

COMP8400: Algorithms and Techniques for Data Mining

Classification and prediction:
Support vector machines,
other classification methods,
and prediction

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)

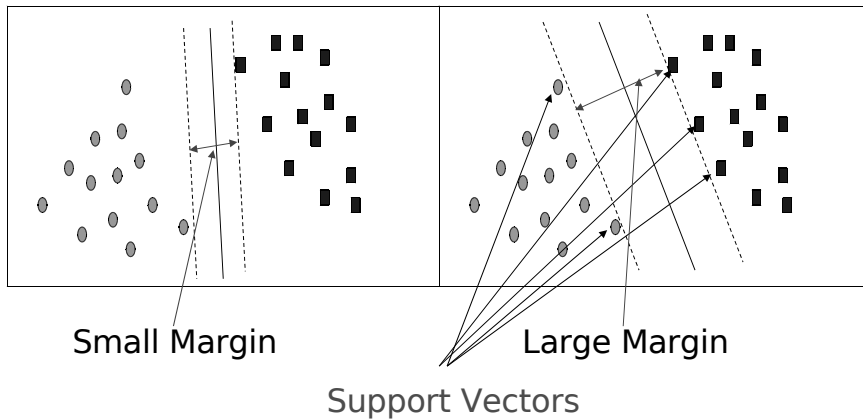
Lecture outline

- Support vector machines
 - History and applications, general philosophy
 - Margins and support vectors
 - Linearly separable and inseparable
 - Kernel functions
 - SVM versus neural network
- Lazy versus eager learners
 - Nearest neighbour based classification
- Genetic algorithms
- Rough and fuzzy set approaches
- Prediction
 - Linear and non-linear regression
 - Regression trees and model trees

SVM—History and applications

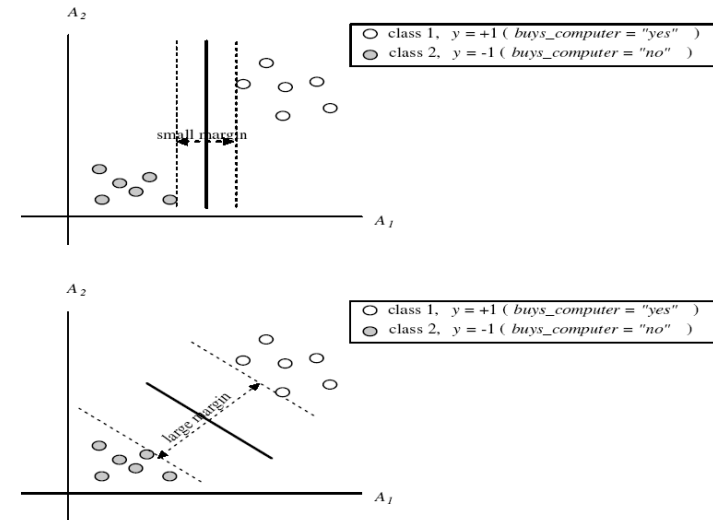
- Vapnik and colleagues (1992)—groundwork from Vapnik and Chervonenkis’ statistical learning theory in 1960s
- Features: training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximisation)
- Used both for classification and prediction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

SVM—General philosophy



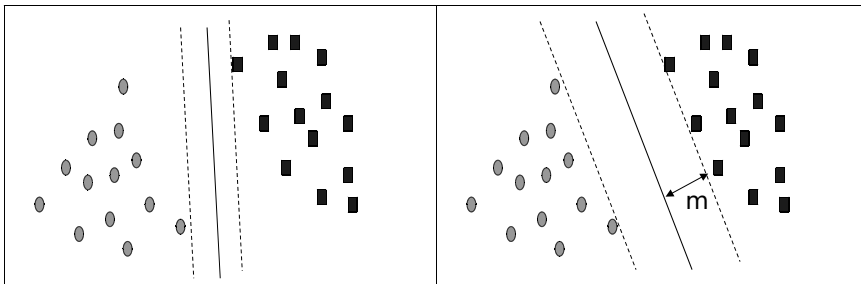
Source: Han and Kamber, DM Book, 2nd Ed. (Copyright © 2006 Elsevier Inc.)

