MONASH University
Information Technology

MONASH University
Information Technology

FIT5047 – Intelligent Systems

Machine Learning
Chapters 18.1-4, 18.6.1, 18.6.3-4, 18.7, 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
  - Artificial Neural Networks
- **Unsupervised learning**
  - Clustering
    - > Similarity measures
    - > The K-means algorithm

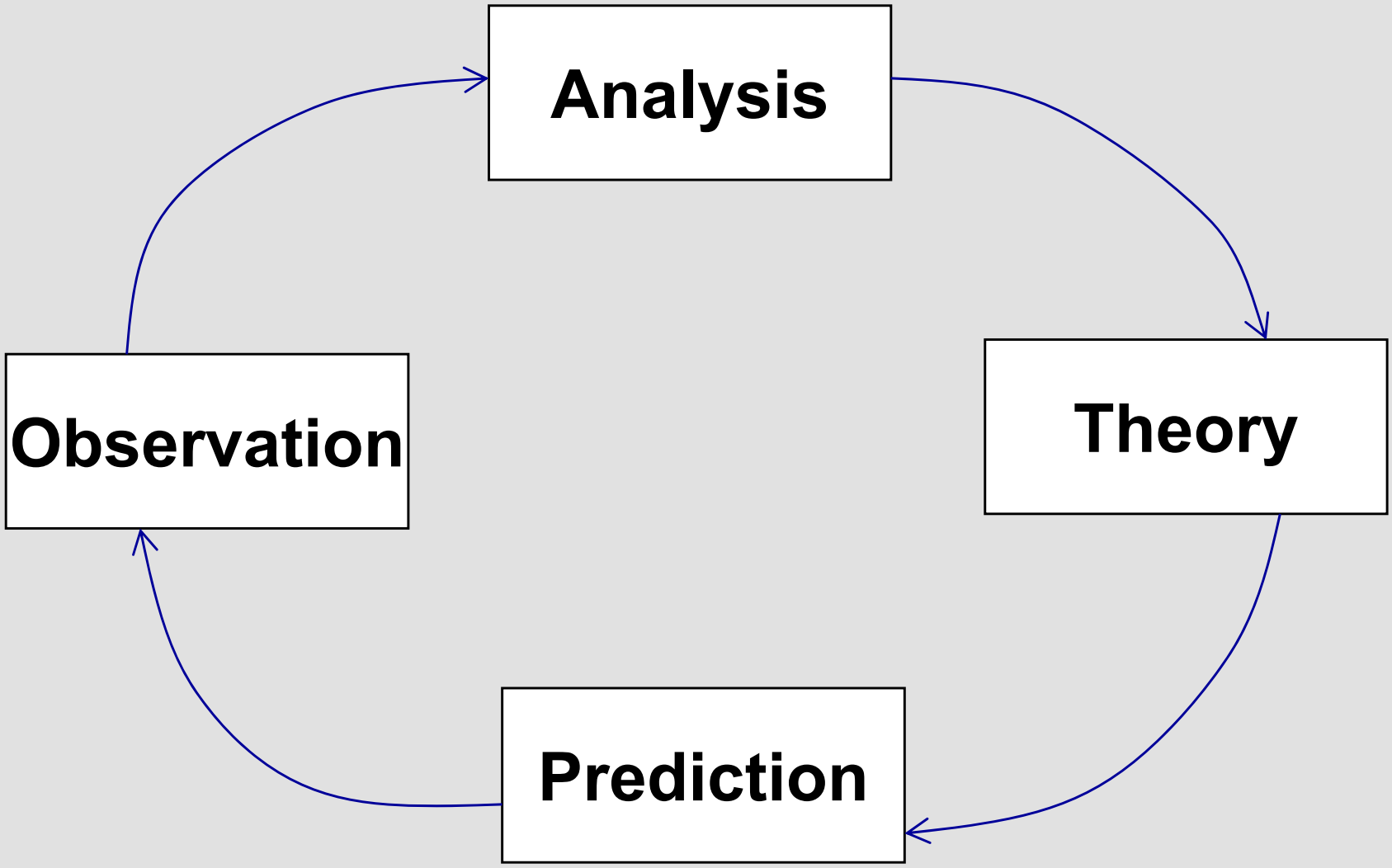
Why Learning?

- **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,** e.g., Classification and Regression
    - > **Unsupervised learning,** e.g., Clustering, Association Analysis, Deviation/Outlier Detections
    - > **Reinforcement learning,** e.g., Q-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 solution**

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
- **"All models are wrong, but some models are useful."**

  George Box
- **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 **)**

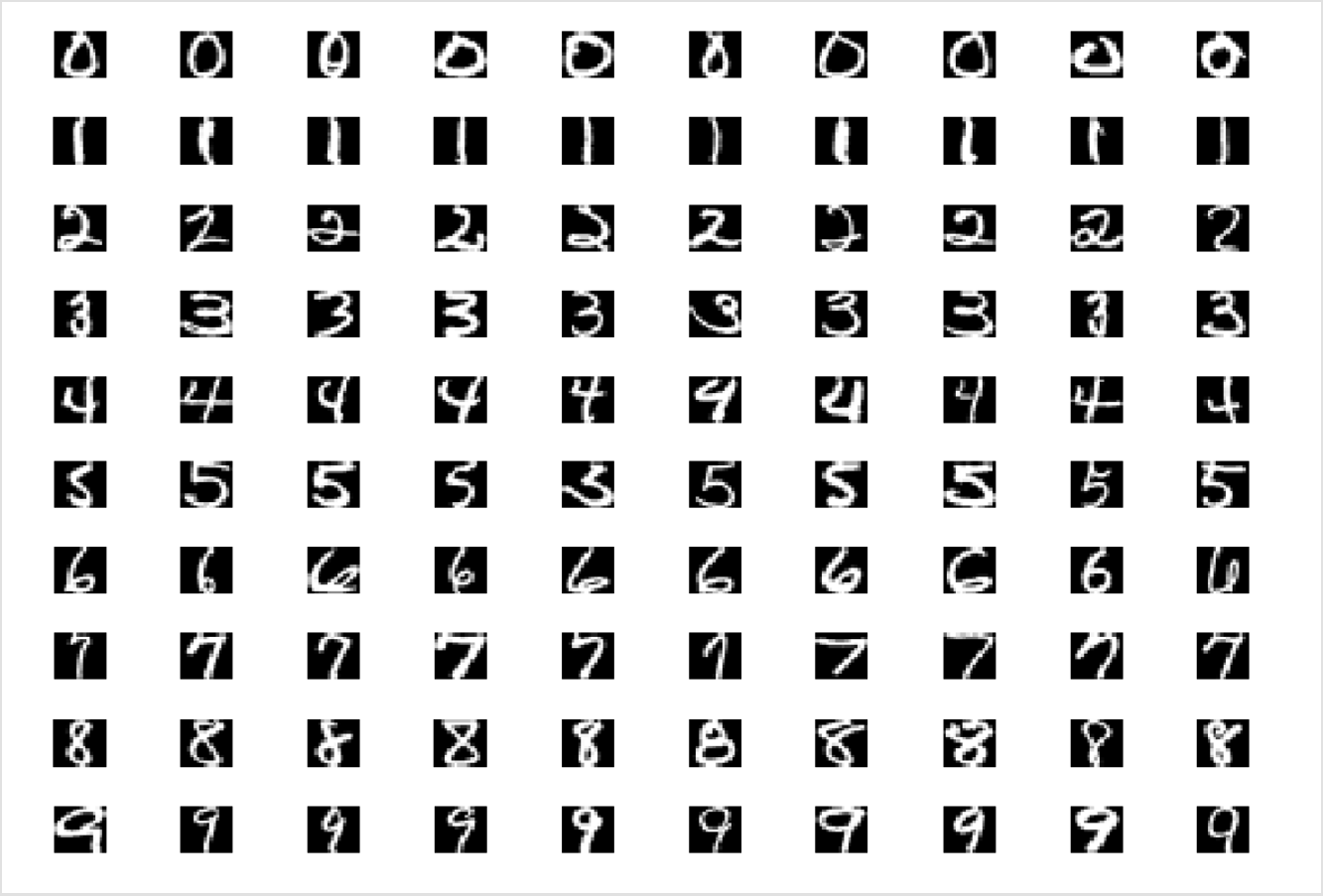
MONASH University
Information Technology

FIT5047 – Intelligent Systems

Mathematical Principles of Machine Learning

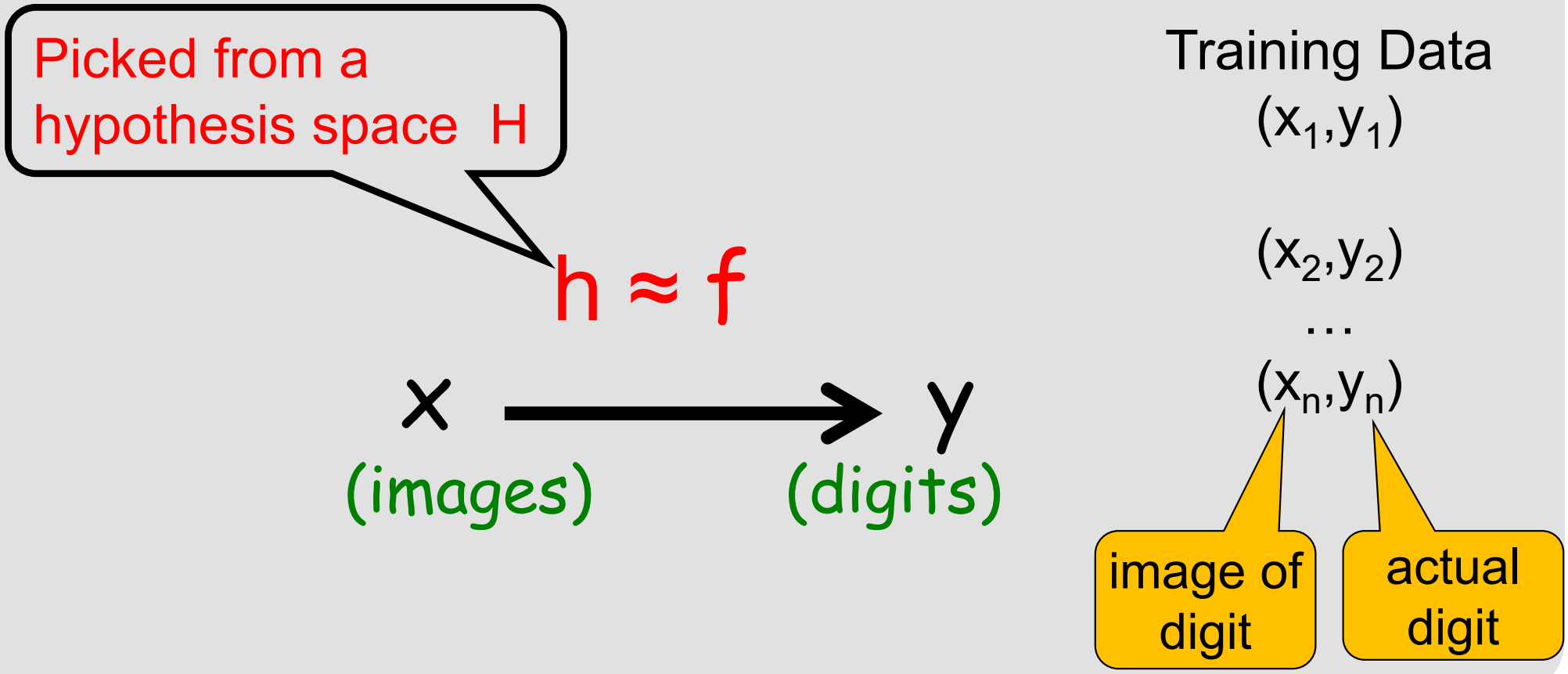
Supervised Learning Example: Handwritten Digit Recognition (I)

- **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**



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 ≈ f (h approximates f as best as possible)
    - > **h must generalize well on unseen examples**

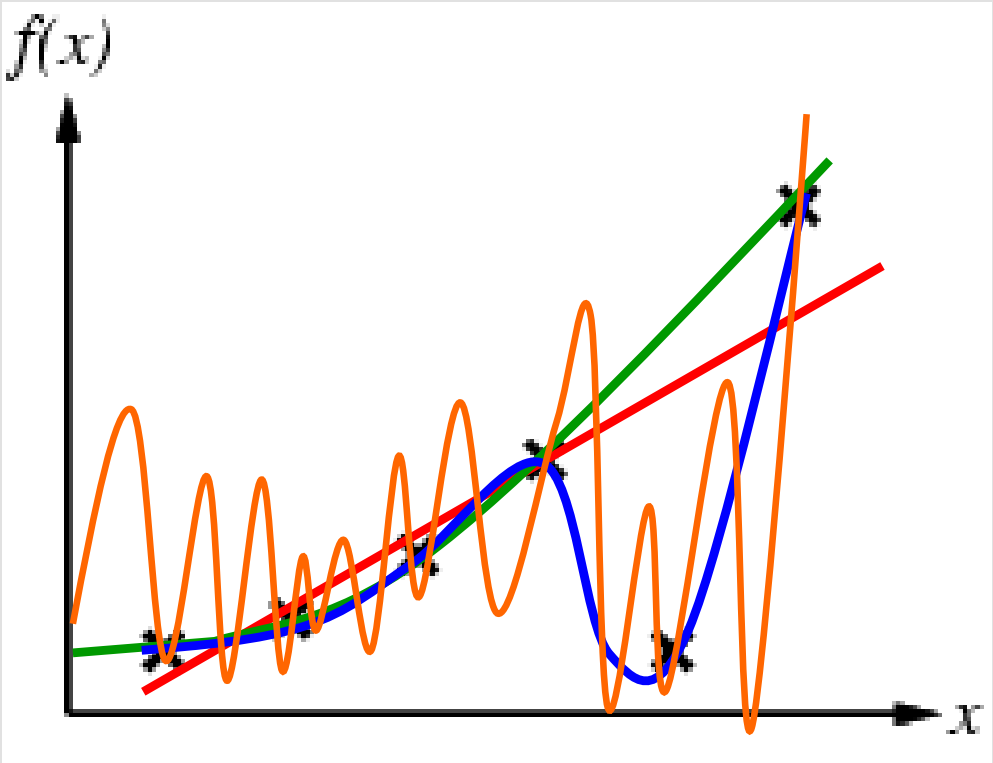
Picking the Best Hypothesis h

- **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")***

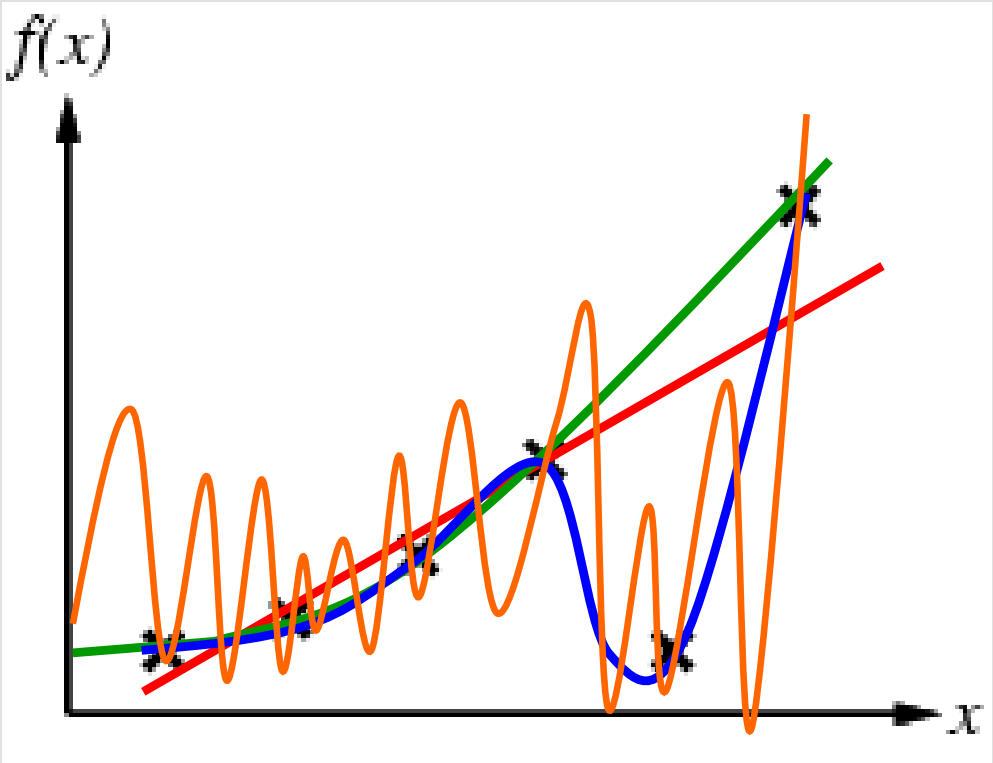
  William of Ockham (1287–1347)

- **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

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

Bias

Idea 1: Error on the training data

Learning is indeed Search/Optimization

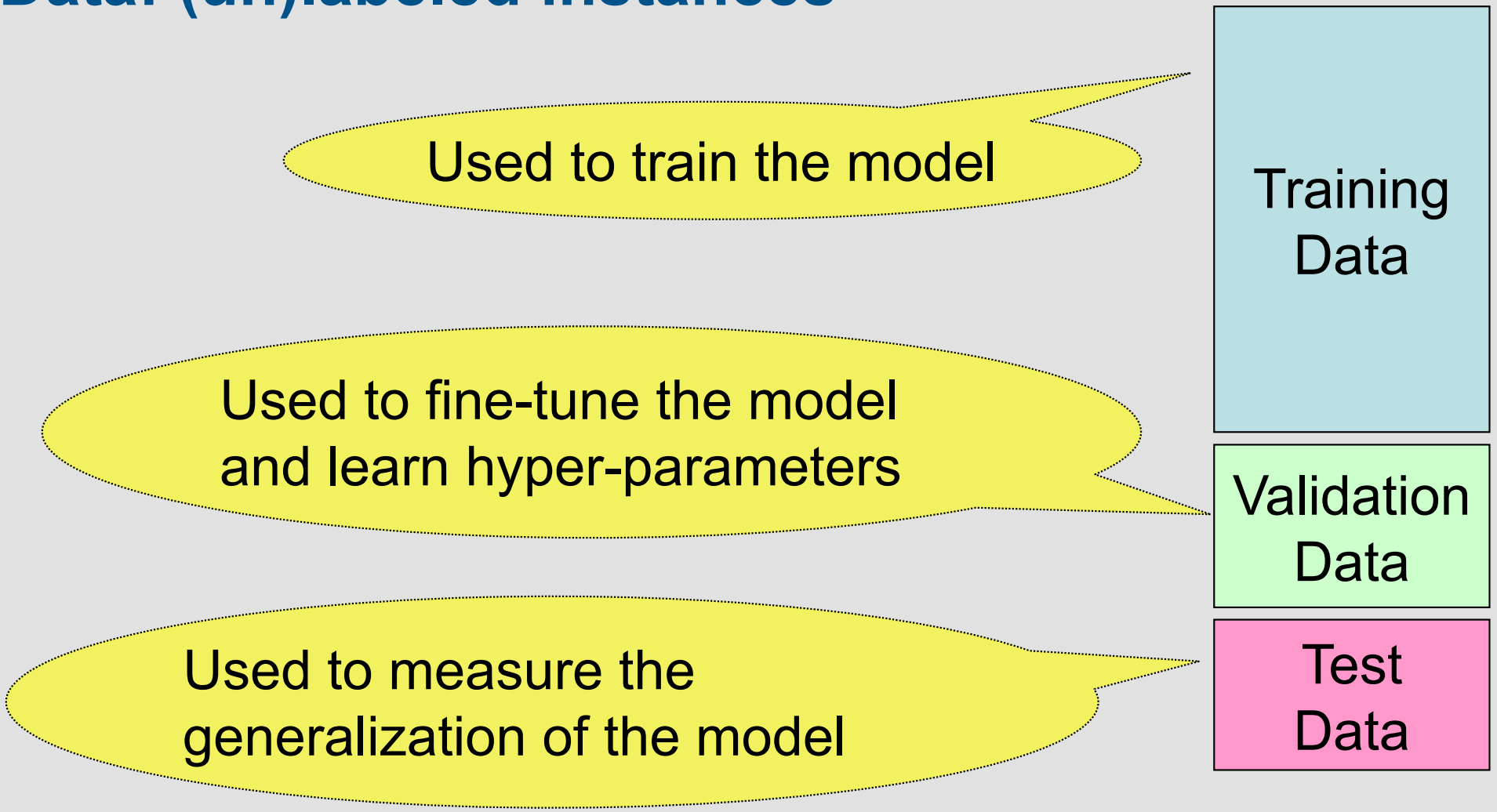
MONASH University
Information Technology

FIT5047 – Intelligent Systems

Concepts in Machine Learning

Types of Data

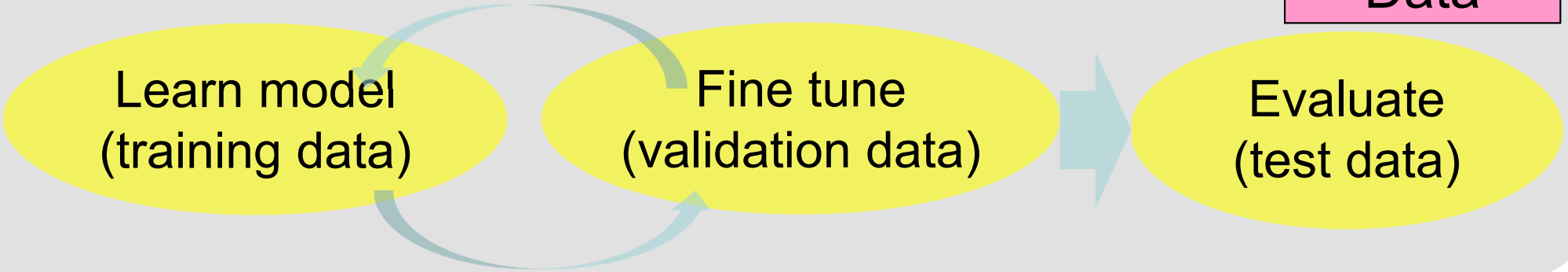
- **Data: (un)labeled instances**



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!

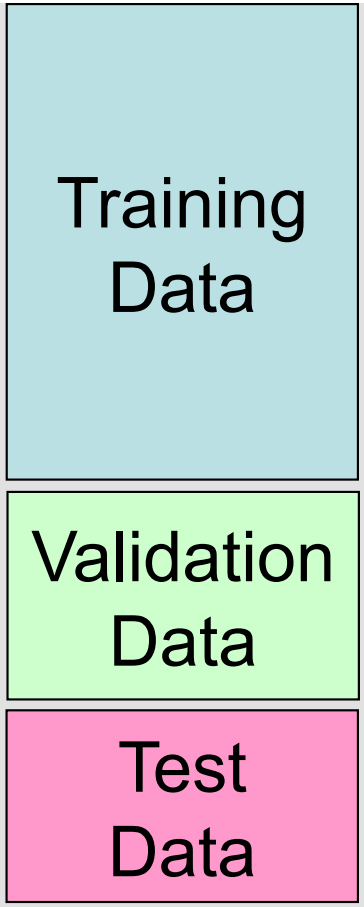


Evaluation Metrics

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

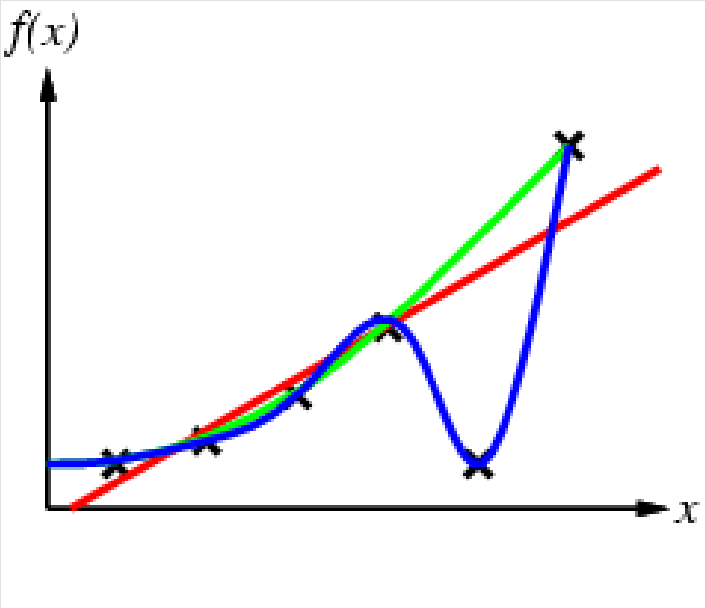
If 80 predictions are correct out of 100 then accuracy = 80/100 =0.8

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



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

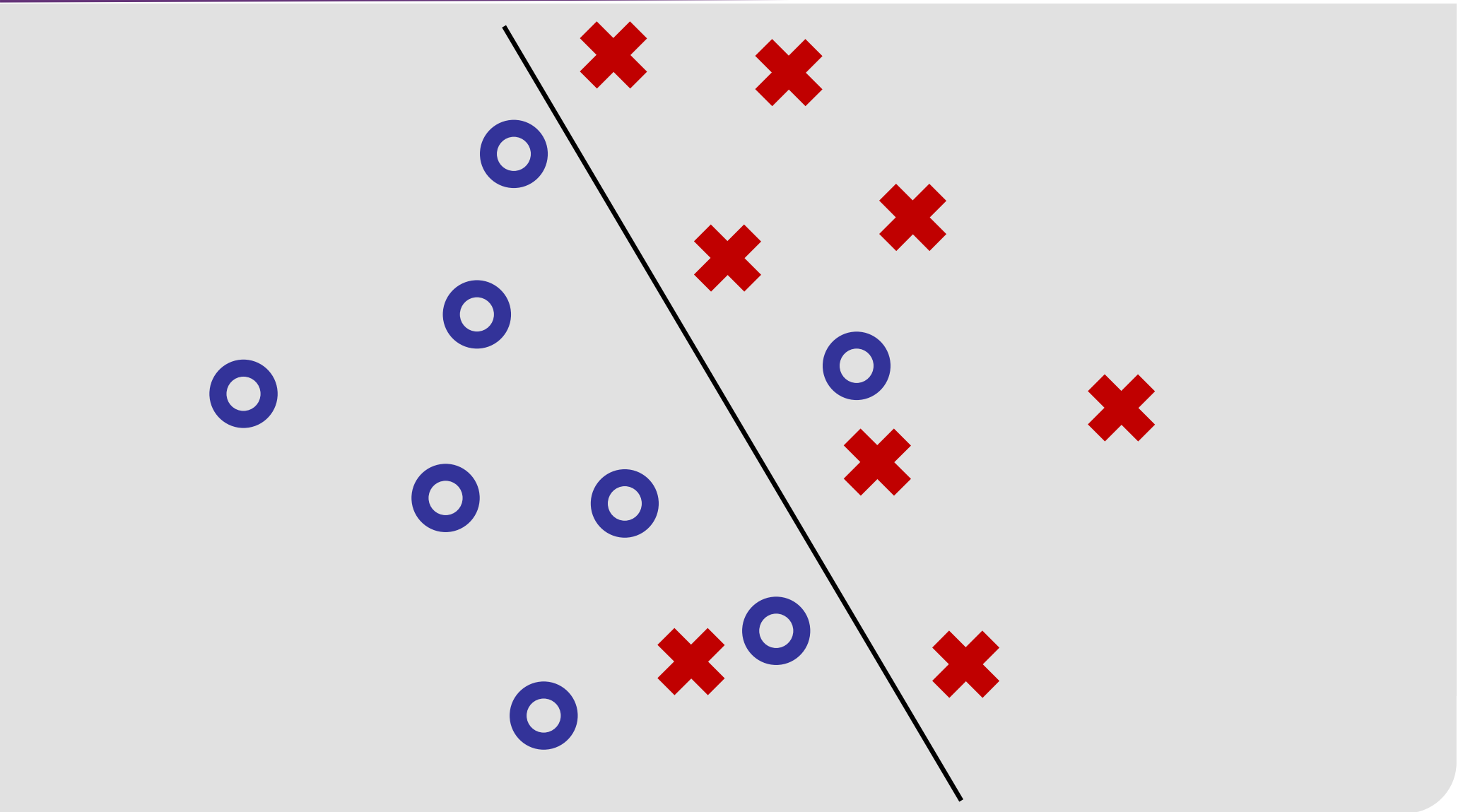
Discovery Driven – Supervised Learning

- **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
    - > It can also infer discrete values

