Principles of Knowledge Discovery in Data

Winter 2007

Chapter 4: Data Classification

Dr. Osmar R. Zaïane

University of Alberta

Course Content

- Introduction to Data Mining
- Association Analysis
- Sequential Pattern Analysis
- Classification and prediction
- Contrast Sets
- Data Clustering
- Outlier Detection
- Web Mining
- Other topics if time permits (spatial data, biomedical data, etc.)

Chapter 4 Objectives

Learn basic techniques for data classification and prediction.

Introduce techniques such as **Neural Networks, Naïve Bayesian Classification, k-Nearest Neighbors, Decision Trees & Associative Classifiers**

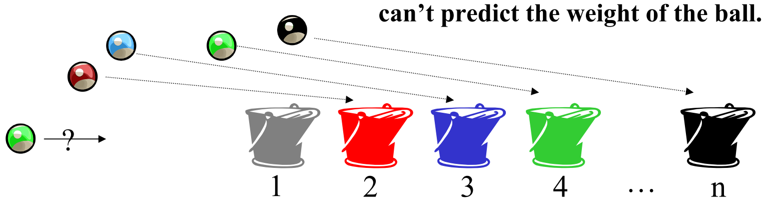
Realize the difference between supervised classification, prediction and unsupervised classification of data.

What is Classification?

The goal of data classification is to organize and categorize data in distinct classes.

- A model is first created based on the data distribution.
- The model is then used to classify new data.
- Given the model, a class can be predicted for new data.

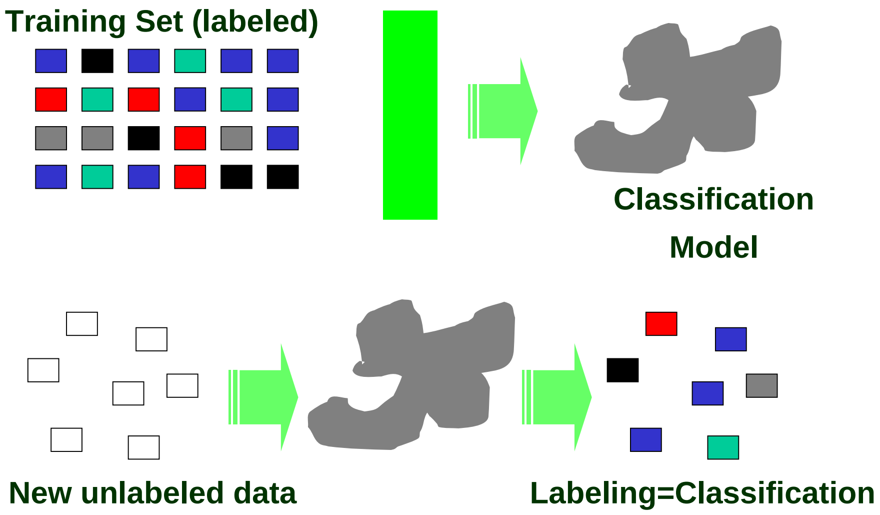
With classification, I can predict in which bucket to put the ball, but I can't predict the weight of the ball.



Application

- Credit approval
- Target marketing
- Medical diagnosis
- Defective parts identification in manufacturing
- Crime zoning
- Treatment effectiveness analysis
- Etc.

Classification = Learning a Model



Training Set (labeled)

Classification Model

New unlabeled data

Labeling=Classification

Classification is a three-step process

1. Model construction (**Learning**):

- Each tuple is assumed to belong to a predefined class, as determined by one of the attributes, called the **class label**.
- The set of all tuples used for construction of the model is called **training set**.
- The model is represented in the following forms:

> - **Classification rules, (IF-THEN statements),**
> - **Decision tree**
> - **Mathematical formulae**

Classification is a three-step process

2. Model Evaluation (**Accuracy**):

Estimate accuracy rate of the model based on a **test set**.

