

1. **initialize w^0 arbitrarily**
2. **for t = 1,2,…**
   a. $\boldsymbol{w}^t \leftarrow \boldsymbol{w}^{t-1} - \alpha \nabla_{\mathbf{w}} E(\boldsymbol{w})$
   b. if $|\boldsymbol{w}^t - \boldsymbol{w}^{t-1}| < \varepsilon$ then break

Learning rate

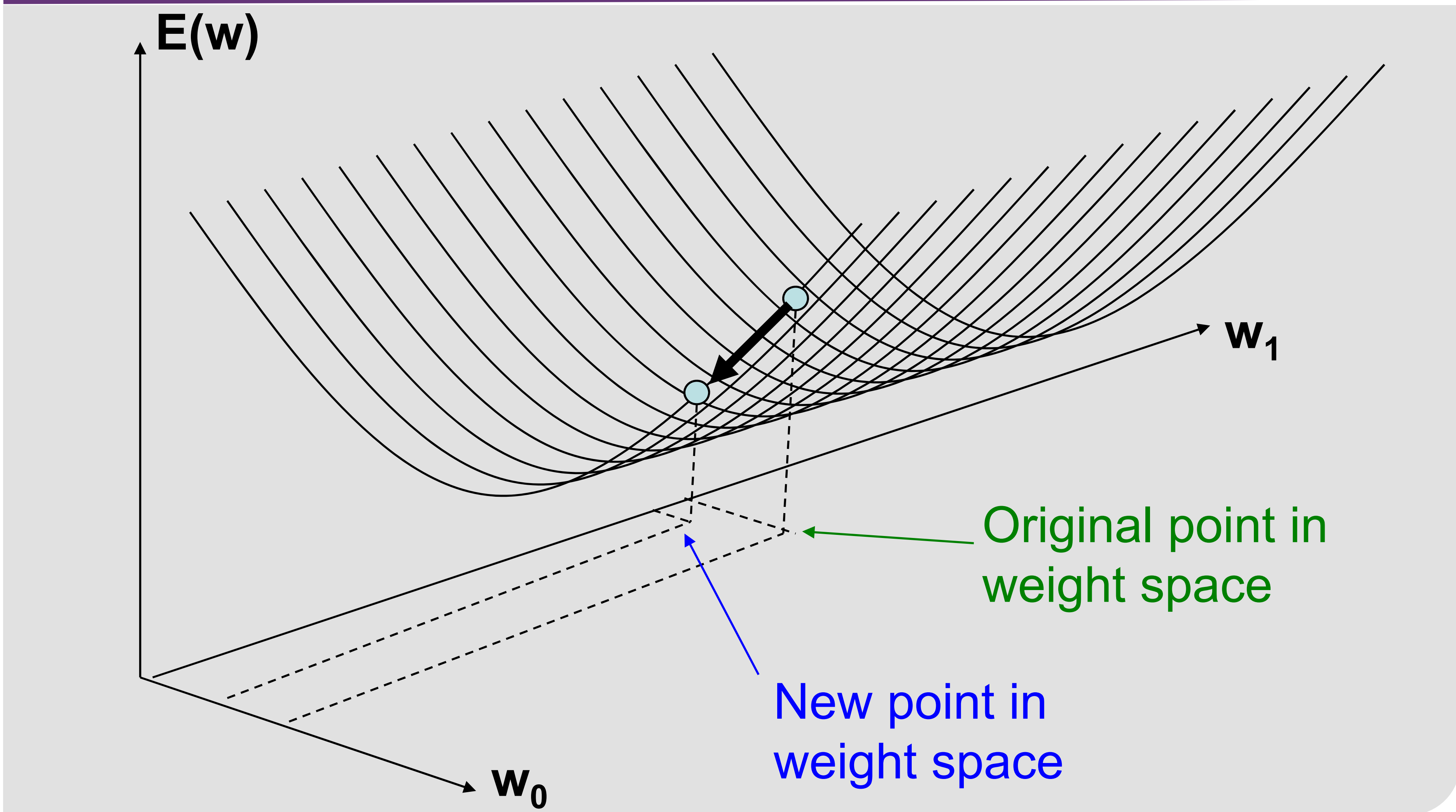
Gradient vector: stack up the partial derivatives $\frac{\partial E(w)}{\partial w_i}$ in a vector

Illustration of Gradient Descent (I)



Direction of steepest descent = direction of negative gradient

Illustration of Gradient Descent (II)



Linear Regression – Example (II)

- **Training data: {(1,3),(2.1,0.5),(-5,6.2)}**
- **Linear regression model:** $t = w_1 x + w_0$
- **Error function:**

$$E(w) = (3 - 1w_1 - w_0)^2 + (0.5 - 2.1w_1 - w_0)^2 + (6.2 - (-5)w_1 - w_0)^2$$

- **The gradient:**

$$\nabla_w E(w) = \begin{bmatrix} \frac{\partial E}{\partial w_0} \\ \frac{\partial E}{\partial w_1} \end{bmatrix} = -2 \times \begin{bmatrix} (3 - w_1 - w_0) + (0.5 - 2.1w_1 - w_0) + (6.2 + 5w_1 - w_0) \\ (3 - w_1 - w_0) + 2.1(0.5 - 2.1w_1 - w_0) - 5(6.2 + 5w_1 - w_0) \end{bmatrix}$$