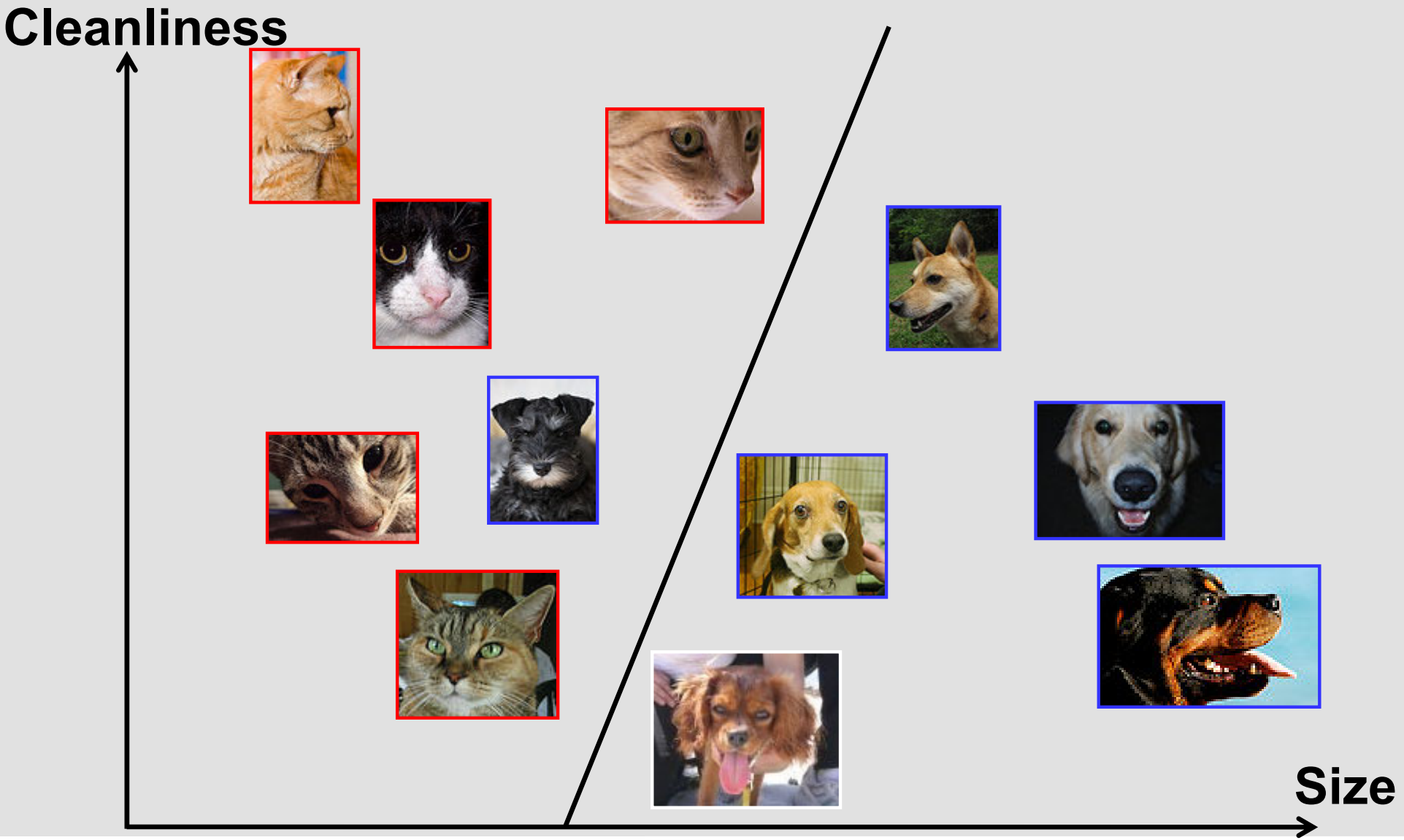
Classification



Classification – Example (I)

Cat **?** **Dog**

Classification – Example (II)



Regression – Example (I)

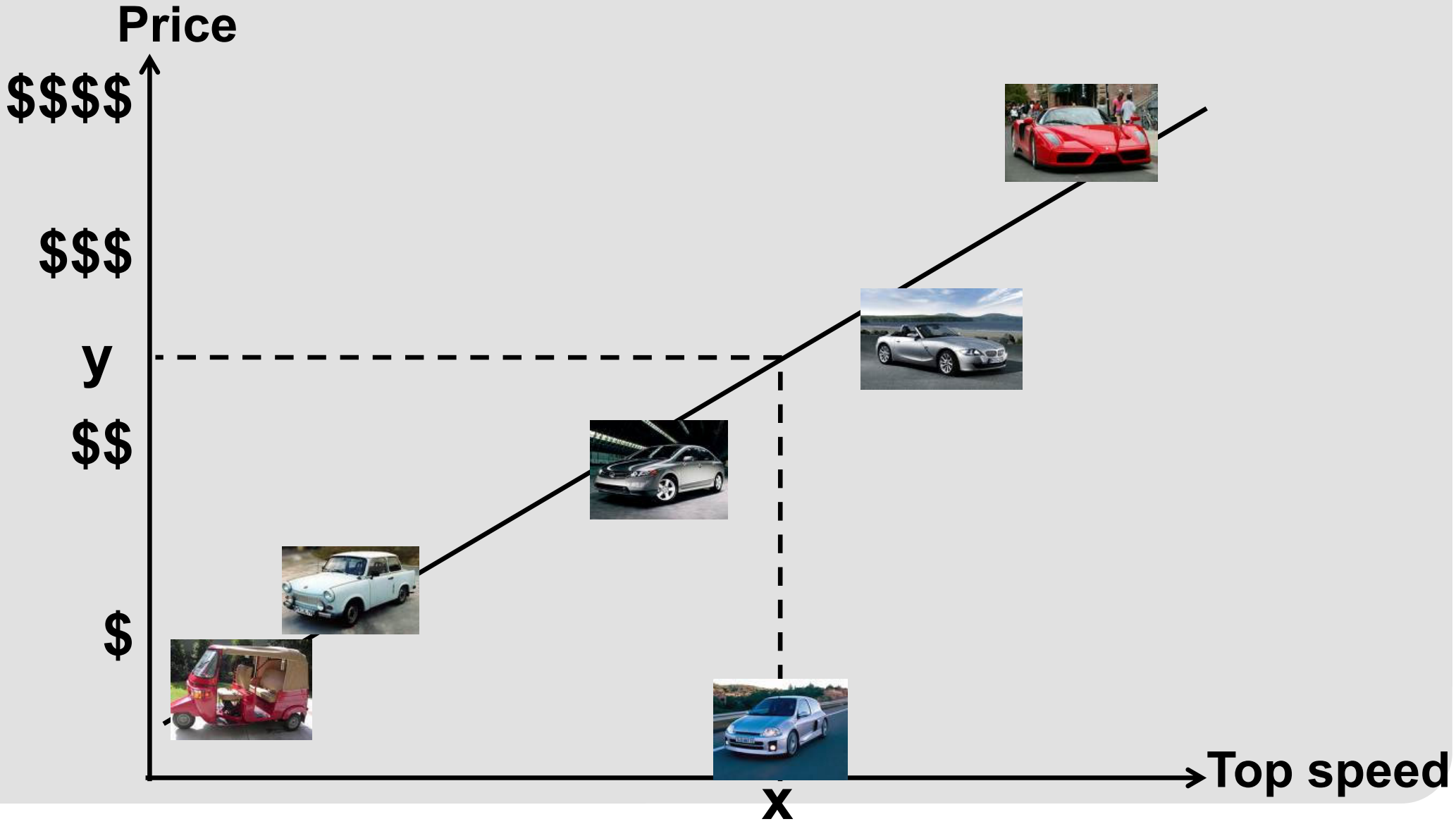
$ $$ $$$ $$$$





?

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**
- **General Techniques:**
  - Clustering
  - Association Analysis
  - Outlier detection
- **Lack of supervision:**
  - given a set of observations (***training data***), infer patterns in the data
  - training data is unlabelled – there are no pre-defined classes

General Techniques

- **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*
- **Outlier detection –**
  - Discovers exceptions to normal patterns in the data

MONASH University
Information Technology

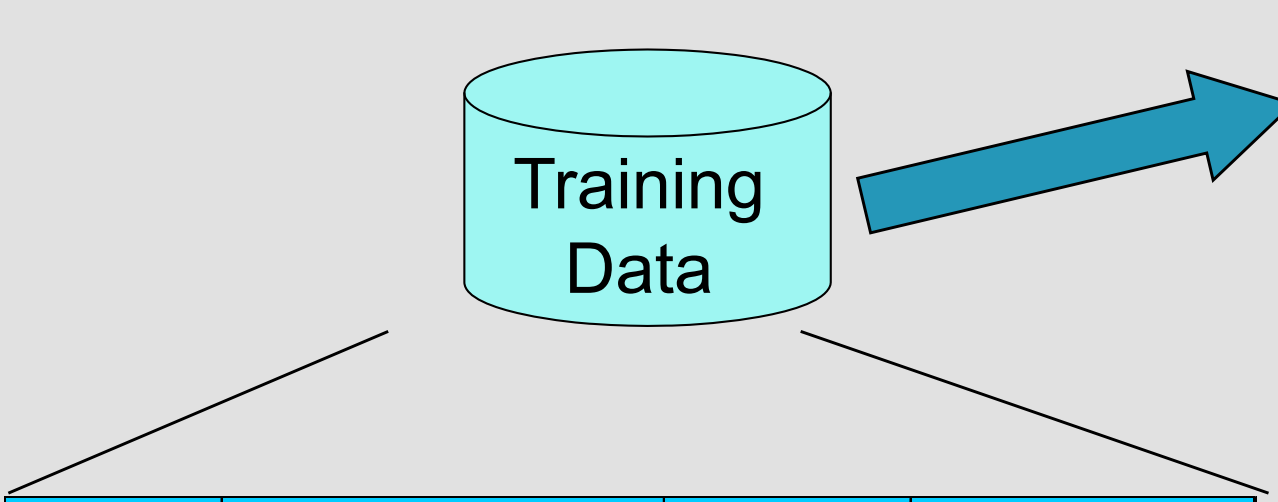
FIT5047 –Intelligent Systems

Supervised Machine Learning – Classification

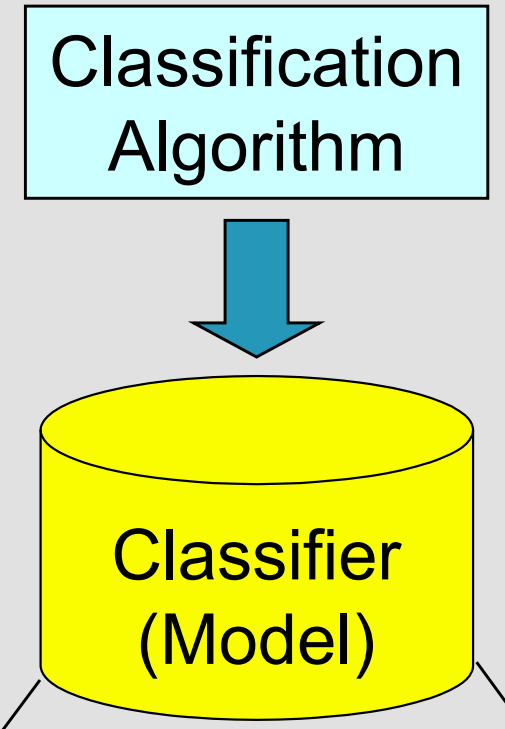
Classification Process

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

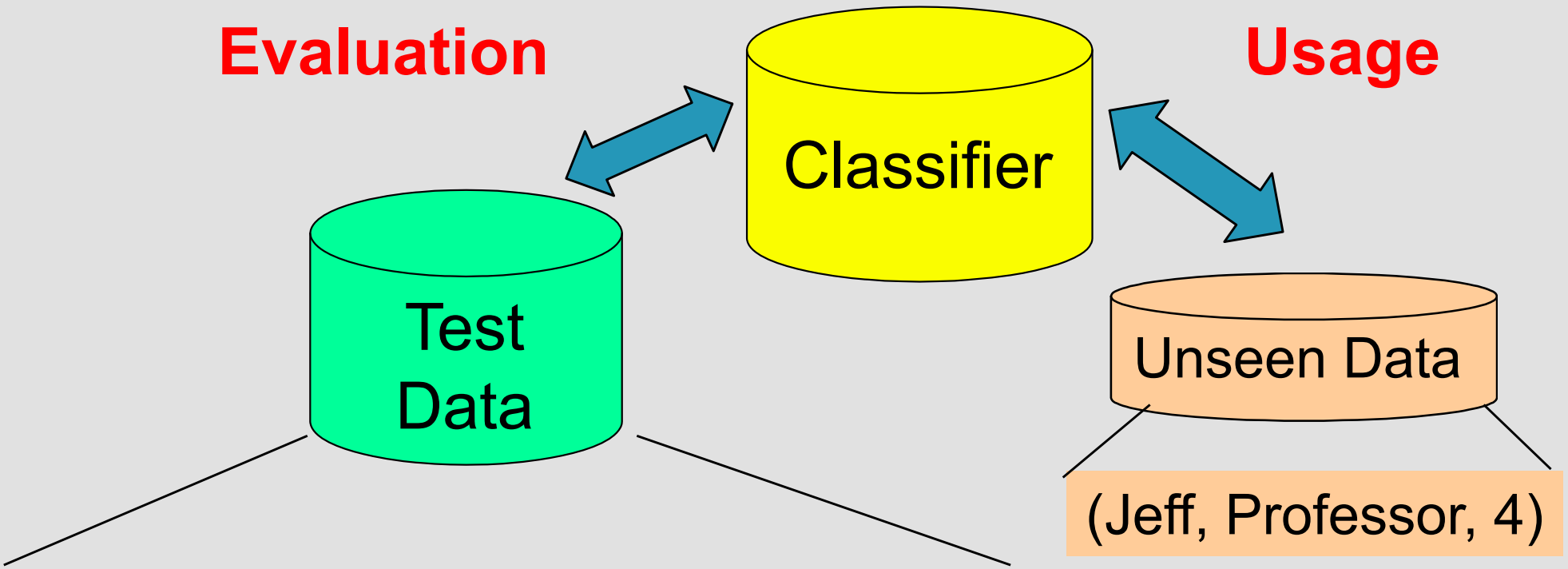


NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no



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

Model Evaluation and Usage



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 runs on different training/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

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

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

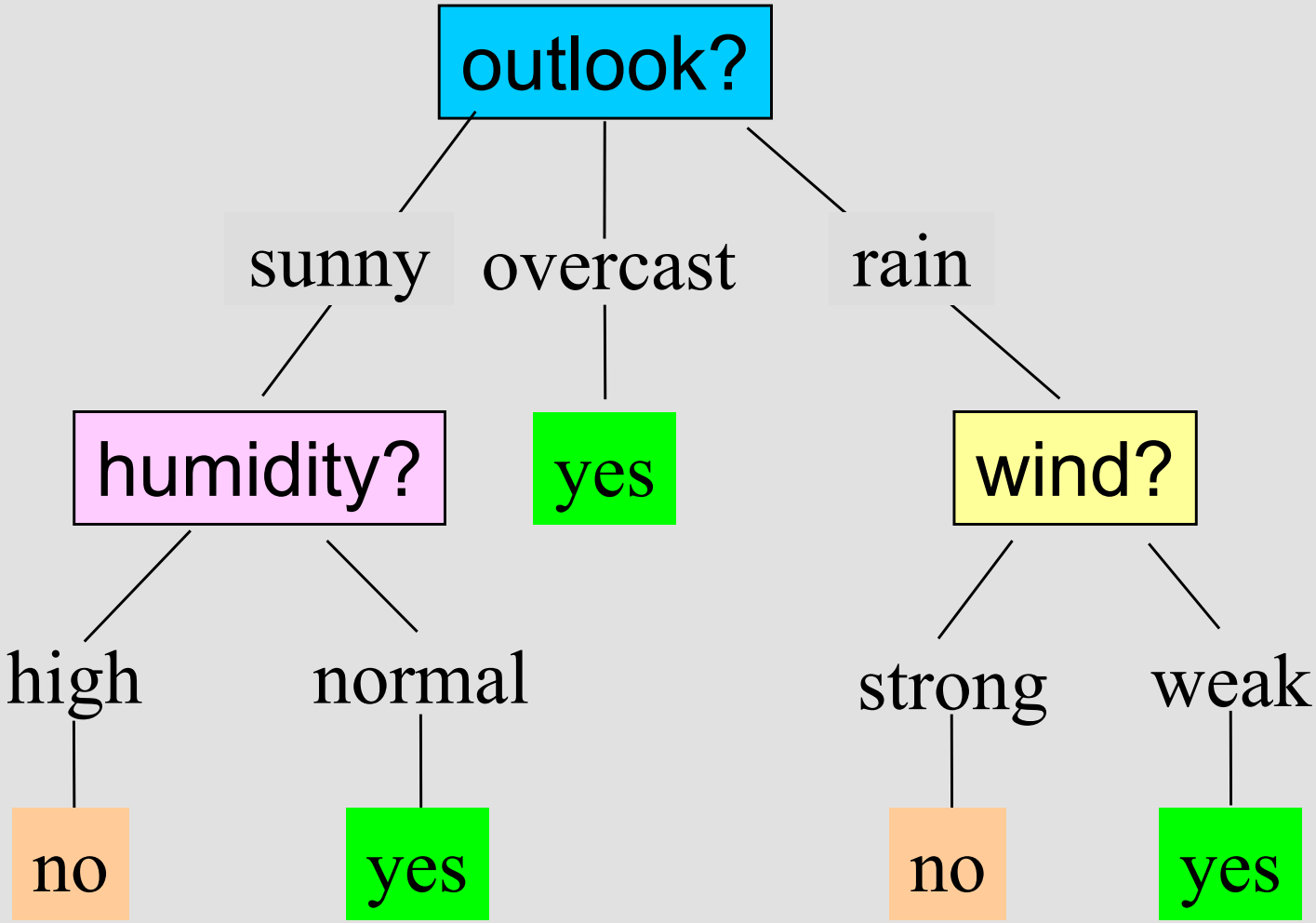
Classification by DT Induction

- **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

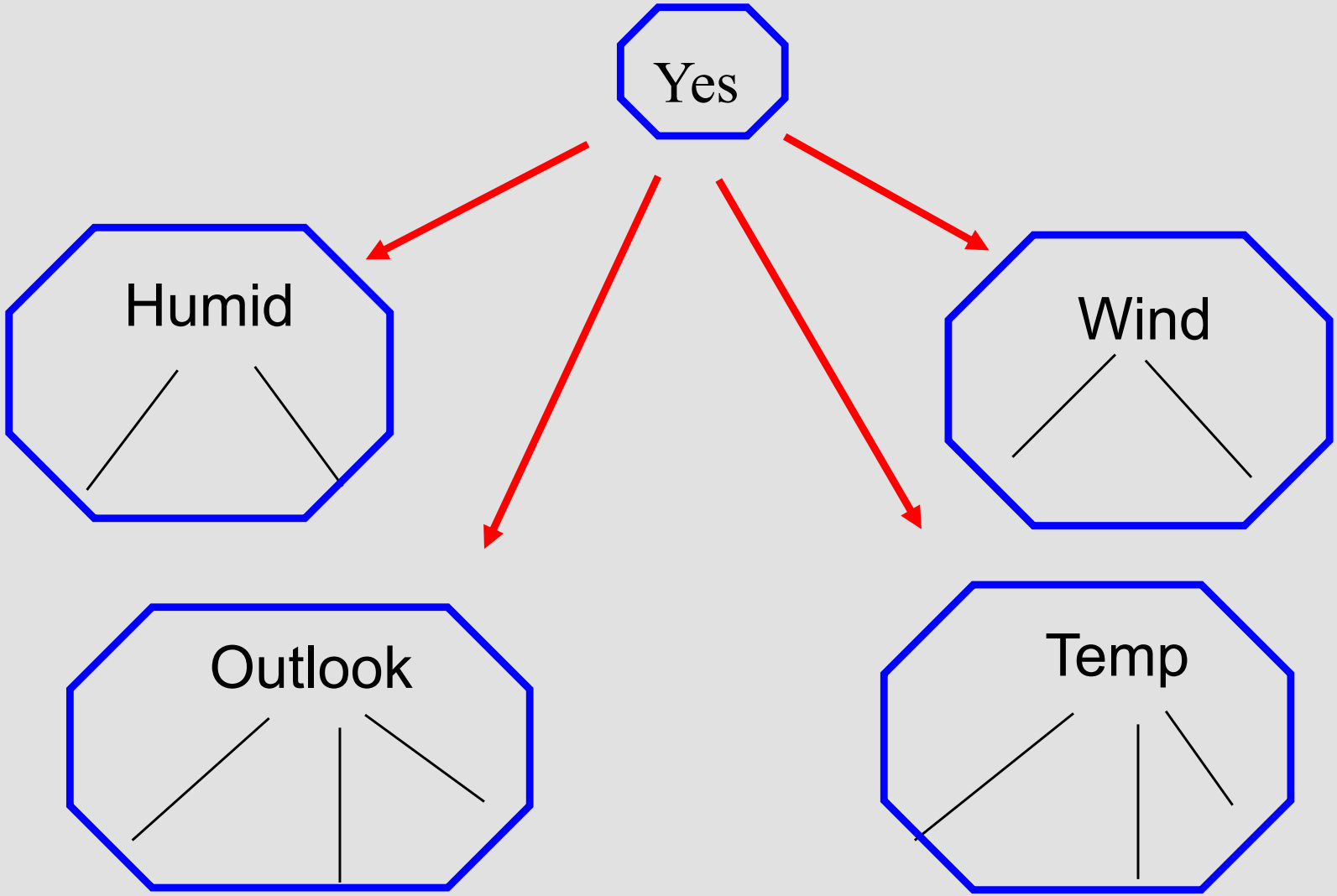
- **Approach – choose the attribute that best separates training examples into targeted classes**
- **Examples of proposed techniques**
  - ***Information Gain*** [Quinlan, 1975]
  - ***Message Length*** [Wallace and Patrick, 1993]

What is the "simplest" Tree? – Example

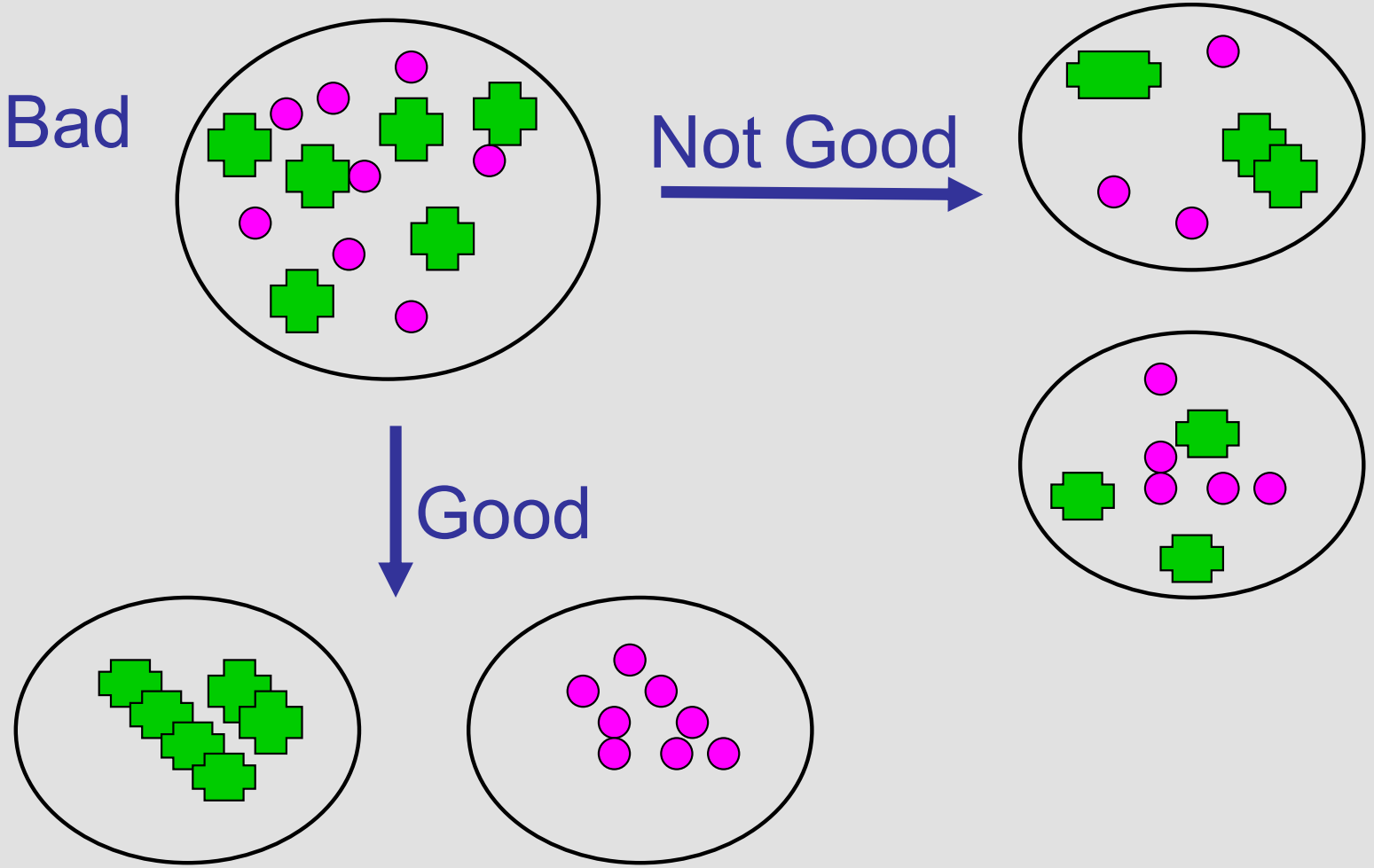
- **Majority class (not a tree)**
  - Always predict "yes"
  - How good it is?
    - > Correct on 9 examples
    - > Wrong on 5 examples
- **Decision stump**
  - A tree with one decision node
    - > Splits on one attribute

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

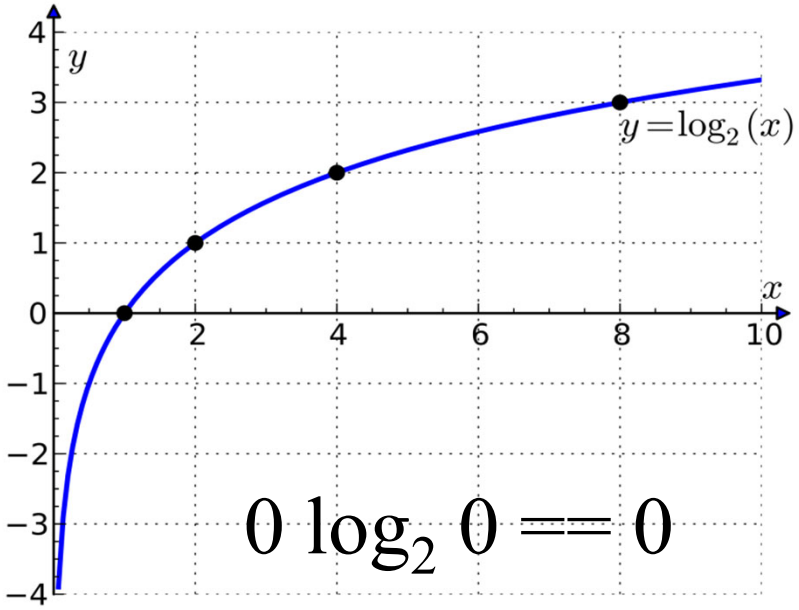
- ***Entropy*** **measures the amount of uncertainty in a probability distribution**
- **Given a discrete random variable on a finite set $X=\{x_1, \ldots, x_n\}$, with probability distribution function $\Pr(x)=\Pr(X=x)$, the *entropy* $H(X)$ of X is defined as**

