Data Warehousing

Lecture 9 Frequent Pattern based Classification and Supervised Learning

CITS3401
CITS5504

Zeyi Wen

Computer Science and Software Engineering

School of Maths, Physics and Computing

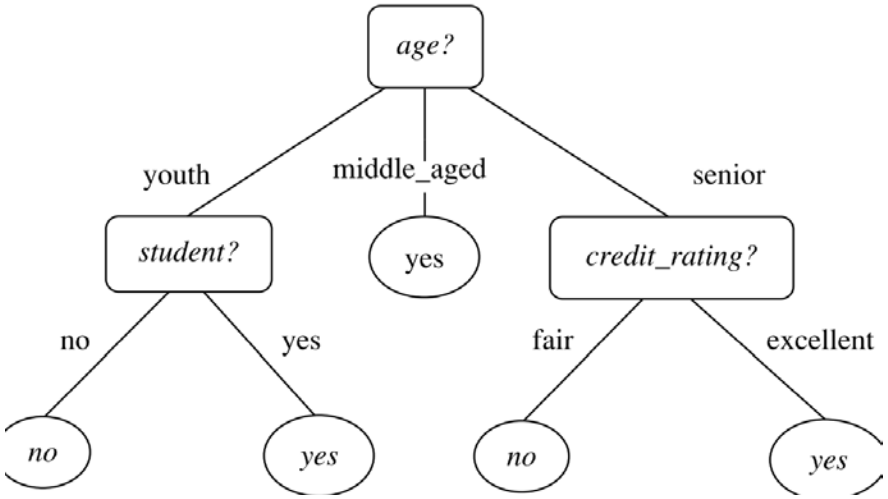
Acknowledgement: The lecture slides are based on online sources.

Lecture Outline

- Overview of Learning Models
- Model Evaluation
- Decision Trees
- Frequent Pattern based Classification
- k-Nearest Neighbours
- Other Common Machine Learning Models
  - Bayesian Classification
  - Artificial Neural Networks as Classifiers
  - Support Vector Machines
- Overfitting and underfitting

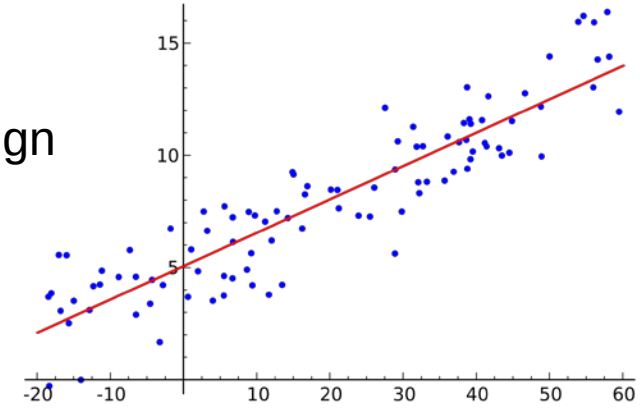
Basic Concepts

- **The purpose of data analysis is to**
  - Design *models* describing important data trends
  - What does a model look like?
    - A function
    - A decision-tree
    - An artificial neural network
- **Two major forms of data analysis**
  - Classification
    - Predicts categorical (class) labels
  - Regression
    - Models continuous valued functions
- **Applications**
  - target marketing, performance prediction, medical diagnosis, manufacturing, fraud detection, webpage categorisation, …



Classification vs. Regression

- **Classification**
  - Predict categorical class labels (discrete or nominal)
  - Construct a model based on the training set and the **class labels** (the values in a classifying attribute) and use it in classifying new data
- **Regression**
  - Model continuous-valued functions (i.e. predict unknown or missing values)
- **Typical applications of classification**
  - Credit/loan approval
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorisation: which category it is



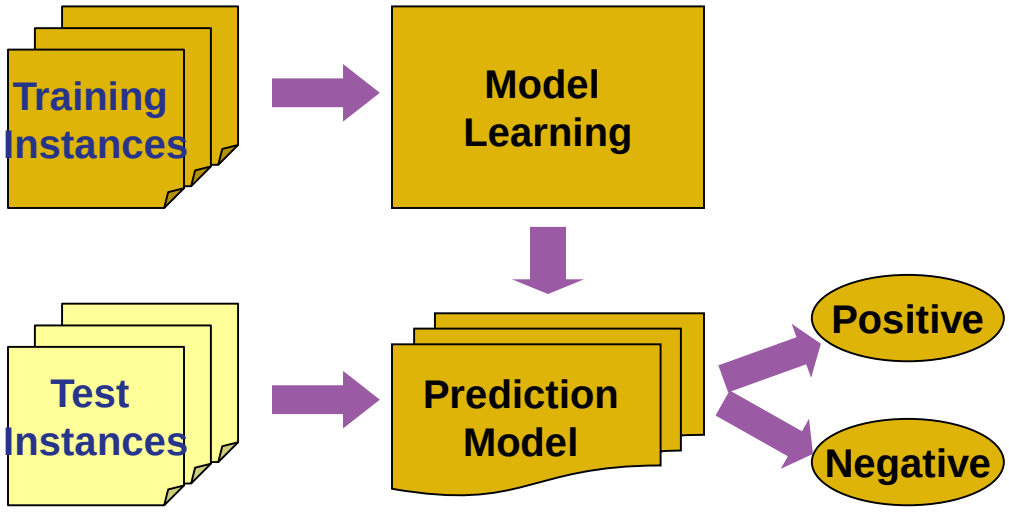
Supervised vs. Unsupervised learning

THE UNIVERSITY OF WESTERN AUSTRALIA

- **Supervised learning (classification)**
  - Supervision: The training data such as observations or measurements are accompanied by **labels** indicating the classes which they belong to
  - New data is classified based on the models built from the training set

Training Data with class label:

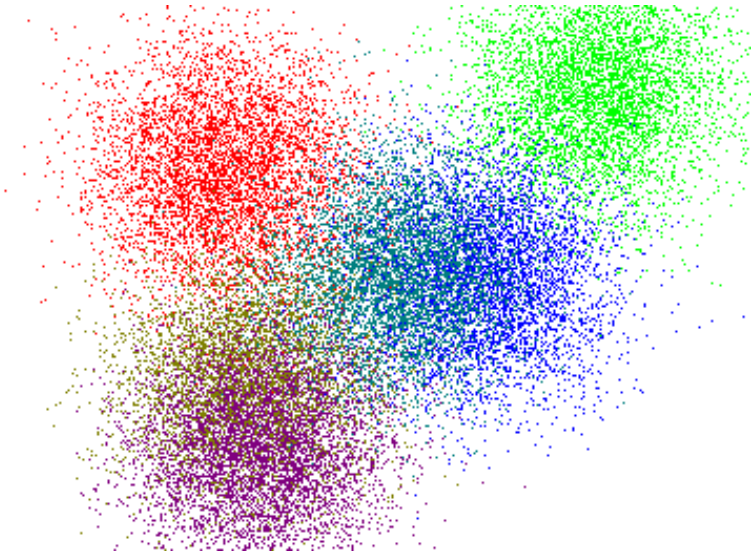
age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_age	high	no	fair	yes
senior	medium	no	fair	no
senior	low	yes	fair	no
senior	low	yes	excellent	yes
middle_age	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_age	medium	no	excellent	yes
middle_age	high	yes	fair	yes
senior	medium	no	excellent	yes



Supervised vs. Unsupervised learning

THE UNIVERSITY OF WESTERN AUSTRALIA

- **Unsupervised learning (clustering)**
  - The class labels of training data are unknown
  - Given a set of observations or measurements, establish the possible existence of classes or clusters in the data



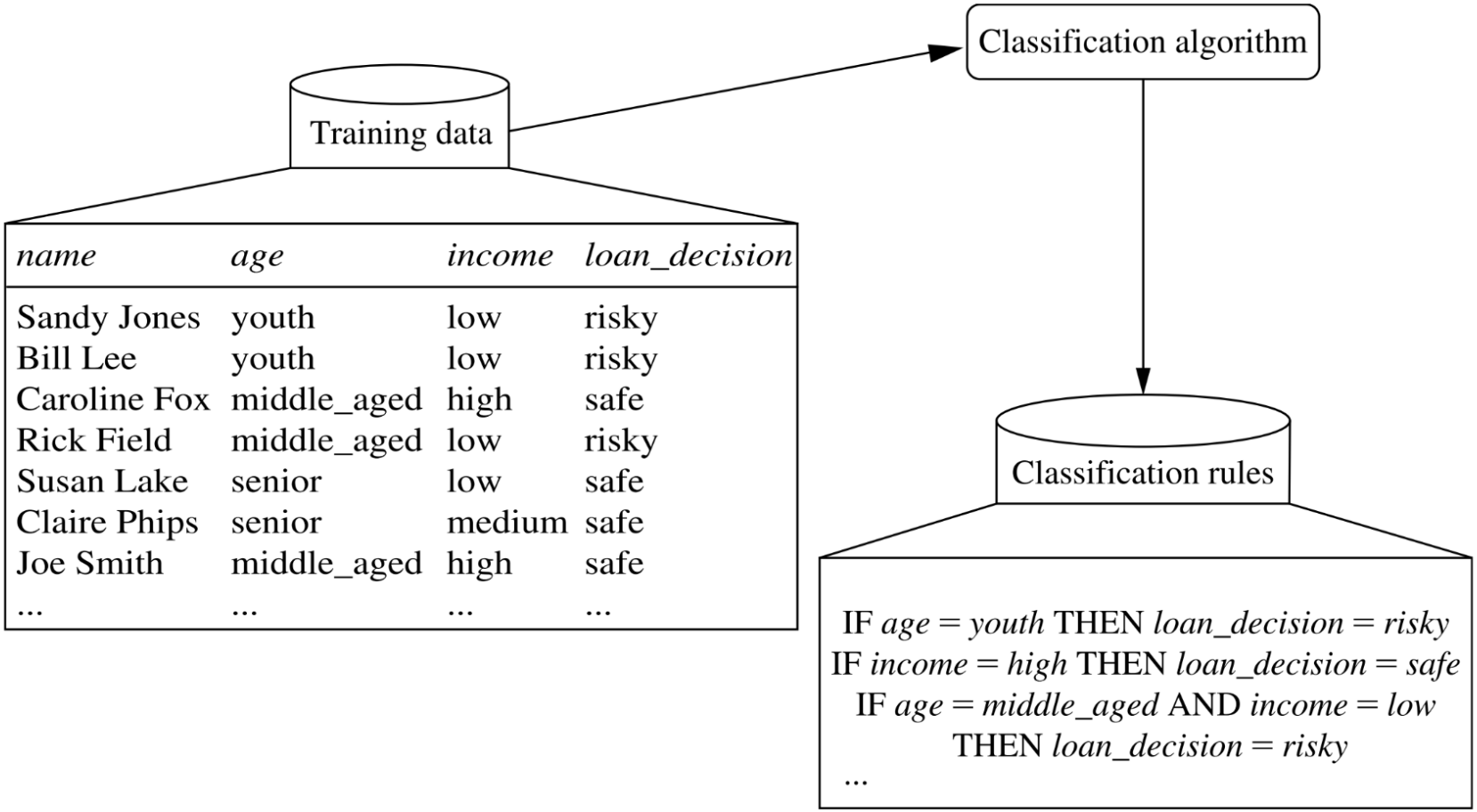
Supervised vs. Unsupervised learning

- **Supervised learning (e.g. classification)**
  - Supervision: The training data (e.g. observations or measurements) are accompanied by labels indicating the class which they belong to.
  - New data is classified using the model built from the training set.
- **Unsupervised learning (e.g. clustering)**
  - The class labels of training data are unknown.
  - Given a set of measurements or observations, establish the existence of classes or clusters in the data.

