COMP3420: Advanced Databases and Data Mining

Classification and prediction: Artificial neural networks, other classification methods, and prediction

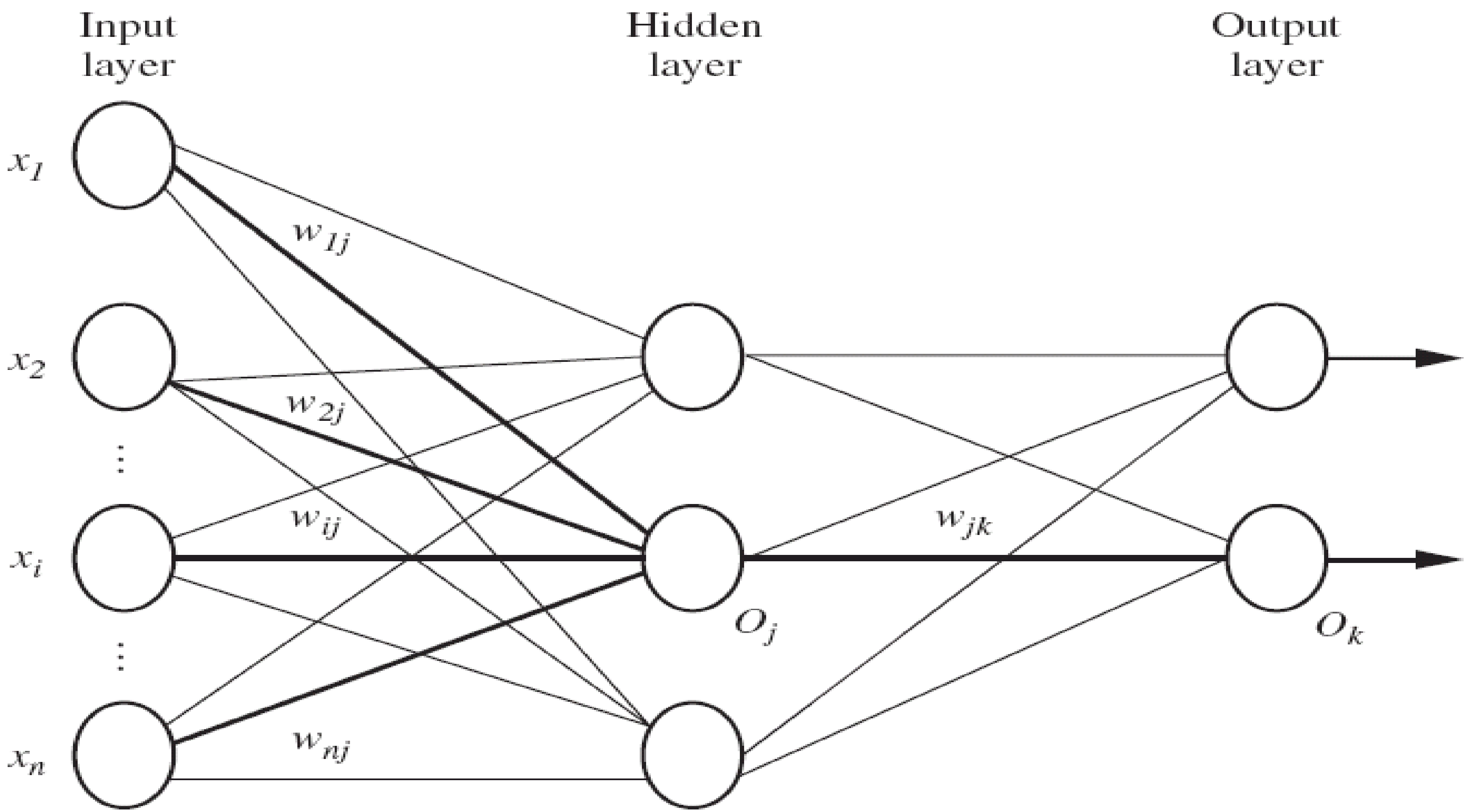
Lecture outline

- Artificial neural networks
  - Classification through backpropagation
  - Neural network as a classifier
  - A multi-layer feed-forward neural network
  - Defining a network topology
  - Backpropagation and interpretability
- Lazy versus eager learners
  - Nearest neighbour based classification
- Genetic algorithms
- Fuzzy set approaches
- Prediction
  - Linear and non-linear regression
  - Regression trees and model trees