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

where $0 \leq H(X)$

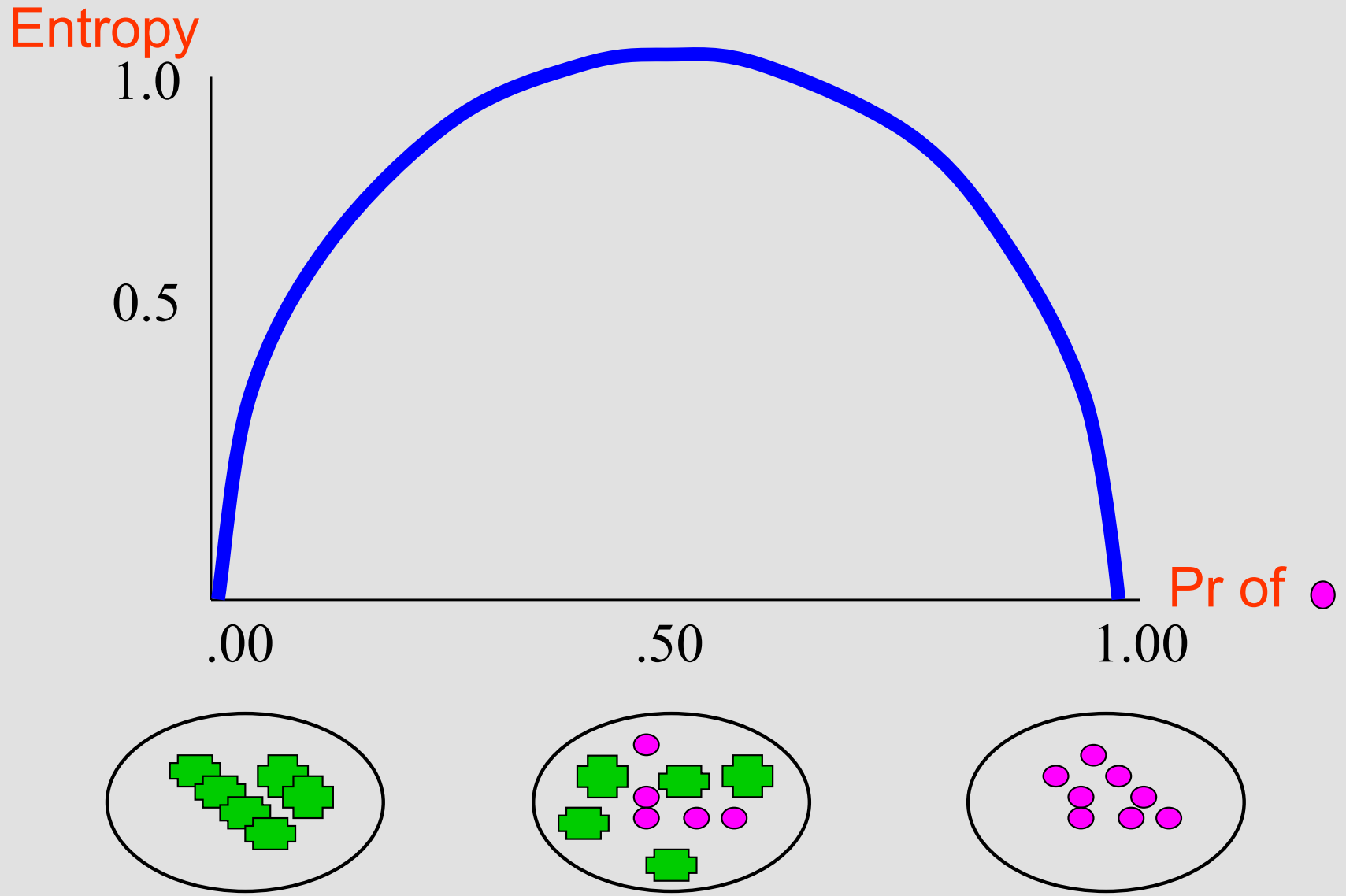
Entropy – Example

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



- **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 maximizes ***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} = [6+, 2-]

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

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

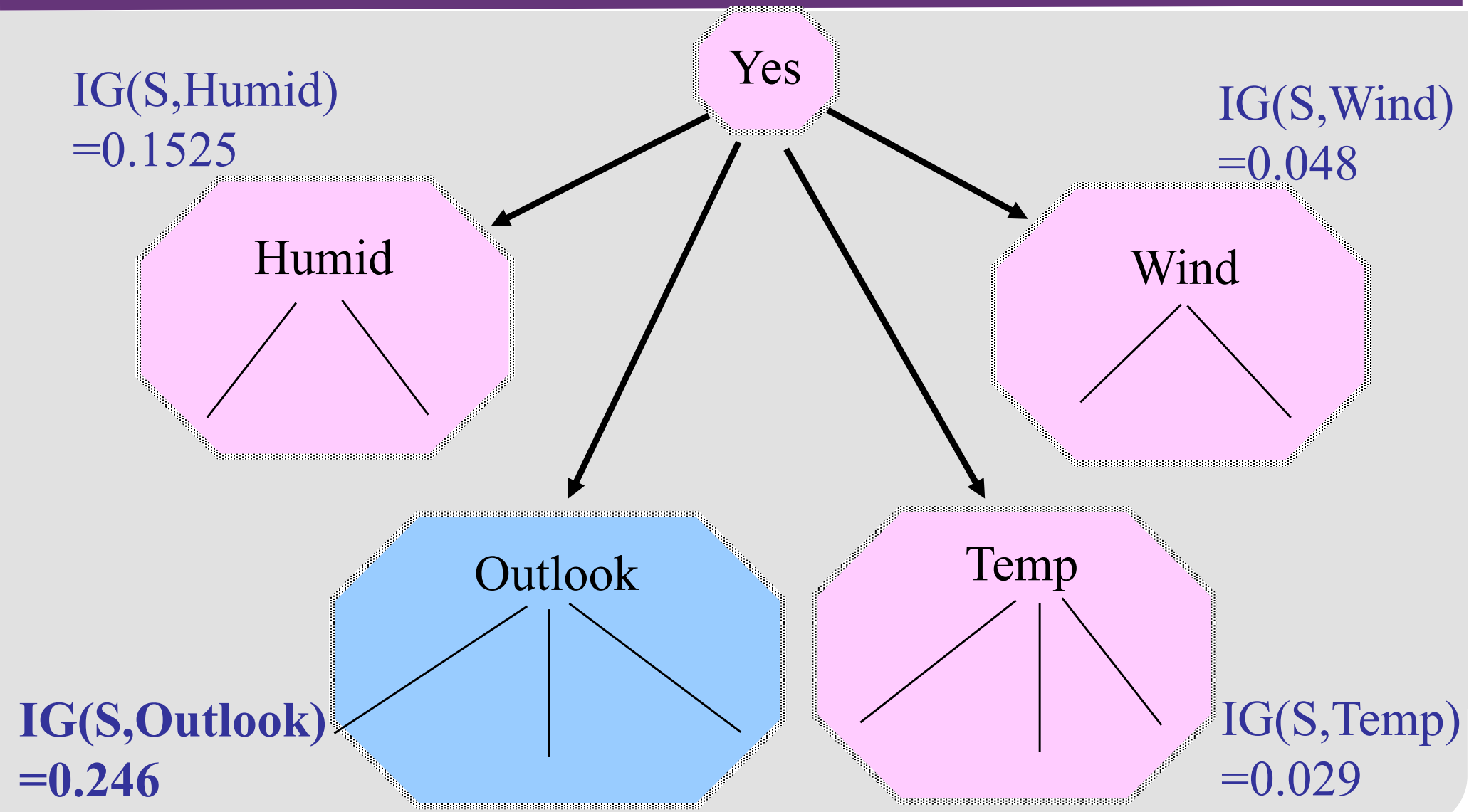
= H(S) - 8/14 H(S_{weak}) - 6/14 H(S_{strong})

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

= 0.048

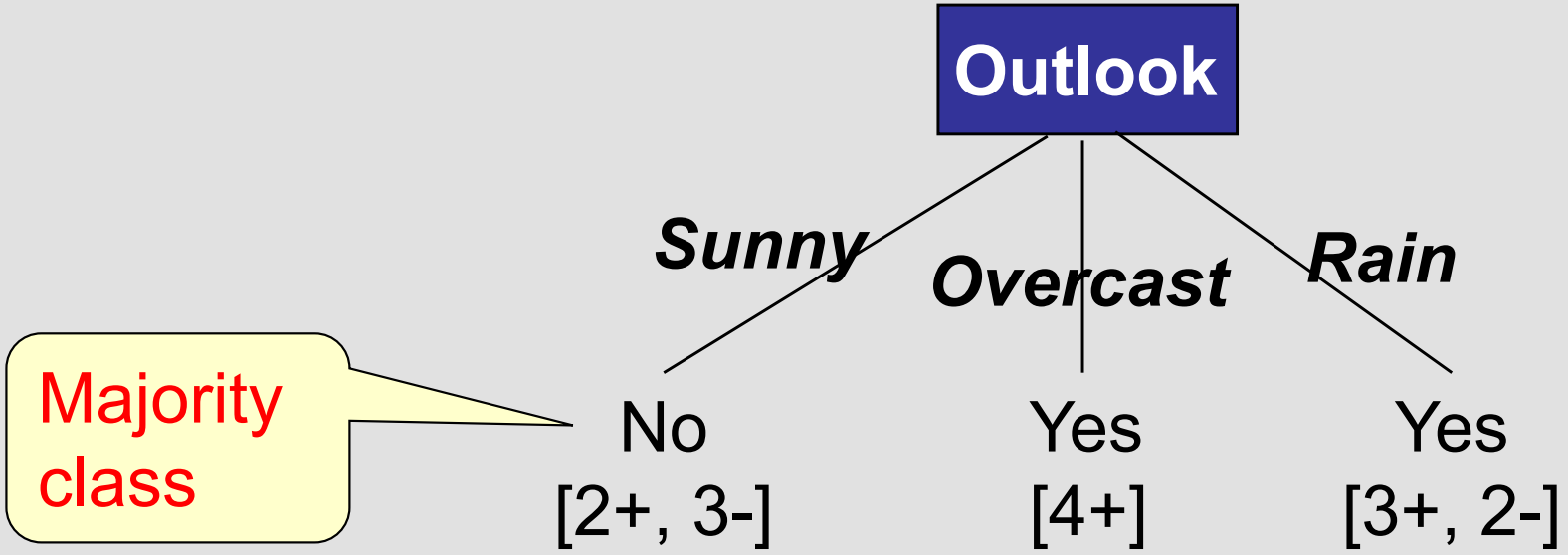
Day	Wind	Play ball?
d1	weak	no
d2	strong	no
d3	weak	yes
d4	weak	yes
d5	weak	yes
d6	strong	no
d7	strong	yes
d8	weak	no
d9	weak	yes
d10	weak	yes
d11	strong	yes
d12	strong	yes
d13	weak	yes
d14	strong	no

Evaluating Attributes

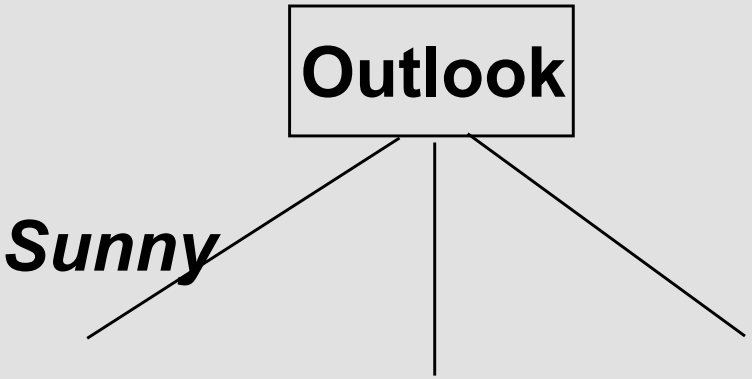


Building a Decision Tree (I)

Good day for playing ball?

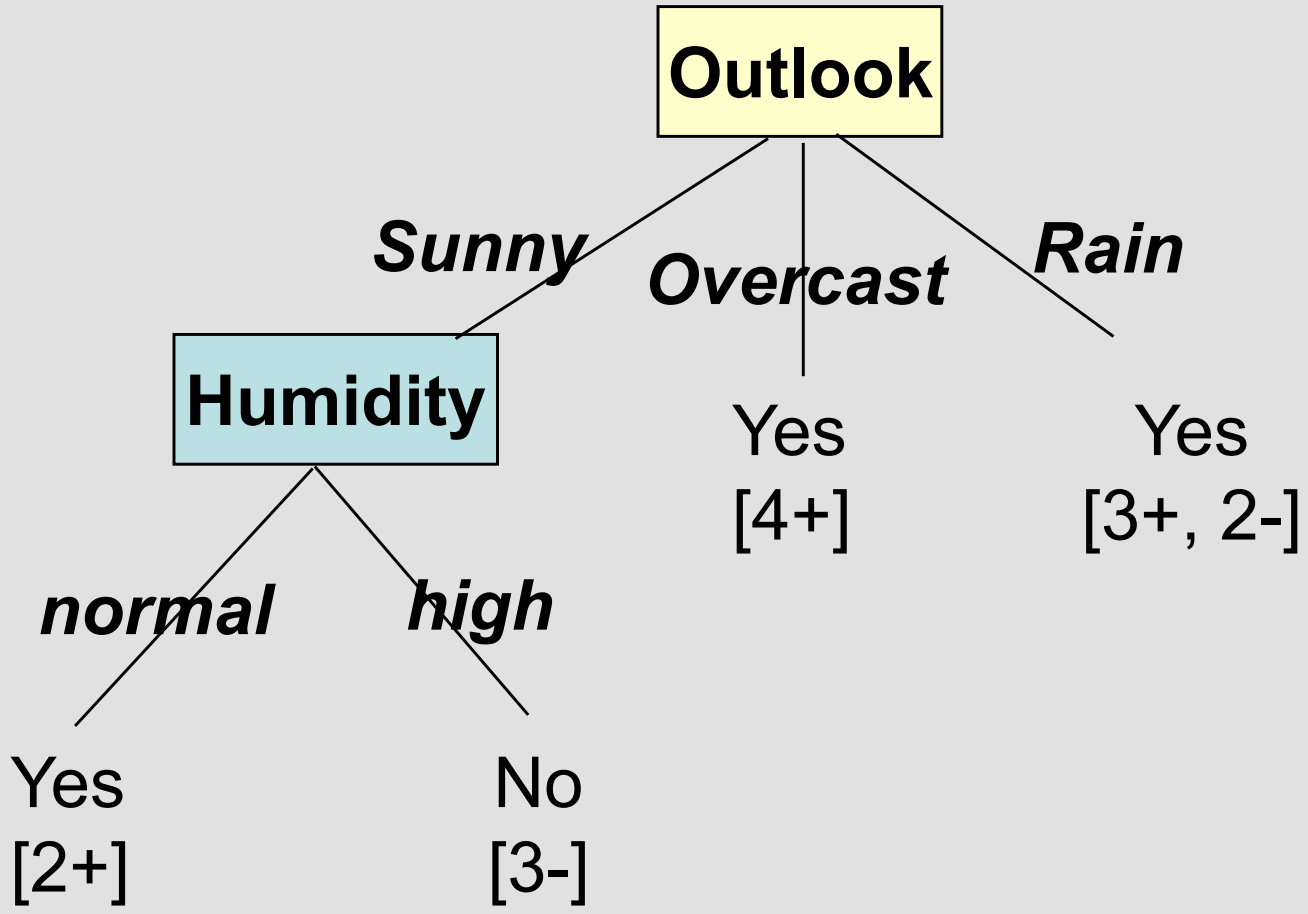


Building a Decision Tree (II)



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)

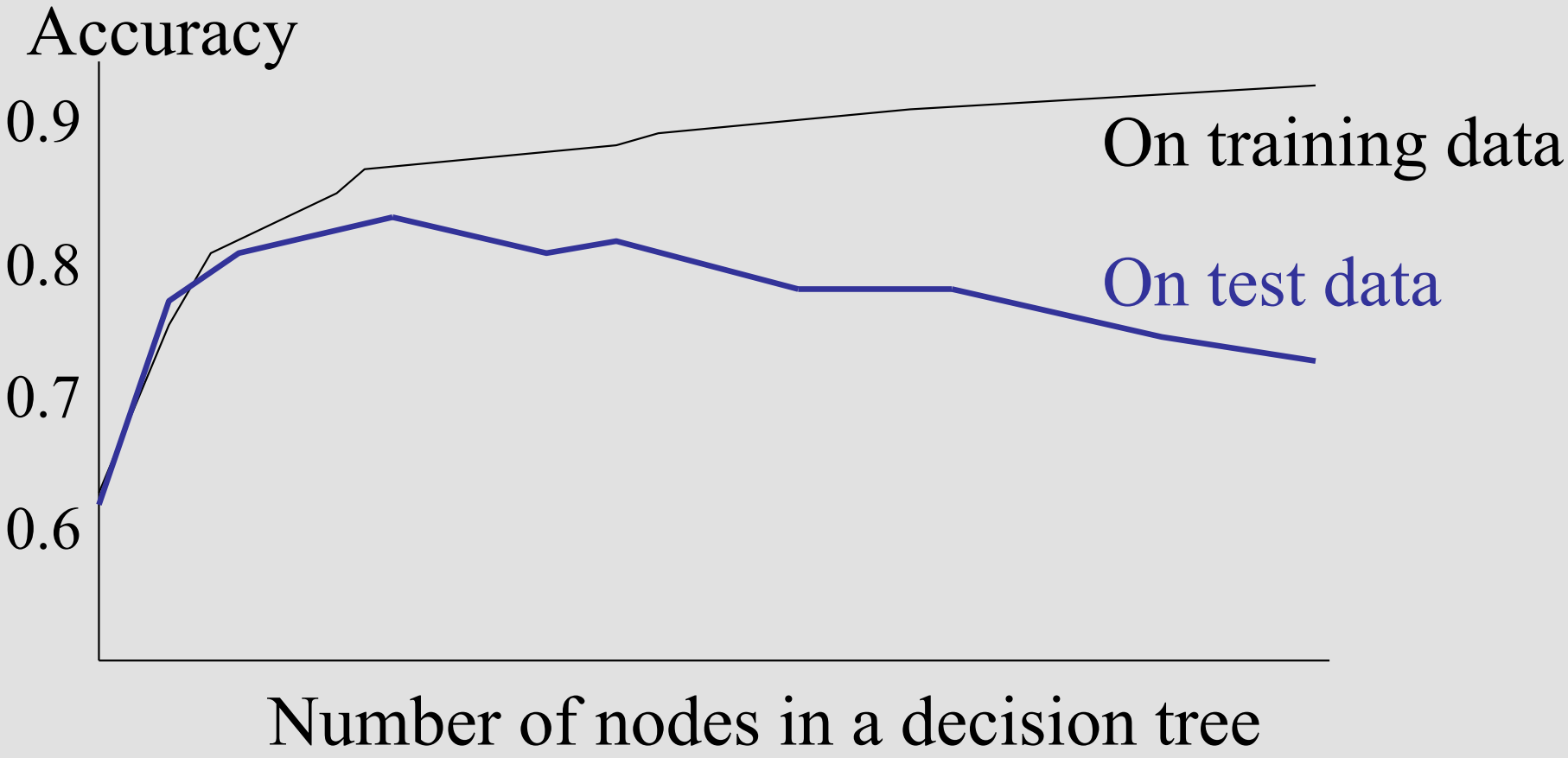
1. **If** all points in D are of the same class
   Then Return
2. **For each** attribute A
   a. Evaluate splits on attribute A
3. Use best split to partition D into D1, D2, …
   a. Split (D1)
   b. Split (D2)
   . . .

recursive call

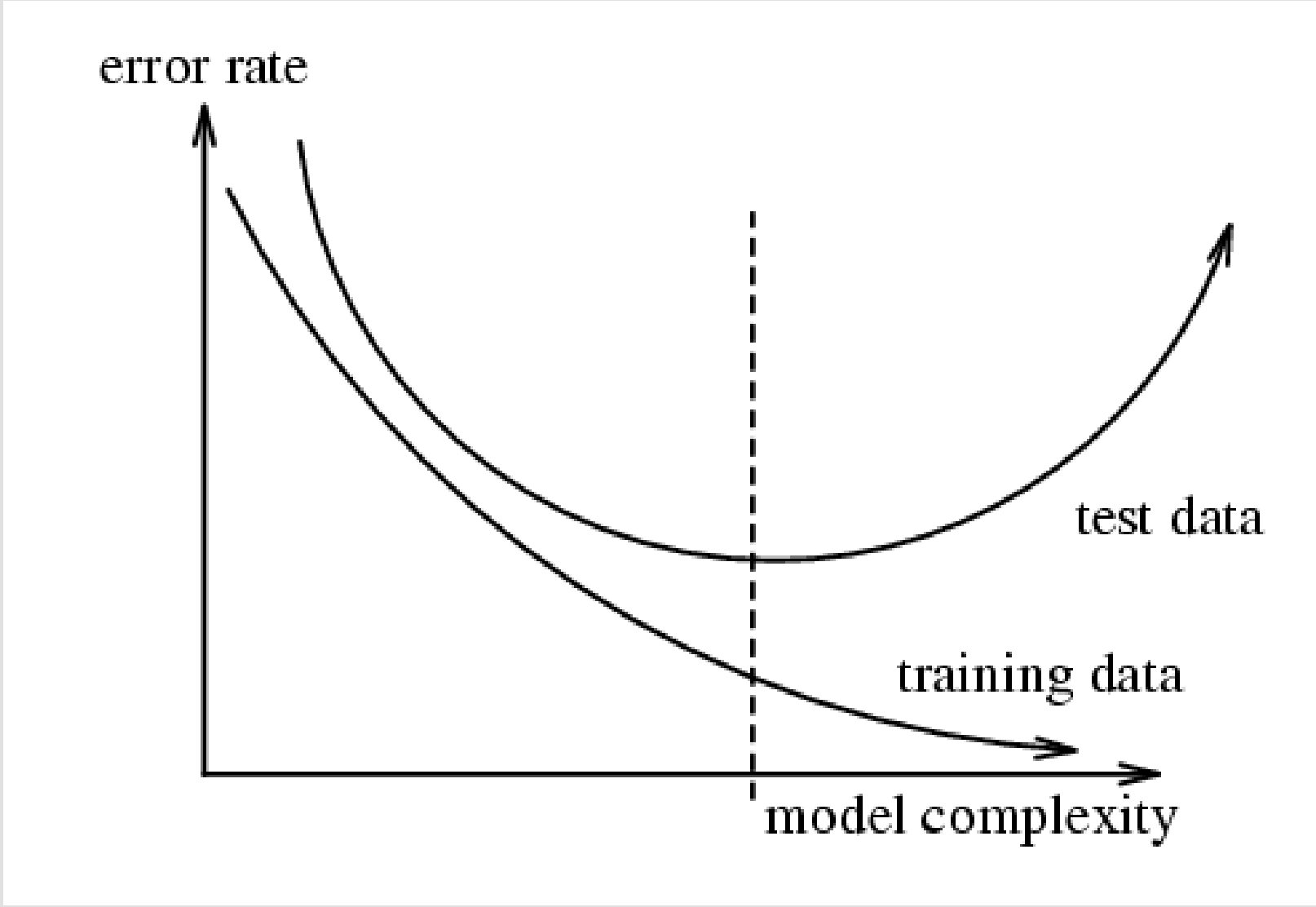
Overfitting (I)

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

→ reduced accuracy for unseen samples



Overfitting (II)



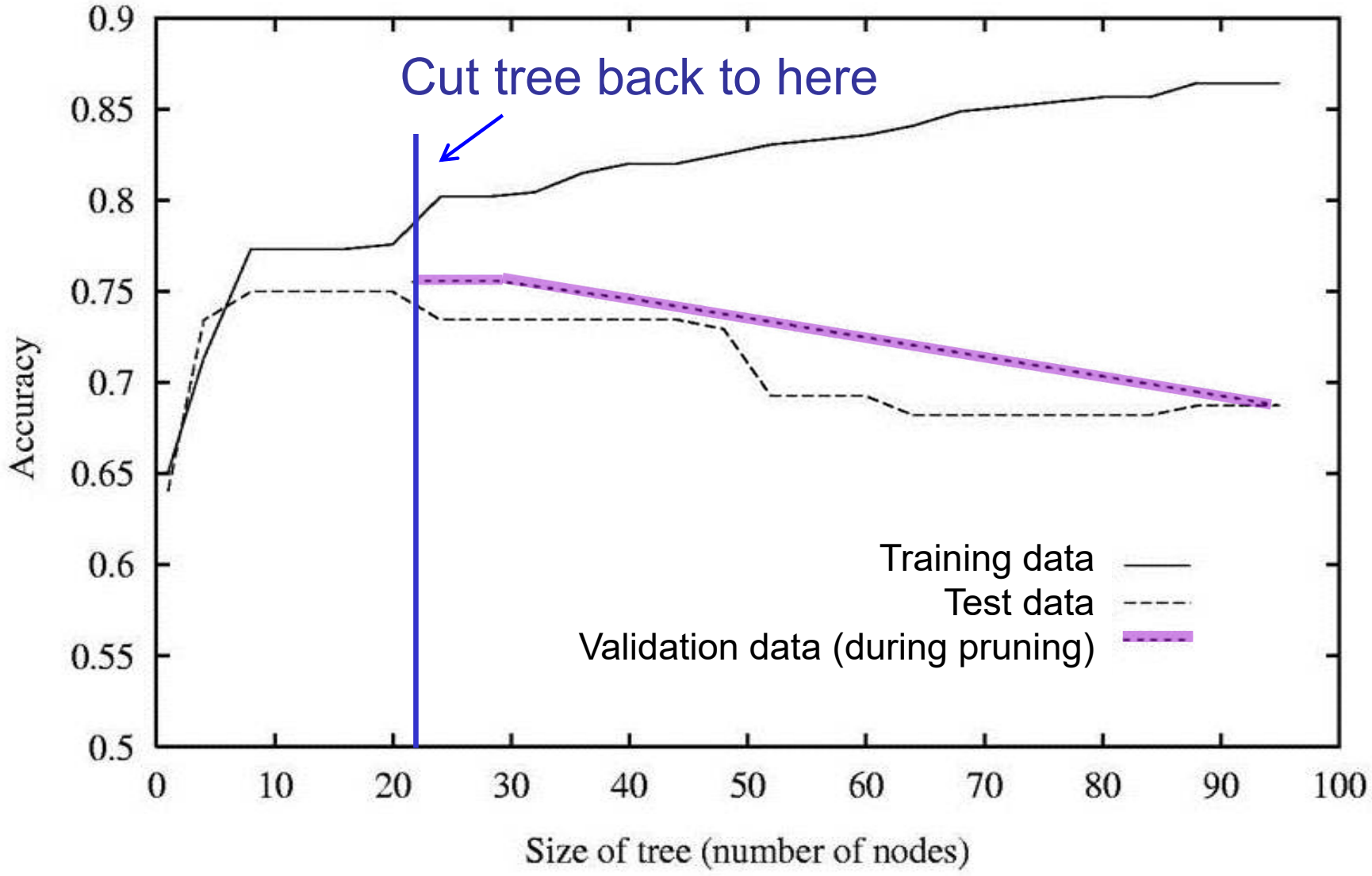
Overfitting (III)