Classification – A two-step process

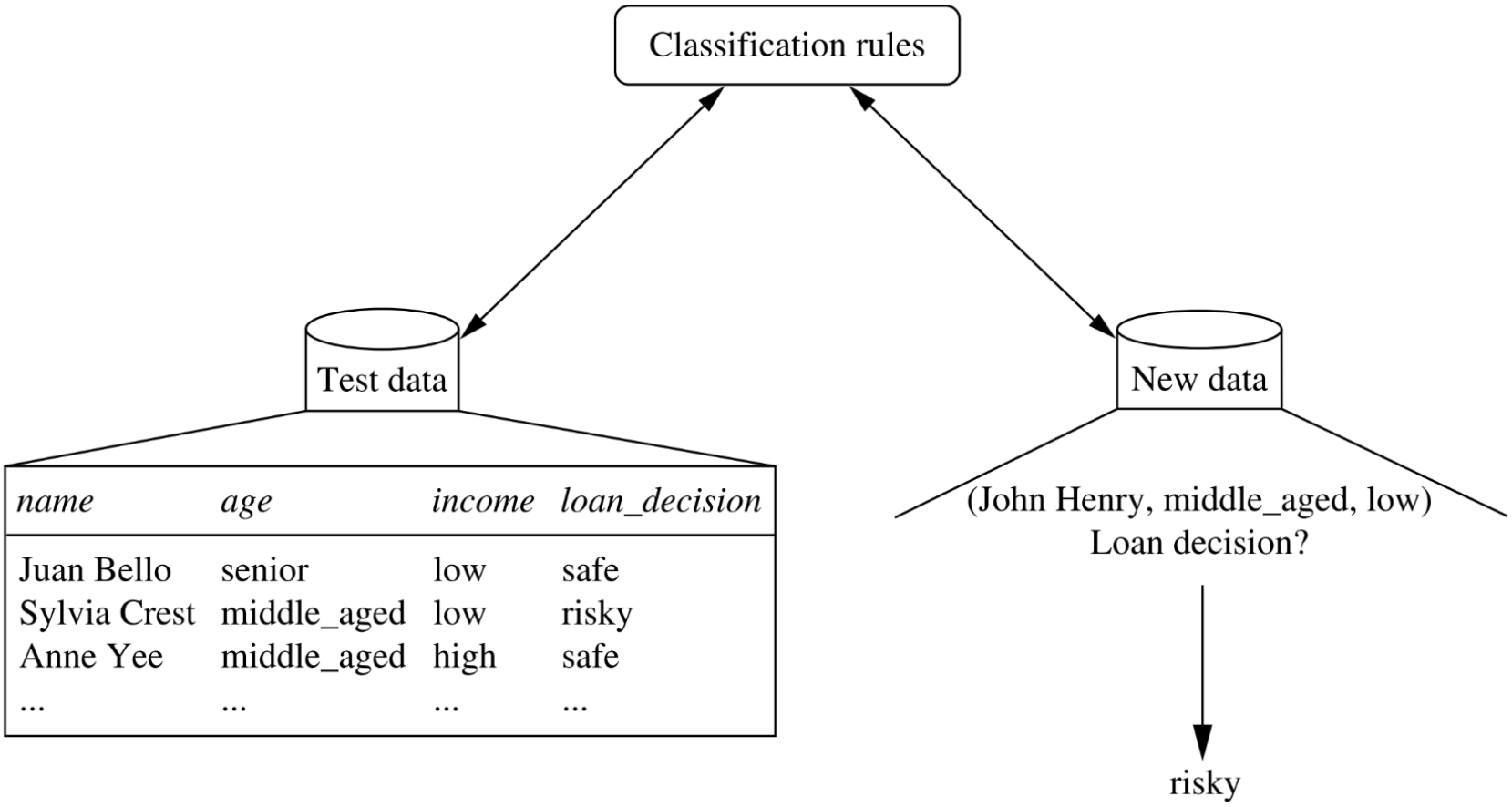
- **Model construction: describing a set of predetermined classes**
  - Each instance is assumed to belong to a predefined class (shown by the class label)
  - The set of instances used for model construction is training set
  - The model is represented as decision trees, rules or mathematical formulas
- **Model usage: for classifying future or unknown objects**
  - Estimate accuracy of the model
    - The known label of a test instance is compared with the classified result from the model
    - Accuracy: % of test instances that are correctly classified by the model
    - Test set is independent of training set (otherwise overfitting)
  - If the accuracy is acceptable, use the model to classify new data
- **Note: If the test set is used to select models, it is called validation (test) set**

Step 1: Build a classification model



name	*age*	*income*	*loan_decision*
Sandy Jones	youth	low	risky
Bill Lee	youth	low	risky
Caroline Fox	middle_aged	high	safe
Rick Field	middle_aged	low	risky
Susan Lake	senior	low	safe
Claire Phips	senior	medium	safe
Joe Smith	middle_aged	high	safe
...	...	...	...

Step 2: Use the model on test data



Lecture Outline

- Overview of Learning Models
- **Model Evaluation**
- Decision Trees
- Frequent Pattern based Classification
- k-Nearest Neighbours
- Other Common Machine Learning Models
  - Bayesian Classification
  - Artificial Neural Networks as Classifiers
  - Support Vector Machines
- Overfitting and underfitting

Model Evaluation and Selection

- Evaluation metrics: How can we measure accuracy? Other metrics to consider?
- Use validation test set of class-labeled instances instead of training set when assessing accuracy
- Methods for estimating a classifier's accuracy:
  - Holdout method, random subsampling
  - Cross-validation
  - Bootstrap (i.e. sampling with replacement)
- Comparing classifiers:
  - Confidence intervals
  - Cost-benefit analysis and ROC Curves

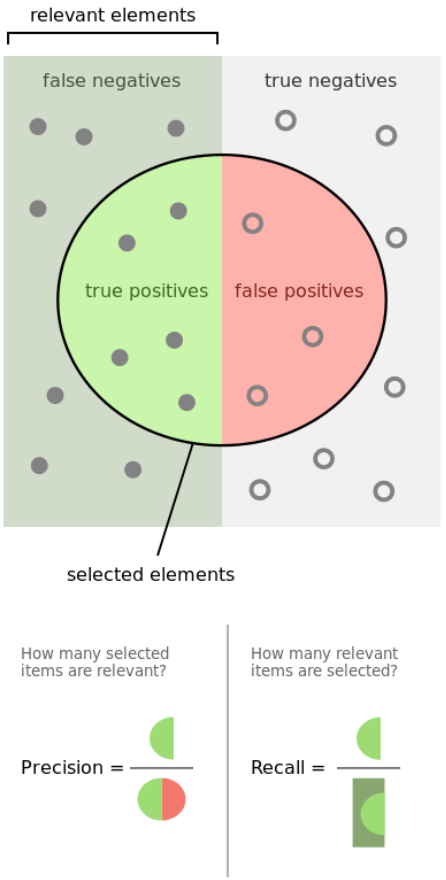
Classifier Evaluation: Confusion Matrix

- **Confusion Matrix (two classes)**

Truth\Predicated	Model Yes	Model No
Actual Yes	True Positives (TP)	False Negatives (FN)
Actual No	False Positives (FP)	True Negatives (TN)

- **Example of Confusion Matrix**

Truth\Predicated	buy_computer = yes	buy_computer = No	Total
buy_computer=yes	**6954**	**46**	7000
buy_computer=no	**412**	**2588**	3000
Total	7366	2634	10000



Confusion Matrix for Multiple Classes

- Given m classes, an entry, $CM_{i,j}$ in a confusion matrix indicates number of instances in class i that were labeled by the classifier as class j
- May have extra rows/columns to provide totals

	class 1	…	class i	…	class m	Total
class 1						
…						
class j			$CM_{i,j}$			
…						
class m						
Total						

Accuracy, Error Rate, Sensitivity and Specificity

- Classifier Accuracy, or recognition rate: percentage of test set instances that are correctly classified
  - $Accuracy = \frac{TP+TN}{ALL}$
- Error rate:
  - $Error\ Rate = 1 - Accuracy$
    $= \frac{FP+FN}{ALL}$

- One class may be *rare*, e.g. fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class
- Sensitivity: True Positive recognition rate
  - $Sensitivity = \frac{TP}{P}$
- Specificity: True Negative recognition rate
  - $Specitivity = \frac{TN}{N}$

Precision, Recall and F-measures

THE UNIVERSITY OF WESTERN AUSTRALIA

- Precision: exactness – what % of instances that the classifier labeled as positive are actually positive
  - $precision = \frac{TP}{TP+FP}$
- Recall: completeness – what % of positive instances did the classifier label as positive?
  - $recall = \frac{TP}{TP+FN}$
- Perfect score is 1.0
- Inverse relationship between precision & recall
- *F* measure (F_1 or *F*-score): harmonic mean of precision and recall,
  - $F = \frac{2 \times precision \times recall}{precision+recall}$
- F_β: weighted measure of precision and recall
  - assigns ß times as much weight to recall as to precision
  - $F_\beta = \frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision+recall}$

Classifier Evaluation Metrics: example

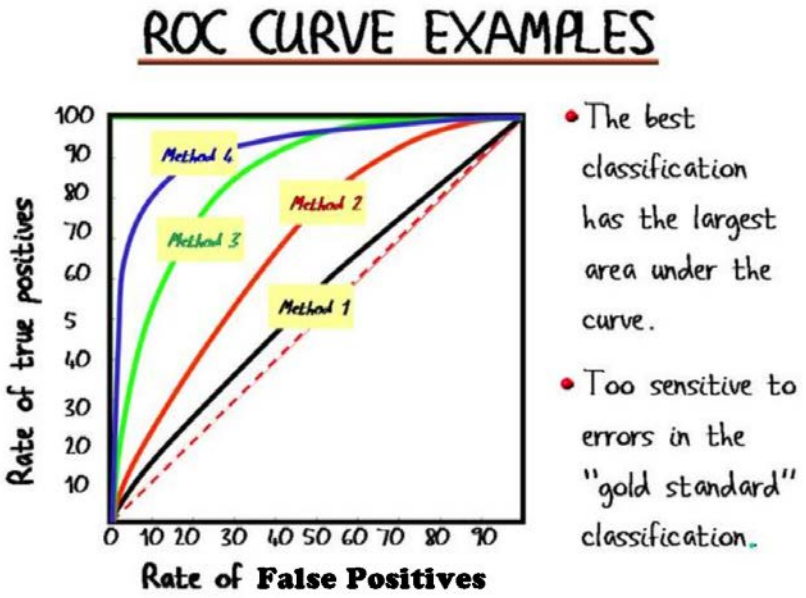
THE UNIVERSITY OF WESTERN AUSTRALIA

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	**90**	**210**	300	30.00 (*sensitivity)*
cancer = no	**140**	**9560**	9700	98.56 (*specificity)*
Total	230	9770	10000	96.40 (*accuracy*)

- Sensitivity = TP/P = 90/300 = 30%
- Specificity = TN/N = 9560/9700 = 98.56%
- Accuracy = (TP + TN)/All = (90+9560)/10000 = 96.50%
- Error rate = (FP + FN)/All = (140 + 210)/10000 = 3.50%
- Precision = TP/(TP + FP) = 90/(90 + 140) = 90/230 = 39.13%
- Recall = TP/ (TP + FN) = 90/(90 + 210) = 90/300 = 30.00%
- F1 = 2 P × R /(P + R) = 2 × 39.13% × 30.00%/(39.13% + 30%) = 33.96%

ROC Curves

- ROC (Receiver Operating Characteristics) curves: for visual comparison of classification models
  - Originated from signal detection theory
  - Shows the trade-off between the true positive rate and the false positive rate
  - The area under the ROC curve is a measure of the accuracy of the model
- Rank the test instances in decreasing order: the one that is most likely to belong to the positive class appears at the top of the list
- The closer to the diagonal line (i.e. the closer the area is to 0.5), the less accurate is the model