- The known label of test sample is compared with the classified result from the model.
- Accuracy rate is the percentage of test set samples that are correctly classified by the model.
- Test set is independent of training set otherwise over-fitting will occur.

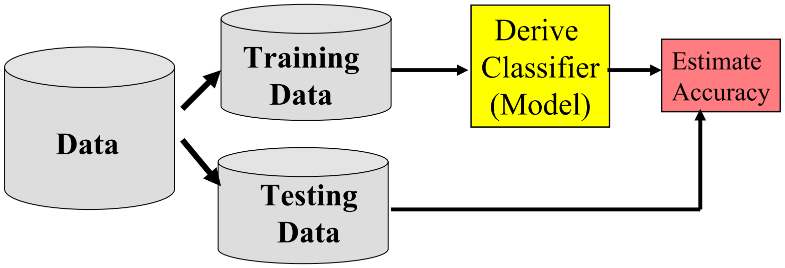
Classification is a three-step process

3. **Model Use** (Classification):

The model is used to classify unseen objects.

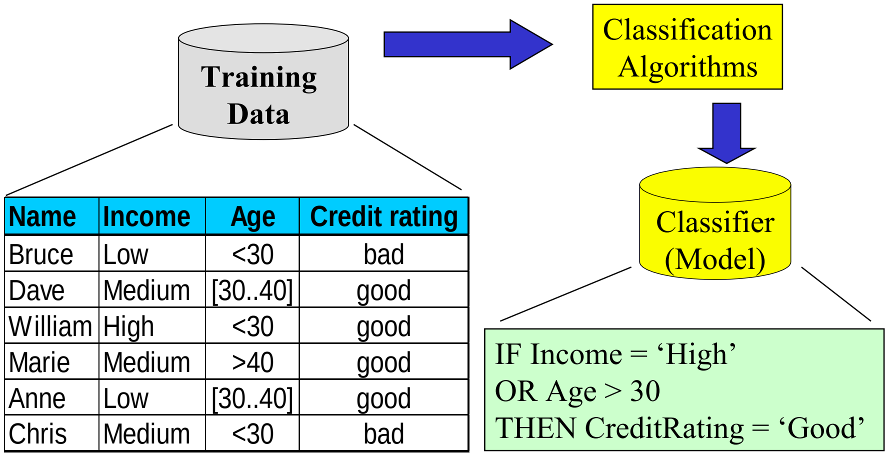
- Give a class label to a new tuple
- Predict the value of an actual attribute

Classification with Holdout



- Holdout (partition D in two; one for test one for training)
- Random sub-sampling (repeated holdouts with different partitioning)
- K-fold cross validation (K partitions of D; do k experiments each with a different test partition. No overlap as in Random sub-sampling. Typically k=10)
- Bootstrapping (N samples (typically of size 63% of D) with replacements)
- Leave-one-out (|D| experiments with |D|-1 for training and one for test)

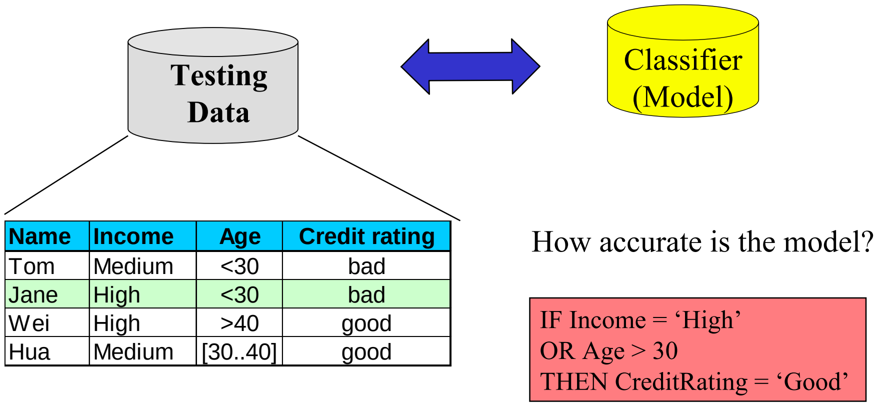
1. Classification Process (Learning)