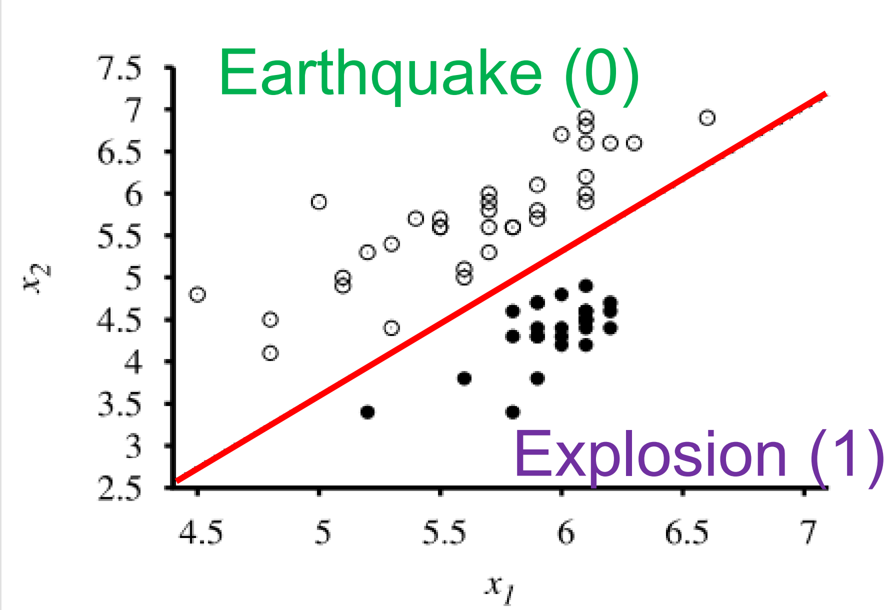
Classification with Linear Regression

- **The regressor is a *decision boundary* between two *separable* classes**
  – A linear decision boundary is called a ***linear separator,*** in this case the data are ***linearly separable***
- **Example:**
  – Linear separator

$$-4.9+1.7x_1-x_2$$

  – Classification

$$h(x_1,x_2)=\begin{cases}1 & \text{if } \boldsymbol{w}\cdot\boldsymbol{x}>0\\ 0 & \text{otherwise}\end{cases}$$



Summary – Supervised Machine Learning

- **Various techniques for supervised machine learning**
  - where we are given labeled training data
- **Discrete:**
  - Decision trees (non-parametric)
  - Naïve-Bayes (parametric)
  - K nearest neighbour (non-parametric)
- **Continuous:**
  - Regression
    - > Models based on least square error (parametric)

MONASH University
Information Technology

FIT5047 – Intelligent Systems

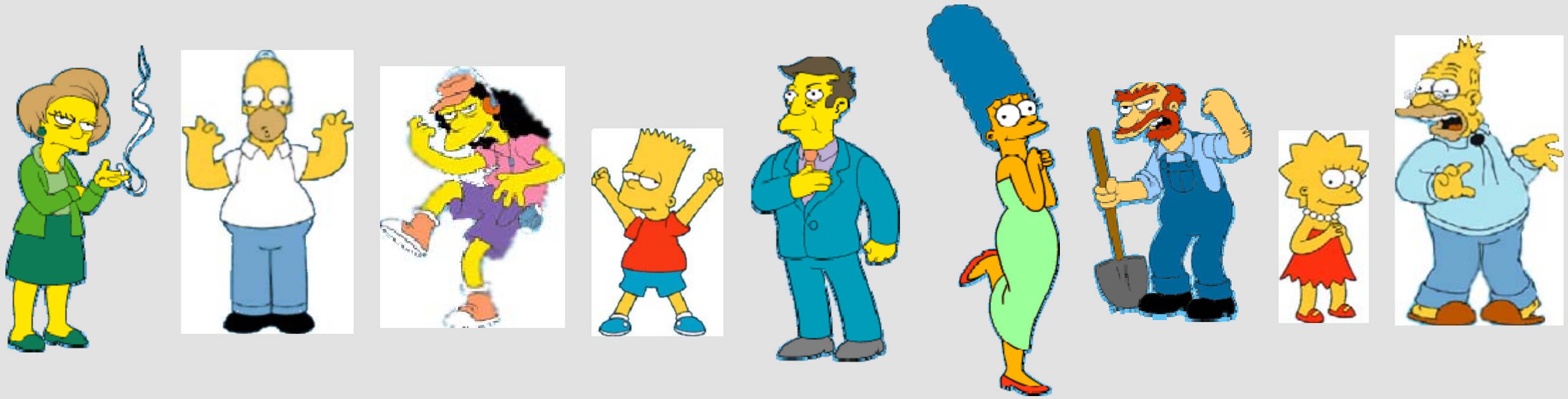
Unsupervised Machine Learning – Clustering

Slides adapted from Eamonn Keogh, Dan Klein and Kevin Korb

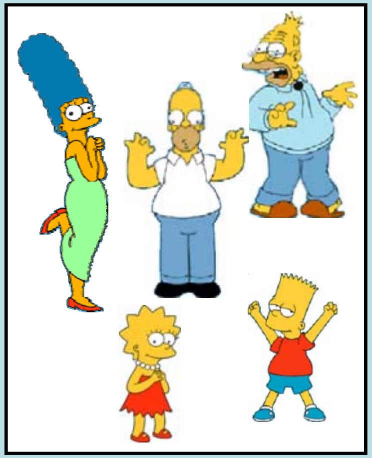
What is Clustering?

- **Organizing data into classes such that there is**
  - high intra-class similarity
  - low inter-class similarity
- **Finding the class labels and the number of classes directly from the data**

What is a Natural Grouping of these Objects?