- **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 a 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

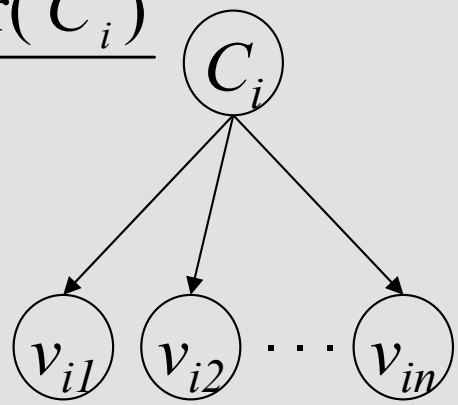
FIT5047 –Intelligent Systems

Naïve Bayes Classifier

Naïve Bayes Classifier (I)

- **Based on Bayes rule**

$$\Pr(C_i \mid V_i) = \frac{\Pr(V_i \mid C_i)\Pr(C_i)}{\Pr(V_i)}$$



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

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

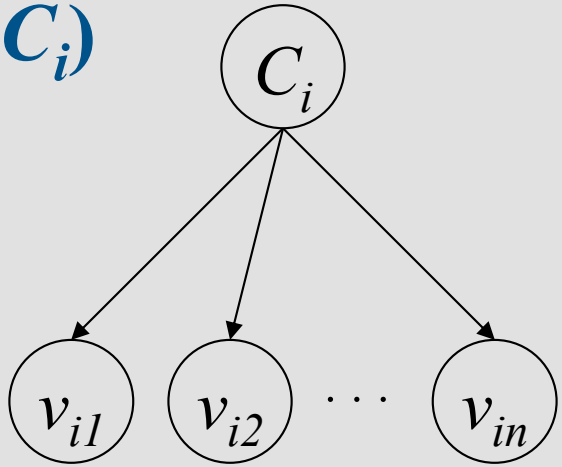
- **Assumes conditional independence of the attribute values for different classes**

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

  - where α is a normalizing constant

Naïve Bayes Classifier (II)

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



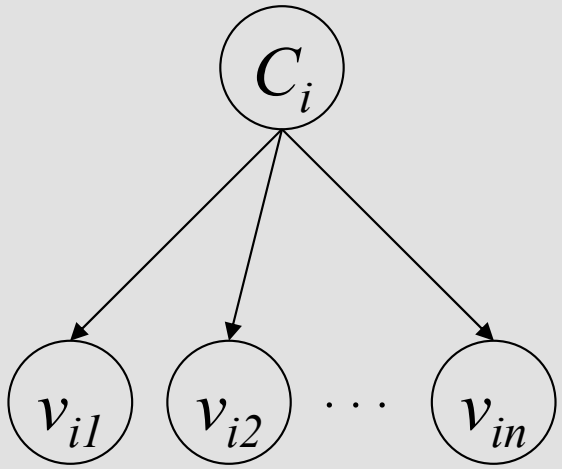
- **BIG questions:**
  - **Prediction/classification: Given that we know the parameters, how do we predict or classify an instance?**
  - **Learning: How do we learn the parameters?**

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

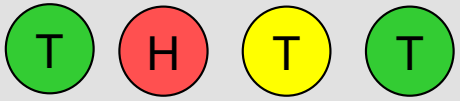
- **Maximum Likelihood Estimator (MLE):**

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

count

  - where m is the number of values for var

- **Example:**



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

$$\frac{\partial}{\partial\theta}[\log \Pr_\theta(D) = 0] \rightarrow \frac{count(var = w_a)}{count\ all\ instances\ of\ var}$$

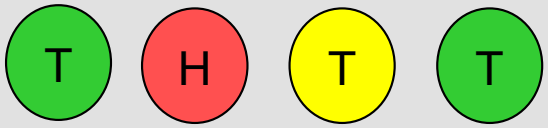
The Sparse Data Problem – Smoothing

- **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



$$\Pr_{ML}(c = H) \cong \frac{|H|}{\sum_1^2 |C_i|} = \frac{1}{1+3} = \frac{1}{4} \qquad \Pr_{ML}(C) = \left\langle \overset{\text{H}}{\frac{1}{4}}, \overset{\text{T}}{\frac{3}{4}} \right\rangle$$

$$\Pr_{EL}(c = H) \cong \frac{|H| + \varepsilon}{\sum_1^2 |C_i + \varepsilon|} = \frac{|H| + \varepsilon}{\sum_1^2 |C_i| + 2\varepsilon} = \frac{1+\varepsilon}{\{1+3\} + 2\varepsilon} = \frac{1+\varepsilon}{4+2\varepsilon}$$

for ε=1 $\qquad \Pr_{EL}(C) = \left\langle \overset{\text{H}}{\frac{1}{3}}, \overset{\text{T}}{\frac{2}{3}} \right\rangle$

Enhancements to Naïve Bayes

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

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_j**
   - use a Normal density function

$$\Pr(v_{iA} = \mathrm{t} | C_j = \mathrm{c}) \cong \Pr(\mathrm{t} - \frac{\varepsilon}{2} \leq v_{iA} \leq \mathrm{t} + \frac{\varepsilon}{2})$$

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

Continuous-valued Attributes – Example

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

$$\Pr(v_{i2} = 85 | C_j = YES) = \varepsilon \frac{1}{\sqrt{2\pi} \cdot 5.23} e^{-\frac{(85-73)^2}{2 \cdot 5.23^2}}$$

$$= 0.0055\varepsilon$$

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

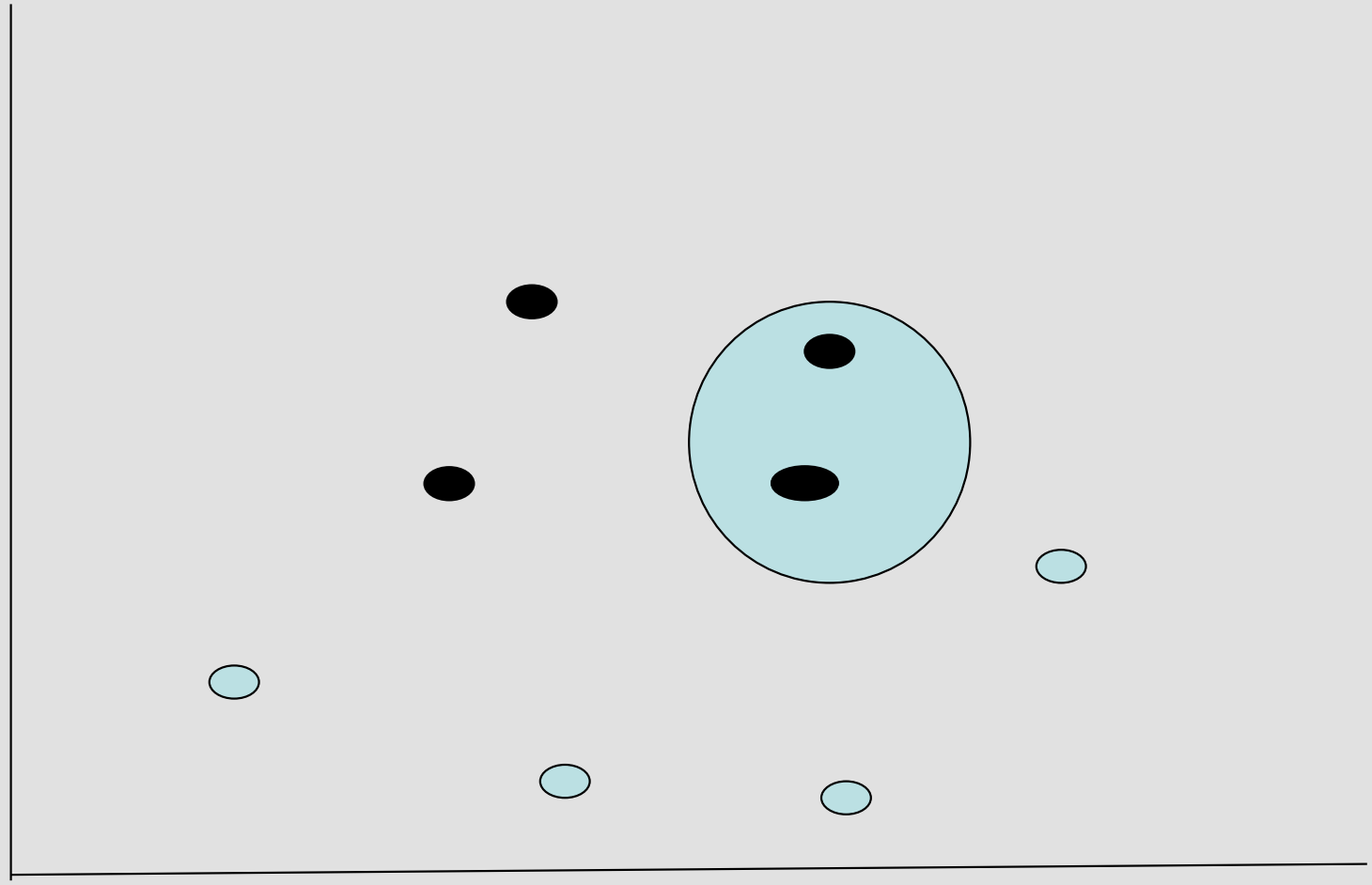
FIT5047 –Intelligent Systems

K Nearest Neighbour (k-NN)

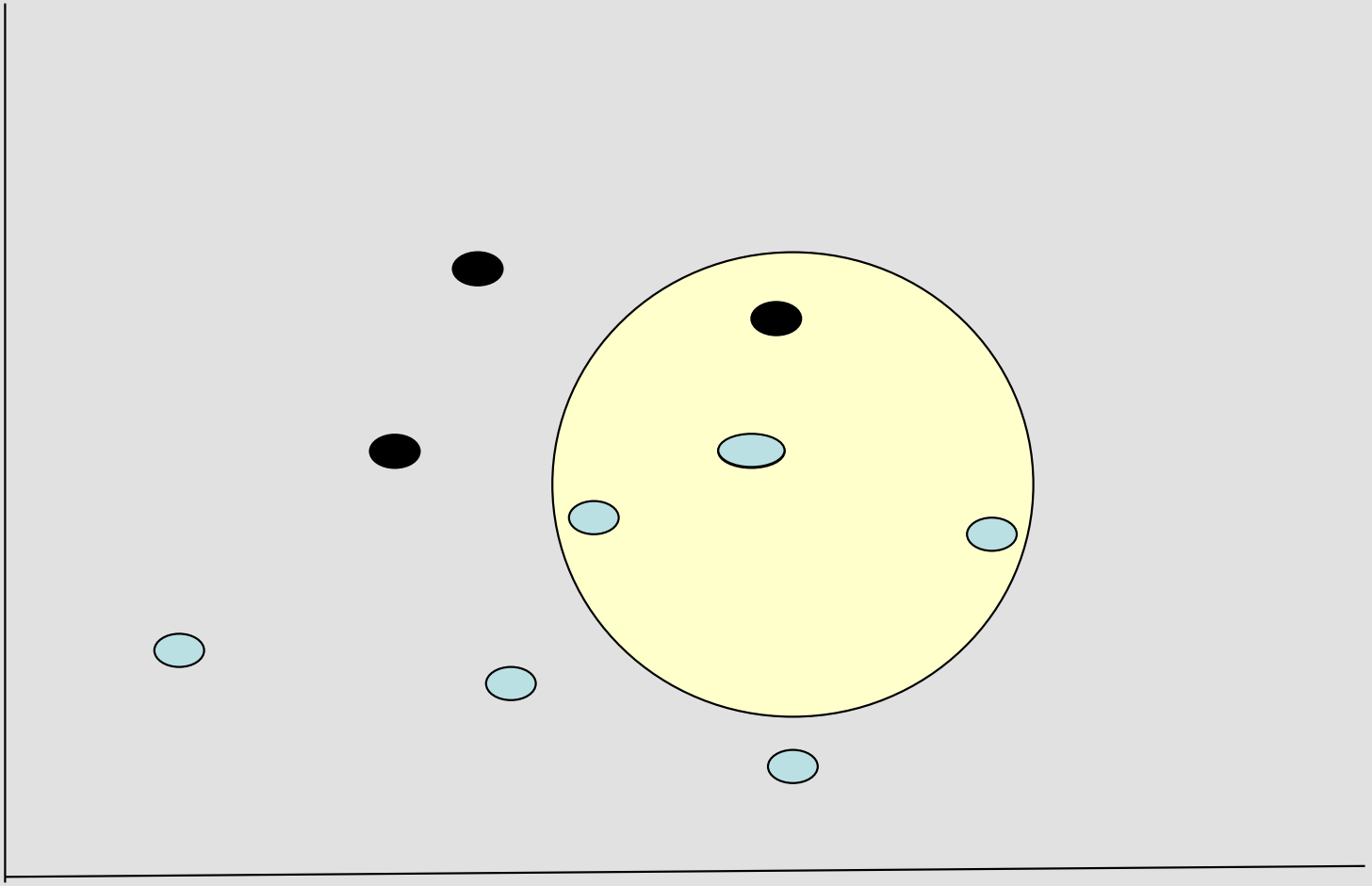
k-Nearest Neighbour (I)

- **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 $(f_{i1}, f_{i2}, ..., f_{in})$**
  - $f_{i,k}$ 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

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

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!

  Need to normalize features

- **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 combine categorical and continuous features**

Parameter Learning

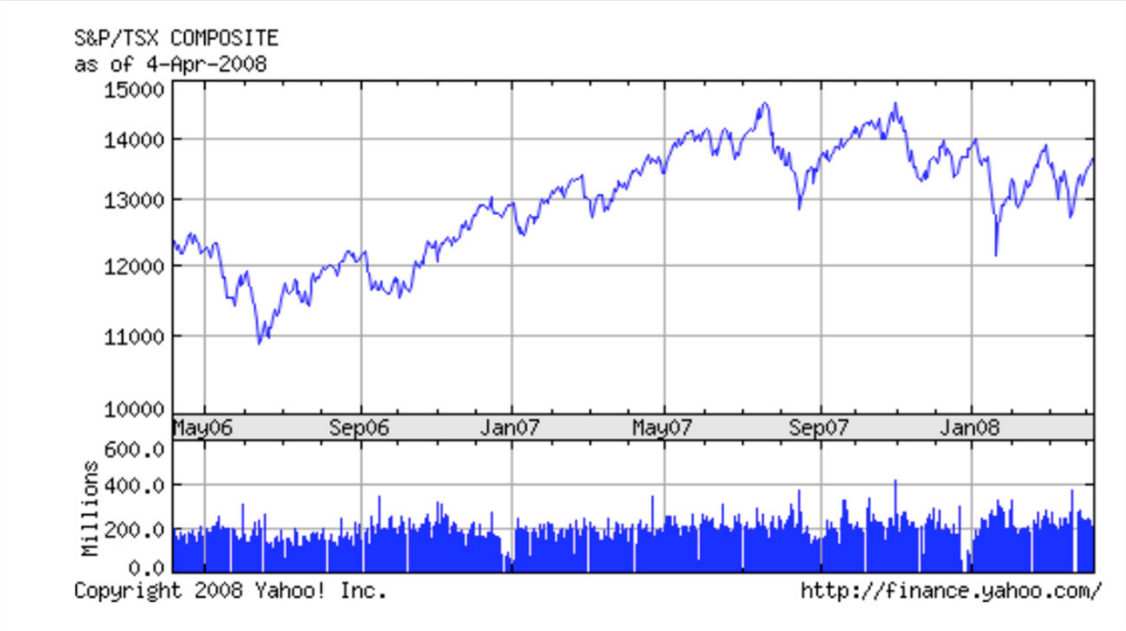
- **Given the training data and the distance function, there is no training**
  - The algorithm memorizes the training data
  - As data comes in, the model size grows
- **In some cases, the distance function is parameterized and its parameters are learnt**

MONASH University
Information Technology

FIT5047 – Intelligent Systems

Regression

Regression



- **Given a training set $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$**
  - $\mathbf{x}_j$ and y_j are continuous → regression
  - assume $\mathbf{x}_j \in R^m, y_j \in R$
- **E.g., y_j is stock price, $\mathbf{x}_j$ contains company profit, debt, cash flow, …**

Univariate Linear Regression

- **Given a training set $\{(x_1, y_1), \ldots, (x_N, y_N)\}$**
  - assume $x_j \in R, y_j \in R$

$$h_{\mathbf{w}}(x) = w_1 x + w_0$$

  - we want to learn the parameters w_0 and w_1

- **Loss: Square of the difference between the true and predicted target value for x_j**

$$Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(x_j))^2 = \sum_{j=1}^{N} (\underbrace{y_j}_{\text{truth}} - \underbrace{(w_1 x_j + w_0)}_{\text{prediction}})^2$$

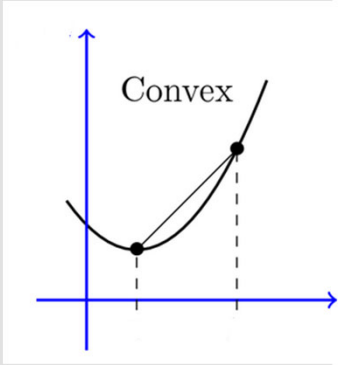
  - where *N* is the number of samples

Learning the Parameters

- **Find $\mathbf{w}^*$ that minimizes the loss function**

$$\mathbf{w}^* = \text{argmin}_{\mathbf{w}} Loss(h_{\mathbf{w}})$$

  - $Loss(h_{\mathbf{w}})$ is a convex function, so $\mathbf{w}^*$ is unique



- **How to find $\mathbf{w}^*$?**
  - Set the derivatives to zero: $\frac{\partial}{\partial w_i} Loss(h_{\mathbf{w}}) = 0$
    - > For a linear function $h_{\mathbf{w}} = w_1 x + w_0$ the solution is:

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

  - Use an iterative algorithm such as ***gradient descent***

Univariate Linear Regression – Example (I)

$x_1, y_1 \quad x_2, y_2 \quad x_3, y_3$

- **Training data:** $\{(1,3),(2.1,0.5),(-5,6.2)\}$
- **Linear regression model:** $h_{\mathbf{w}} = w_1 x + w_0$
- **Loss function:**

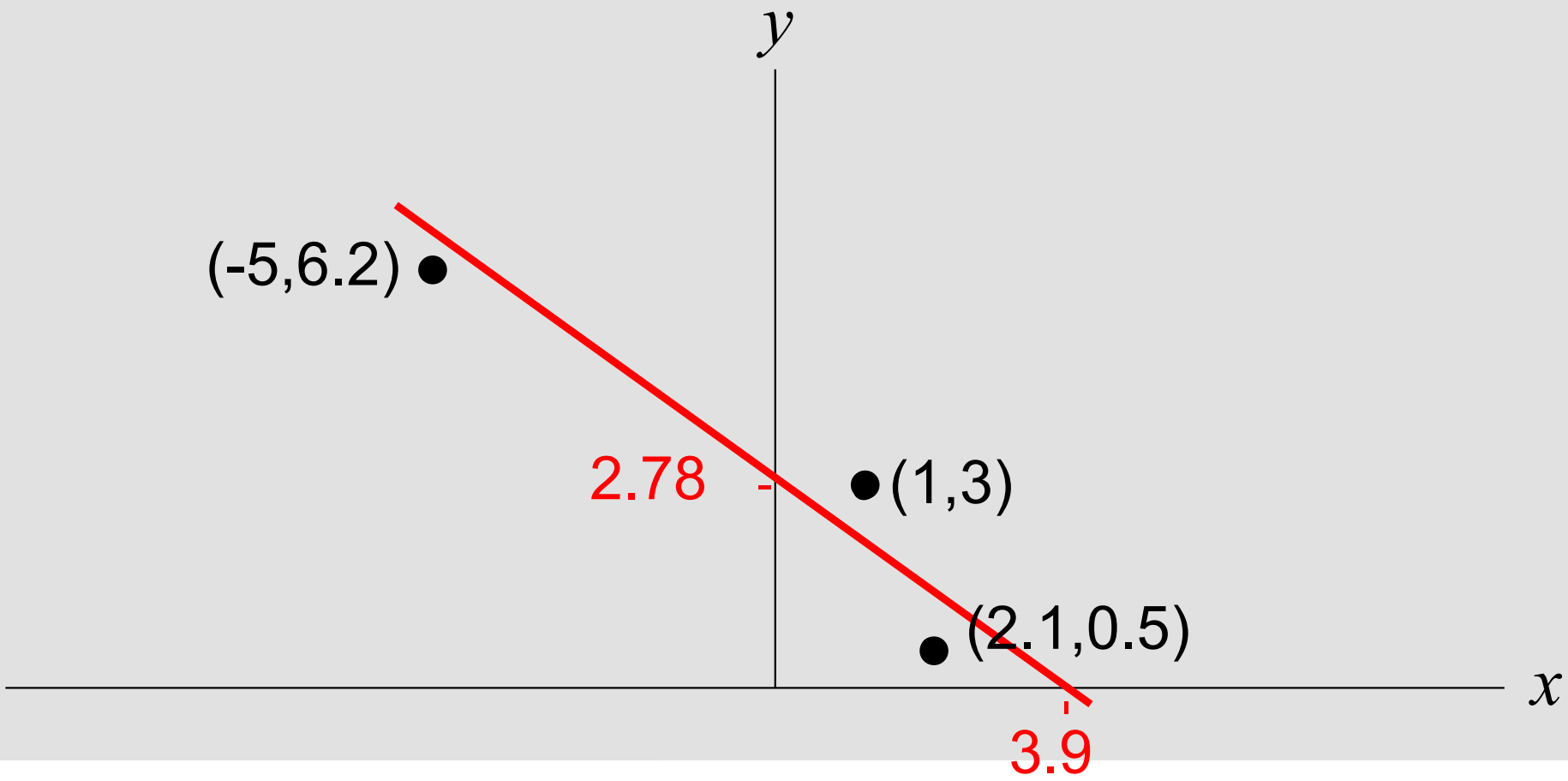
$$Loss(h_{\mathbf{w}}) = (3 - (1w_1 + w_0))^2 + (0.5 - (2.1w_1 + w_0))^2 + (6.2 - (-5w_1 + w_0))^2$$

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

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

Univariate Linear Regression – Example (II)

- **Training data:** {(1,3),(2.1,0.5),(−5,6.2)}
- **Function:** $h_{\mathbf{w}} = -0.712x + 2.78$



Gradient Descent (GD) Algorithm

1. **$\mathbf{w}^0 \leftarrow$ any point in the parameter space; t = 1**
2. **Until convergence ($|| \mathbf{w}^t - \mathbf{w}^{t-1} || < \varepsilon$) do**
   a. For each w_i in **w** do

$$w_i^t \leftarrow w_i^{t-1} - \alpha \frac{\partial}{\partial w_i} Loss(h_{\mathbf{w}})$$

Learning rate

Index in vector **w**

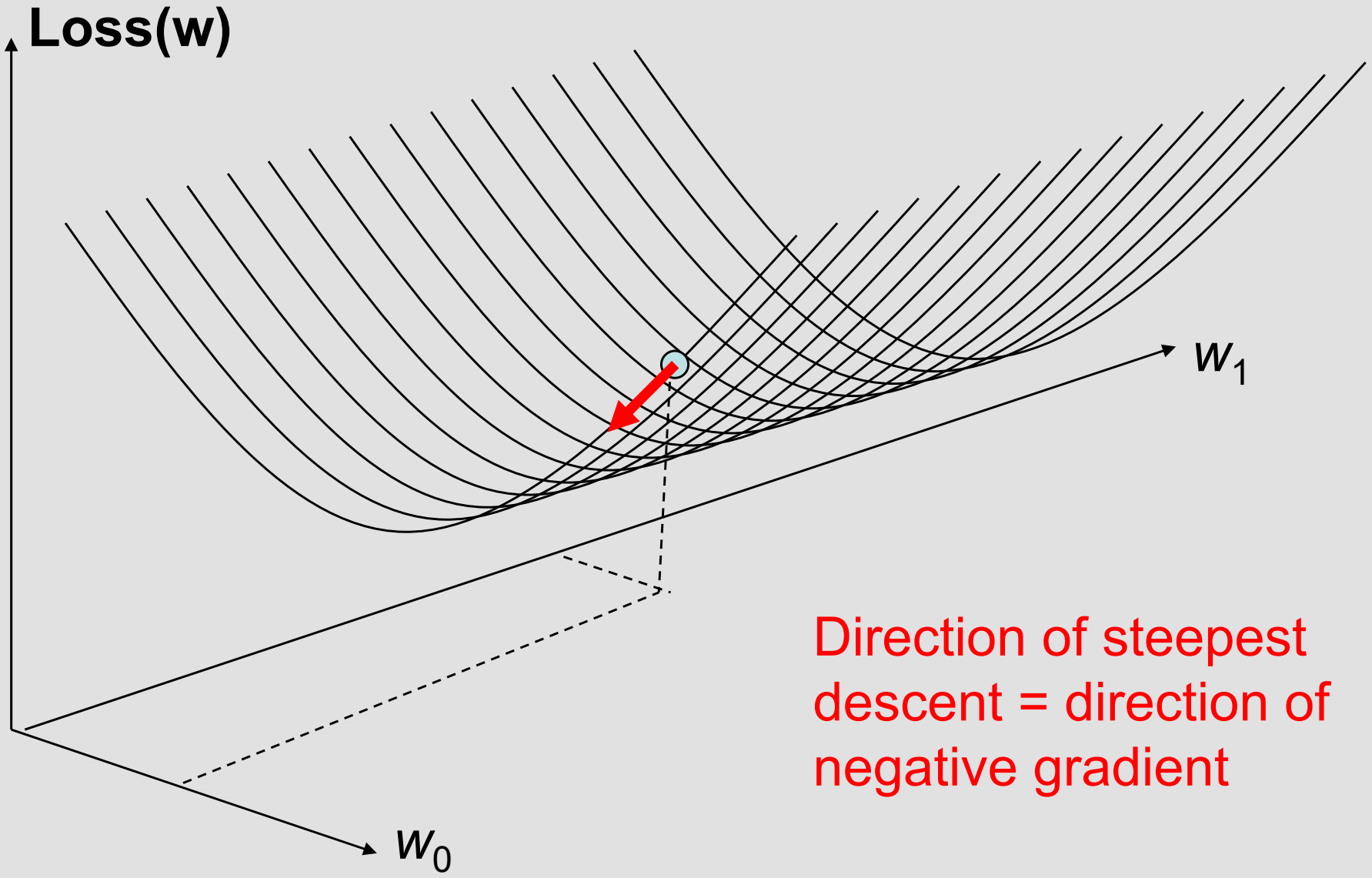
   b. $t \leftarrow t + 1$

where

$$Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} \left(y_j - h_{\mathbf{w}}(x_j)\right)^2$$