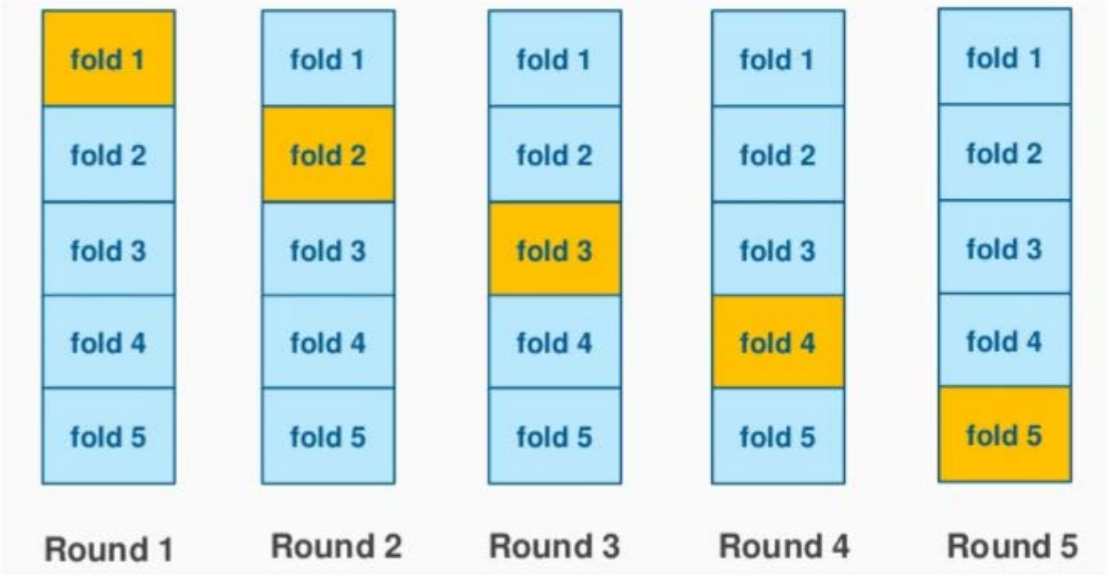


Holdout & Cross-Validation Methods

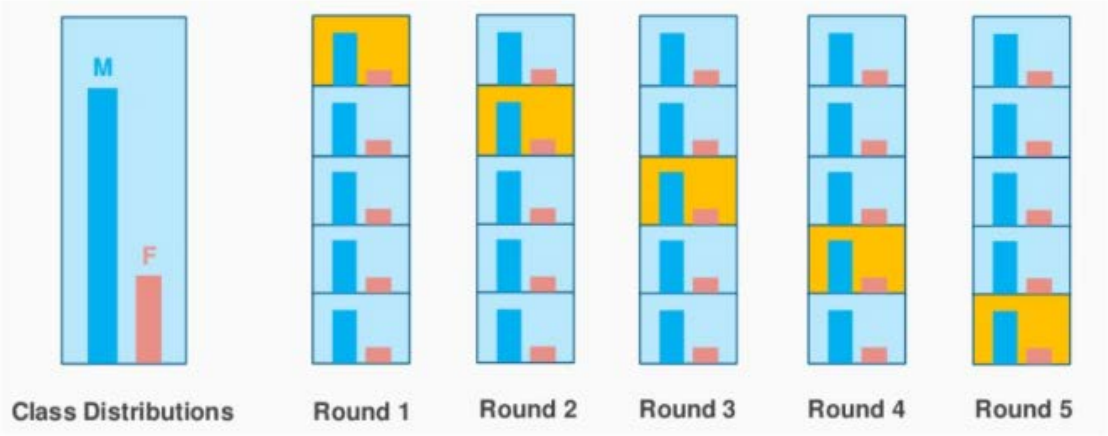
- **Holdout method**
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
- **Random sampling: a variation of holdout**
  - Repeat holdout k times, accuracy = avg. of the accuracies obtained
- **Cross-validation (k-fold, where k = 10 is most popular)**
  - Randomly partition the data into k mutually exclusive subsets, each approximately equal size
  - At *i*-th iteration, use D_i as test set and others as training set
  - Leave-one-out: k folds where k = # of instances, for small sized data
  - *Stratified cross-validation*: folds are stratified so that class distribution in each fold is approximately the same as that in the initial data.

Stratified cross-validation

Example of 5 fold **Cross Validation**:



Example of 5 folds **Stratified Cross Validation**:



Bootstrap

- **Bootstrap**
  - Works well with small data sets
  - Samples the given training instances uniformly *with replacement*
    - i.e. each time an instance is selected, it is equally likely to be selected again and re-added to the training set
- **Several bootstrap methods, and a common one is .632 boostrap**
  - A data set with *d* instances is sampled *d* times, with replacement, resulting in a training set of *d* samples. The instances that did not make it into the training set end up forming the test set. About 63.2% of the original data end up in the bootstrap, and the remaining 36.8% form the test set (since $(1 - 1/d)^d \approx e^{-1} = 0.368$)
  - Repeat the sampling procedure *k* times, overall accuracy of the model:

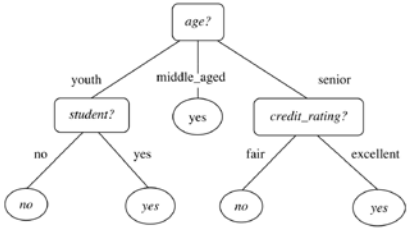
$$Acc(M) = \frac{1}{k}\sum_{i=1}^{k}(0.632 \times Acc(M_i)_{test\_set} + 0.368 \times Acc(M_i)_{train\_set})$$

Lecture Outline

- Overview of Learning Models
- Model Evaluation
- **Decision Trees**
- Frequent Pattern based Classification
- k-Nearest Neighbours
- Other Common Machine Learning Models
  - Bayesian Classification
  - Artificial Neural Networks as Classifiers
  - Support Vector Machines
- Overfitting and underfitting

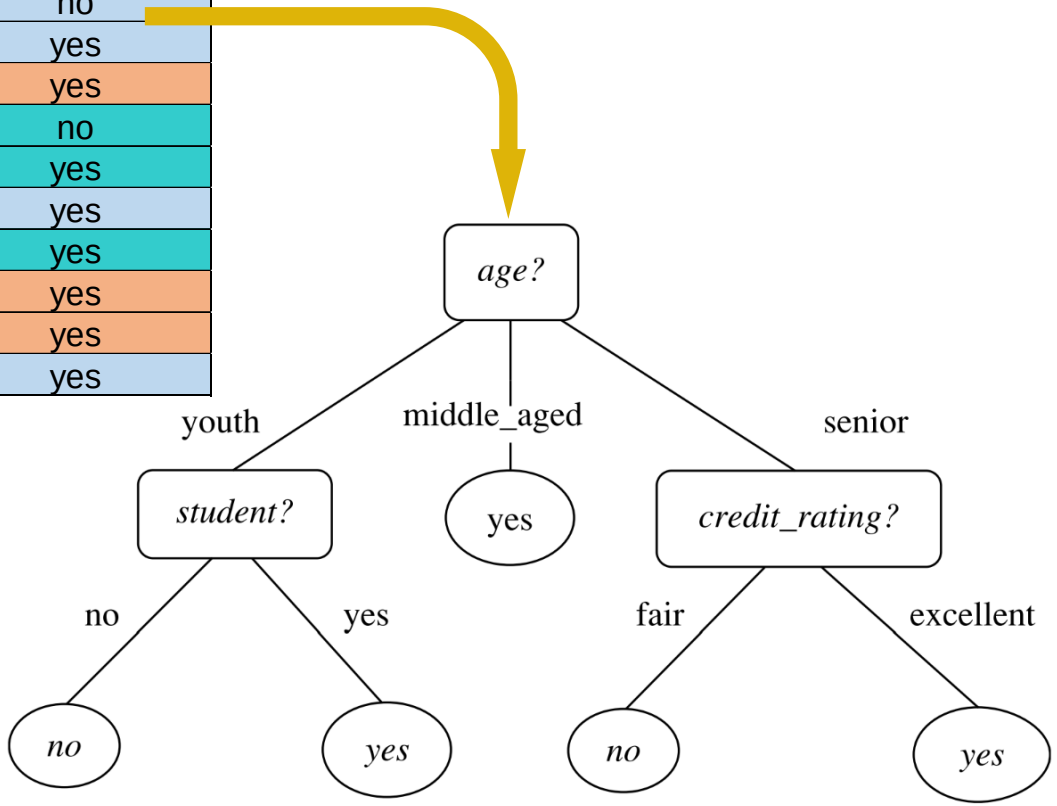
Decision Tree Induction

- **Basic algorithm (a greedy algorithm)**
  - Tree is constructed in a top-down, recursive, divide-and-conquer manner.
  - At start, all the training instances are at the root
  - Attributes are categorical
    - if continuous-valued, they are discretised in advance
  - Instances are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g. information gain)
- **Conditions for stopping partitioning**
  - All instances for a given node belong to the same class
  - no remaining attributes for further partitioning, or the decision tree reaches the maximum depth
    - majority voting is employed for classifying the leaf



From Data to Decision Trees

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_age	high	no	fair	yes
senior	medium	no	fair	no
senior	low	yes	fair	no
senior	low	yes	excellent	yes
middle_age	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_age	medium	no	excellent	yes
middle_age	high	yes	fair	yes
senior	medium	no	excellent	yes



Which attribute to use?

- Age, income, student, and credit_rating can be the attribute to split a node.

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_age	high	no	fair	yes
senior	medium	no	fair	no
senior	low	yes	fair	no
senior	low	yes	excellent	yes
middle_age	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_age	medium	no	excellent	yes
middle_age	high	yes	fair	yes
senior	medium	no	excellent	yes

Information Gain (IG)

- IG calculates effective change in entropy after making a decision based on the value of an attribute.

- For decision trees, it's ideal to base decisions on the attribute that provides the largest change in entropy, the attribute with the highest gain.

  – Information Gain for attribute A on set D is defined by taking the entropy of D and subtracting from it the summation of the entropy of each subset of D, determined by values of A, multiplied by each subset's proportion of D.

$$Info(D) = \sum_{i=1}^{m} (p(D_i|A = i) * Info(Di|A = i))$$

Gain Ratio (C4.5)

- Information gain measure is biased towards attributes with a large number of unique values.
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalisation to information gain)
  - $SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2(\frac{|D_j|}{|D|})$
  - $GainRatio(A) = Gain(A)/SplitInfo_A(D)$

- The attribute with the maximum gain ratio is selected as the splitting attribute

Gini Index (CART)

- If a data set D contains instances from n classes, gini index, $gini(D)$ is defined as
  - $gini(D) = 1 - \sum_{j=1}^{n} p_j^2$
  - where p_j is the relative frequency of class j in D.
- If a data set D is split on A into two subsets D_1 and D_2, the *gini* index $gini_A(D)$ is defined as
  - $gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$
- Reduction in impurity:
  - $gini_{split,A}(D) = \Delta gini(A) = gini(D) - gini_A(D)$
- The attribute provides the smallest $gini_{split}(D)$ or the largest reduction in impurity) is chosen to split the node.
- *need to enumerate all the possible splitting points for each attribute*

Lecture Outline

- Overview of Learning Models
- Model Evaluation
- Decision Trees
- **Frequent Pattern based Classification**
- k-Nearest Neighbours
- Other Common Machine Learning Models
  - Bayesian Classification
  - Artificial Neural Networks as Classifiers
  - Support Vector Machines
- Overfitting and underfitting