Name	Income	Age	Credit rating
Bruce	Low	<30	bad
Dave	Medium	[30..40]	good
William	High	<30	good
Marie	Medium	>40	good
Anne	Low	[30..40]	good
Chris	Medium	<30	bad

IF Income = 'High'
OR Age > 30
THEN CreditRating = 'Good'

2. Classification Process (Accuracy Evaluation)

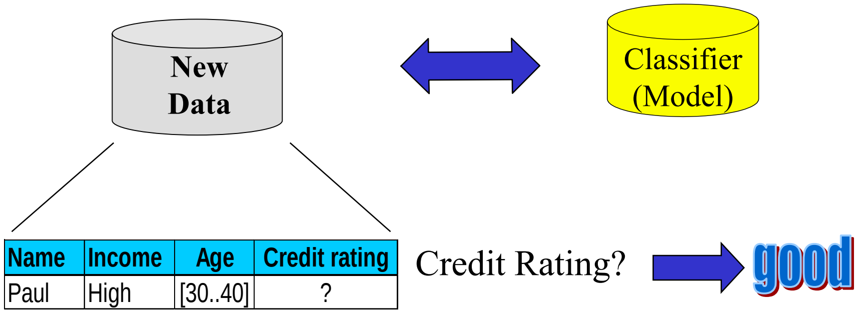


Name	Income	Age	Credit rating
Tom	Medium	<30	bad
Jane	High	<30	bad
Wei	High	>40	good
Hua	Medium	[30..40]	good

How accurate is the model?

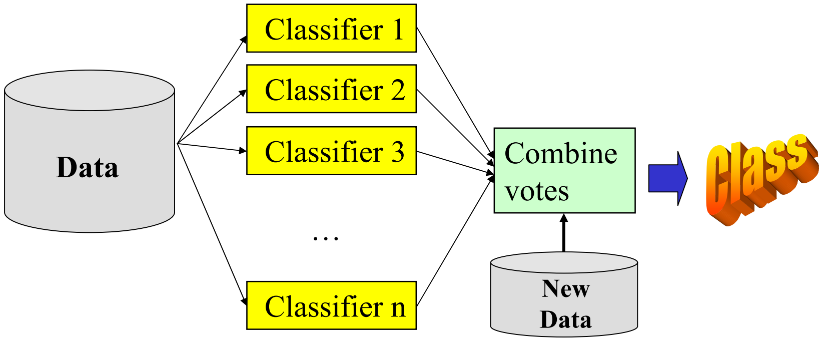
IF Income = 'High'
OR Age > 30
THEN CreditRating = 'Good'

3. Classification Process (Classification)



Name	Income	Age	Credit rating
Paul	High	[30..40]	?