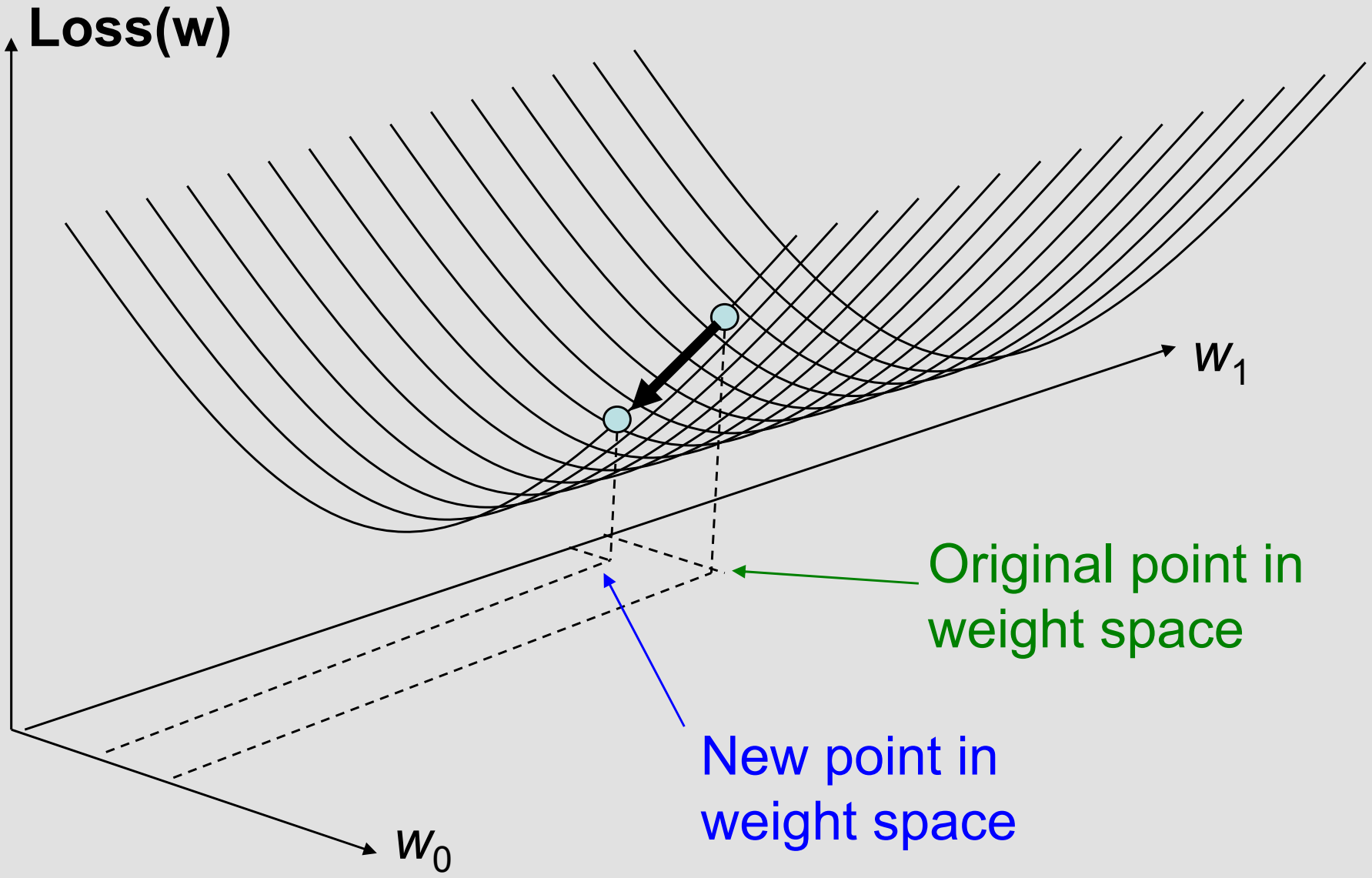
$$\frac{\partial}{\partial w_i} Loss(h_{\mathbf{w}}) = -2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i} h_{\mathbf{w}}(x)$$

Illustration of Gradient Descent (I)



Direction of steepest descent = direction of negative gradient

Illustration of Gradient Descent (II)



Univariate Linear Regression – GD

$$\frac{\partial}{\partial w_0} Loss(h_{\mathbf{w}}) = -2(y - h_{\mathbf{w}}(x))$$

$$\frac{\partial}{\partial w_1} Loss(h_{\mathbf{w}}) = -2(y - h_{\mathbf{w}}(x)) \times x$$

Updates for one training example (x_j, y_j):

- $w_0^t \leftarrow w_0^{t-1} + \alpha \left(y_j - h_w(x_j)\right)$
- $w_1^t \leftarrow w_1^{t-1} + \alpha \left(y_j - h_w(x_j)\right) \times x_j$

Updates for *N* training examples:

- $w_0^t \leftarrow w_0^{t-1} + \alpha \sum_{j=1}^{N} \left(y_j - h_w(x_j)\right)$
- $w_1^t \leftarrow w_1^{t-1} + \alpha \sum_{j=1}^{N} \left(y_j - h_w(x_j)\right) \times x_j$

Univariate Linear Regression – Example (III)

- **Training data:** {(1,3),(2.1,0.5),(−5,6.2)}
- **Linear regression model:** $h_{\mathbf{w}} = w_1 x + w_0$
- **Loss function:**

$$Loss(\boldsymbol{w}) = (3 - (1w_1 + w_0))^2 + (0.5 - (2.1w_1 + w_0))^2 + (6.2 - (-5w_1 + w_0))^2$$

- **The gradient:**

$$\nabla_{\mathbf{w}} Loss(\mathbf{w}) = \begin{bmatrix} \frac{\partial Loss}{\partial w_0} \\ \frac{\partial Loss}{\partial w_1} \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{3} [-2(y_j - h_w(x_j))] \\ \sum_{j=1}^{3} [-2(y_j - h_w(x_j)) \times x_j] \end{bmatrix}$$

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

Multivariable Linear Regression – GD

$$h_{\mathbf{w}}(\mathbf{x}_j) = w_0 + w_1 x_{j,1} + \cdots + w_m x_{j,m} = w_0 + \sum_{i=1}^{m} w_i x_{j,i}$$

We set $x_{j,0} = 1$ for all the samples $j=1, \ldots, N$

$$h_{\mathbf{w}}(\mathbf{x}_j) = \mathbf{w} \cdot \mathbf{x}_j = \sum_{i=0}^{m} w_i x_{j,i}$$

For one training example $(\mathbf{x}_j, y_j)$:

$$w_i^t \leftarrow w_i^{t-1} + \alpha \left(y_j - h_{\mathbf{w}}(\mathbf{x}_j) \right) \times x_{j,i}$$

For *N* training examples:

$$w_i^t \leftarrow w_i^{t-1} + \alpha \sum_{j=1}^{N} \left(y_j - h_{\mathbf{w}}(\mathbf{x}_j) \right) \times x_{j,i}$$

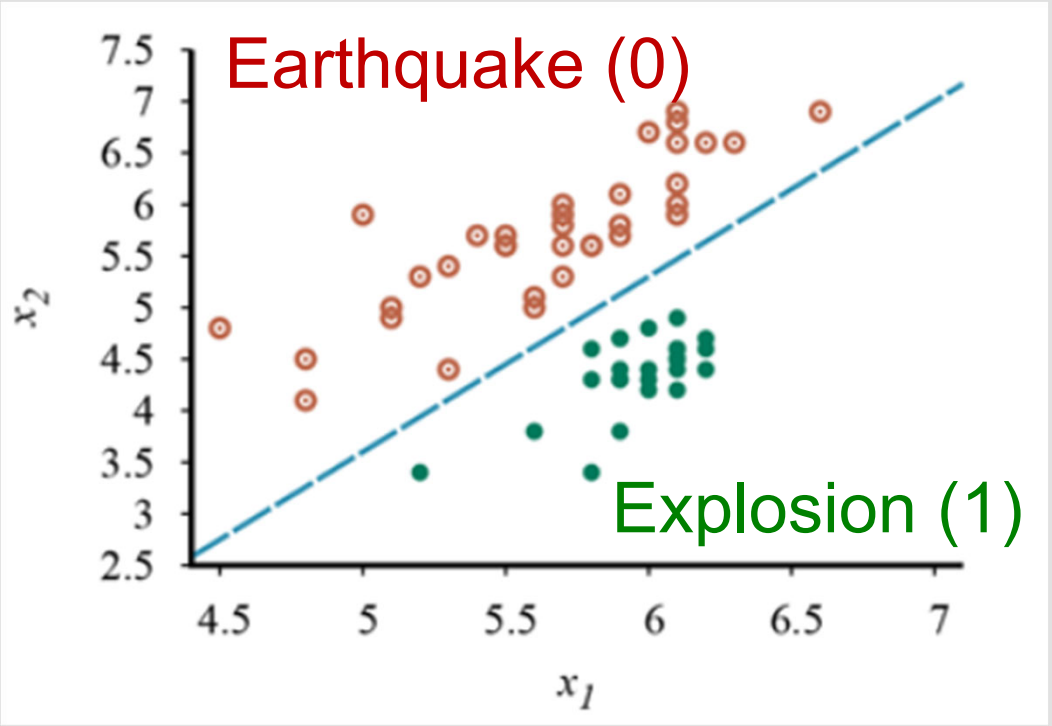
MONASH University
Information Technology

FIT5047 – Intelligent Systems

Classifiers with Thresholds

Linear Classifier (I)

- **The regressor is a *decision boundary* between two *linearly separable* classes**
  - A linear decision boundary is called a ***linear separator***
- **Example:**
  - **Linear separator**
    $-4.9 + 1.7x_1 - x_2 = 0$
  - **Classification**
    - Explosions:
      $x_2 \leq -4.9 + 1.7x_1$
    - Earthquakes
      $x_2 > -4.9 + 1.7x_1$



Linear Classifier (II)

- Using the convention of a dummy input $x_0 = 1$, we can write the classification hypothesis as

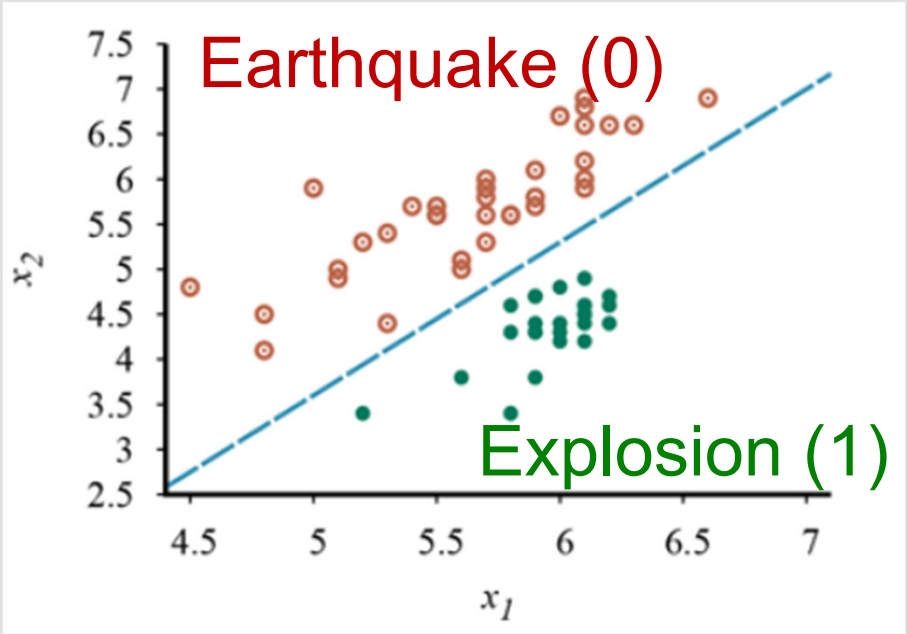
$$h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

where

$$\mathbf{w} = \langle -4.9, 1.7, -1 \rangle$$

to represent

$$-4.9 + 1.7x_1 - x_2 = 0$$



- Alternative representation

$$h_{\mathbf{w}}(\mathbf{x}) = Threshold(\mathbf{w} \cdot \mathbf{x})$$

$$Threshold(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Learning the Parameters

Objective: choose the weights w to minimize ***loss***

- We cannot perform gradient descent
- There is a weight update rule that converges on the solution, *provided the data are linearly separable*
  - For a single example $(\mathbf{x},y)$, where y is 0 or 1, we have

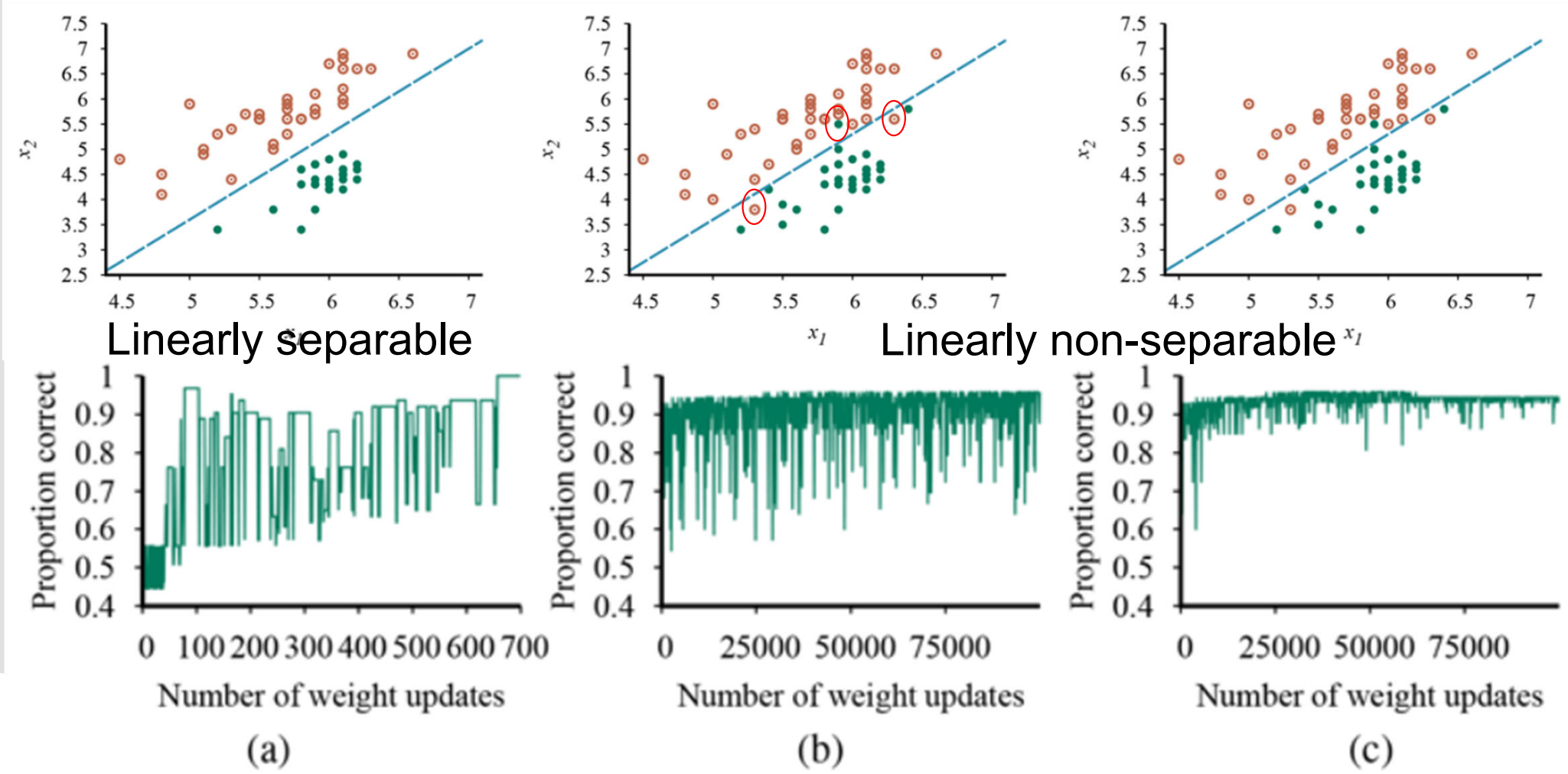
$$w_i \leftarrow w_i + \alpha(y - h_{\mathbf{w}}(\mathbf{x})) \times x_i$$

	x_i **is positive**	x_i **is negative**
$y = 1$ & $h_{\mathbf{w}} = 0$	w_i is increased	w_i is decreased
$y = 0$ & $h_{\mathbf{w}} = 1$	w_i is decreased	w_i is increased

  - This rule is called the **perceptron learning rule**
  - It is applied one example at a time, choosing examples at random

Learning a Classifier – Training Curves

- A training curve measures the classifier performance on a fixed training set as the learning process proceeds one update at a time



Linear Classifier – Disadvantages

- The hypothesis $h_{\mathbf{w}}(\mathbf{x})$ is not differentiable, and is a discontinuous function of its inputs and its weights
  - ➔ This makes learning with the perceptron rule unpredictable
- The linear classifier always announces a completely confident prediction of 1 or 0

MONASH University
Information Technology

FIT5047 – Intelligent Systems

Logistic Regression

Logistic Function

$$Logistic(z) = \frac{1}{1 + e^{-z}}$$

- For the threshold function we get:

$$h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$

  where $h_{\mathbf{w}}(\mathbf{x})$ can be interpreted as the probability of sample $\mathbf{x}$ belonging to the class labeled 1

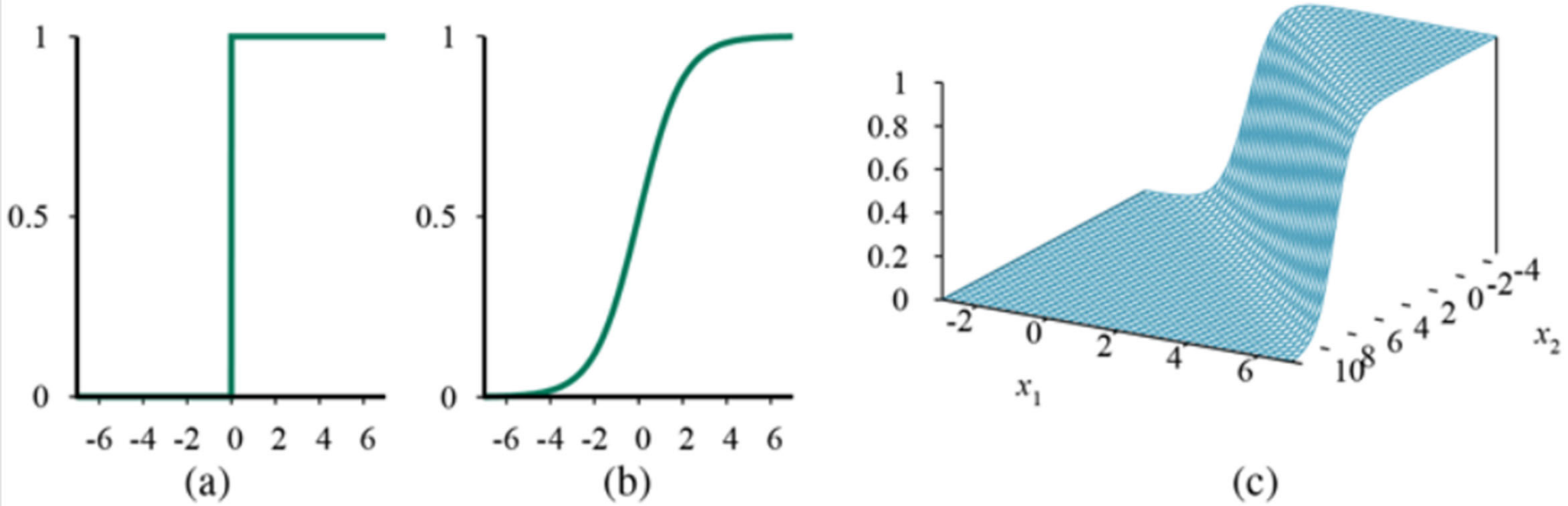
- The process of fitting the weights of this model to minimize loss on a data set is called ***logistic regression***
  - Calculated by gradient descent

Logistic and Linear Thresholds

0/1 threshold

logistic (sigmoid) function

logistic regression function for the example



Learning the Parameters (GD)

$$h_w(\boldsymbol{x}) = Logistic(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$

- **Gradient of the Loss function**

vector

$$\nabla_{\mathbf{w}} Loss(\mathbf{w}) = \left[\frac{\partial}{\partial w_i} Loss(\mathbf{w})\right] = \left[\frac{\partial}{\partial w_i} (y - h_{\mathbf{w}}(\mathbf{x}))^2\right]$$

- **Adjusting the weights**

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} Loss(\mathbf{w})$$

$$w_i \leftarrow w_i + \alpha \underbrace{\underbrace{(y - h_w(\boldsymbol{x}))}_{Error} \times \underbrace{h_w(\boldsymbol{x}) \times (1 - h_w(\boldsymbol{x}))}_{g'(\mathbf{w} \cdot \mathbf{x})} \times x_i}_{\Delta}$$

Learning rate

Learning Weights – Earthquake Example

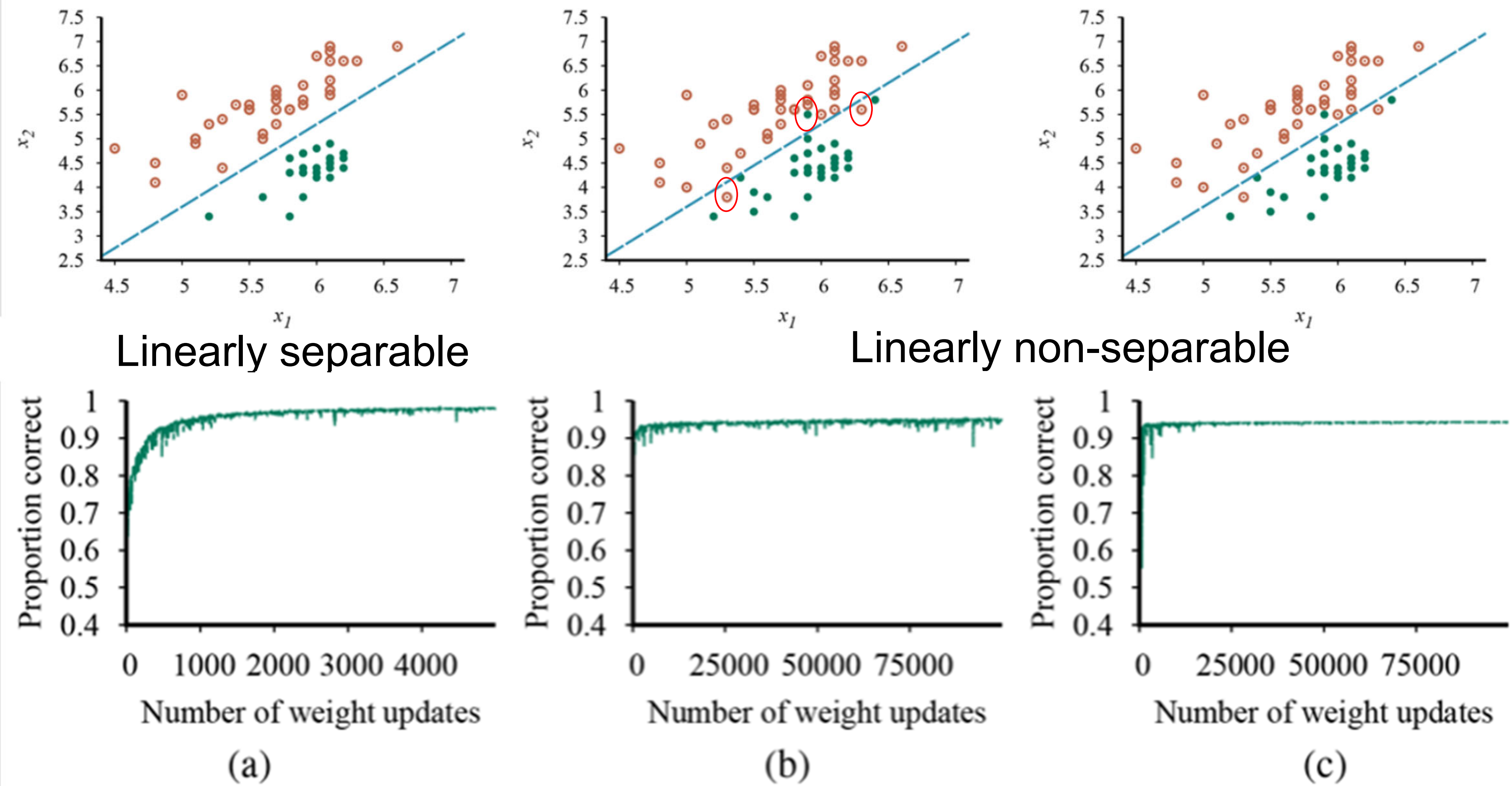
$$h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$

$$w_i \leftarrow w_i + \underbrace{\alpha(y - h_{\mathbf{w}}(\mathbf{x})) \times h_{\mathbf{w}}(\mathbf{x}) \times (1 - h_{\mathbf{w}}(\mathbf{x}))}_{\Delta} \times x_i$$

$\alpha = 0.2$

t	w_0	w_1	w_2	x_0	x_1	x_2	$\mathbf{w} \cdot \mathbf{x}$	$h_{\mathbf{w}}(\mathbf{x})$	y	Δ
0	0	0	0	1	5.2	3.4	0	0.5	1	0.025
1	0.025	0.13	0.085	1	5	6	1.185	0.765846	0	-0.02747
2	-0.00247	-0.00734	-0.0798	1	6	6.6	-0.57319	0.360502	0	-0.01662
3	-0.01909	-0.10707	-0.18951	1	6	4.9	-1.59009	0.169371	1	0.023371

Learning the Classifier – Training Curves



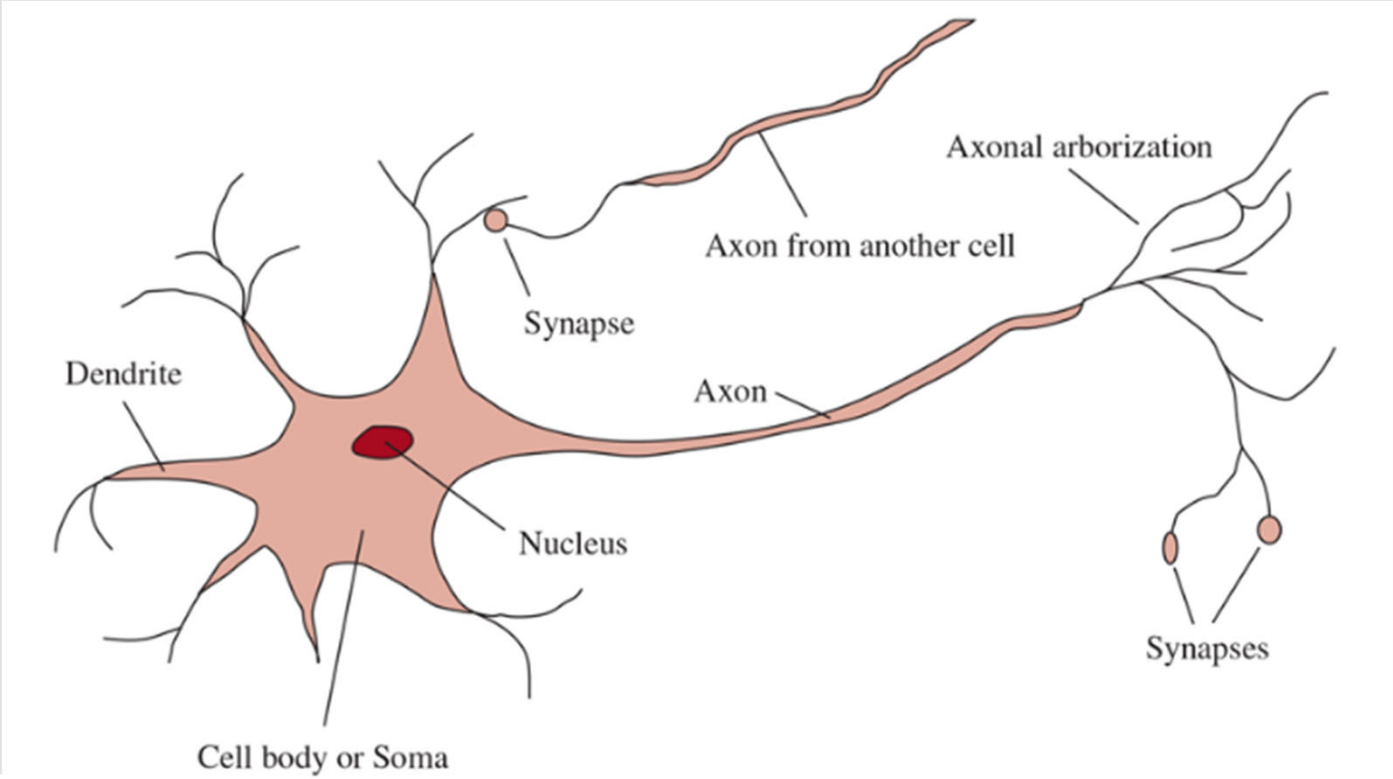
MONASH University
Information Technology