IF-THEN Rules for Classification

- Represent the knowledge in the form of **IF-THEN** rules

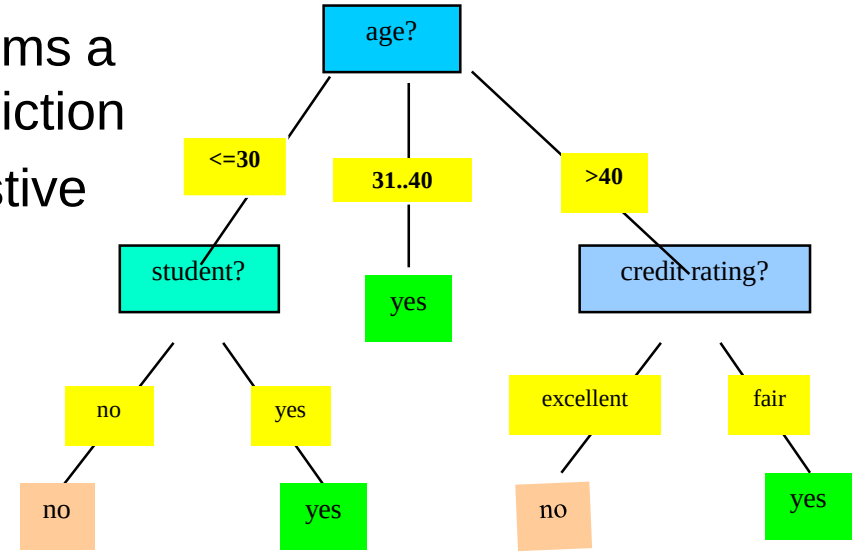
  R_1: IF *age* = youth AND *student* = yes THEN *buys_computer* = yes
  - Rule antecedent/precondition vs. rule consequent
- Assessment of a rule: *coverage* and *accuracy*
  - n_{covers} = # of tuples covered by R_1
  - $n_{correct}$ = # of tuples correctly classified by R_1

    $coverage(R_1) = n_{covers} / |D|$ /* D: training data set */

    $accuracy(R_1) = n_{correct} / n_{covers}$
- If more than one rule are triggered, need **conflict resolution**
  - **Size ordering**: assign the highest priority to the triggering rules that has the "toughest" requirement (i.e. with the *most attribute tests*)
  - **Class-based ordering**: decreasing order of *prevalence or misclassification cost per class*
  - **Rule-based ordering** (**decision list**): rules are organised into one long priority list, according to some measure of rule quality or by experts

Rule Extraction from a Decision Tree

- Rules are *easier to understand* than large trees
- 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 holds the class prediction
- Rules are mutually exclusive and exhaustive



- Example: Rule extraction from our *buys_computer* decision-tree

IF *age* = young AND *student* = *no* THEN *buys_computer* = *no*

IF *age* = young AND *student* = *yes* THEN *buys_computer* = *yes*

IF *age* = mid-age THEN *buys_computer* = *yes*

IF *age* = old AND *credit_rating* = *excellent* THEN *buys_computer* = *no*

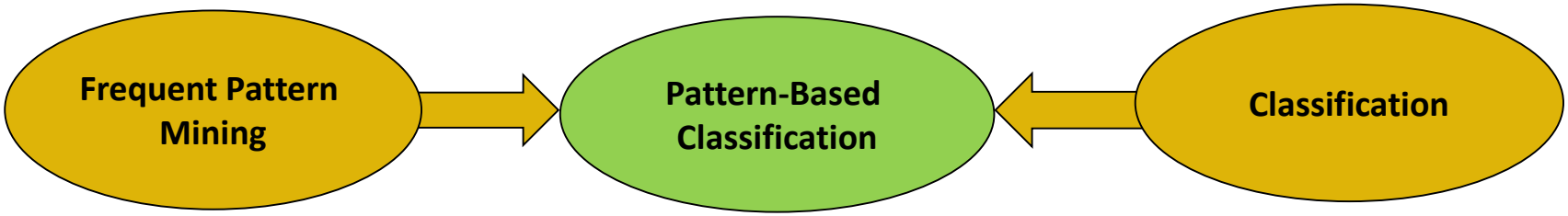
IF *age* = old AND *credit_rating* = *fair* THEN *buys_computer* = *yes*

Rule Induction: Sequential Covering Method

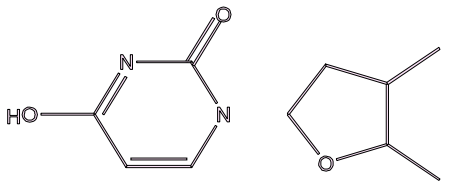
THE UNIVERSITY OF WESTERN AUSTRALIA

- Sequential covering algorithm: Extracts rules directly from training data
- Typical sequential covering algorithms: FOIL, AQ, CN2, RIPPER
- Rules are learned *sequentially*, each for a given class C_i will cover many instances of class C_i but none (or few) of the examples of other classes
- Steps:
  - Rules are learned one at a time
  - Each time a rule is learned, instances covered by the rules are removed
  - Repeat the process on the remaining instances until *termination condition*, e.g. when no more training instances or when the quality of a rule returned is below a user-specified threshold
- In comparison, decision-tree induction learns a set of rules *simultaneously*

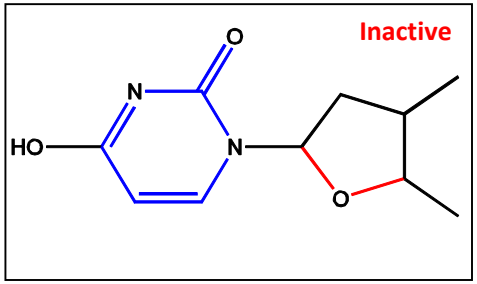
Pattern-based Classification, Why?

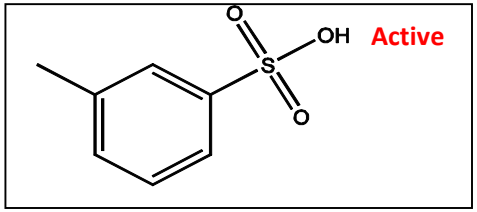


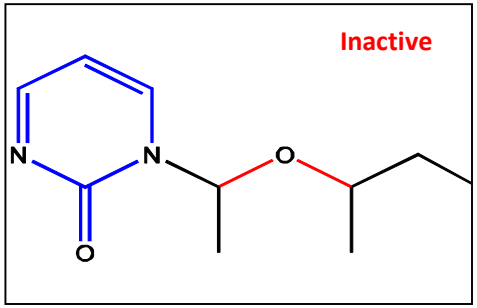
- **Pattern-based classification:** An integration of both themes
- **Why pattern-based classification?**
  - **Feature construction**
    - Higher order; compact; discriminative
    - E.g. single word → phrase (Apple pie, Apple i-pad)
  - **Complex data modeling**
    - Graphs (no predefined feature vectors)
    - Sequences
    - Semi-structured/unstructured Data



Pattern-based Classification on Graphs



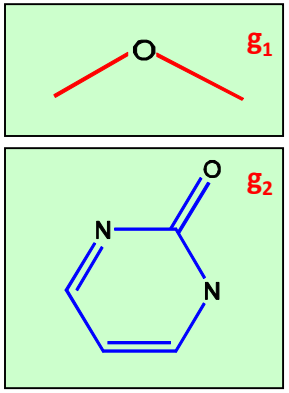




Mining

min_sup=2

Frequent subgraphs



Transform

Use frequent patterns as features for classification

g_1	g_2	Class
1	1	0
0	0	1
1	1	0

Classification based on Associations

- Method (Classification Based on Association [Liu, Hsu and Ma, KDD'98])
  - Mine high-confidence, high-support class association rules
  - LHS: conjunctions of attribute-value pairs; RHS: class labels

    $p_1 \wedge p_2 \ldots \wedge p_l \rightarrow$ "$A_{\text{class-label}} = C$" (confidence, support)
  - Rank rules in descending order of confidence and support
  - Classification: apply the first rule that matches a test case
  - Effectiveness: Often found more accurate than some traditional classification methods, such as C4.5
  - Why? — Exploring high confident associations among multiple attributes may overcome some constraints introduced by some classifiers that consider only one attribute at a time

Discriminative Pattern-based Classification

THE UNIVERSITY OF WESTERN AUSTRALIA

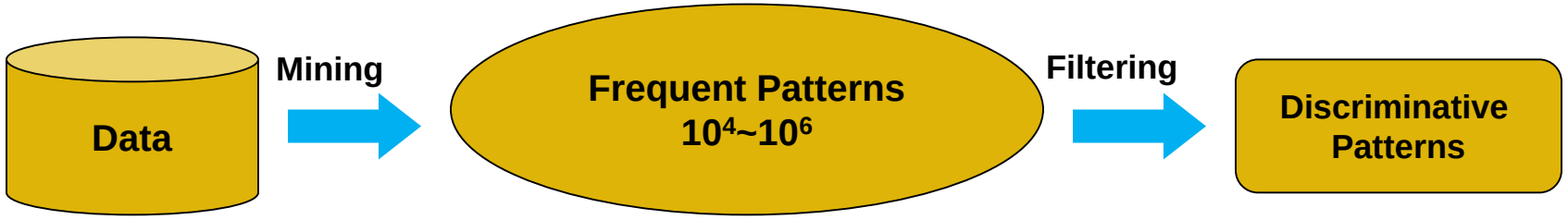
- Discriminative patterns as features for classification [Cheng et al., ICDE'07]
- **Principle:** Mining discriminative frequent patterns as high-quality features and then apply any classifier
- **Framework (PatClass)**
  - Feature construction by *frequent itemset mining*
  - Feature selection (e.g. using Maximal Marginal Relevance (MMR))
    - Select discriminative features (i.e. that are relevant but minimally similar to the previously selected ones)
    - Remove redundant or closely correlated features
  - Model learning
    - Apply a general classifier, such as SVMs, to build a classification model

K-itemsets are often more informative than single features (1-itemsets) in classification

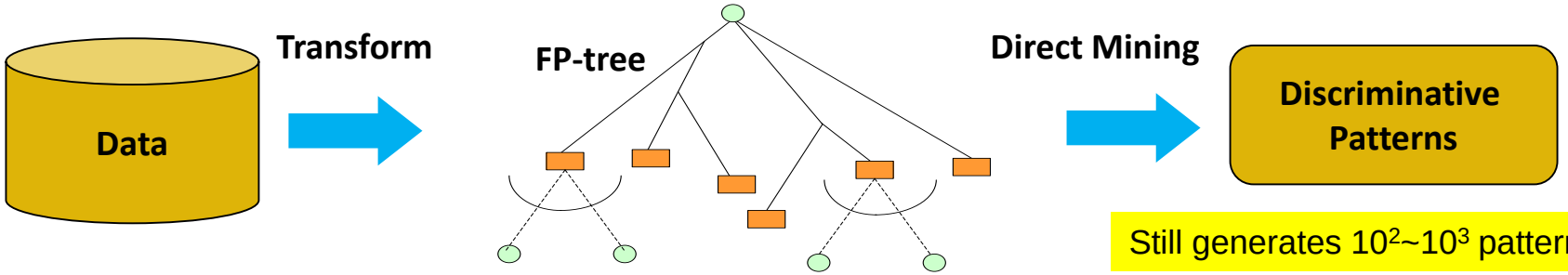
Mining Concise Set of Discriminative Patterns

THE UNIVERSITY OF WESTERN AUSTRALIA

Frequent pattern mining, then getting discriminative patterns: Expensive, large model



DDPMine [Cheng et al., ICDE'08]: Direct mining of discriminative patterns: Efficient



Still generates 10^2~10^3 patterns

DPClass [Shang et al, SDM'16]: A better solution—Efficient, effective, and generating a very limited number of (such as only 20 or so) patterns

A Comparison on Classification Accuracy

THE UNIVERSITY OF WESTERN AUSTRALIA

- DPClass: Discriminative & frequent at the same time, then select top-k
  - Only top-20 patterns are used in DPClass
- Two methods on pattern selection
  - Forward vs. LASSO
  - DPClass has higher accuracy than DDPMine and Random Forest