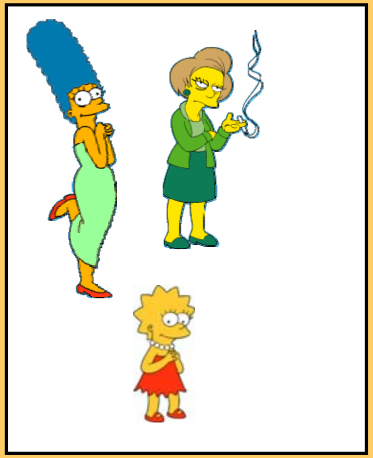


Clustering is subjective



Simpson's Family

School Employees



Females

Males

What is Similarity?

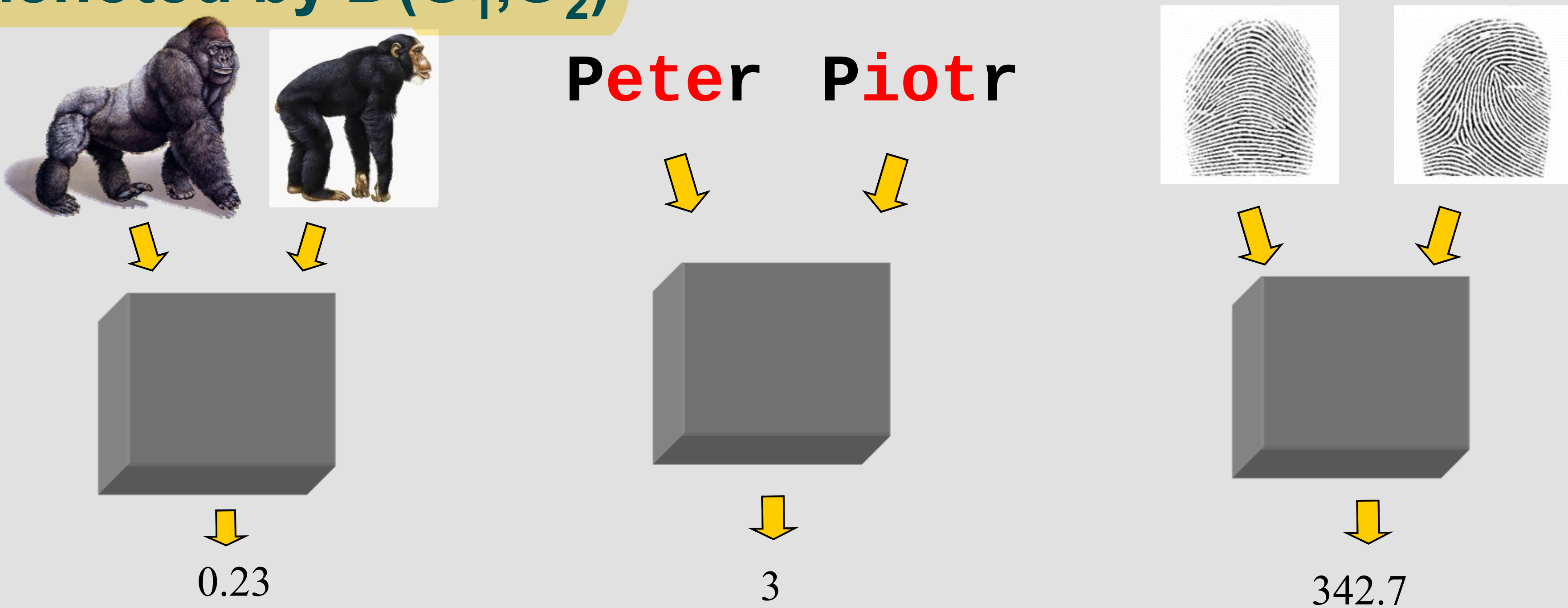
- ***The quality or state of being similar; likeness; resemblance; as, a similarity of features.***

  Webster's dictionary

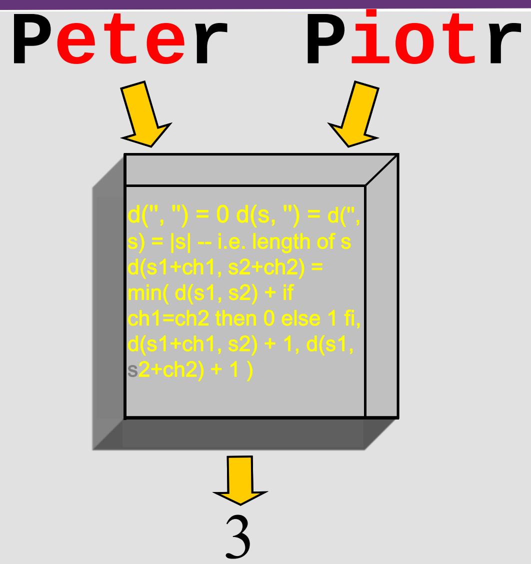
- **Similarity is hard to define, but…**
  "*We know it when we see it*"

Distance Measures (I)

Definition: Let O_1 and O_2 be two objects from the universe of possible objects. The distance (dissimilarity) between O_1 and O_2 is a real number denoted by $D(O_1,O_2)$



Distance Measures (II)



When we peek inside one of these black boxes, we see a function of two variables

What properties should a distance measure have?