FIT5047 –Intelligent Systems

Artificial Neural Networks

Brains

- 10^{11} neurons of > 20 types
- 10^{14} synapses
- 1ms–10ms cycle time
- Signals are noisy “spike trains” of electrical potential

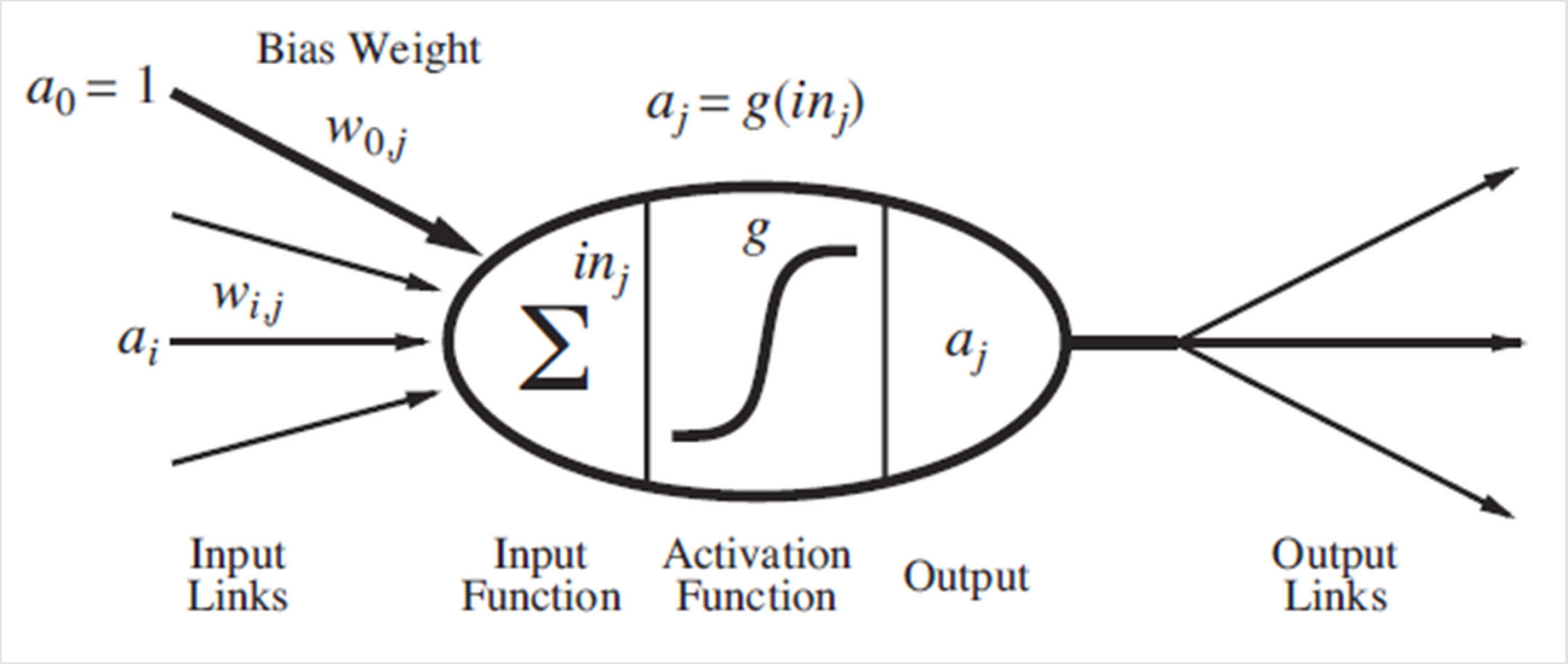


McCulloch-Pitts (1943) – Simplified Neuron

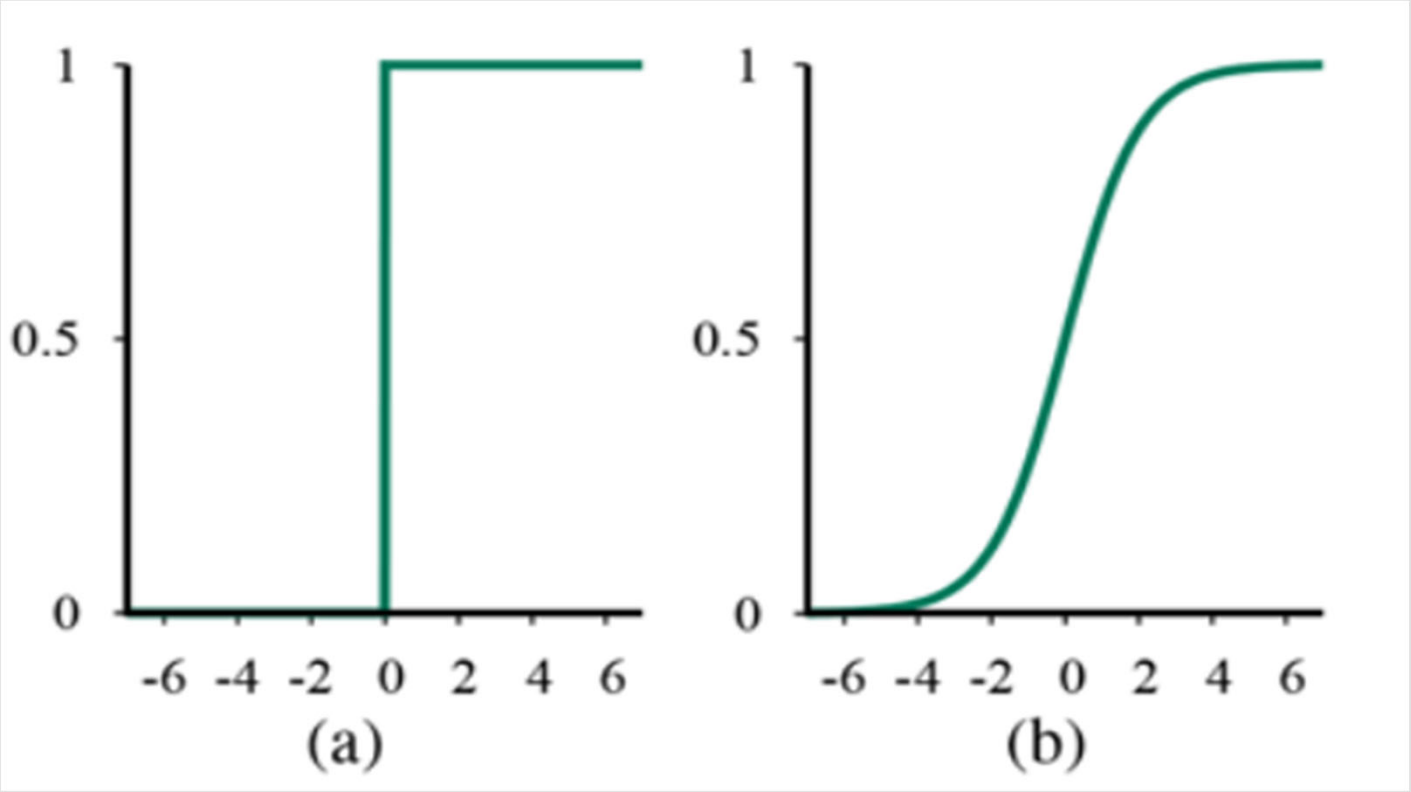
- Output is a function of the inputs:

$$a_j = g(in_j) = g(\mathbf{w}_j \cdot \mathbf{a}) = g\left(\sum_{i=0}^{n} w_{i,j}\, a_i\right)$$

activation function



Activation Functions



$$g(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{else} \end{cases}$$

Perceptron

$$g(x) = \frac{1}{1 + e^{-x}}$$

Sigmoid perceptron

Network Structure (I)

- **Feed-forward network**
  - Has connections only in one direction
  - Represents a function of its current input
    → Has no internal state
  - May have several outputs
  - Types
    - Single-layer perceptrons
    - Multi-layer networks
      - Arranged in **layers** – each unit receives input only from units in the immediately preceding layer
      - Have one or more **layers** of **hidden units** that are ***not*** connected to outputs of the network

Network Structure (II)

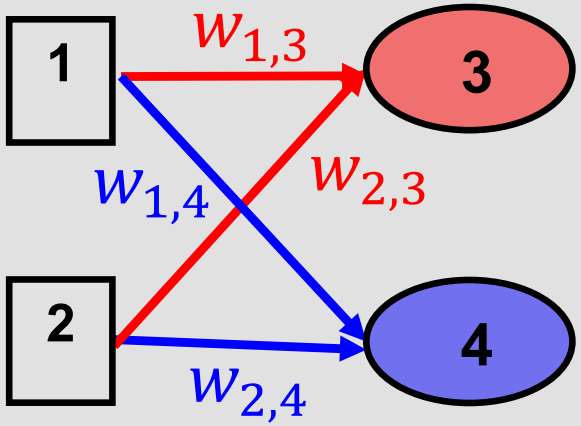
- **Recurrent network**
  - Feeds its outputs back into its own inputs
    - the network's response to an input depends on its initial state, which may depend on previous inputs
      → Can support short-term memory
  - Has an internal state

Single-layer Feed-forward NN (Perceptrons)

- **A network with all the inputs connected directly to the outputs**
  - When there are *m* outputs
    - Each output unit corresponds to a separate function
    - Each weight affects only one output
      → there are *m* training processes
- **Example:**

$$a_3 = w_{1,3}a_1 + w_{2,3}a_2$$

$$a_4 = w_{1,4}a_1 + w_{2,4}a_2$$



Example: Spam Filter (I)

- **Input: email**
- **Output: SPAM/HAM**
- **Setup:**
  - Get a large collection of sample emails, each labeled "SPAM" or "HAM"
    - The samples have been labeled
  - **Goal**: learn to predict labels for new, future emails
- **Features: The attributes used to make a prediction**
  - words in the email's text

Dear Sir.
First, I must solicit your confidence in this transaction, this is by virture of its nature as being utterly confidencial and top secret. …

TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.
99 MILLION EMAIL ADDRESSES FOR ONLY $99

Ok, Iknow this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

Example: Spam Filter (II)

- **Four features**
  - BIAS (always has value 1)
  - free (# occurrences of “free”)
  - money (# occurrences of “money”)
  - the (# occurrences of “the”)

$$\boldsymbol{w} \cdot f(\boldsymbol{x})$$

$$\sum_i w_i \cdot f_i(x)$$

x	$f(\mathbf{x})$	w
“free money”	BIAS : 1 free : 1 money : 1 the : 0 ...	BIAS : -3 free : 4 money : 2 the : 0 ...

$$\begin{aligned} &(1)(-3) + \\ &(1)(4) + \\ &(1)(2) + \\ &(0)(0) + \\ &\cdots \\ &= 3 \end{aligned}$$

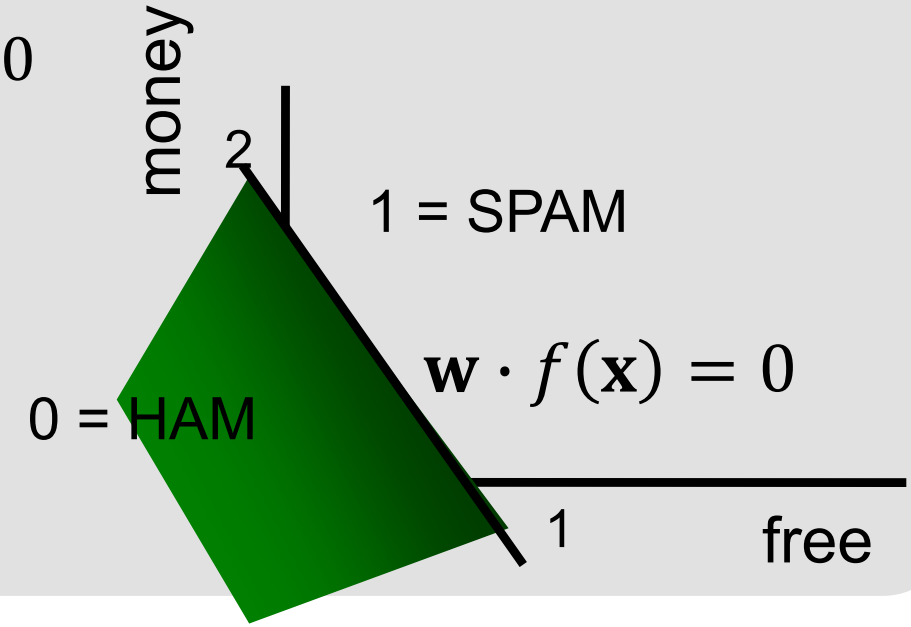
Prediction: Binary Decision Rule

- **In the space of feature vectors**
  - Any weight vector is a hyperplane
    - One side will be class 1
    - Other side will be class 0
- **Example: Spam filter**

$\mathbf{w}$

BIAS :	-3
free :	4
money :	2
the :	0
...	

$$g(x) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \boldsymbol{f} \geq 0 \\ 0 & \text{else} \end{cases}$$



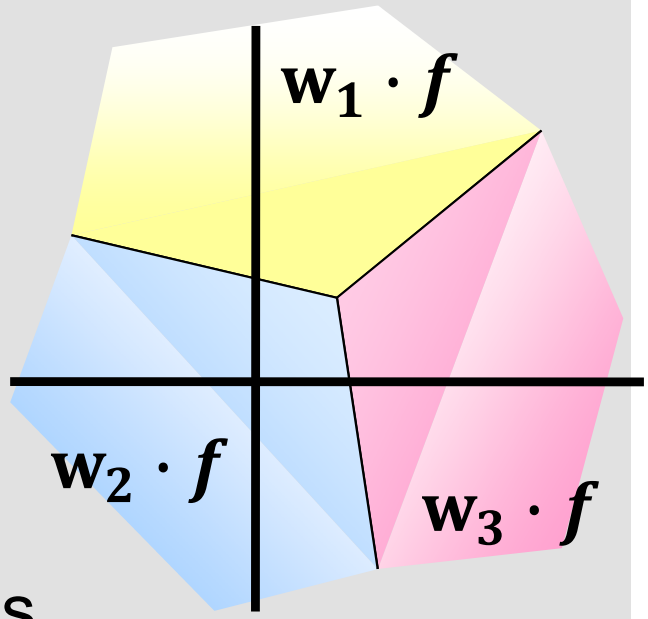
Prediction: Multiclass Decision Rule

- **For more than two classes**
  - Have a weight vector for each class
  - Calculate the activation for each class

$$\text{activation}_w(x, c) = \sum_i w_{c,i} \cdot f_i(x)$$

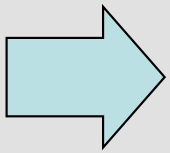
  - The class with the highest activation wins

$$c = \arg\max_c \ (\text{activation}_w(x, c))$$



Example: Multiclass Decision Rule

"win the vote"



```
BIAS   :  1
win    :  1
game   :  0
vote   :  1
the    :  1
...
```

w_{SPORTS}

```
BIAS   : -2
win    :  4
game   :  4
vote   :  0
the    :  0
...
```

$w_{POLITICS}$

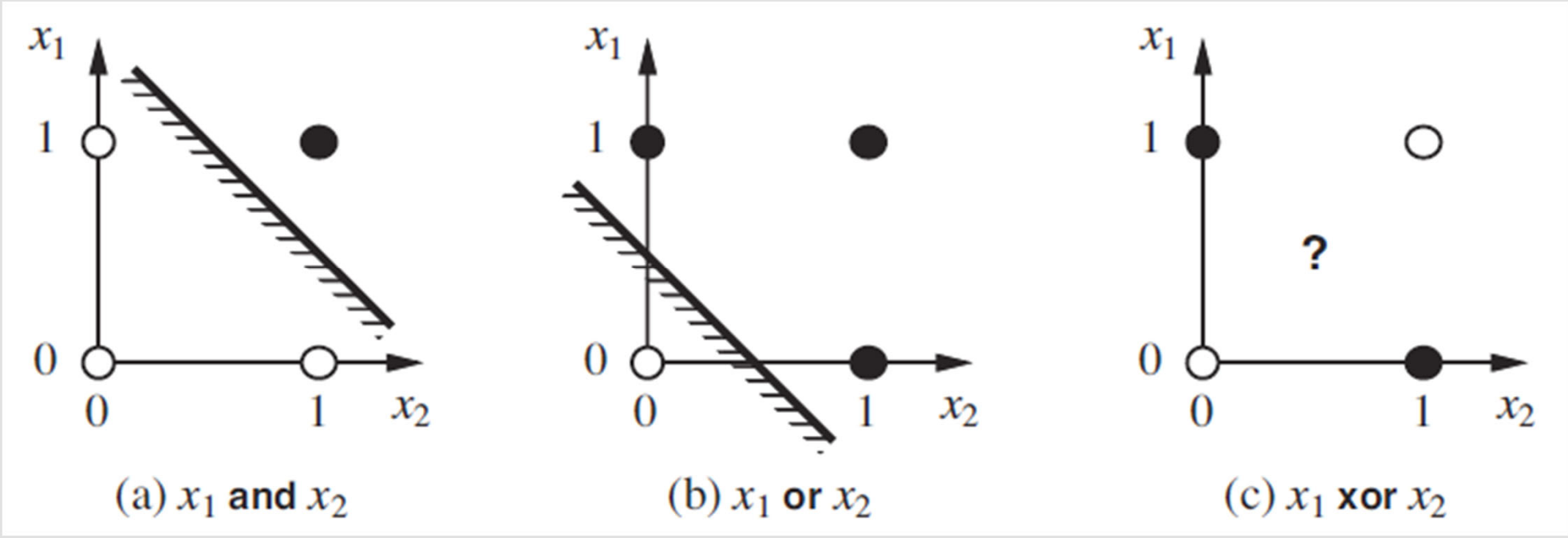
```
BIAS   :  1
win    :  2
game   :  0
vote   :  4
the    :  0
...
```

w_{TECH}

```
BIAS   :  2
win    :  0
game   :  2
vote   :  0
the    :  0
...
```

Expressiveness of Perceptrons

- **Can learn only linearly separable functions!**
  - Examples – AND, OR, NOT, majority, but not XOR



Learning Perceptrons

- **Each weight affects only one output**
  → there are *m* training processes
- **Depending on the activation function, the training process will be either**
  - **perceptron learning rule**

$$w_i \leftarrow w_i + \alpha(y - h_{\mathbf{w}}(\mathbf{x})) \times x_i$$

α is the learning rate

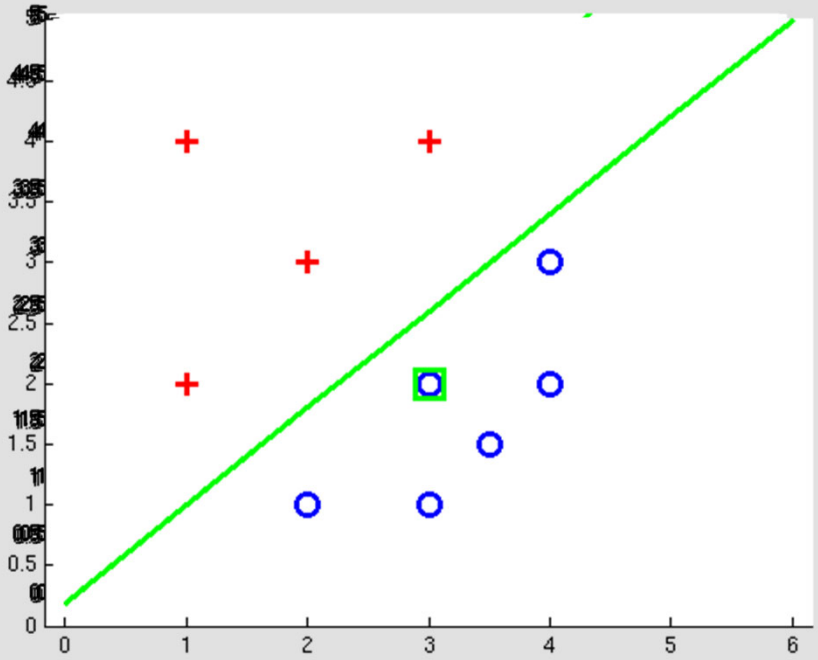
	x_i **is positive**	x_i **is negative**
$y = 1$ & $h_{\mathbf{w}} = 0$	w_i is increased	w_i is decreased
$y = 0$ & $h_{\mathbf{w}} = 1$	w_i is decreased	w_i is increased

  - gradient descent rule for **logistic regression**

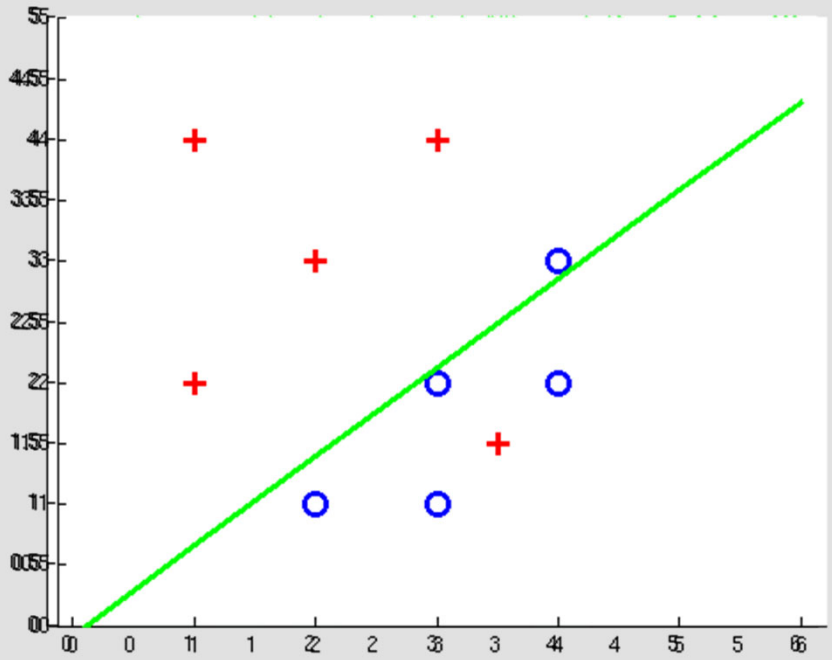
$$w_i \leftarrow w_i + \underbrace{\alpha(y - h_{\mathbf{w}}(\mathbf{x}))}_{Error} \times \underbrace{h_{\mathbf{w}}(\mathbf{x}) \times (1 - h_{\mathbf{w}}(\mathbf{x}))}_{g'(\mathbf{w} \cdot \mathbf{x})} \times x_i$$

Perceptrons – Separability

- **Separable Case**

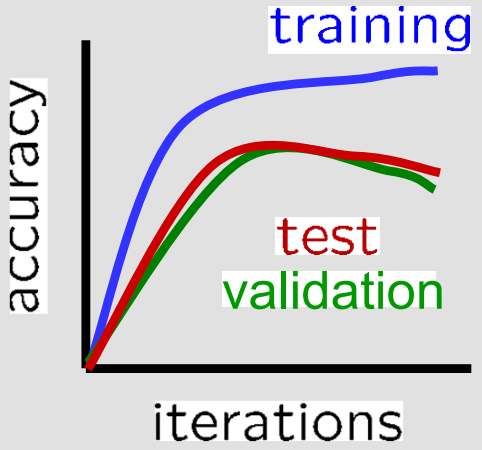


- **Non-Separable Case**

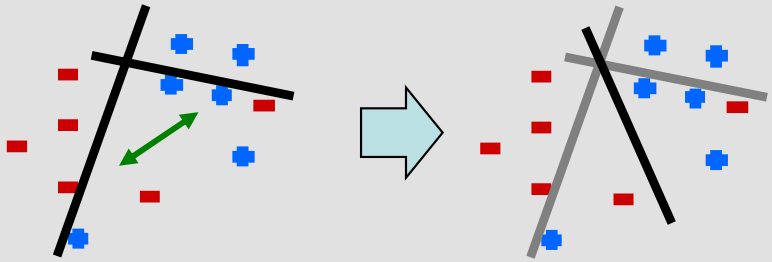


Issues with Perceptrons

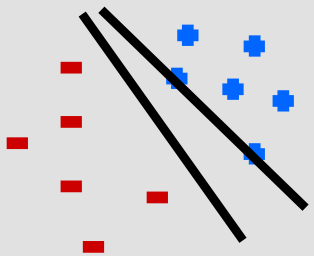
- **Overtraining**
  - Test / validation accuracy usually rises, then falls



- **Non-separability**
  - If the data aren't separable, weights might thrash around



- **Mediocre generalization**
  - Finds a "barely" separating solution

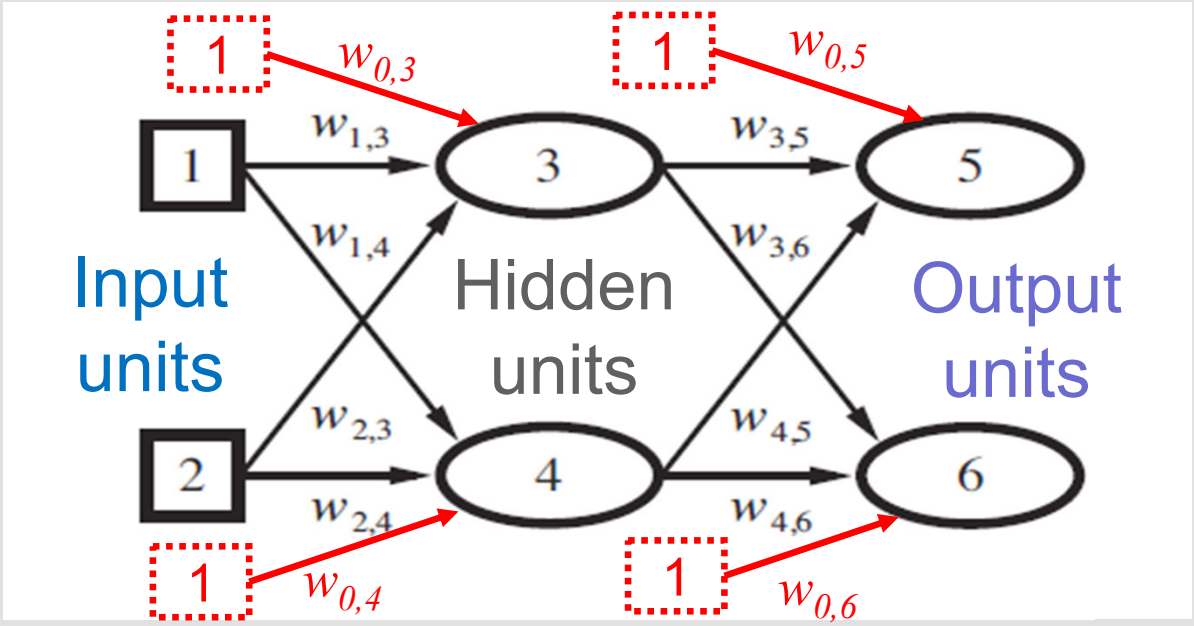


Multi-layer Feed-forward NN

- **A network with one or more hidden layers**
  - Given an input vector $\mathbf{x} = (x_1, x_2)$
    - The activations of the input units are set to $(a_1, a_2) = (x_1, x_2)$
    - The output of unit 5 is

$$a_5 = g(w_{0,5} + w_{3,5}a_3 + w_{4,5}a_4)$$

$$a_3 = g(w_{0,3} + w_{1,3}a_1 + w_{2,3}a_2) \qquad a_4 = g(w_{0,4} + w_{1,4}a_1 + w_{2,4}a_2)$$