SVM—Margins and support vectors



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

SVM—When data is linearly separable



- Let data D be $(\mathbf{X}_1, y_1), \dots, (\mathbf{X}_{|D|}, y_{|D|})$, where \mathbf{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

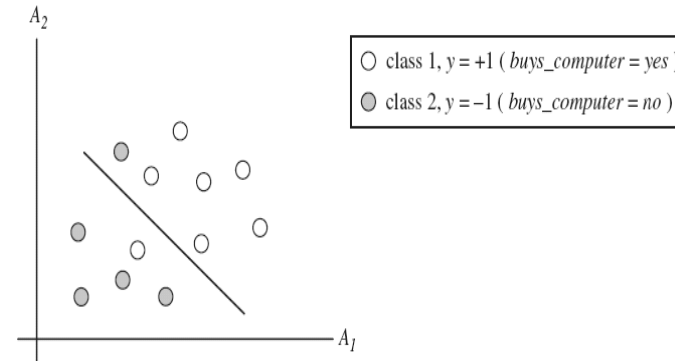
- A separating hyperplane can be written as $\mathbf{W} \cdot \mathbf{X} + b = 0$ (dot product), where $\mathbf{W} = \{w_1, w_2, \dots, w_n\}$ is a weight vector and b a scalar (bias)
 - For 2-D it can be written as $w_0 + w_1 x_1 + w_2 x_2 = 0$
- The hyperplane defining the sides of the margin:
 - $H_1: w_0 + w_1 x_1 + w_2 x_2 \geq 1$ 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* $K(\mathbf{X}_i, \mathbf{X}_j)$ to the original data, i.e., $K(\mathbf{X}_i, \mathbf{X}_j) = \Phi(\mathbf{X}_i) \cdot \Phi(\mathbf{X}_j)$
 - $\Phi(\mathbf{X}_i)$ is a non-linear mapping function applied to transform training tuples
- Typical Kernel Functions

$$\text{Polynomial kernel of degree } h : K(\mathbf{X}_i, \mathbf{X}_j) = (\mathbf{X}_i \cdot \mathbf{X}_j + 1)^h$$

$$\text{Gaussian radial basis function kernel : } K(\mathbf{X}_i, \mathbf{X}_j) = e^{-\|\mathbf{X}_i - \mathbf{X}_j\|^2 / 2\sigma^2}$$

$$\text{Sigmoid kernel : } K(\mathbf{X}_i, \mathbf{X}_j) = \tanh(\kappa \mathbf{X}_i \cdot \mathbf{X}_j - \delta)$$

- 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
 - 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)

Lazy versus eager learning

• Lazy versus eager learning

- Lazy learning (for example, instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
- Eager learning (previously discussed methods): Given a set of training set, constructs a classification model 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 its 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:

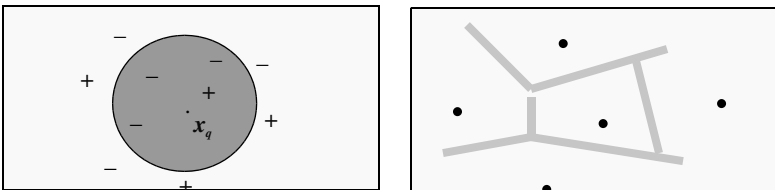
- Store training examples and delay the processing (“lazy evaluation”) until a new instance must be classified

• Typical approaches

- k -nearest neighbour approach (instances represented as points in an Euclidean space)
- Locally weighted regression (constructs local approximation)
- Case-based reasoning (uses symbolic representations and knowledge-based inference)

The k -nearest neighbour algorithm (k -NN)

- All instances correspond to points in the n -dimensional space
- The nearest neighbours are defined in terms of Euclidean distance (for example, other distance functions are possible): $dist(\mathbf{X}_1, \mathbf{X}_2)$
- 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
- Voronoi diagram: the decision surface induced by 1-NN for a typical set of training examples



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

Discussion on the k -NN algorithm

• k -NN for real-valued prediction for a given unknown tuple

- Returns the mean values of the k nearest neighbours

• Distance-weighted nearest neighbour algorithm

- Weight the contribution of each of the k neighbours according to their distance to the query x_q

- Give greater weight to closer neighbours

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

• Robust to noisy data by averaging k -nearest neighbours

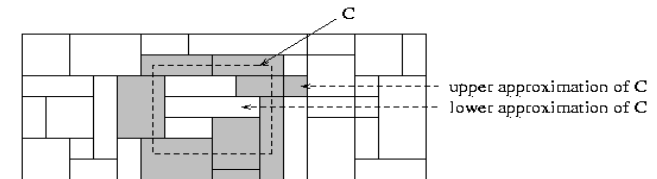
- Curse of dimensionality: distance between neighbours could be dominated by irrelevant attributes

Genetic algorithms (GA)

- Based on an analogy to biological evolution: An initial *population* is created consisting of randomly generated rules
 - Each rule is represented by a string of bits, for example, “if A_1 and $\neg A_2$ then C_2 ” can be encoded as 100
 - If an attribute has $k > 2$ values, k bits can be used
- Based on the notion of survival of the *fittest*, a new population is formed to consist of the fittest rules and their offsprings
 - The fitness of a rule is represented by its *classification accuracy* on a set of training examples
 - Offsprings are generated by *crossover* and *mutation*
- The process continues until a population P evolves when each rule in P satisfies a pre-specified threshold
 - Slow but easily parallelisable

