COMP3420: Advanced Databases and Data Mining

Classification and prediction: Rule-based classification and support vector machines

Lecture outline

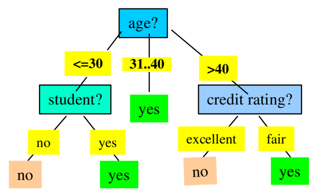
- IF-THEN rule classification
  - Rule extraction from a decision tree
  - Rule extraction from training data
  - Sequential covering algorithm
- Classification: a mathematical mapping
- Linear classification
- Support vector machines
  - History and applications, general philosophy
  - Margins and support vectors
  - Linearly separable and inseparable
  - Kernel functions
  - SVM versus neural network

Using IF-THEN rules for classification

- Represent the knowledge in the form of *IF-THEN* rules
  - Rule *R*: IF age="youth" AND student="yes" THEN buys_computer ="yes"
  - Rule antecedent/precondition versus rule consequent
- Assessment of a rule: *coverage* and *accuracy*
  - n_{covers} = Number of tuples (records) covered by a rule *R*
  - $n_{correct}$ = Number of tuples correctly classified by a rule *R*
  - $coverage(R) = n_{covers} / |D|$ and $accuracy(R) = n_{correct} / n_{covers}$
    (with *D* the training data set and | | = number of records)
- If more than one rule is triggered, need *conflict resolution*
  - Size ordering: assign the highest priority to the triggering rule 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* <= 30 AND *student* = *no* THEN *buys_computer* = *no*
  IF *age* <= 30 AND *student* = *yes* THEN *buys_computer* = *yes*
  IF *age* = 31..40 THEN *buys_computer* = *yes*
  IF *age* > 40 AND *credit_rating* = *excellent* THEN *buys_computer* = *no*
  IF *age* > 40 AND *credit_rating* = *fair* THEN *buys_computer* = *yes*

Rule extraction from training data

- 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 tuples of C_i but none (or few) of the tuples of other classes
  - Rules are learned one at a time
  - Each time a rule is learned, the tuples covered by the rules are removed
  - The process repeats on the remaining tuples unless *termination condition*, for example, when no more training examples or when the quality of a rule returned is below a user-specified threshold
- Decision-tree induction: learning a set of rules *simultaneously*

Sequential covering algorithm

Algorithm: Sequential covering. Learn a set of IF-THEN rules for classification

Input:

- *D*, a data set class-labeled tuples;
- *Att-vals*, the set of all attributes and their possible values.

Output: A set of IF-THEN rules.

Method:

```
Rule_set = { }; // initial set of rules learned is empty
for each class c do
    repeat
        Rule = Learn_One_Rule(D, Att–vals, c);
        remove tuples covered by Rule from D;
    until terminating condition;
    Rule_set = Rule_set + Rule // add new rule to rule set
endfor
return Rule_set;
```

Source: Han and Kamber, DM Book, 2nd Ed. (Copyright © 2006 Elsevier Inc.)

How to learn-one-rule?

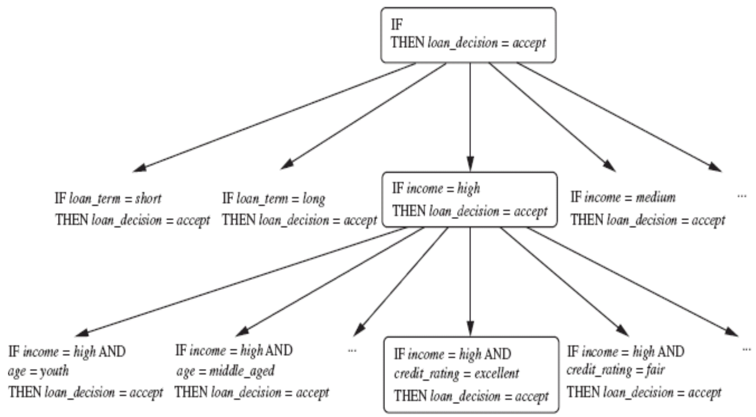
- Start with the most general rule possible: *condition = empty*
- Adding new attributes by adopting a greedy depth-first strategy
  - Picks the one that most improves the rule quality
- Rule-quality measures: consider both *coverage* and *accuracy*
  - *Foil$_{gain}$* (in *FOIL* and *RIPPER*): assesses *info_gain* by extending condition (*pos'*, *neg'* positive and negative tuples covered by the extended rule *R'*)

$$FOIL_{Gain} = pos' \times \left(\log_2 \frac{pos'}{pos'+neg'} - \log_2 \frac{pos}{pos+neg}\right)$$