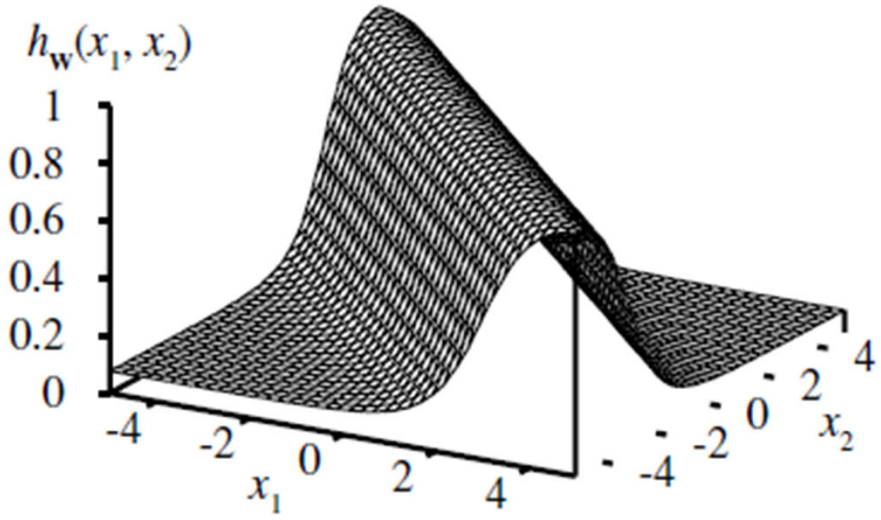


Dummy inputs = 1

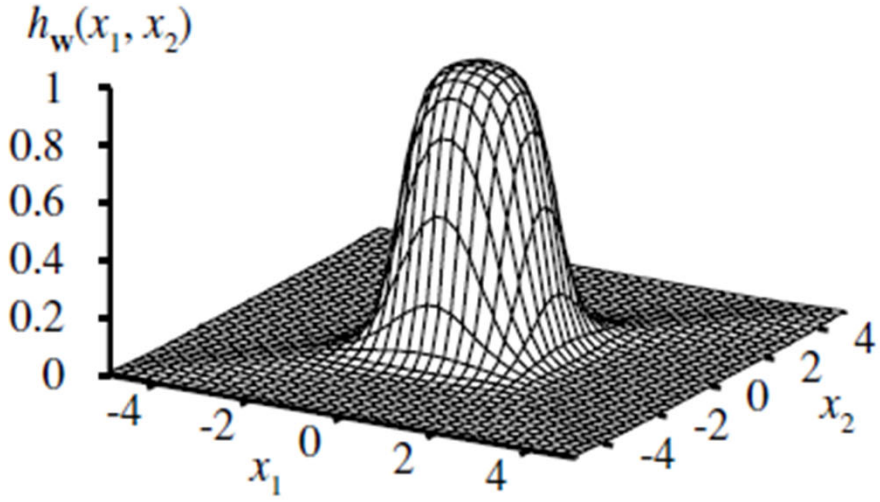
Expressiveness of Multi-layer FF NNs

- **Each unit in a sigmoid network represents a soft threshold**
- **Can create arbitrarily complex functions**

Two opposite-facing soft threshold functions



Four "opposite facing" soft threshold functions at right angles



Learning Multi-layer Feed-forward NNs

- **Problems:**
  - Interactions among learning problems when the network has multiple outputs
    - For an *additive* loss function, we add the gradients of the individual losses

$$\frac{\partial}{\partial w} Loss(\boldsymbol{w}) = \frac{\partial}{\partial w} \sum_{k=1}^{m} (y_k - a_k)^2 = \sum_{k=1}^{m} \frac{\partial}{\partial w} (y_k - a_k)^2$$

nodes in the output layer

      ➔ we can decompose an *m*-output learning problem into *m* learning problems: when updating each weight, add up the gradient contributions from each output
  - How do we calculate the error in the hidden layers?
    - **Back-propagation**

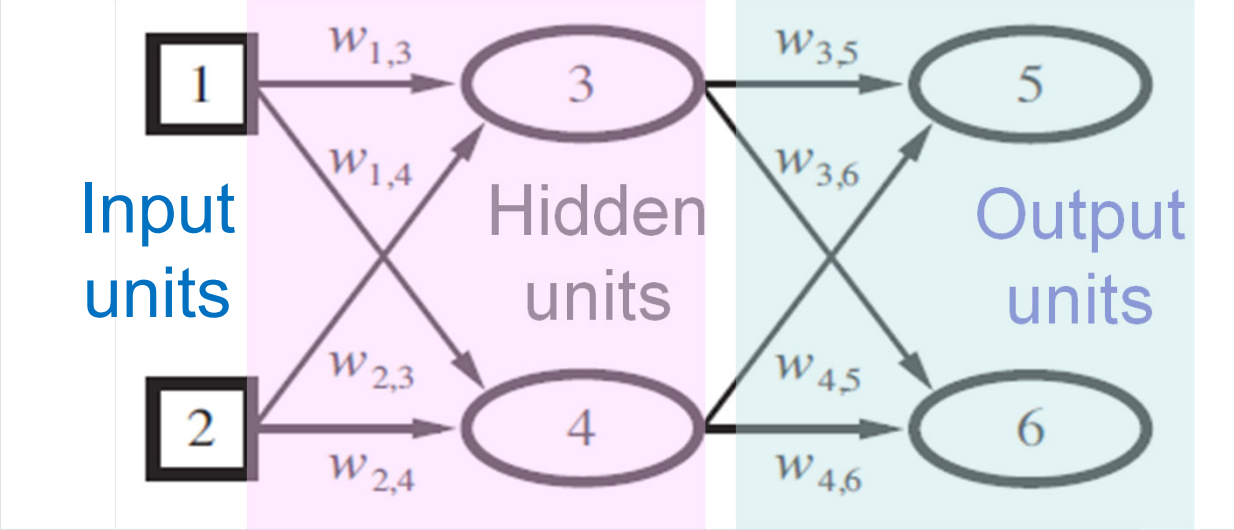
Back-propagation

- Weight adjustment hidden unit j → output unit k

$$\Delta_k \leftarrow (y_k - a_k) g'(in_k) \qquad w_{j,k} \leftarrow w_{j,k} + \alpha \times a_j \times \Delta_k$$

- Weight adjustment input/hidden unit i → hidden unit j

$$\Delta_j \leftarrow g'(in_j) \boxed{\sum_k w_{j,k} \ \Delta_k} \qquad w_{i,j} \leftarrow w_{i,j} + \alpha \times a_i \times \Delta_j$$



Back-propagation Algorithm

1. Compute the Δ values for the output units, using the observed error
2. Starting with the output layer, for each layer in the network **do** until the earliest hidden layer is reached
   a. Propagate the Δ values back to the previous layer
   b. Update the weights between the two layers

Summary – Supervised Machine Learning

- **Techniques for supervised machine learning**
  - where we are given labeled training data
- **Discrete:**
  - Decision Trees — non-parametric — non-linear
  - Naïve-Bayes — parametric — linear
  - K nearest neighbour — non-parametric — non-linear
  - Logistic regression — parametric — linear
  - Perceptron — parametric — linear
  - Multi-layer perceptron — parametric — non-linear
- **Continuous:**
  - Regression
    - > Models based on least square error (parametric)

MONASH University
Information Technology

FIT5047 – Intelligent Systems

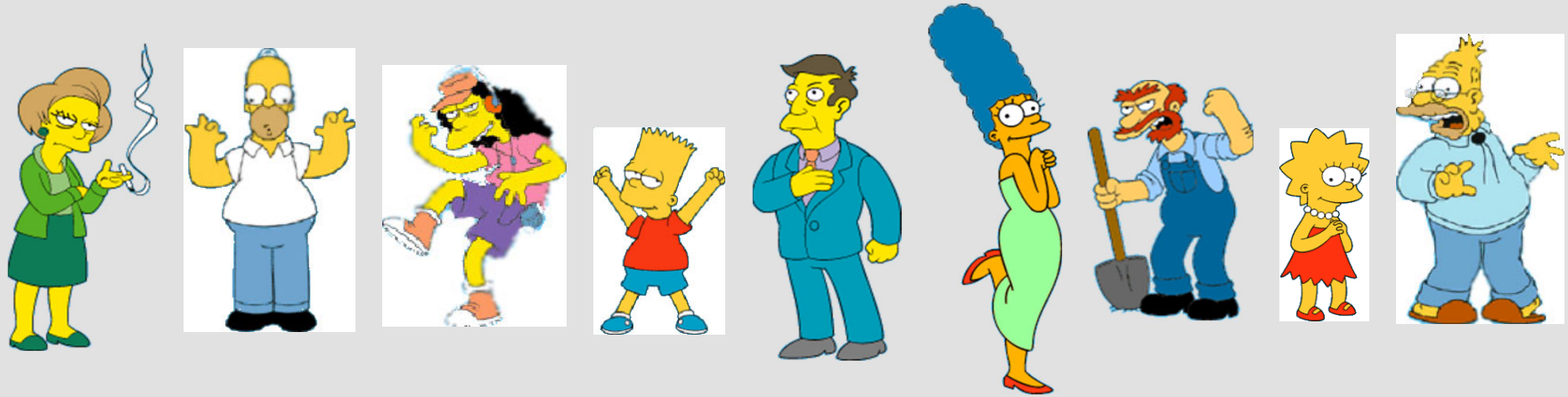
Unsupervised Machine Learning – Clustering

Slides adapted from Eamonn Keogh, Dan Klein and Kevin Korb

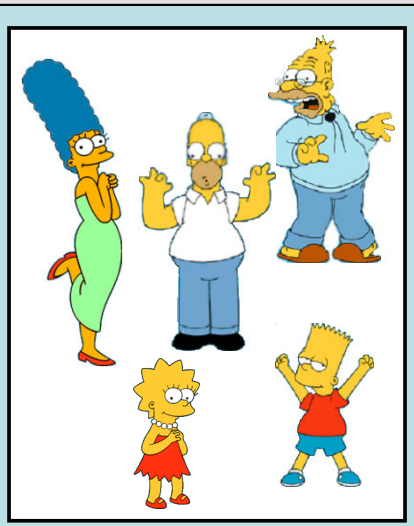
What is Clustering?

- **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?

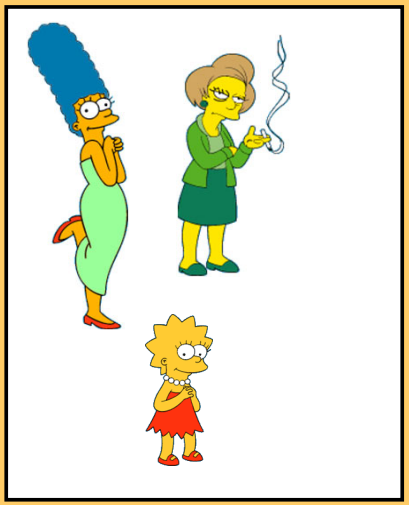


Clustering is subjective



Simpson's Family

School Employees



Females

Males

What is Similarity?

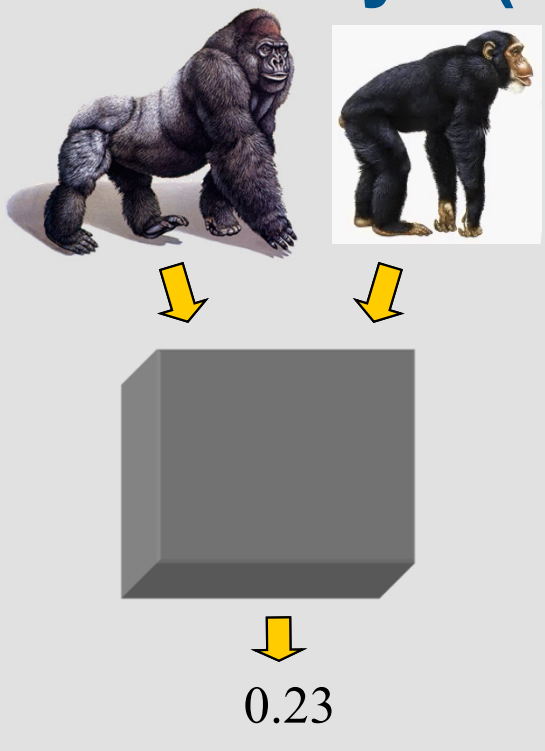
- ***The quality or state of being similar; likeness; resemblance; as, a similarity of features.***

  Webster's dictionary

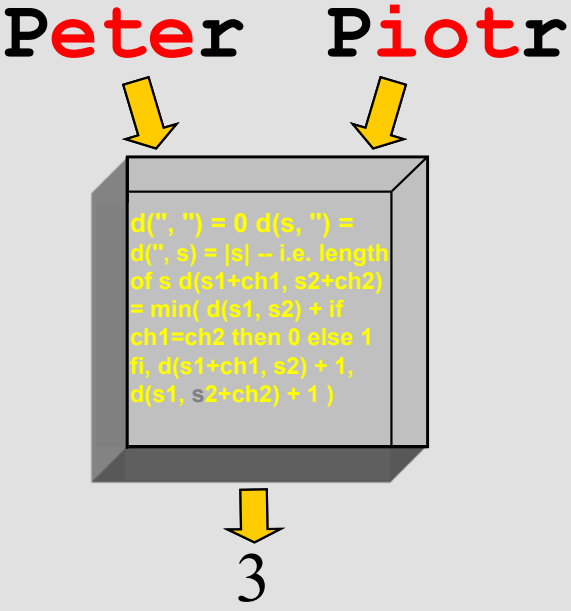
- **Similarity is hard to define, but…**
  "*We know it when we see it*"

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)*
- $D(A,B) \leq D(A,C) + D(B,C)$ *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:

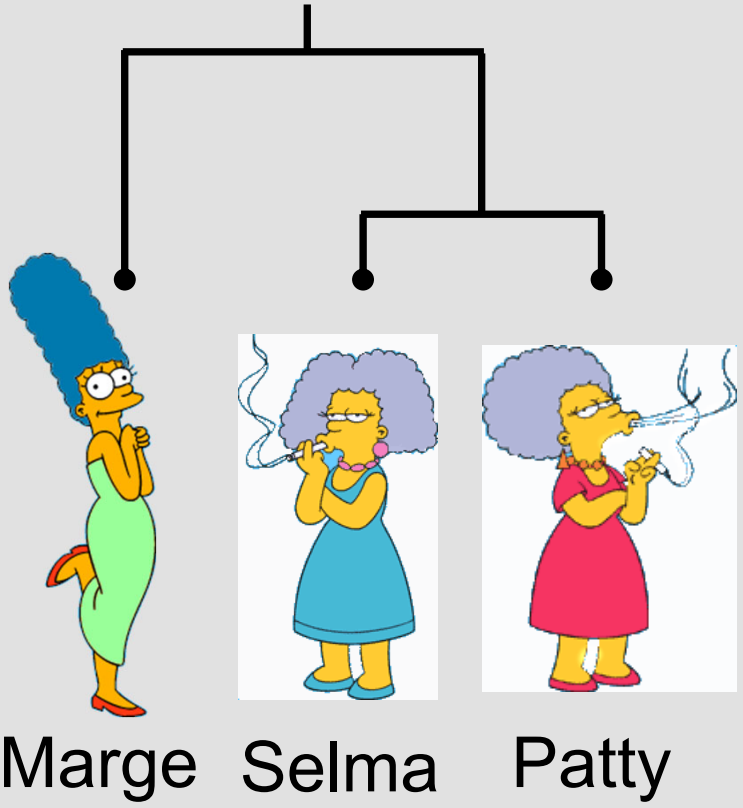
Change dress color	1 point
Change earring shape	1 point
Change hair part	1 point

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

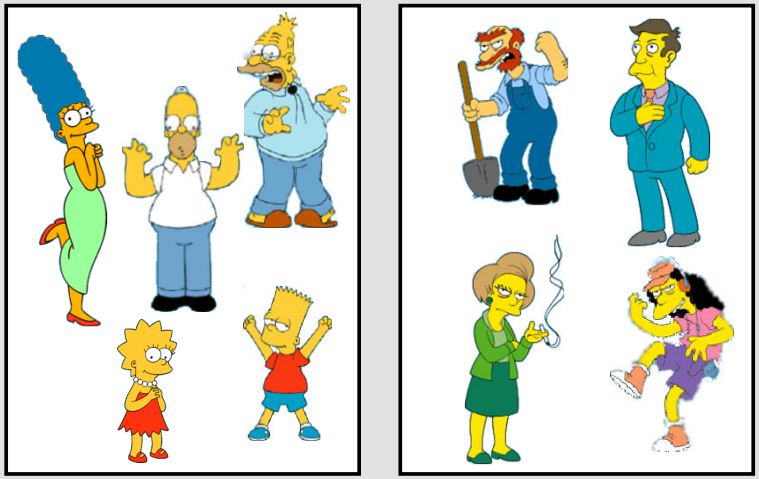
D(Marge,Selma) = 5



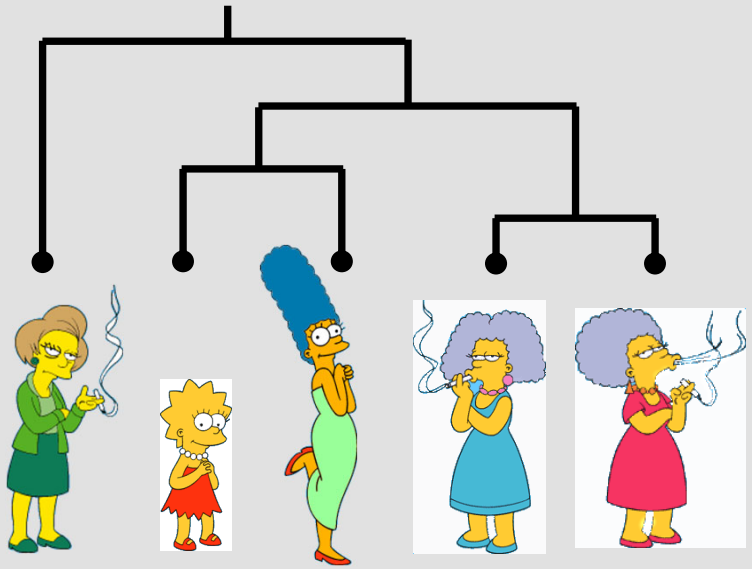
Two Types of Clustering

- **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



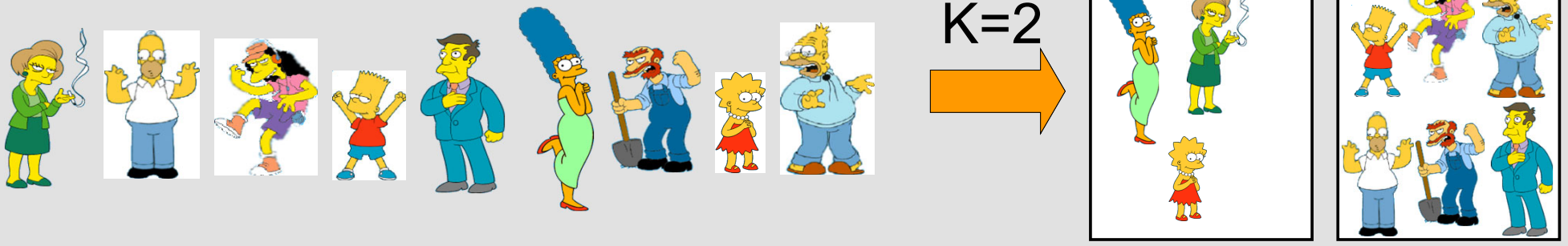
MONASH University
Information Technology

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

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

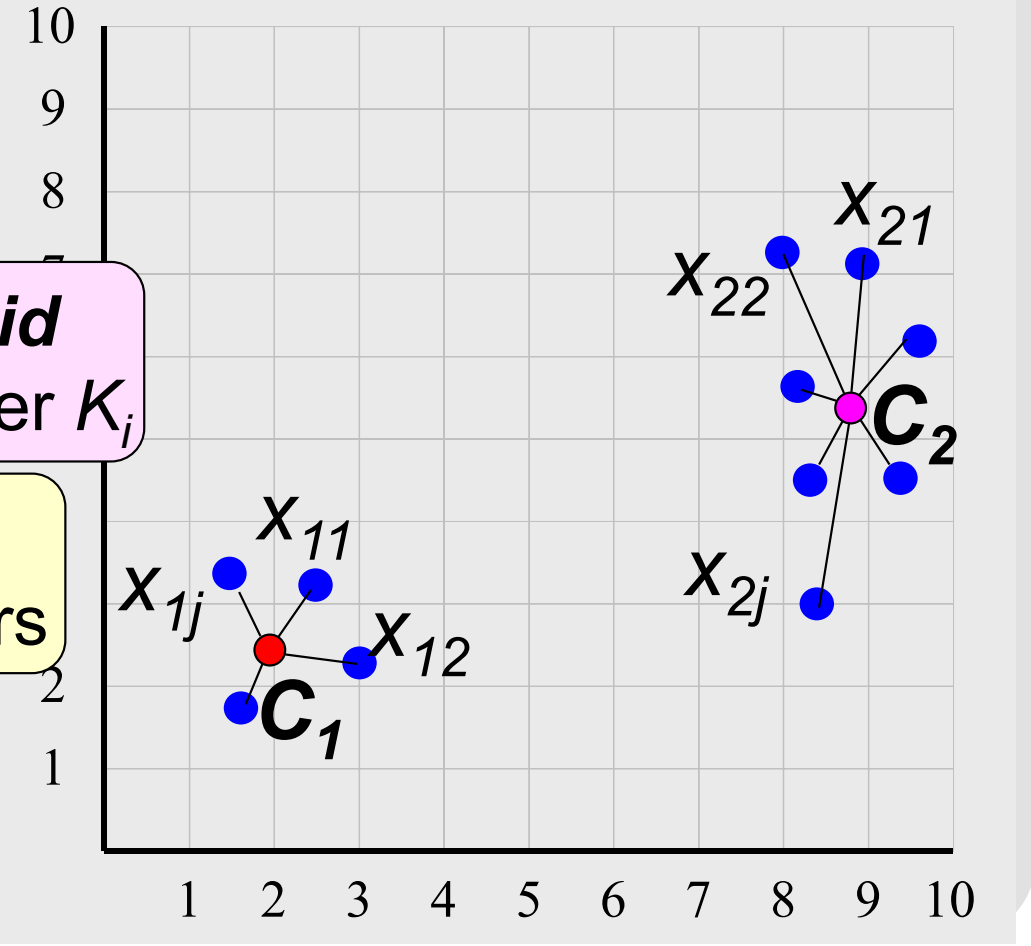
Centroid of cluster K_i

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

Number of clusters

Objective Function:

min se_K

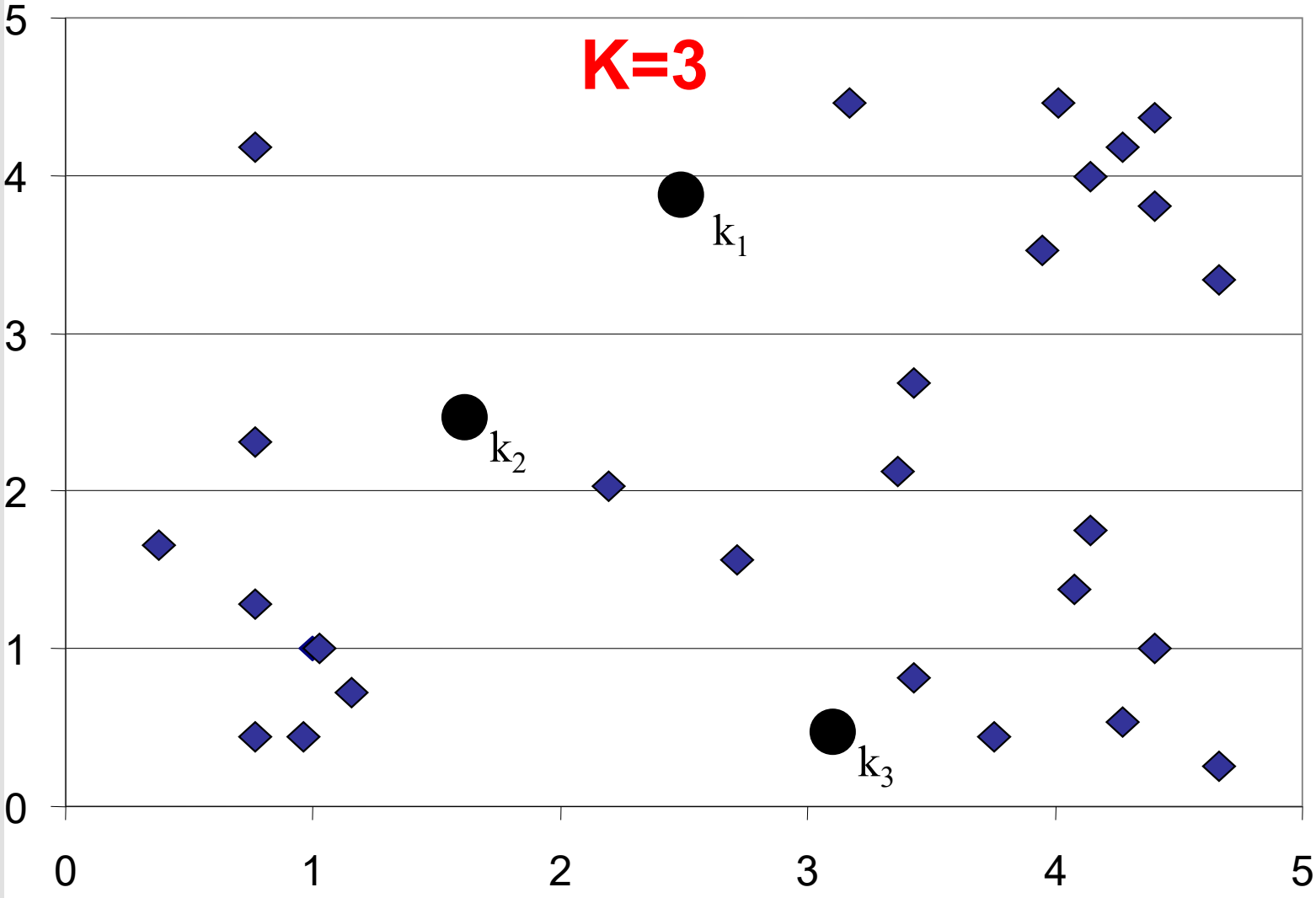


K-means Algorithm

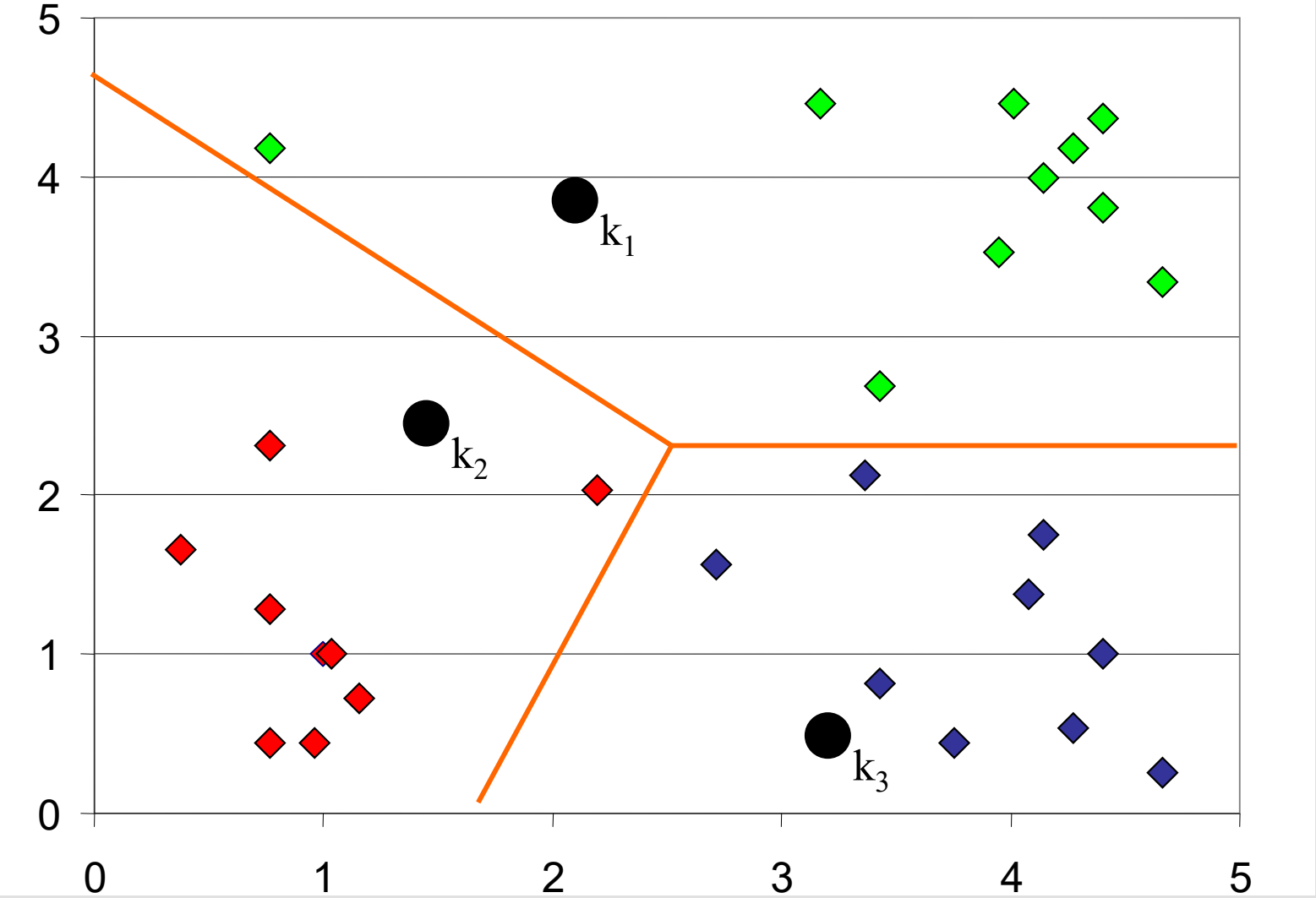
1. Decide on a value for *k*
2. Initialize the *k* cluster *centroids* (randomly, if necessary)
3. Decide the class memberships of the *N* objects by assigning them to the nearest cluster centroid
4. Re-estimate the *k* cluster centroids, assuming the memberships found above are correct
5. If none of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