- *D*(A,B) = *D*(B,A) — *Symmetry*
- *D*(A,A) = 0 — *Constancy of Self-Similarity*
- *D*(A,B) = 0 iff A= B — *Positivity (Separation)*
- *D*(A,B) ≤ *D*(A,C) + *D*(B,C) — *Triangle Inequality*

Measuring Similarity

Edit distance: Transform one of the objects into the other, and measure how much effort it took

Distance between Patty and Selma:

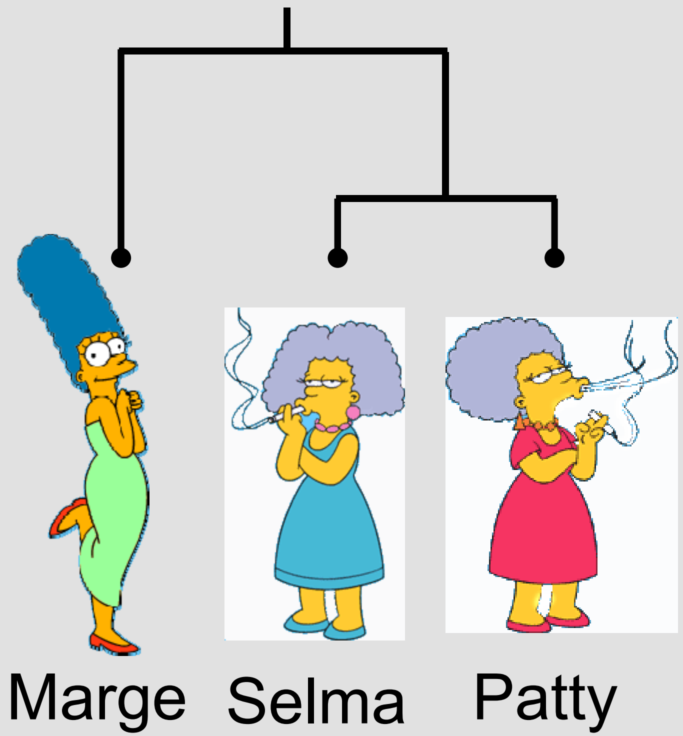
Change dress color	1 point
Change earring shape	1 point
Change hair part	1 point

D(Patty,Selma) = 3

Distance between Marge and Selma:

Change dress color	1 point
Add earrings	1 point
Decrease height	1 point
Take up smoking	1 point
Gain weight	1 point

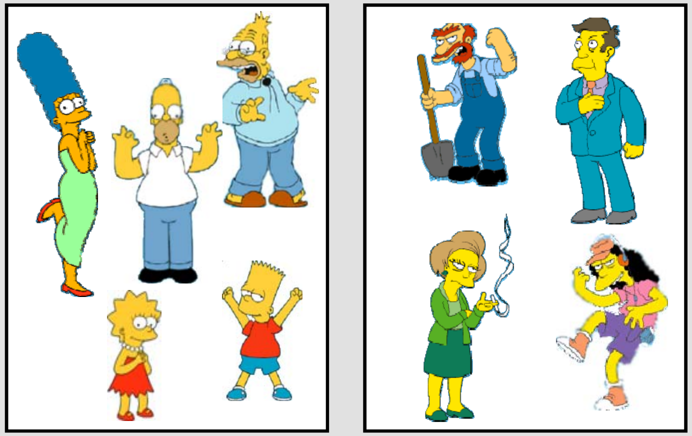
D(Marge,Selma) = 5



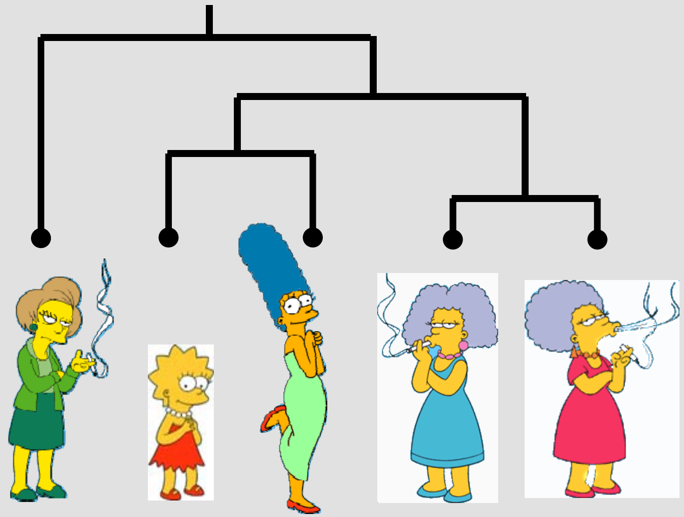
Two Types of Clustering

- **Partitional:** Construct various partitions and then evaluate them by some criterion
- **Hierarchical:** Create a hierarchical decomposition of the set of objects using some criterion

Partitional



Hierarchical



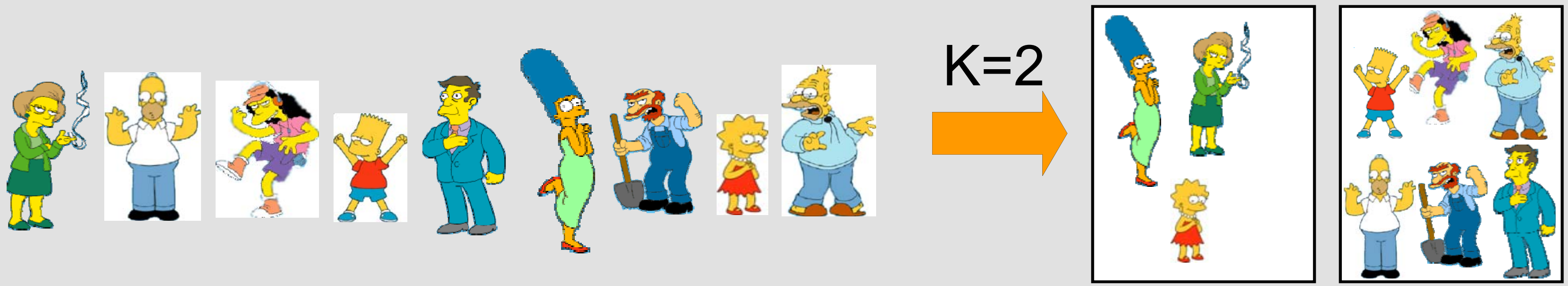
MONASH University
Information Technology