Improving Accuracy

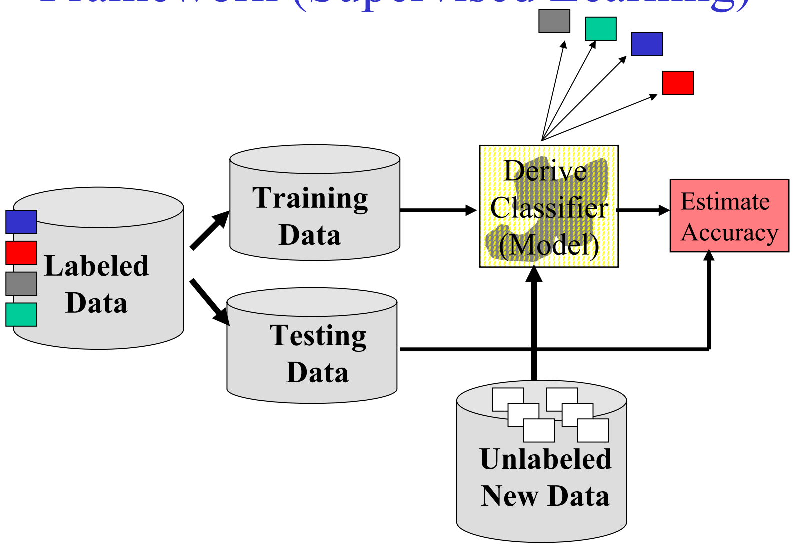


Composite classifier

Evaluating Classification Methods

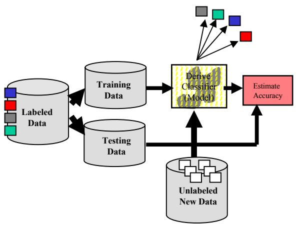
- Predictive accuracy
  - Ability of the model to correctly predict the class label
- Speed and scalability
  - Time to construct the model
  - Time to use the model
- Robustness
  - Handling noise and missing values
- Scalability
  - Efficiency in large databases (not memory resident data)
- Interpretability:
  - The level of understanding and insight provided by the model
- Form of rules
  - Decision tree size
  - The compactness of classification rules

Framework (Supervised Learning)



Classification Methods

- ❖ Neural Networks
- ❖ Bayesian Classification
- ❖ K-Nearest Neighbour
- ❖ Decision Tree Induction
- ❖ Associative Classifiers
- ❖ Support Vector Machines
- ❖ Case-Based Reasoning
- ❖ Genetic Algorithms
- ❖ Rough Set Theory
- ❖ Fuzzy Sets
- ❖ Etc.



Lecture Outline

Part I: Artificial Neural Networks (ANN) *(1 hour)*

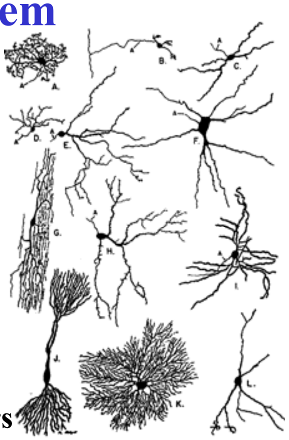
- **Introduction to Neural Networks**
  - Biological Neural System
  - What is an artificial neural network?
  - Neuron model and activation function
  - Construction of a neural network
- **Learning: Backpropagation Algorithm**
  - Forward propagation of signal
  - Backward propagation of error
  - Example

Part II: Bayesian Classifiers (Statistical-based) *(1 hour)*

- **What is Bayesian Classification**
- **Bayes theorem**
- **Naïve Bayes Algorithm**
  - Using Laplace Estimate
  - Handling Missing Values and Numerical Data
- **Belief Networks**

Human Nervous System

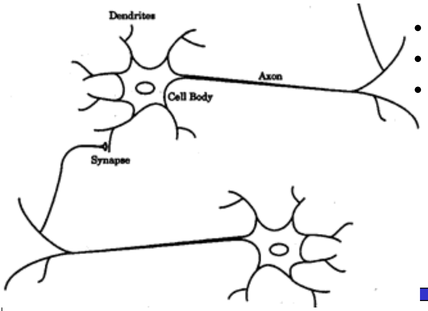
- We have only just began to understand how our neural system operates
- A huge number of *neurons* and *interconnections* between them
  - 100 billion (i.e. 10^{10}) neurons in the brain
    - a full Olympic-sized swimming pool contains 10^{10} raindrops; the number of stars in the Milky Way is of the same magnitude
  - 10^4 connections per neuron



- Biological neurons are slower than computers
  - Neurons operate in 10^{-3} seconds , computers in 10^{-9} seconds
  - The brain makes up for the slow rate of operation by a single neurone by the large number of neurons and connections (think about the speed of face recognition by a human, for example, and the time it takes fast computers to do the same task.)