Classification through backpropagation

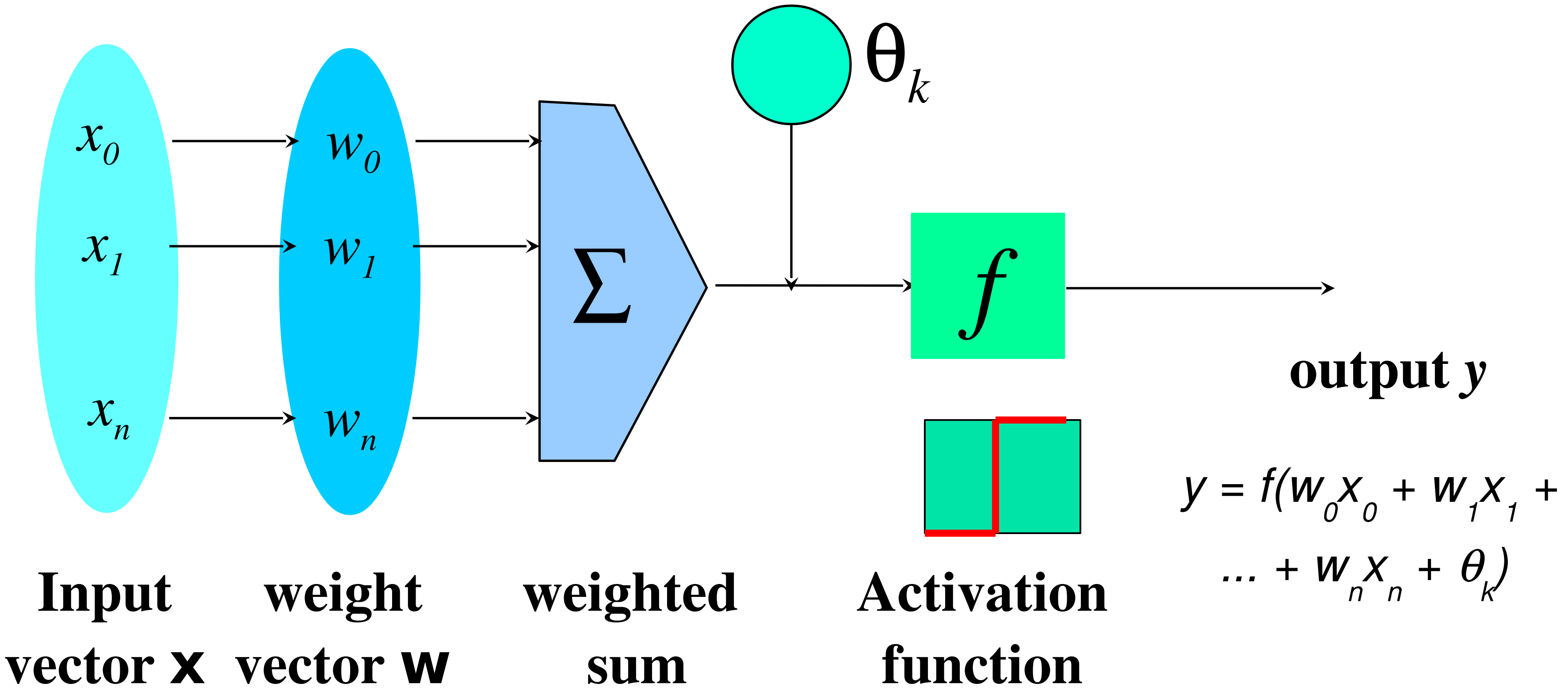
- Backpropagation: An *artificial neural network* learning algorithm
- Started by psychologists and neurobiologists to develop and test computational analogues of neurons
- A neural network: A set of connected input/output units where each connection has a *weight* associated with it
  - A data structure that simulates the behaviour of neuron in a biological brain
- During the learning phase, the *network learns by adjusting the weights* so as to be able to predict the correct class label of the input tuples
  - Also referred to as *connectionist learning* due to the connections between units