FIT 5047 – Intelligent Systems

K-means Algorithm

Partitional Clustering

- **Non-hierarchical, each instance is placed in exactly one of K non-overlapping clusters**
- **Produces only one set of clusters**
  ➔ the user normally has to input the desired number of clusters K



Squared Error

Squared error for cluster K_i

# of items in cluster K_i

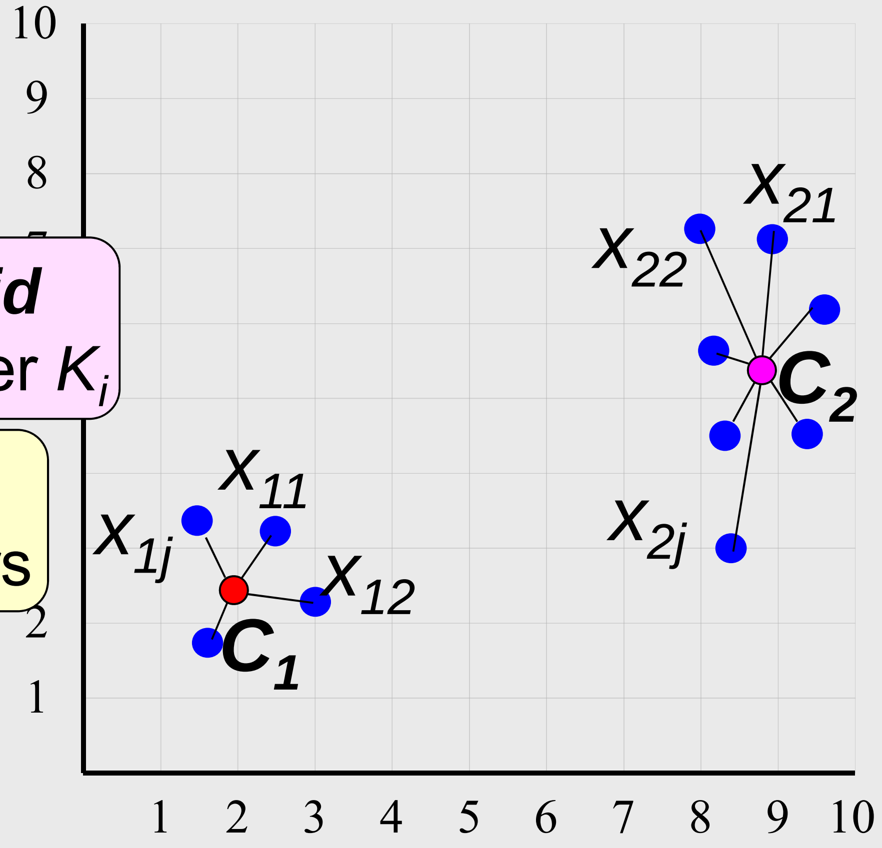
Data point j in cluster K_i

$$se_{K_i} = \sum_{j=1}^{m_i} \| x_{ij} - C_i \|^2$$

Centroid of cluster K_i

$$se_K = \sum_{i=1}^{K} se_{K_i}$$

Number of clusters



Objective Function:

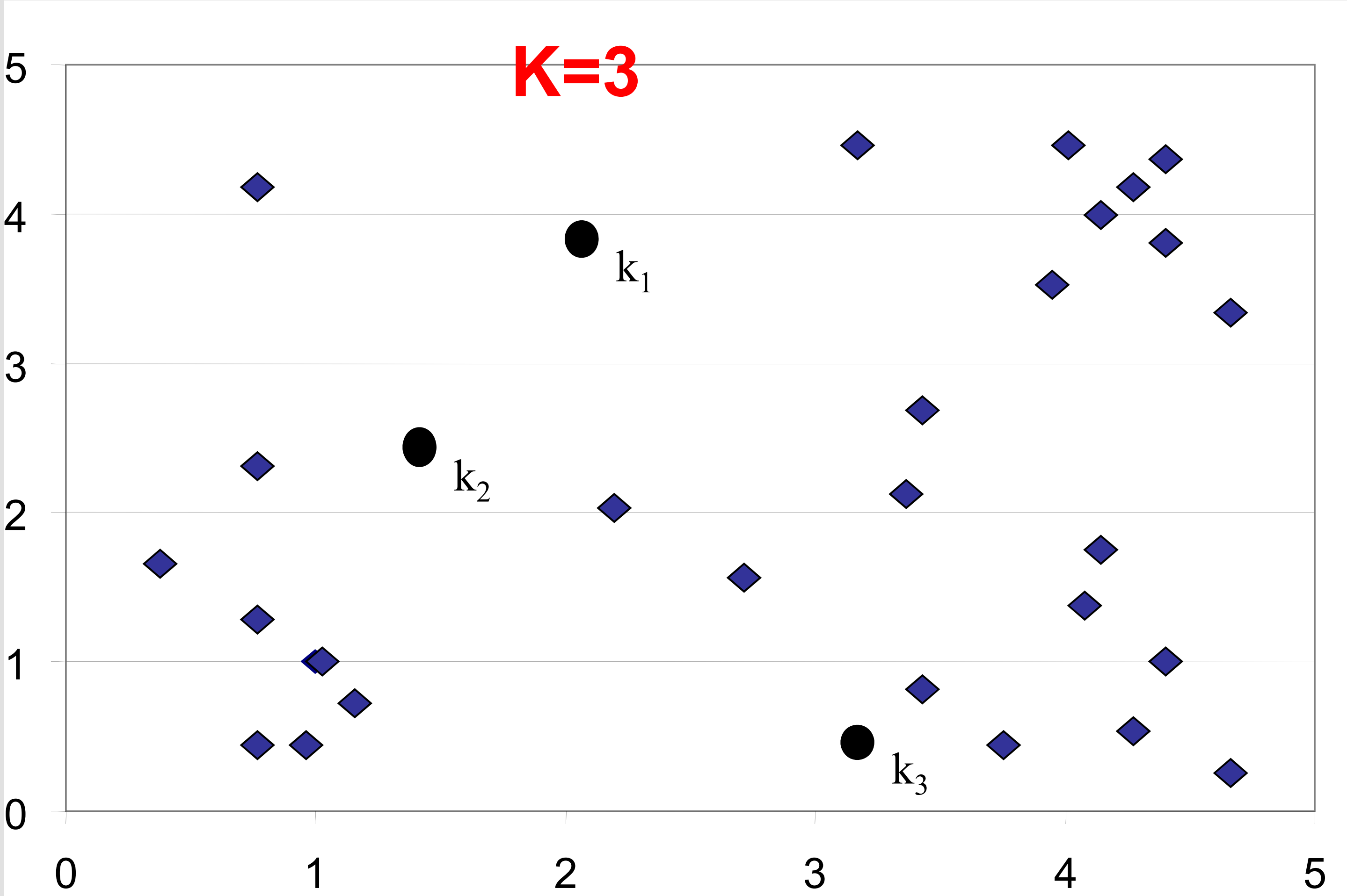
min se_K

K-means Algorithm

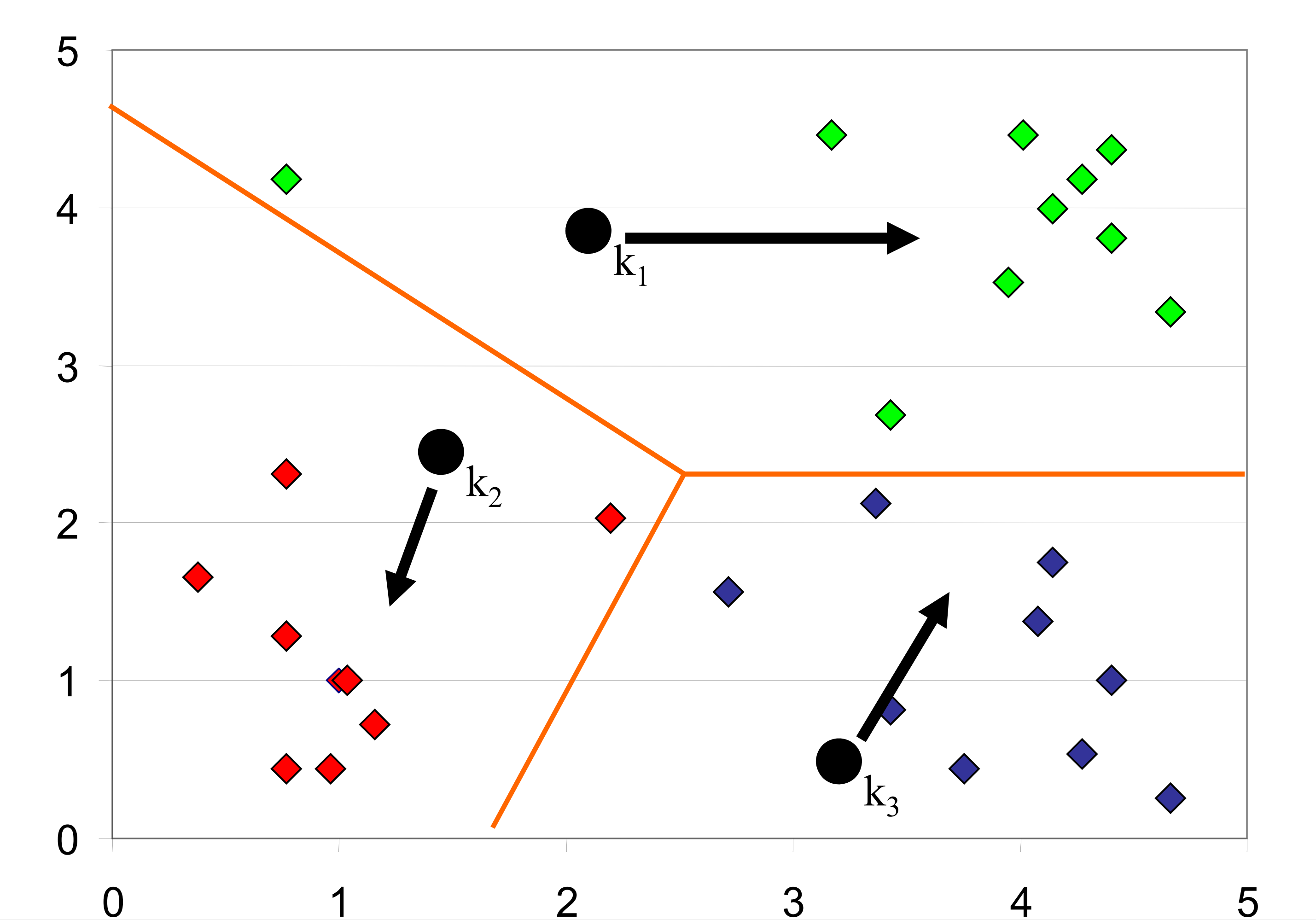
1. **Decide on a value for *k***
2. **Initialize the *k* cluster *centroids* (randomly, if necessary)**
3. **Decide the class memberships of the *N* objects by assigning them to the nearest cluster centroid**
4. **Re-estimate the *k* cluster centroids, assuming the memberships found above are correct**
5. **If none of the *N* objects changed membership in the last iteration, exit**
6. **(Otherwise) goto Step 3**