Dataset	DPClass (Forward)	DPClass (LASSO)	DDPMine	Random Forest
adult	**85.66%**	84.33%	83.42%	85.45%
hypo	**99.58%**	99.28%	92.69%	97.22%
sick	98.35%	**98.87%**	93.82%	94.03%
crx	**89.35%**	87.96%	87.96%	**89.35%**
sonar	**85.29%**	83.82%	73.53%	83.82%
chess	92.25%	92.05%	90.04%	**94.22%**
namao	97.17%	96.94%	96.83%	**97.86%**
musk	95.92%	95.71%	93.29%	**96.60%**
madelon	74.50%	**76.00%**	59.84%	56.50%

- An extension of DPClass has been applied to health study
  - Cheng et al, "Mining Discriminative Patterns to Predict Health Status for Cardiopulmonary Patients", *ACM-BCB'16*

Lecture Outline

- Overview of Learning Models
- Model Evaluation
- Decision Trees
- Frequent Pattern based Classification
- **k-Nearest Neighbours**
- Other Common Machine Learning Models
  - Bayesian Classification
  - Artificial Neural Networks as Classifiers
  - Support Vector Machines
- Overfitting and underfitting

Lazy vs. Eager Learning

THE UNIVERSITY OF WESTERN AUSTRALIA

- Lazy vs. eager learning
  - **Lazy learning** (e.g. instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - **Eager learning** (the above discussed methods): Given a set of training instances, constructs a classifier before receiving new (e.g. test) data to classify
- Lazy: less time in training but more time in predicting
- Accuracy
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form an implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space

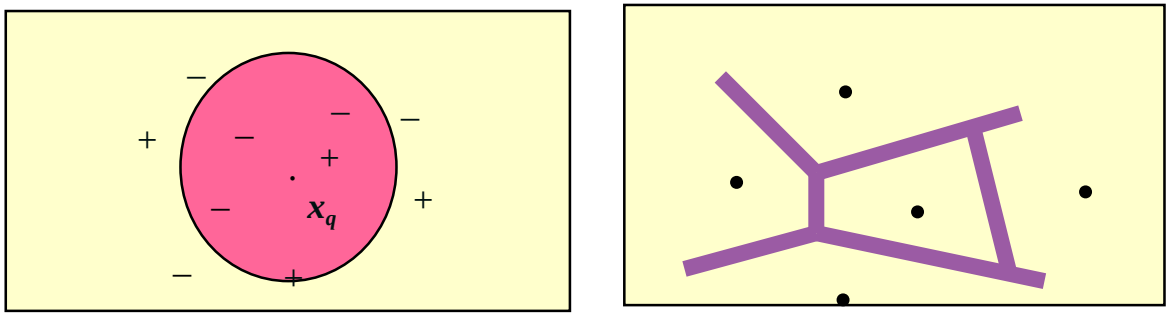
Lazy Learner: Instance-Based Methods

- Instance-based learning:
  - Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified
- Typical approaches
  - *k*-nearest neighbor approach
    - Instances represented as points in a Euclidean space.
  - Locally weighted regression
    - Constructs local approximation
  - Case-based reasoning
    - Uses symbolic representations and knowledge-based inference

The *k*-Nearest Neighbor Algorithm

THE UNIVERSITY OF WESTERN AUSTRALIA

- All instances correspond to points in the n-D space
- Nearest neighbours are defined in terms of Euclidean distance, dist($\mathbf{X_1}$, $\mathbf{X_2}$)
- Target function could be discrete- or real- valued
- For discrete-valued, *k*-NN returns the most common value among the *k* training examples nearest to x_q
- Vonoroi diagram: the decision surface induced by 1-NN for a typical set of training instances



Discussion on the *k*-NN Algorithm

THE UNIVERSITY OF WESTERN AUSTRALIA

- *k*-NN for real-valued prediction for a given unknown instance
  - Returns the mean values of the *k* nearest neighbors
- Distance-weighted nearest neighbor algorithm
  - Weight the contribution of each of the *k* neighbors according to their distance to the query x_q
    - Give greater weight to closer neighbors

$$w \equiv \frac{1}{d(x_q, x_i)^2}$$

- Robust to noisy data by averaging *k*-nearest neighbors
- Curse of dimensionality: distance between neighbours could be dominated by irrelevant attributes
  - To overcome it, axes stretch or elimination of the least relevant attributes

Case-Based Reasoning (CBR)

THE UNIVERSITY OF WESTERN AUSTRALIA

- **CBR**: Uses a database of problem solutions to solve new problems
- Store symbolic description (e.g. cases)—not points in a Euclidean space
- Applications: Customer-service (product-related diagnosis), legal ruling
- Methodology
  - Instances represented by rich symbolic descriptions (e.g. function graphs)
  - Search for similar cases, multiple retrieved cases may be combined
  - Tight coupling between case retrieval, knowledge-based reasoning, and problem solving
- Challenges
  - Find a good similarity metric
  - Indexing based on syntactic similarity measure, and when failure, backtracking, and adapting to additional cases

Lecture Outline

- Overview of Learning Models
- Model Evaluation
- Decision Trees
- Frequent Pattern based Classification
- k-Nearest Neighbours
- **Other Common Machine Learning Models**
  - Bayesian Classification
  - Artificial Neural Networks as Classifiers
  - Support Vector Machines
- Overfitting and underfitting

Bayesian Classification

- **A statistical classifier:**
  - performs *probabilistic prediction, i.e.* predicts class membership probabilities
- **Foundation:**
  - Based on Bayes' Theorem.
- **Performance:**
  - A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
- **Incremental:**
  - Each training instance can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- **Standard:**
  - Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Training Set and Class Labels

- Let D be a set of training instances and their associated class labels, and each instance is represented by an n-D dimensional vector $X = (x_1, x_2, \ldots, x_n)$. The label of X is denoted by Y.

- Suppose there are *m* classes C_1, C_2, …, C_m.

Bayes' Theorem: Basics

- **Total probability Theorem:** $\boldsymbol{P(B) = \sum_{i=1}^{v} P(B|A_i)P(A_i)}$

- **Bayes' Theorem:** $\boldsymbol{P(Y = Ci|X) = \frac{P(X|Y = Ci)P(Y=Ci)}{P(X)}}$
  - Let **X** be an instance ("*evidence*"): class label is unknown
  - "$\boldsymbol{Y = Ci}$" means that X belongs to class $\boldsymbol{C_i}$.
  - Classification is to determine $P(Y = \boldsymbol{C_i}|\boldsymbol{X})$, (i.e. *posteriori probability):* the probability of "$\boldsymbol{Y = Ci}$" given the observation $\boldsymbol{X}$
  - $P(Y = \boldsymbol{C_i})$ (*prior probability*): the initial probability
    - E.g., $\boldsymbol{X}$ will buy computer, regardless of age, income, …
  - $P(\boldsymbol{X})$: probability that the instance $\boldsymbol{X}$ is observed
  - $P(\boldsymbol{X}|\boldsymbol{Y = Ci})$ (likelihood): the probability of observing the instance $\boldsymbol{X}$, given that $\boldsymbol{Y = Ci}$
    - E.g., Given $\boldsymbol{X}$ will buy computer, the probability that $\boldsymbol{X}$ is 31..40, medium income

Prediction Based on Bayes' Theorem

- Given instance X, *posteriori probability of a hypothesis* $Y = C_i$, $P(Y = C_i|X)$, follows the Bayes' theorem
  - $P(Y = C_i|X) = \frac{P(X|Y = C_i)P(Y=C_i)}{P(X)}$

- Informally, this can be viewed as

$$posterior = likelihood \times prior/evidence$$

- Predicts X belongs to C_i iff the probability $P(Y = C_i|X)$ is the highest among all the $P(Y = C_k|X)$ for all the *k* classes

Classification Is to Derive the Maximum Posteriori

- Suppose there are *m* classes C_1, C_2, …, C_m.
- Classification is to derive the maximum posteriori, i.e., the maximal $P(Y = C_i|X)$
- This can be derived from Bayes' theorem
  - $P(Y = C_i|X) = \frac{P(X|Y = C_i)P(Y = C_i)}{P(X)}$
- Since $P(X)$ is constant for all classes, only
  - $\boldsymbol{P}(Y = C_i|\boldsymbol{X}) \propto \boldsymbol{P}(\boldsymbol{X}|Y = C_i)\boldsymbol{P}(Y = C_i)$

  needs to be maximised

Bayes' Theorem: Summary

- **Total probability Theorem**:

$$P(B) = \sum_{i=1}^{M} P(B|A_i)P(A_i)$$

- **Bayes' Theorem**:

$$P(Y = Ci|X) = \frac{P(X|Y = Ci)P(Y = Ci)}{P(X)} \propto P(X|Y = Ci)P(Y = Ci)$$

posteriori probability — What we should choose

likelihood — What we just see

prior probability — What we knew previously

  - **X:** an instance ("*evidence*")
  - $Y = Ci$: **X** belongs to class C_i

Prediction can be done based on Bayes' Theorem:

Classification is to derive the maximum posteriori

Practical difficulty: It requires initial knowledge of many probabilities, involving significant computational cost.

Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence between attributes):
  - $P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_1|C_i) \times \cdots \times P(x_n|C_i)$
  - Where $P(x_k|C_i)$ is shorthand for $P(x_k|Y = C_i)$

- This greatly reduces the computation cost: Only counts the class distribution
  - If the *k*-th attribute A_k is categorical, $P(x_k|Ci)$ is the # of instances in C_i having value x_k for A_k divided by $|Ci_{,D}|$ (# of instances of C_i in D)
  - If A_k is continuous-valued, $P(x_k|Ci)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ
    - $g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
  - and $P(x_k|C_i) = g(xk, \mu_{C_i}, \sigma_{C_i})$

Dealing with zero probabilities

- Naïve Bayesian prediction requires each conditional probability be non-zero. Otherwise, the predicted probability will be zero.
  - $P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_1|C_i) \times \cdots \times P(x_n|C_i)$
  - Ex. Suppose a dataset with 1000 instances, income=low (0), income= medium (990), and income = high (10)
- Use Laplacian correction (or Laplacian estimator)
  - *Adding 1 to each case*

    Prob(income = low) = 1/1003

    Prob(income = medium) = 991/1003

    Prob(income = high) = 11/1003
  - The "corrected" probability estimates are close to their "uncorrected" counterparts.