  - It favors rules that have high accuracy and cover many positive tuples
- Rule pruning based on an independent set of test tuples
  - *pos/neg* are number of positive/negative tuples covered by rule *R*
  - If *FOIL$_{Prune}$* is higher for the pruned version of *R*, prune *R*

$$FOIL_{Prune}(R) = \frac{pos - neg}{pos + neg}$$

General to specific search through rule space

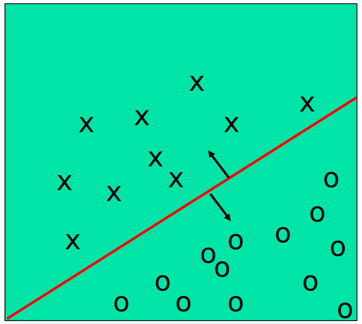


Source: Han and Kamber, DM Book, 2nd Ed. (Copyright © 2006 Elsevier Inc.)

Classification: A mathematical mapping

- Classification:
  - Predicts categorical class labels
- For example, personal homepage classification
  - $x_i = (x_1, x_2, x_3, \ldots), y_i = +1$ or -1
  - x_1 : number of occurrences of a word "homepage"
  - x_2 : number of occurrences of a word "welcome"
- Mathematically
  - $x \in X = \Re^n, y \in Y = \{+1, -1\}$
  - We want a function $f: X \rightarrow Y$

Linear classification



- Binary classification problem
- The data above the red line belongs to class 'x'
- The data below red line belongs to class 'o'
- Examples: Support vector machines (SVM), perceptron, probabilistic classifiers

Source: Han and Kamber, DM Book, 2nd Ed. (Copyright © 2006 Elsevier Inc.)

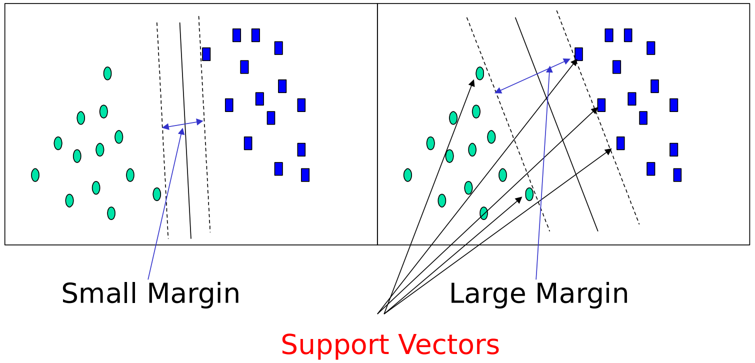
SVM—Support vector machines

- A new classification method for both linear and nonlinear data
- It uses a *nonlinear mapping* to transform the original training data into a higher dimension
- 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 tuples) and margins (defined by the support vectors)

SVM—History and applications

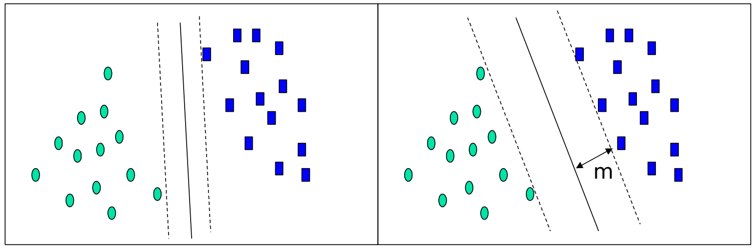
- Vapnik and colleagues (1992)—groundwork from Vapnik and Chervonenkis' statistical learning theory in 1960s
- Features: training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximisation)
- Used both for classification and prediction
- Applications: Handwritten digit recognition, object recognition, speaker identification, Web page classification, text classification, etc.

SVM—General philosophy



Source: Han and Kamber, DM Book, 2nd Ed. (Copyright © 2006 Elsevier Inc.)