K-means Clustering: Steps 1 and 2

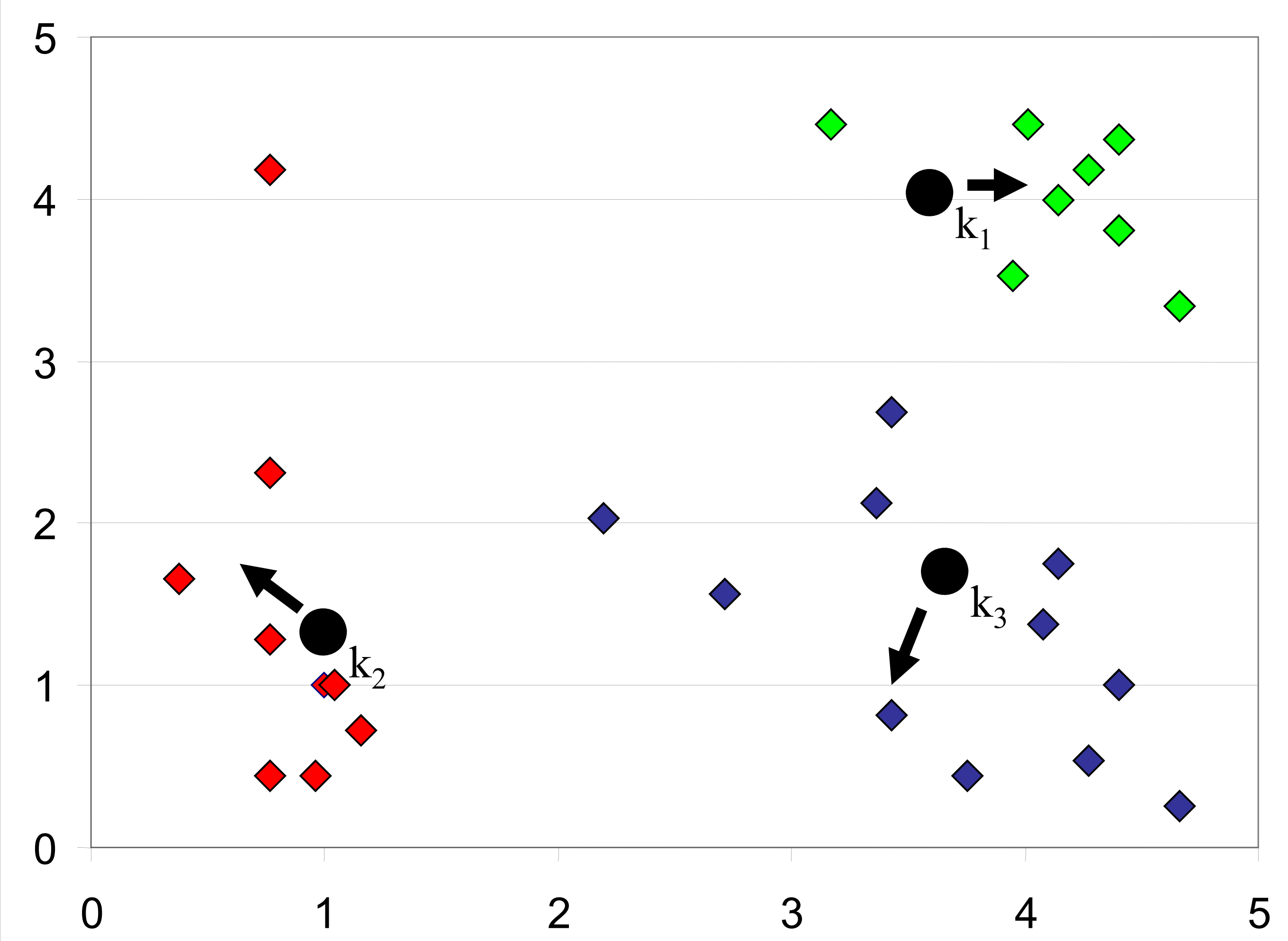
Distance Metric: Euclidean Distance



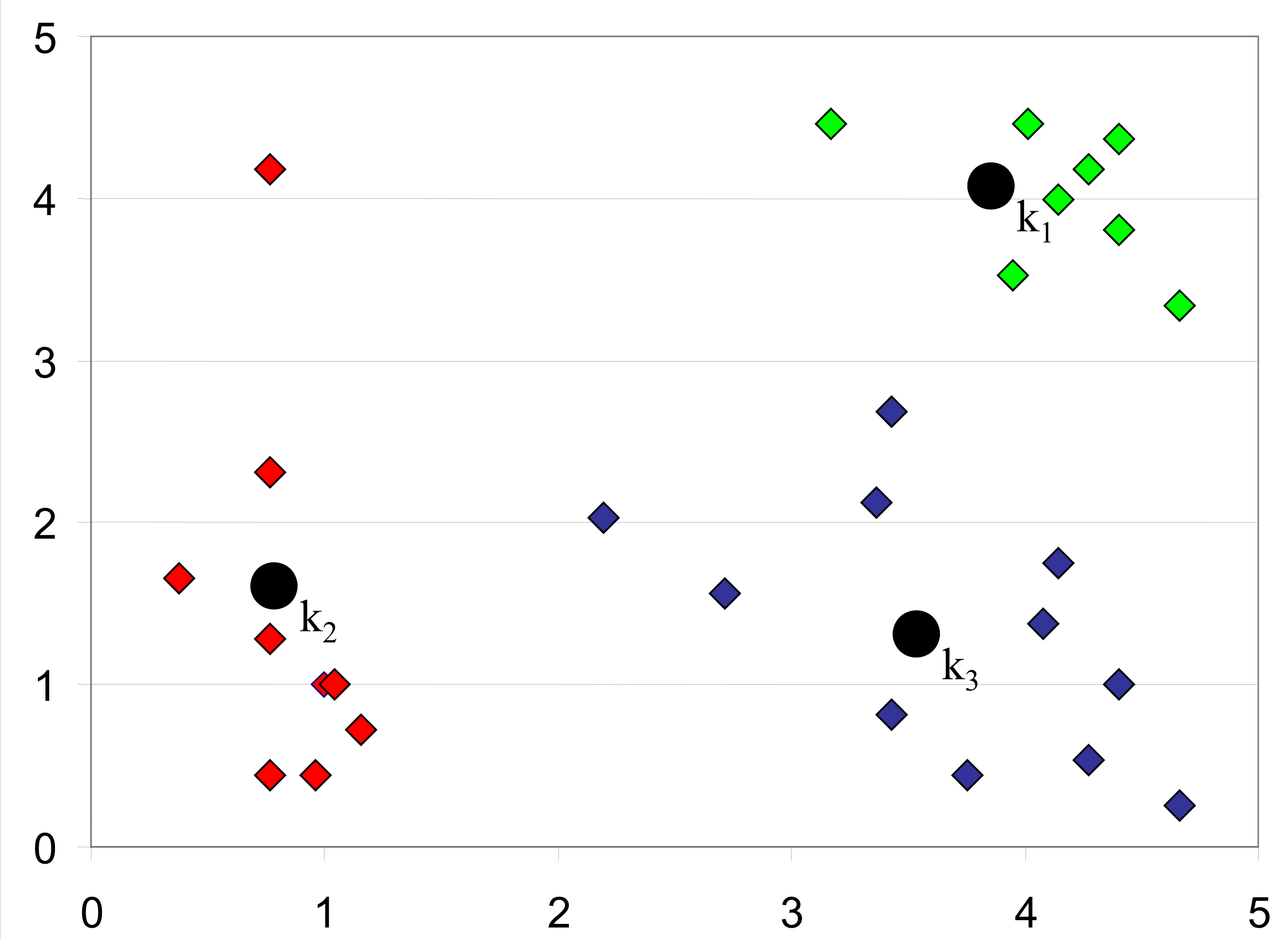
K-means Clustering: Steps 3 and 4



K-means Clustering: Steps 3 and 4

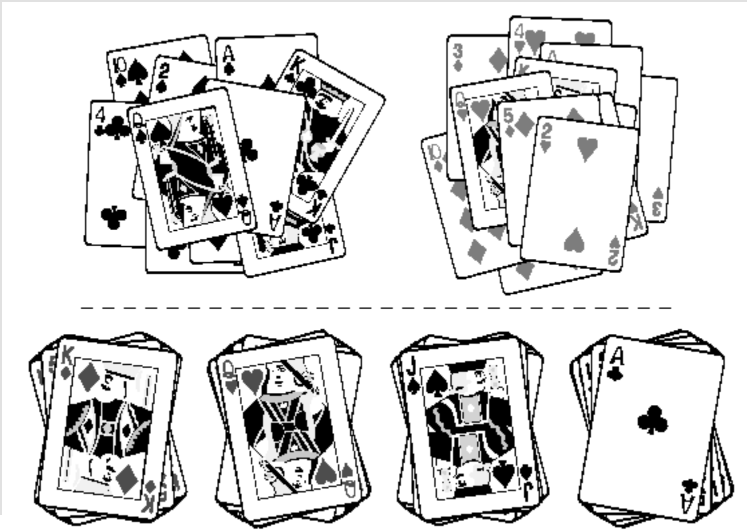


K-means Clustering: Step 5



The Meaning of K

- **Clusters describe underlying structure in data**
  - but structures in data can exist at different levels
- **In many cases, there is no a priori reason to select a particular value for k**
  - Should consider several k-s, but different values of k can lead to different clusterings



k = 2

k = 4

Normalization

- **Problem: distance will be dominated be attributes with large magnitude**
- **Example: in which cluster do we put age=25 & income=$30,000?**
  - Centroid 1: age=26 & income=$25,000
  - Centroid 2: age=80 & income=$34,500
- **Solution: normalize the data**

Summary of the K-Means Algorithm

- **Advantages**
  - ***Relatively efficient***: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, k, t << n
  - ***Global optimum*** may be found using techniques such as *deterministic annealing* and *genetic algorithms*
- **Disadvantages**