Naïve Bayes Classifier: Comments

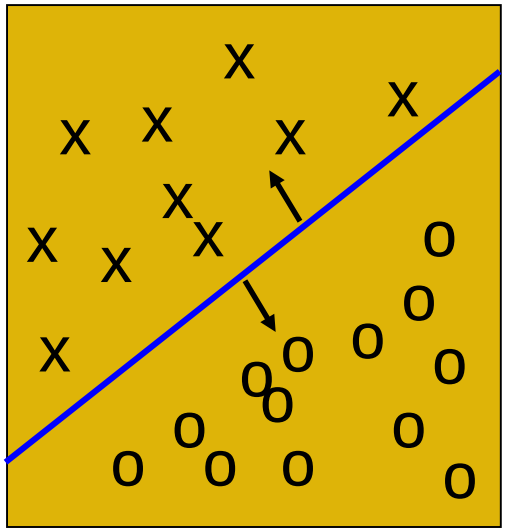
- **Advantages**
  - Easy to implement
  - Good results obtained in most of the cases
- **Disadvantages**
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among attributes
    - E.g. Patients: Profile: age, family history, etc.
      Symptoms: fever, cough, etc.,
      Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayes Classifier
- **(Aside) How to deal with these dependencies? Bayesian Belief Networks.**

Lecture Outline

- Overview of Learning Models
- Model Evaluation
- Decision Trees
- Frequent Pattern based Classification
- k-Nearest Neighbours
- Other Common Machine Learning Models
  - Bayesian Classification
  - **Artificial Neural Networks as Classifiers**
  - Support Vector Machines
- Overfitting and underfitting

Classification: A Mathematical Mapping

- **Classification: predicts categorical class labels**
  - E.g. $\boldsymbol{x}_i = (x_1, x_2, x_3, \ldots), y = +1 \; or \; -1$
- **Mathematically,** $x \in X = \boldsymbol{R}^n, y \in Y = \{+1, -1\}$,
  - We want to derive a function $f: X \longrightarrow Y$
- **Linear Classification**
  - Binary Classification problem
  - Formulate a linear discriminant **hyperplane**.
  - Data above the blue line belongs to class 'x'
  - Data below blue line belongs to class 'o'
  - Examples: SVM, Perceptron, Probabilistic Classifiers

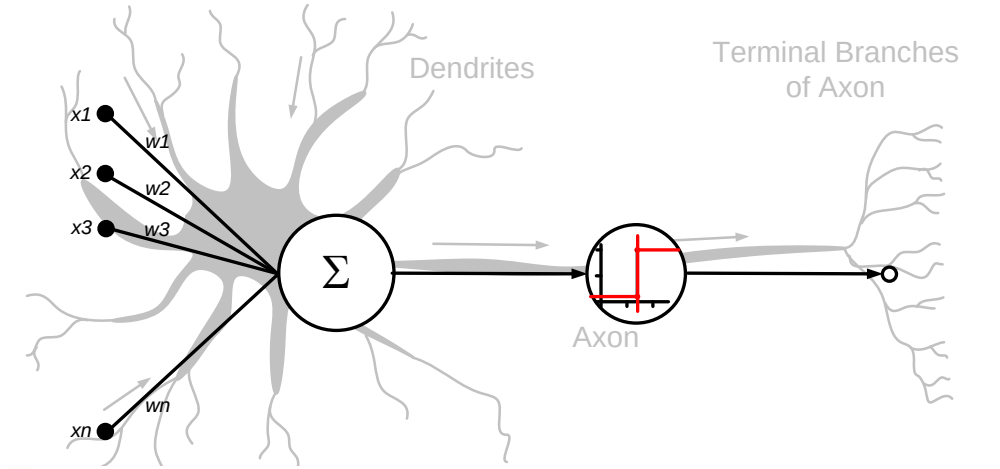


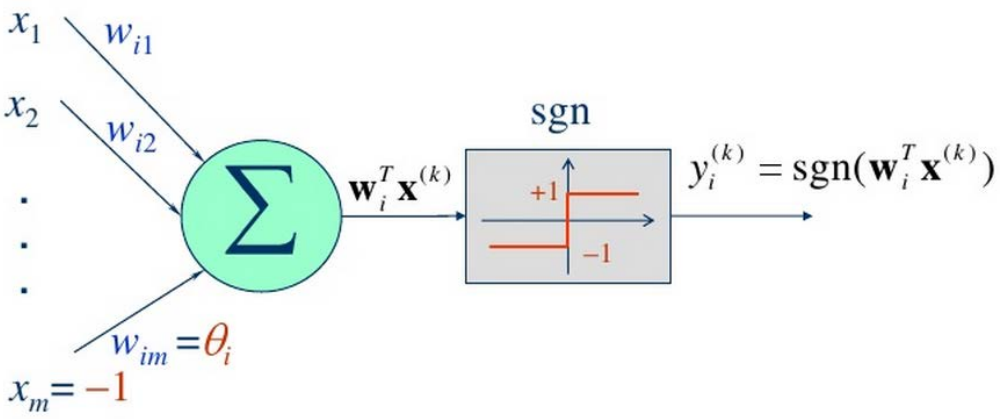
What is Neural Computing

- ANN (artificial neural network) is a model inspired by biological neural network.
- Network functions collectively and in massive parallelism.
  - Good learning ability
  - Adaptive

A single perceptron

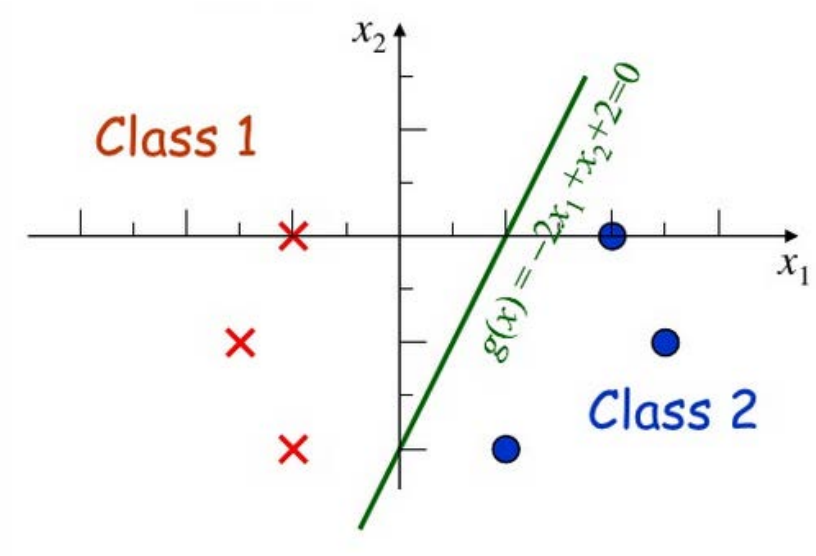
- **Output is scaled sum of inputs.**
  - Sensory Unit, Association Unit and Response Unit

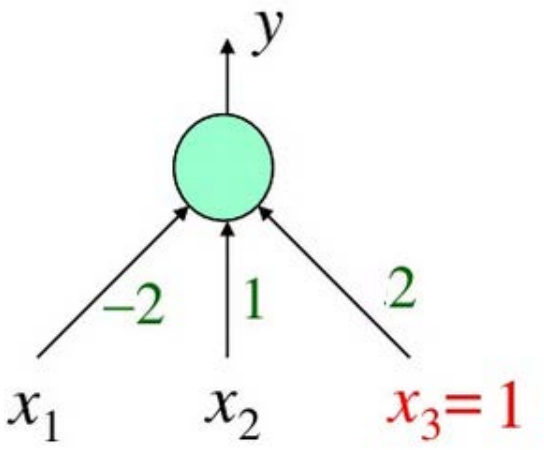




Case 1: Binary Class Linearly Separable

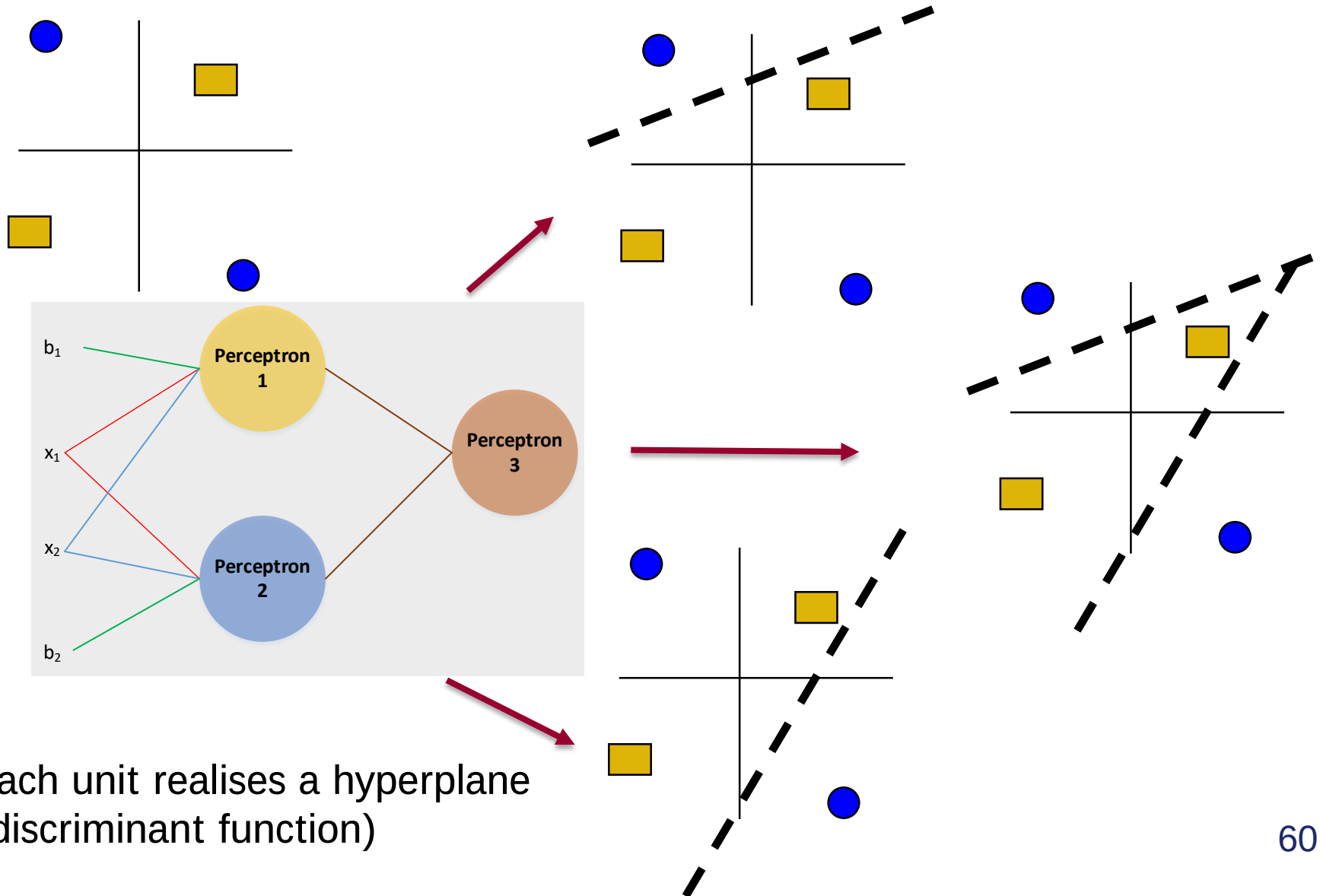
$$Class\ 1\ (+)\ :\ \{\,[-1,0],[-1.5,-1],[-1,-2]\}$$
$$Class\ 2\ (-):\ \{\,[2,0],[2.5,-1],[1,-2]\}$$





➢Without the bias decision boundary passes through the origin.

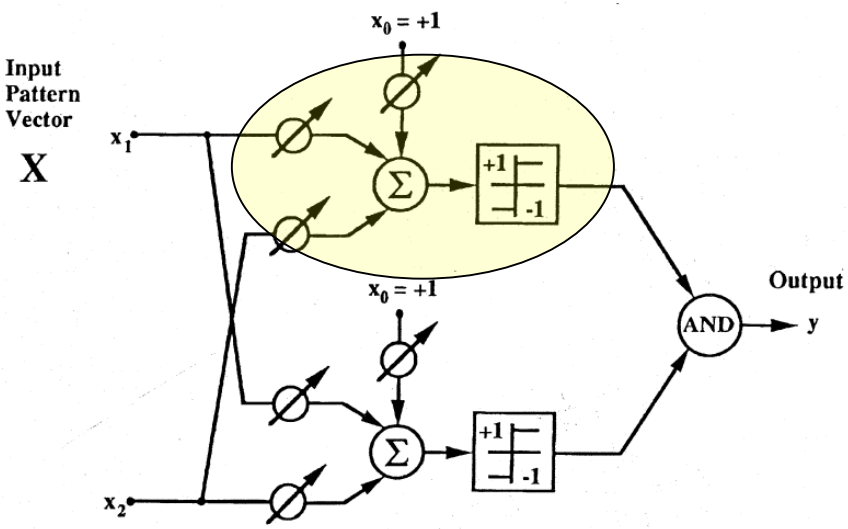
Case II: Binary Class non-linearly separated



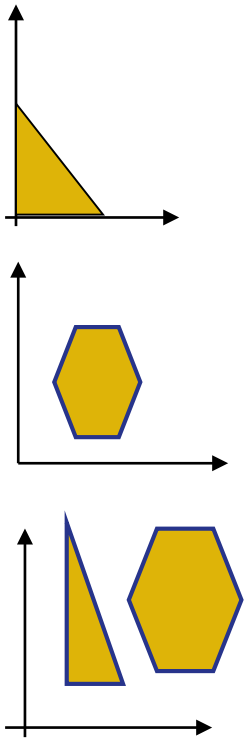
Each unit realises a hyperplane (discriminant function)

What do the multiple layers do?