A multi-layer feed-forward neural network



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

A neuron (= a perceptron)



- The n-dimensional input vector $\boldsymbol{x}$ is mapped into variable y by means of the scalar product and a nonlinear function mapping

How a multi-layer neural network works

- The *inputs* to the network correspond to the attributes measured for each training tuple / record
- Inputs are fed simultaneously into the units making up the *input layer*
- They are then weighted and fed simultaneously to a *hidden layer*
- The number of hidden layers is arbitrary, although usually only one
- The weighted outputs of the last hidden layer are input to units making up the *output layer,* which emits the network's prediction
- The network is *feed-forward* in that none of the weights cycles back to an input unit or to an output unit of a previous layer
- From a statistical point of view, networks perform *nonlinear regression*: Given enough hidden units and enough training samples, they can closely approximate any function

Defining a network topology

- First decide the *network topology: number* of units in the *input layer*, number of *hidden layers* (if > 1), number of units in *each hidden layer*, and number of units in the *output layer*
- Normalise the input values for each attribute measured in the training tuples to [0.0—1.0]
- For discrete (categorical) attributes: one *input* unit per value, each initialised to 0 (e.g. for three categories have three inputs)
- *Output*, if for classification and more than two classes, one output unit per class is used
- Once a network has been trained and its accuracy is *unacceptable*, repeat the training process with a *different network topology* or a *different set of initial weights*

Backpropagation

- Iteratively process a set of training tuples and compare the network's prediction with the actual known target value
- For each training tuple, the weights are modified to *minimise the mean squared error* between the network's prediction and the actual target value
- Modifications are made in the *"backwards"* direction: from the output layer, through each hidden layer down to the first hidden layer, hence *"backpropagation"*
- Steps
  - Initialise weights (to small random numbers) and biases in the network
  - Propagate the inputs forward (by applying activation function)
  - Backpropagate the error (by updating weights and biases)
  - Terminating condition (when error is very small, etc.)

Backpropagation and interpretability

- Efficiency of backpropagation: Each *epoch* (one iteration through the training set) takes $O(|D| * w)$, with $|D|$ tuples and w weights, but the number of epochs can be exponential to n, the number of input units, in the worst case
- Rule extraction from networks: network pruning
  - Simplify the network structure by removing weighted links that have the least effect on the trained network
  - Then perform link, unit, or activation value clustering
  - The set of input and activation values are studied to derive rules describing the relationship between the input and hidden unit layers
- Sensitivity analysis: assess the impact that a given input variable has on a network output. The knowledge gained from this analysis can be represented in rules

Neural network as a classifier

- Weakness
  - Long training time
  - Require a number of parameters typically best determined empirically, for example, the network topology or *structure*
  - 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 a wide array of real-world data
  - Algorithms are inherently parallel
  - Techniques have recently been developed for the extraction of rules from trained neural networks

SVM vs. Neural network

- SVM (yesterday)
  - 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

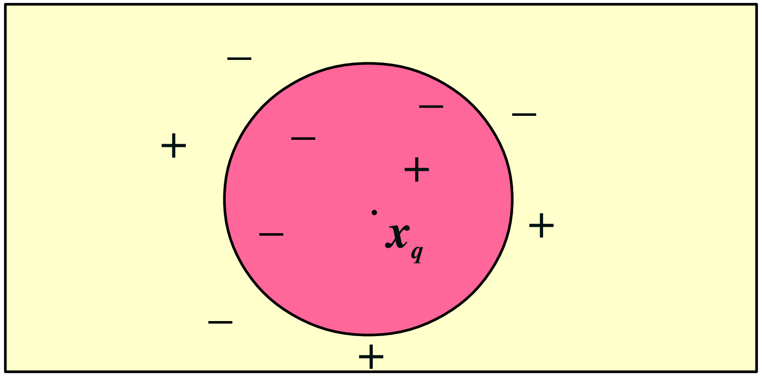
- Lazy versus eager learning
  - Lazy learning (for example, instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - Eager learning (previously discussed methods): Given a set of training set, constructs a classification model before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