SVM—When data is linearly separable



- Let data *D* be ($\boldsymbol{X}_1$, y_1), …, ($\boldsymbol{X}_{|D|}$, $y_{|D|}$), where $\boldsymbol{X}_i$ is the set of training tuples associated with the class labels y_i
- There are infinite lines (hyperplanes) separating the two classes, but we want to find the best one (the one that minimises classification error on unseen data)
- SVM searches for the hyperplane with the largest margin, i.e., *maximum marginal hyperplane* (MMH)

SVM—Linearly separable

- A separating hyperplane can be written as $\boldsymbol{W} \cdot \boldsymbol{X} + b = 0$ (dot product), where $\boldsymbol{W}=\{w_1, w_2, \dots, w_n\}$ is a weight vector and *b* a scalar (bias)
  - For 2-D it can be written as $w_0 + w_1 x_1 + w_2 x_2 = 0$
- The hyperplane defining the sides of the margin:

  H_1: $w_0 + w_1 x_1 + w_2 x_2 \geq 1$ for $y_i = +1$, and

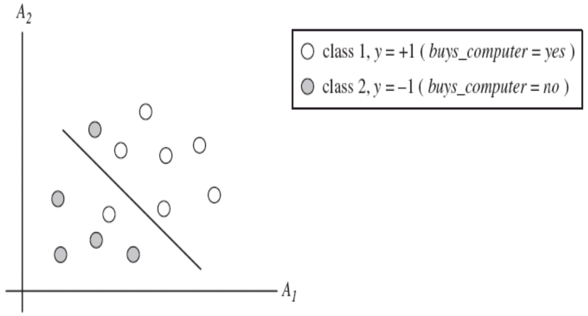
  H_2: $w_0 + w_1 x_1 + w_2 x_2 \leq -1$ for $y_i = -1$
- Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are *support vectors*
- This becomes a *constrained (convex) quadratic optimisation* problem: Quadratic objective function and linear constraints -> *Quadratic Programming (QP)* problem

Why is SVM effective on high dimensional data?

- The complexity of a trained classifier is characterised by the number of support vectors rather than the dimensionality of the data
- The support vectors are the essential or critical training examples, they lie closest to the decision boundary (MMH)
- If all other training examples are removed and the training is repeated, the same separating hyperplane would be found
- The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the SVM classifier, which is independent of the data dimensionality
- Thus, an SVM with a small number of support vectors can have good generalisation, even when the dimensionality of the data is high

SVM—Linearly inseparable

- Transform the original input data into a higher dimensional space
- Search for a linear separating hyperplane in the new space



Source: Han and Kamber, DM Book, 2nd Ed. (Copyright © 2006 Elsevier Inc.)

SVM—Kernel functions

- Instead of computing the dot product on the transformed data tuples, it is mathematically equivalent to instead applying a *kernel function* $K(\boldsymbol{X}_i, \boldsymbol{X}_j)$ to the original data, i.e., $K(\boldsymbol{X}_i, \boldsymbol{X}_j) = \Phi(\boldsymbol{X}_i) \cdot \Phi(\boldsymbol{X}_j)$
  - $\Phi(\boldsymbol{X}_i)$ is a non-linear mapping function applied to transform training tuples
- Typical Kernel Functions

$$\text{Polynomial kernel of degree } h: \quad K(X_i, X_j) = (X_i \cdot X_j + 1)^h$$

$$\text{Gaussian radial basis function kernel}: \quad K(X_i, X_j) = e^{-\|X_i - X_j\|^2/2\sigma^2}$$

$$\text{Sigmoid kernel}: \quad K(X_i, X_j) = \tanh(\kappa X_i \cdot X_j - \delta)$$

- SVM can also be used for classifying multiple (> 2) classes and for regression analysis (with additional user parameters)

SVM vs. Neural network

- SVM
  - Relatively new concept
  - Deterministic algorithm
  - Nice generalisation properties
  - Hard to learn – learned in batch mode using quadratic programming techniques
  - Using kernels can learn very complex functions
- Neural Network (tomorrow)
  - Relatively old
  - Non-deterministic algorithm
  - Generalises well but doesn't have strong mathematical foundation
  - Can easily be learned in incremental fashion
  - To learn complex functions—use multilayer perceptron (not that trivial)

What now... things to do

- Lab 6 next week (last lab)
- Quiz 2 next week
- Read sections 8.4 and 9.3 in text book
- Continue working on assignment 2!
  Due Thursday 19 May 5 pm
  Post any questions on Wattle or ask in labs