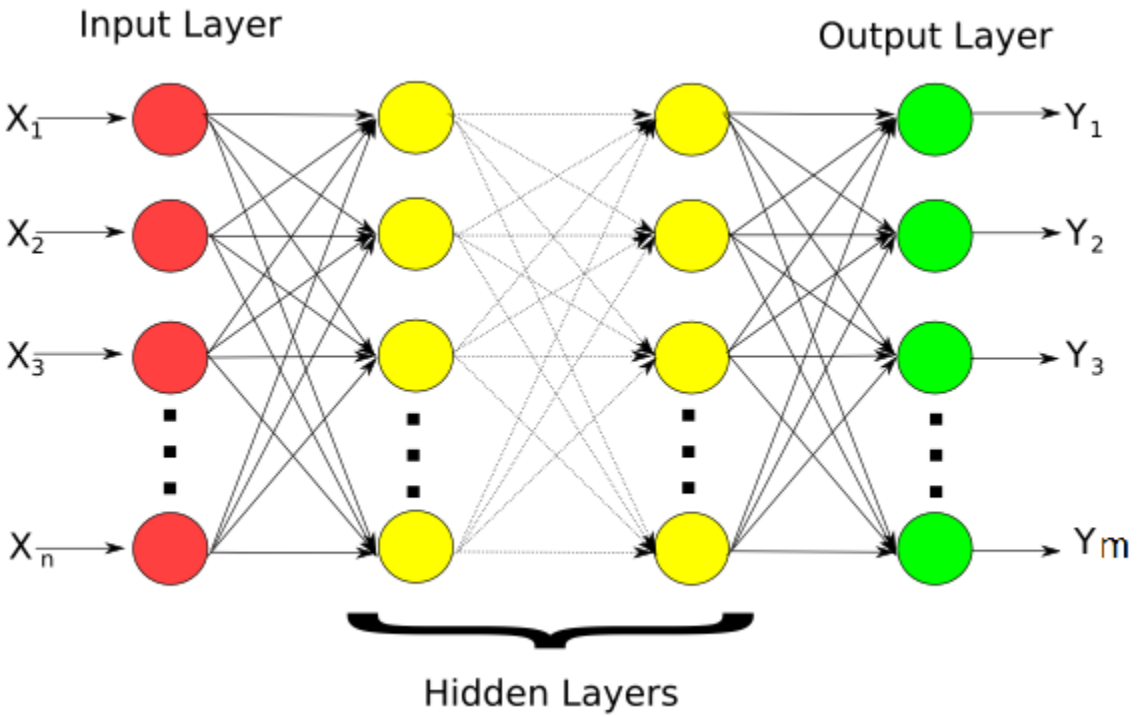
- **Neural networks offer a very powerful and very general framework for**
  - representing non-linear mappings from several input variables to several output variables
  - where the form of the mapping is governed by a number of adjustable weight and bias parameters.



- 1st layer draws linear boundaries;
- 2nd layer combines the boundaries
- More layers for arbitrarily complex boundaries



Feed-forward neural network topology



Neural Network as a Classifier

- **Weakness**
  - Long training time
  - Require a number of parameters typically best determined empirically, e.g. the network topology or "structure", initial values of the weights
  - Poor interpretability: Difficult to interpret the symbolic meaning behind the learned weights and of "hidden units" in the network
- **Strength**
  - High tolerance to noisy data
  - Ability to classify untrained patterns
  - Well-suited for continuous-valued inputs *and outputs*
  - Successful on an array of real-world data, e.g., hand-written letters
  - Algorithms are inherently parallel
  - Techniques have recently been developed for the extraction of rules from trained neural networks

Lecture Outline

- Overview of Learning Models
- Model Evaluation
- Decision Trees
- Frequent Pattern based Classification
- k-Nearest Neighbours
- Other Common Machine Learning Models
  - Bayesian Classification
  - Artificial Neural Networks as Classifiers
  - **Support Vector Machines**
- Overfitting and underfitting

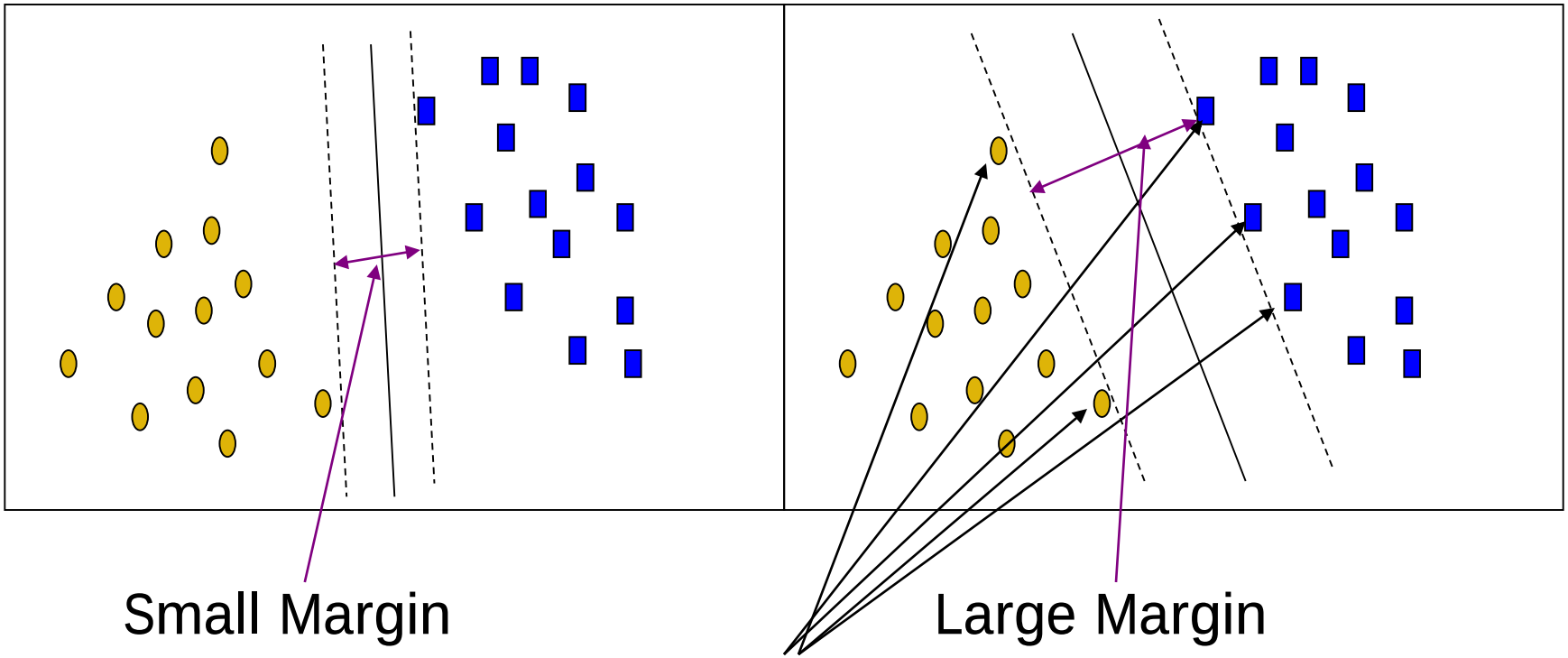
SVM—Support Vector Machines

- It uses a nonlinear mapping to transform the original training data into a higher dimension if required.
- With the new dimension, it searches for the linear optimal separating hyperplane (i.e., “decision boundary”).
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane.
- SVM finds this hyperplane using support vectors (“essential” training instances) and margins (defined by support vectors)

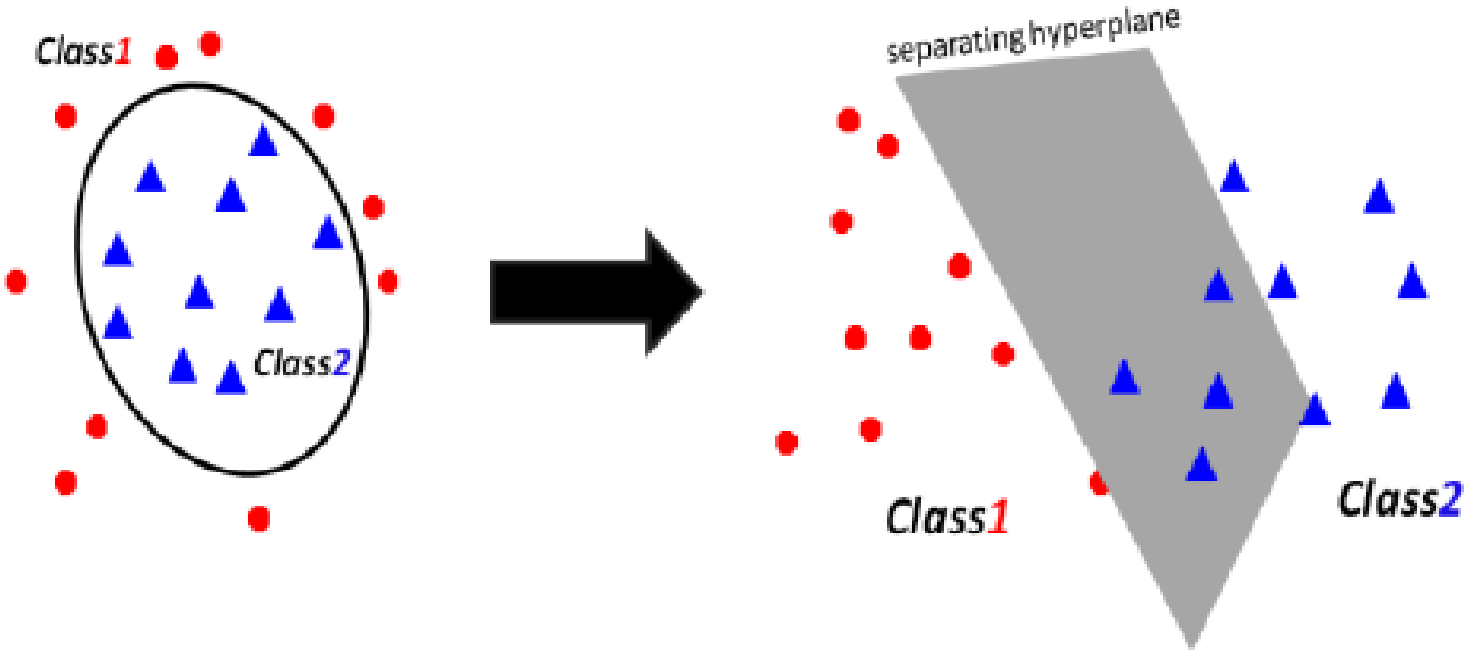
General Philosophy

Infinite number of answers!

Which one is the best?