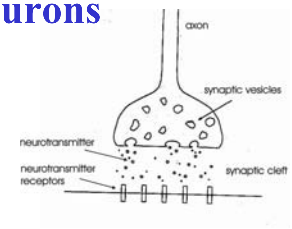
Biological Neurons

- **The purpose of neurons: transmit information in the form of electrical signals**
  - it accepts many inputs, which are all added up in some way
  - if enough active inputs are received at once, the neuron will be activated and fire; if not, it remain in its inactive state
- **Structure of neuron**



- Cell body - contains nucleus holding the chromosomes
- Dendrites
- Axon
- Synapse
  - ➢ couples the axon with the dendrite of another cell;
  - ➢ information is passed from one neuron to another through synapses;
  - ➢ no direct linkage across the junction, it is a chemical one.

Operation of biological neurons

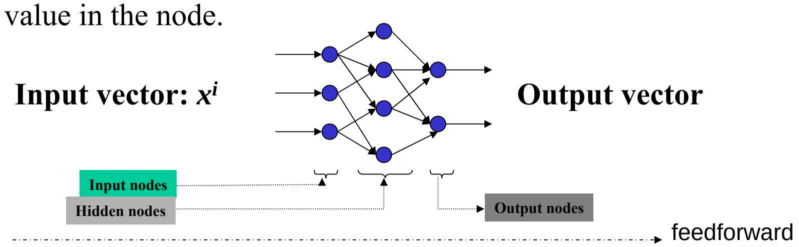


- Signals are transmitted between neurons by electrical pulses (***action potentials, AP***) traveling along the axon;
- When the potential at the synapse is raised sufficiently by the AP, it releases chemicals called neurotransmitters
  - it may take the arrival of more than one AP before the synapse is triggered
- The neurotransmitters diffuse across the gap and chemically activate gates on the dendrites, that allows *charged ions to flow*
- The flow of ions alters the potential of the dendrite and provides a voltage pulse on the dendrite (***post-synaptic-potential, PSP)***
  - some synapses *excite* the dendrite they affect, while others *inhibit* it
  - the synapses also determine the strength of the new input signal
- Each PSP travels along its dendrite and spreads over the soma (cell body)
- The soma sums the effects of thousands PSPs; if the resulting potential exceeds a threshold, the neuron fires and generates another AP.

What is an Artificial Neural Network (NN)?

A neural network is a data structure that supposedly simulates the behaviour of neurons in a biological brain.

A neural network is composed of layers of units interconnected. Messages are passed along the connections from one unit to the other. Messages can change based on the *weight* of the connection and the value in the node.



Input vector: x^i

Output vector

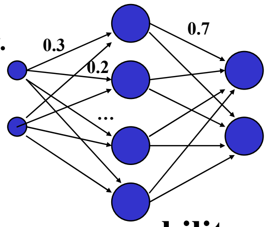
Input nodes

Hidden nodes

Output nodes

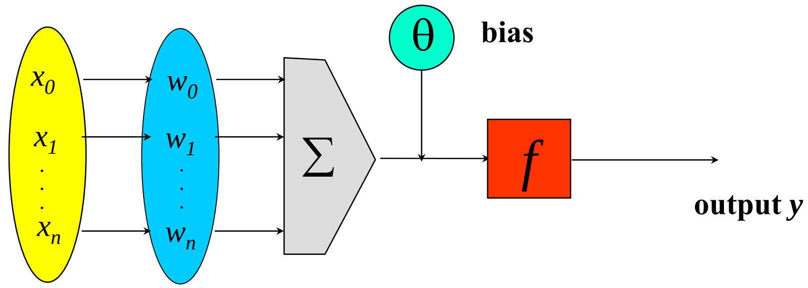
feedforward

What is an Artificial Neural Network (NN)?