  - Need to specify k in advance
  - Applicable only when *mean* is defined
    > What about categorical data?
  - Does not deal well with overlapping clusters
  - Unable to handle noisy data and outliers
    > Outliers can pull cluster centers

WEKA

- www.cs.waikato.ac.nz/ml/weka
- **Several classifiers**
  - weka.classifiers
  - bayes.NaiveBayes – Naïve Bayes
  - trees.DecisionStump – decision trees with one split only
  - trees.J48 – decision trees
  - lazy.IBk – k nearest neighbour
- **Several clustering algorithms**
  - K-means

Reading

- **Supervised learning –** Russell, S. and Norvig, P. (2003), *Artificial Intelligence – A Modern Approach* (3rd ed), Prentice Hall
  - Decision trees – Chapter 18.1-18.2
  - Naïve Bayes – Chapter 20.1-20.2
  - Regression – Chapter 18.6.1
  - k-NN – Chapter 18.8.1
- **Naïve Bayes**
  - John, G. and Langley, P. (1995), Estimating continuous distributions in Bayesian Classifiers. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, pp. 338-345
- **Unsupervised learning and K-means clustering –**
  - http://en.wikipedia.org/wiki/K-means_clustering
  - https://sites.google.com/site/dataclusteringalgorithms/k-means-clustering-algorithm