K-means Clustering (1): Steps 1 and 2

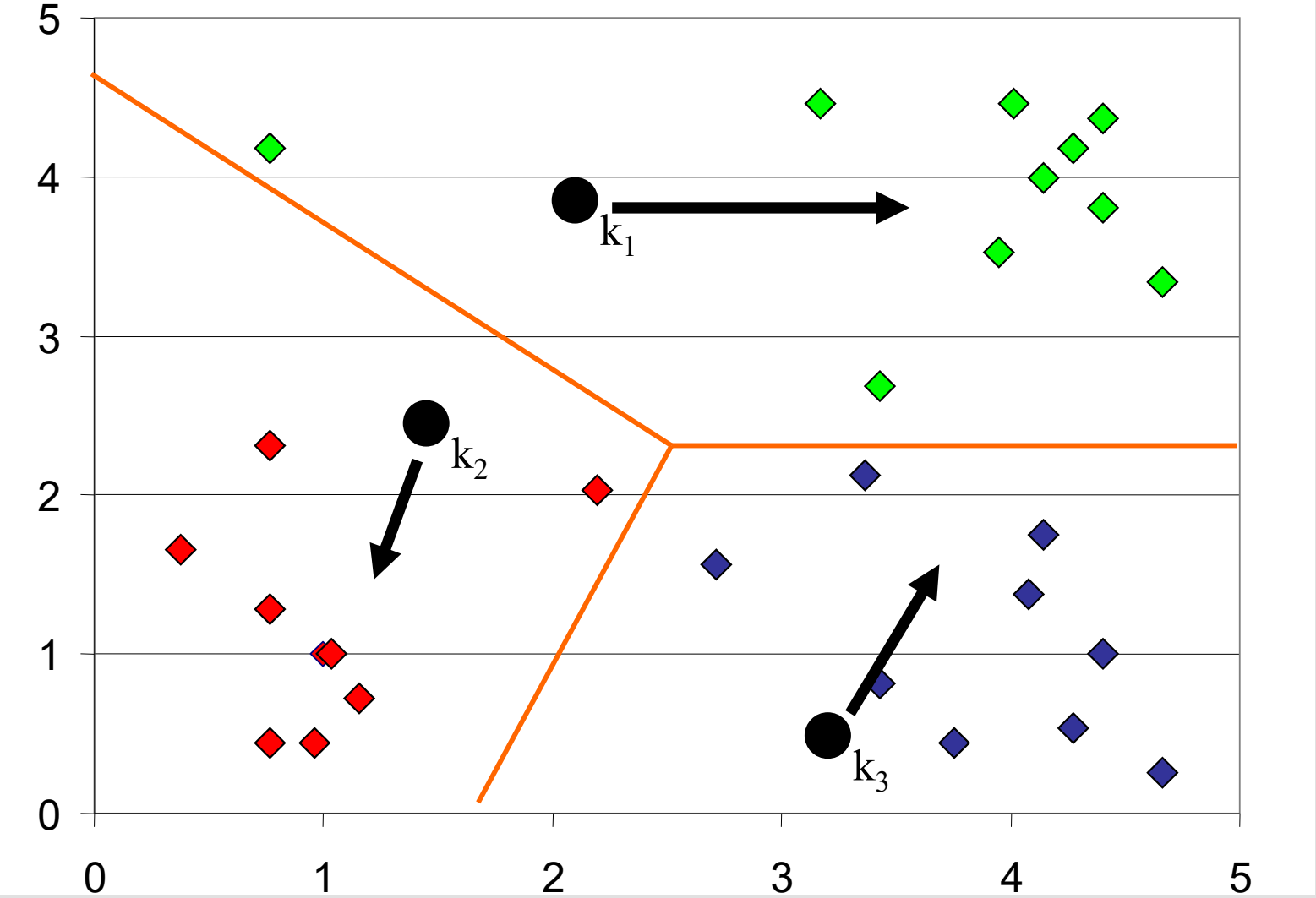
Distance Metric: Euclidean Distance



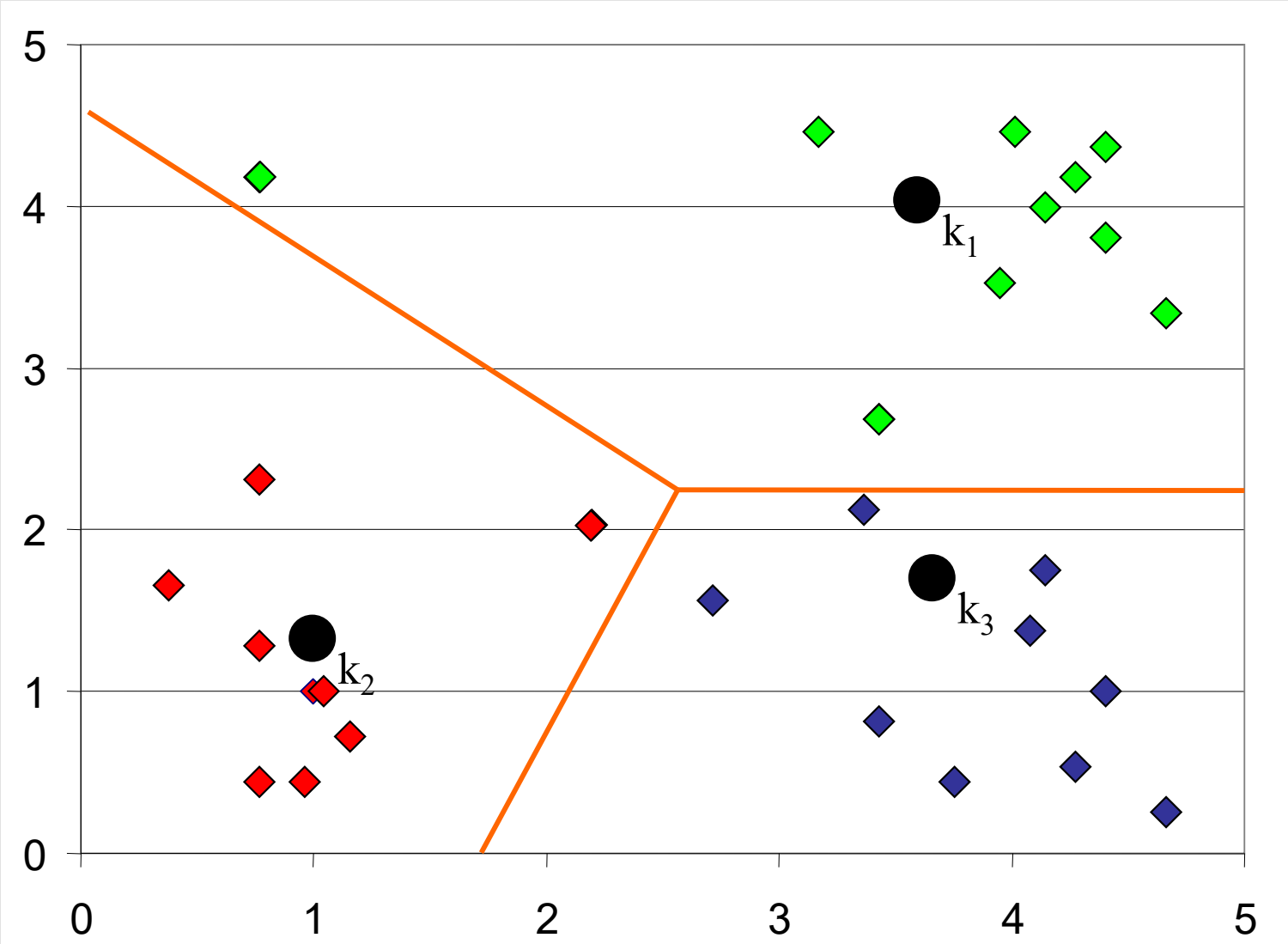
K-means Clustering (1): Step 3



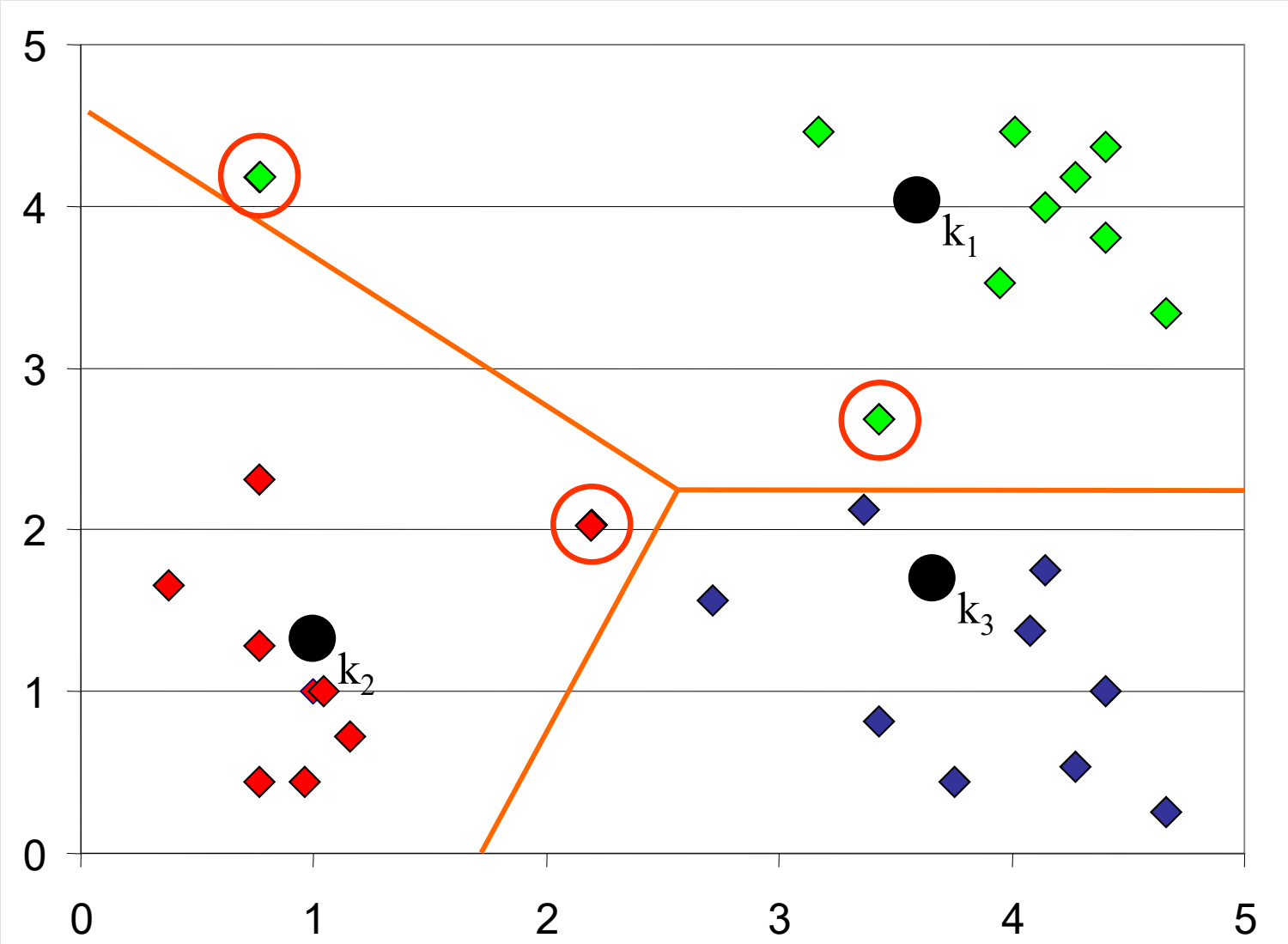
K-means Clustering (1): Step 4



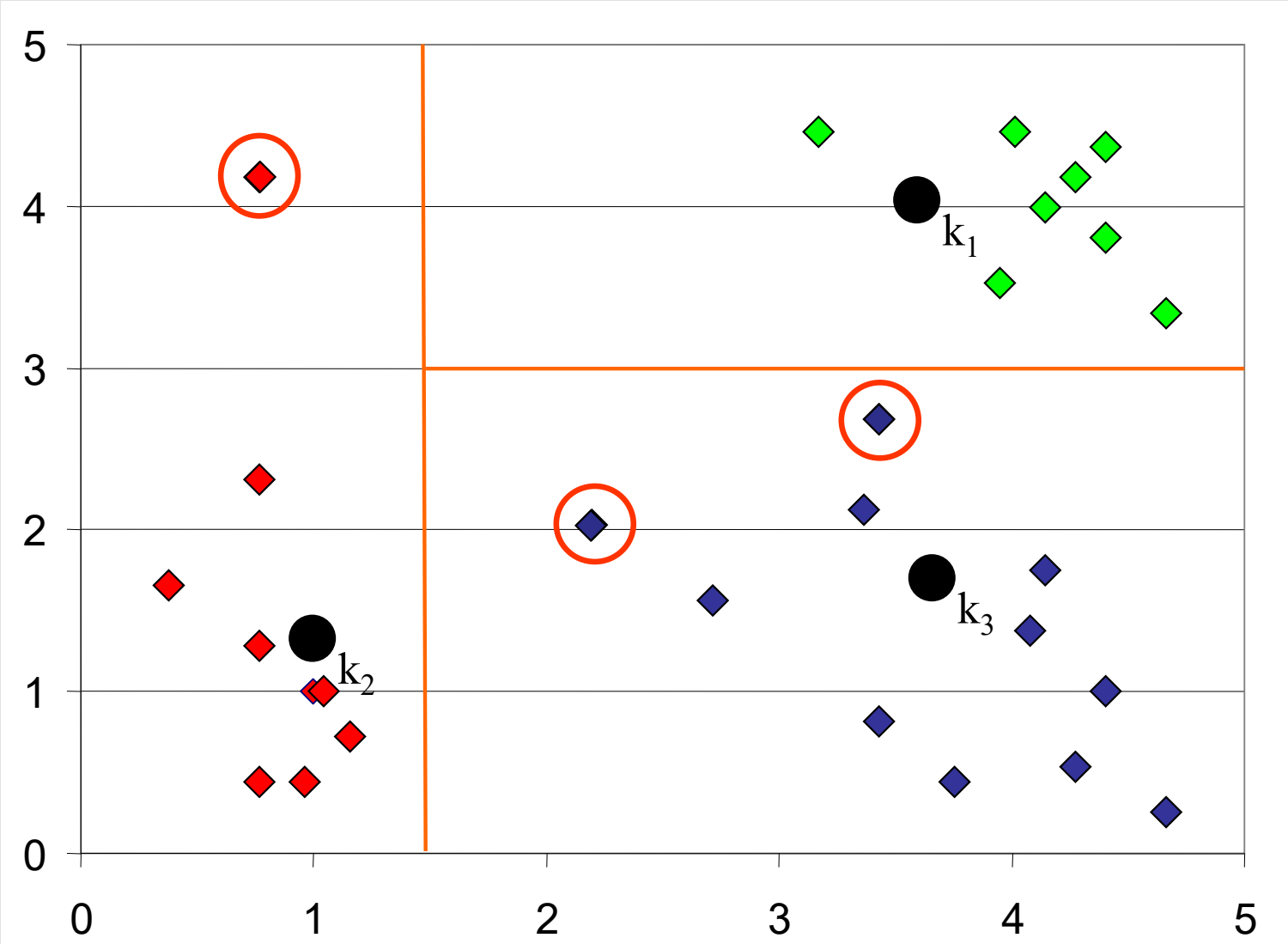
K-means Clustering (1): Step 5



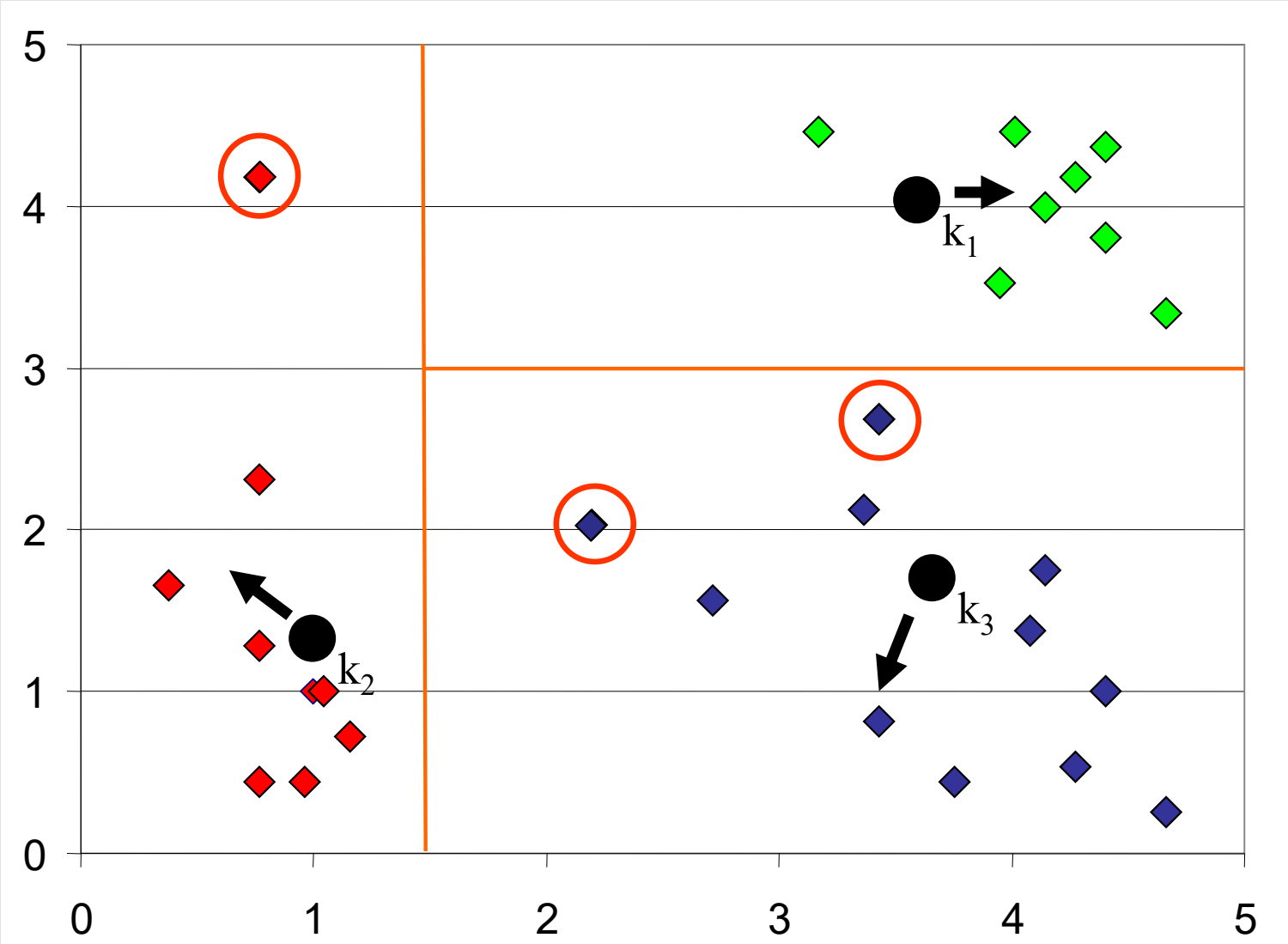
K-means Clustering (2): Step 3



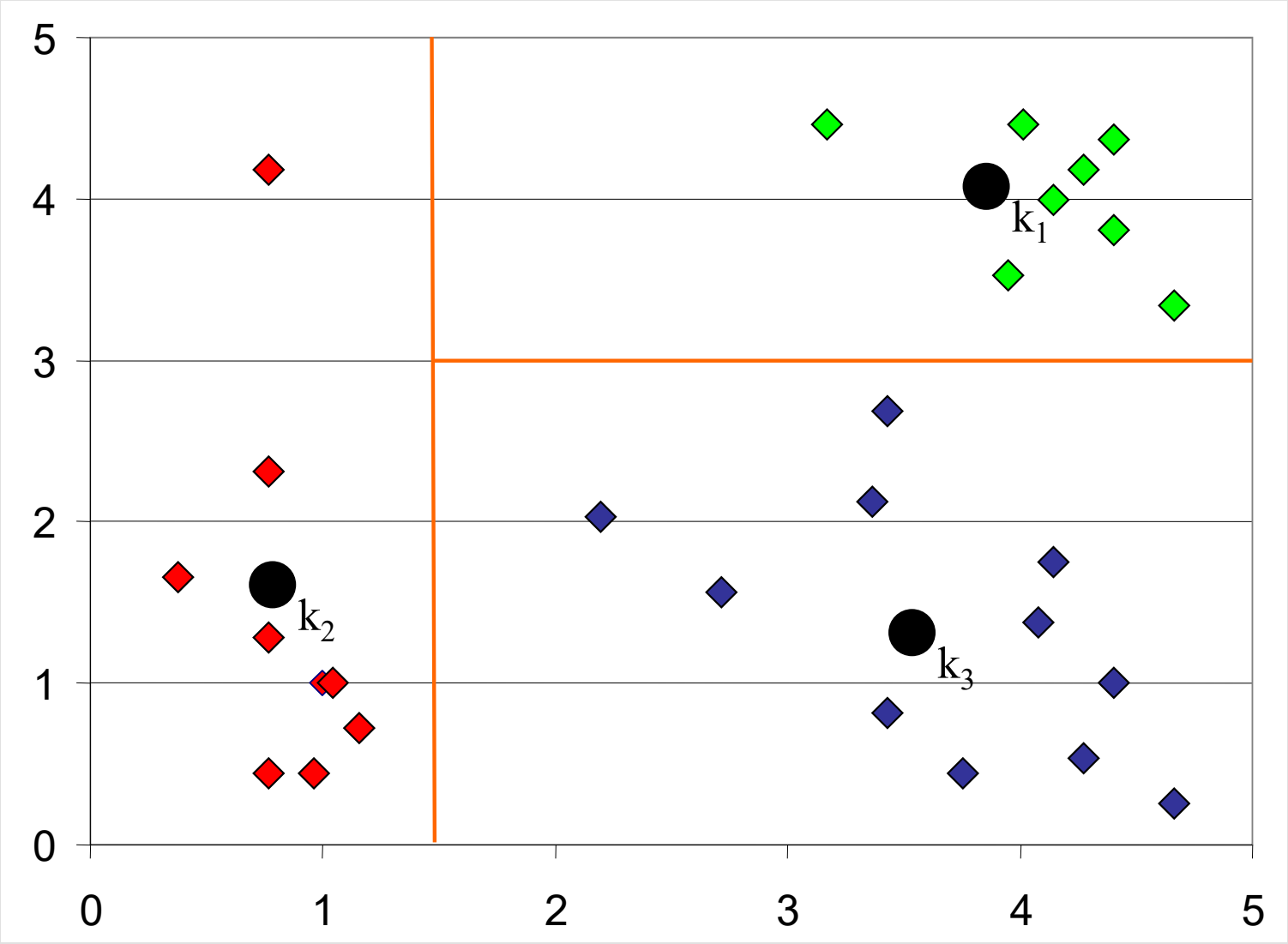
K-means Clustering (2): Step 3



K-means Clustering (2): Step 4

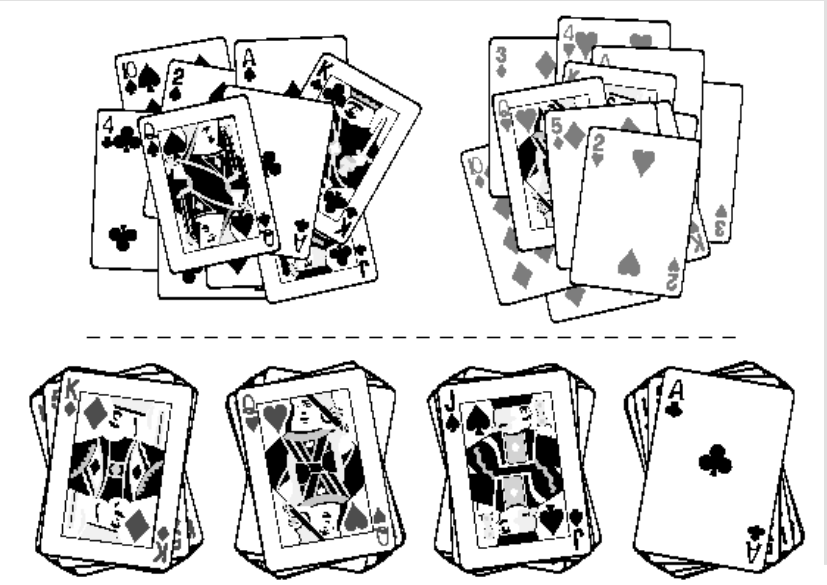


K-means Clustering (2): Step 5



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 by attributes with a 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 features**

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 Neighbor
  - Logistic – Logistic regression
  - MultilayerPerceptron – Multilayer Feed-forward NN
- **Clustering algorithm**
  - K-means

Reading

- **Supervised learning –** Russell, S. and Norvig, P. (2010), *Artificial Intelligence – A Modern Approach* (3rd ed), Prentice Hall
  - Fundamental concepts – Chapter 18.1-18.2
  - Decision trees – Chapter 18.3
  - Evaluating the best hypothesis – Chapter 18.4
  - Naïve Bayes – Chapter 20.1-20.2.3
  - K Nearest Neighbours – Chapter 18.8.1
  - Regression, Logistic regression – Chapter 18.6.1-18.6.4
  - Artificial Neural Networks – Chapter 18.7
- **Unsupervised learning and K-means clustering –**
  - http://en.wikipedia.org/wiki/K-means_clustering
  - https://sites.google.com/site/dataclusteringalgorithms/k-means-clustering-algorithm