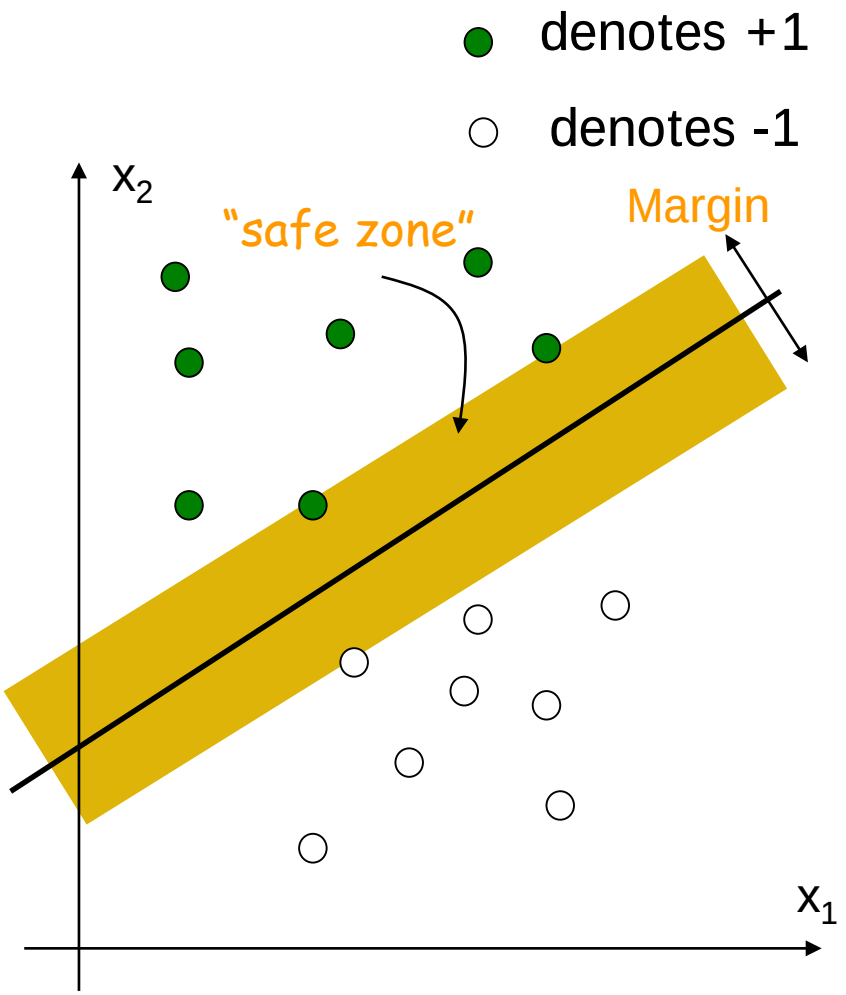


Support Vectors


Class1
Class2
separating hyperplane
Class1
Class2

Largest Margin Classifier

- The linear discriminant function (classifier) with the maximum margin is the “best”
- Margin is defined as the width that the boundary could be shifted by before hitting a data point
- Why it is the best?
  - Robust to noise and outliers and thus strong generalisation ability

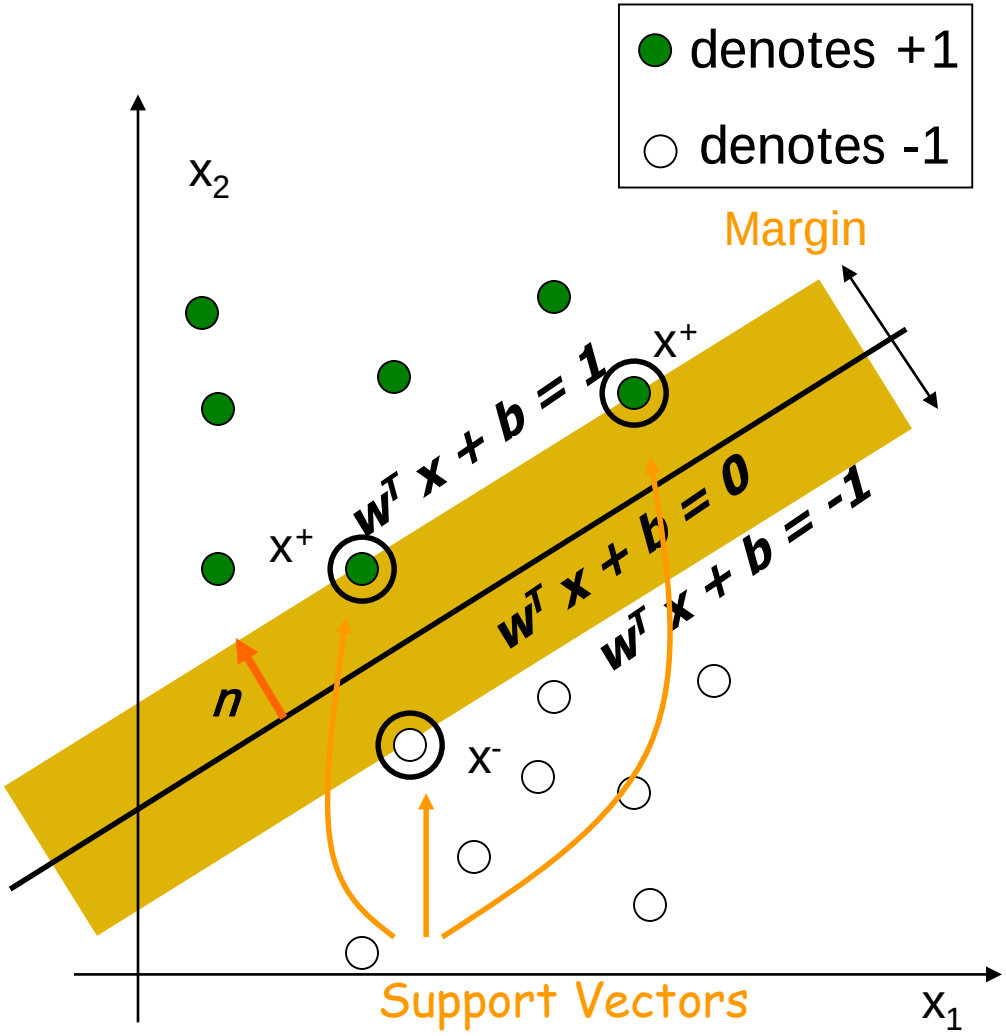


Large Margin Linear Classifier

- **We know that**
  - $\boldsymbol{w}^T x^+ + b = 1$
  - $\boldsymbol{w}^T x^- + b = -1$
- The margin width is
  - $M = (x^+ - x^-) \cdot n$

$$= (x^+ - x^-) \cdot \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} = \frac{2}{\|\boldsymbol{w}\|}$$

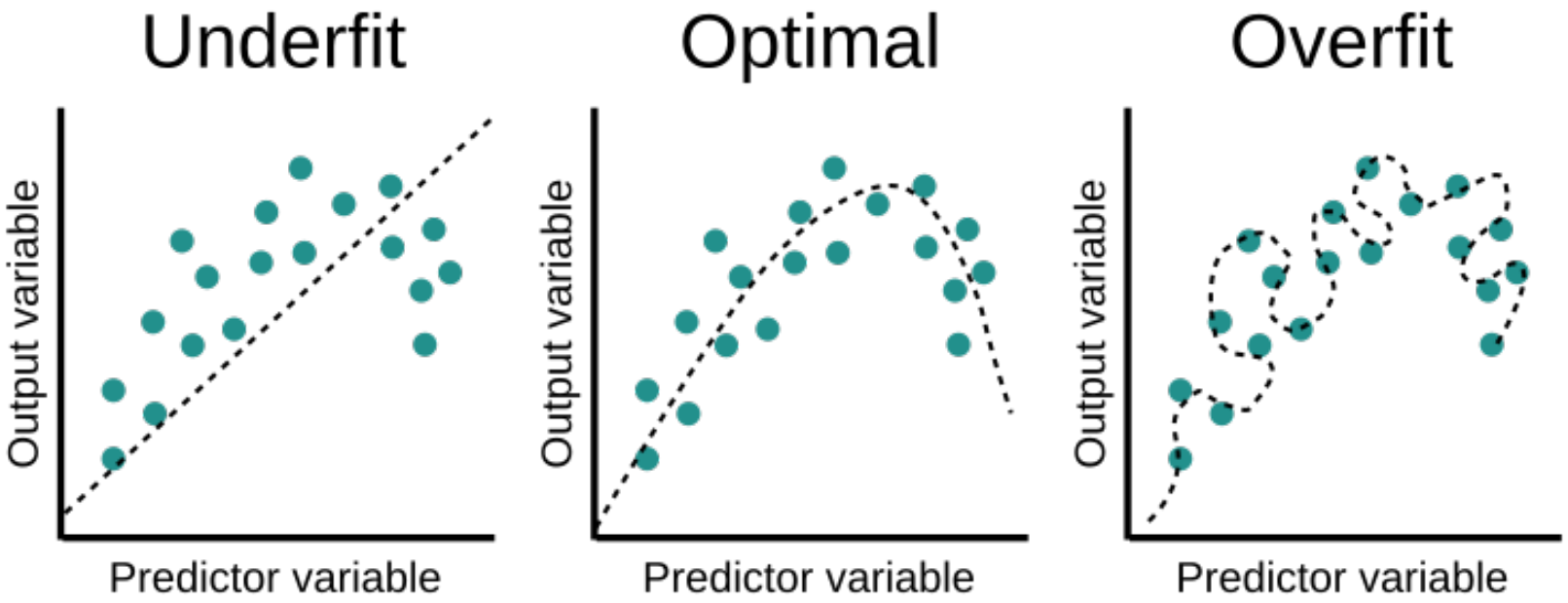
- Formulation
  - Maximise $\frac{2}{\|\boldsymbol{w}\|}$
  - Minimise $\frac{1}{2}\|\boldsymbol{w}\|^2$

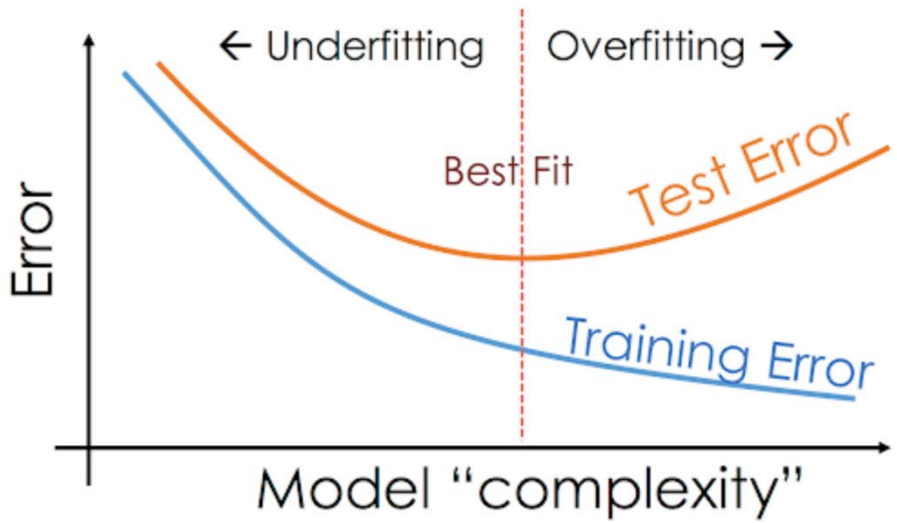


Lecture Outline

- Overview of Learning Models
- Model Evaluation
- Decision Trees
- Frequent Pattern based Classification
- k-Nearest Neighbours
- Other Common Machine Learning Models
  - Bayesian Classification
  - Artificial Neural Networks as Classifiers
  - Support Vector Machines
- **Overfitting and underfitting**

Overfitting





Overfitting and Tree Prunning

- **Overfitting: An induced tree may overfit the training data**
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen instances
- **Two approaches to avoid overfitting**
  - Prepruning: *Halt tree construction early*—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: *Remove branches* from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"

Reference

- Han et al.'s book
  - Chapter 8 and Chapter 9.
- Readings
  - A brief history of machine learning
  - Stratified cross-validation.
  - .632 bootstrap from page 43
  - ThunderSVM: the state-of-art library for SVMs