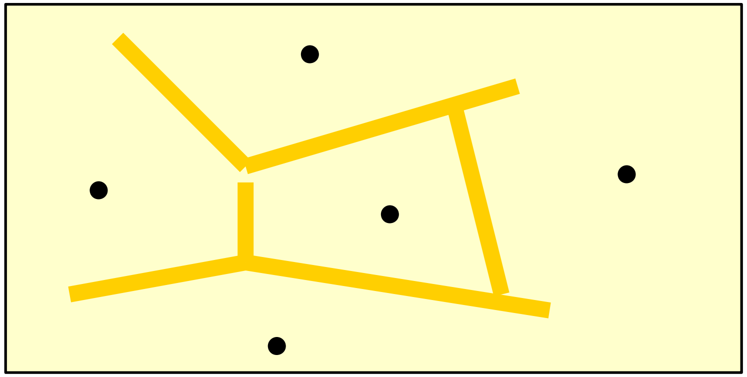
Lazy learner: Instance-based methods

- 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*($\boldsymbol{X_1}$, $\boldsymbol{X_2}$)
- 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 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

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

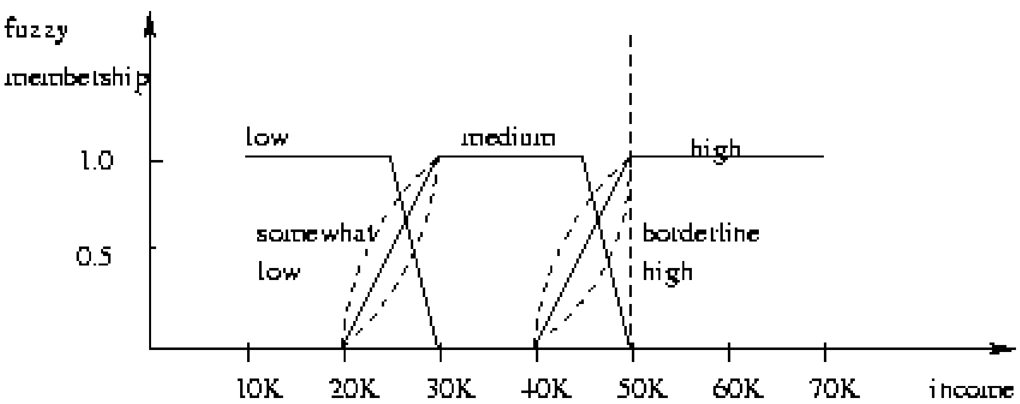
- Robust to noisy data by averaging *k*-nearest neighbours
- Curse of dimensionality: distance between neighbours could be dominated by irrelevant attributes

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

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

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

- Multiple linear regression: involves more than one predictor variable
- Training data is of the form $(\mathbf{X}_1, y_1), (\mathbf{X}_2, y_2), \ldots, (\mathbf{X}_{|D|}, y_{|D|})$
- 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

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

Review question

- Supervised learning techniques allow us to predict the class membership of unseen records.

Yes or No?

- Supervised learning techniques can easily achieve high classification accuracy independent of the quality of the training data used.

Yes or No?

What now... things to do

- Lab 6 next week (last lab)
- Quiz 2 next week

- Read sections 9.2, and 9.5 to 9.7 in text book

- Continue working on assignment 2!

  Due Thursday 19 May 5 pm

  Post any questions on Wattle or ask in labs