Rough set approach

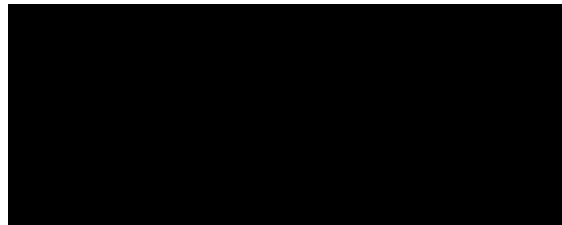
- Rough sets are used to *approximately* or “roughly” define equivalent classes
- A rough set for a given class C is approximated by two sets: a *lower approximation* (certain to be in C) and an *upper approximation* (cannot be described as not belonging to C)
- Finding the minimal subsets (*reducts*) of attributes for feature reduction is NP-hard but a *discernibility matrix* (which stores the differences between attribute values for each pair of data tuples) is used to reduce the computation intensity



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

Fuzzy set approaches

- Fuzzy logic uses truth values between 0.0 and 1.0 to represent the degree of membership (such as using fuzzy membership graph)
- Attribute values are converted to fuzzy values
- For example, income is mapped into the discrete categories *{low, medium, high}* with fuzzy values calculated
- For a given new sample, more than one fuzzy value may apply
- Each applicable rule contributes a vote for membership in the categories
- Typically, the truth values for each predicted category are summed, and these sums are combined



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

What is prediction?

- (Numerical) prediction is similar to classification
 - Construct a model
 - Use model to predict continuous or ordered value for a given input
- Prediction is different from classification
 - Classification refers to predict categorical class label
 - Prediction models continuous-valued functions
- Major method for prediction: *regression*
 - Model the relationship between one or more *independent* or *predictor* variables and a *dependent* or *response* variable
- Regression analysis
 - Linear and multiple regression
 - Non-linear regression
 - Other regression methods: generalised linear model, Poisson regression, log-linear models, regression trees

Linear regression

- Linear regression: involves a response variable y and a single predictor variable x : $y = w_0 + w_1 x$, where w_0 (y -intercept) and w_1 (slope) are regression coefficients
- Method of least squares: estimates the best-fitting straight line

$$w_1 = \frac{\sum_{i=1}^{|D|} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|} (x_i - \bar{x})^2} \quad w_0 = \bar{y} - w_1 \bar{x}$$

- Multiple linear regression: involves more than one predictor variable
- Training data is of the form $(\mathbf{X}_1, y_1), (\mathbf{X}_2, y_2), \dots, (\mathbf{X}_{|D|}, y_{|D|})$
- Example: For 2-D data, we may have: $y = w_0 + w_1 x_1 + w_2 x_2$
- Solvable by extension of least square method
- Many nonlinear functions can be transformed into the above

Nonlinear regression

- Some nonlinear models can be modeled by a polynomial function
- A polynomial regression model can be transformed into linear regression model. For example, $y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$ convertible to linear with new variables: $x_2 = x^2, x_3 = x^3$

$$y = w_0 + w_1 x + w_2 x_2 + w_3 x_3$$

- Other functions, such as power function, can also be transformed to linear model
- Some models are intractable nonlinear (for example, sum of exponential terms)
 - Possible to obtain least square estimates through extensive calculation on more complex formulae

Other regression-based models

- Generalised linear model:
 - Foundation on which linear regression can be applied to modeling categorical response variables
 - Variance of y is a function of the mean value of y , not a constant
 - Logistic regression: models the probability of some event occurring as a linear function of a set of predictor variables
 - Poisson regression: models the data that exhibit a Poisson distribution
- Log-linear models: (for categorical data)
 - Approximate discrete multidimensional probability distributions
 - Also useful for data compression and smoothing
- Regression trees and model trees
 - Trees to predict continuous values rather than class labels

Regression trees and model trees

- Regression tree: proposed in CART system (Breiman et al. 1984)
 - CART: Classification And Regression Trees
 - Each leaf stores a *continuous-valued prediction*
 - It is the *average value of the predicted attribute* for the training tuples that reach the leaf
- Model tree: proposed by Quinlan (1992)
 - Each leaf holds a regression model—a multivariate linear equation for the predicted attribute
 - A more general case than regression tree
- Regression and model trees tend to be more accurate than linear regression when the data are not represented well by a simple linear model

Next.. what to do..

- Peter will be away 10 – 30 April (holidays and PAKDD conference in Thailand)
- **No lectures or labs/tutorials in first week after break (semester week 8, 27 - 30 April)**
- **Assignment 1 is due on Wednesday 6 May, 5 pm**
(e-mail to: comp8400@cs.anu.edu.au)
- Start looking at paper presentation and select a paper
- Get and read papers for tutorial 3 (after semester break)
- Read text book sections 6.5 to 6.7, and 6.9 to 6.11