- **A network of many simple *units (neurons, nodes*)**
  - **The units are connected by *connections.***
  - **Each connection has an associated numeric *weight*.**
  - **Units receive inputs (from the environment or other units) via the connections. They produce output using their weights and the inputs (i.e. they operate locally).**
  - **A NN can be represented as a directed graph.**
- **NNs *learn from examples* and exhibit some capability for generalization beyond the training data.**
  - **knowledge is acquired by the network from its environment via *learning* and is stored in the *weights* of the connections.**
  - **the *training (learning) rule* – a procedure for modifying the weights of connections in order to perform a certain task.**
  - **There are also some sophisticated techniques that allow learning by adding and pruning connections (between nodes).**

A Neuron



Input vector *x*

weight vector *w*

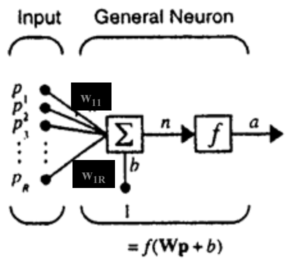
weighted sum

Activation function

- The *n*-dimensional input vector *x* is mapped into variable *y* by means of the scalar product and a nonlinear function mapping.

Neuron Model

- **Each connection from unit *i* to *j* has a numeric weigh w_{ij} associated with it, which determines the strength and the sign of the connection**
- **Each neuron first computes the weighed sum of its inputs w_p, and then applies an activation function *f* to derive the output (activation) *a***
- **A neuron may have a special weight called *bias weight b* .**
- **NNs represent a function of their weights (parameters). By adjusting the weights, we change this function. This is done by using a learning rule.**



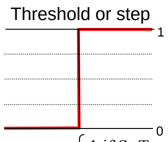
if there are 2 inputs $p_1=2$ and $p_2=3$, and if $w_{11}= 3$, $w_{12}=1$, $b = -1.5$, then $a = f(2*3+3*1\ -1.5) = f(7.5)$

What is *f*?

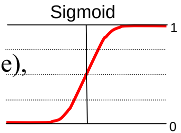
$f(P_1*w_{11} + p_2*w_{12} + b)$

Activation function

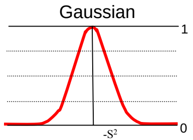
- Activation function, processing element, squashing function, firing rule…
- Is applied by each neuron to its input values and weights (as well as the bias) $S_i= \theta_i + \Sigma_{i=1..n} (x_{ij} * W_{ij})$
- Can be unipolar [0,1] bipolar [-1, 1]
- The function can be
  - Linear (***f*** (S)=cS),
  - Thresholded (***f*** (S)=1 if S>T; 0 otherwise),
  - a Sigmoid (***f*** $(S)=1/(1+e^{-cS})$),
  - a Gaussian (***f*** $(S)=e^{-S2/v}$), etc.



$f(S)=\begin{cases}1 \text{ if } S>T\\ 0 \text{ otherwise}\end{cases}$



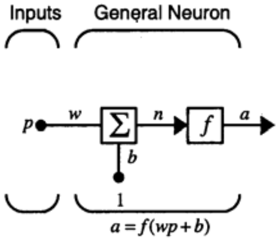
$f(S)= \frac{1}{1 + e^{-cS}}$

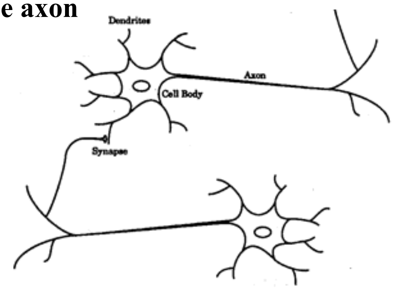


$f(S)= e^{\frac{-S^2}{v}}$

Correspondence Between Artificial and Biological Neurons

- **How this artificial neuron relates to the biological one?**
  - **input *p* (or input vector p) – input signal (or signals) at the dendrite**
  - **weight *w* (or weight vector w) - strength of the synapse (or synapses)**
  - **summer & transfer function - cell body**
  - **neuron output *a* - signal at the axon**





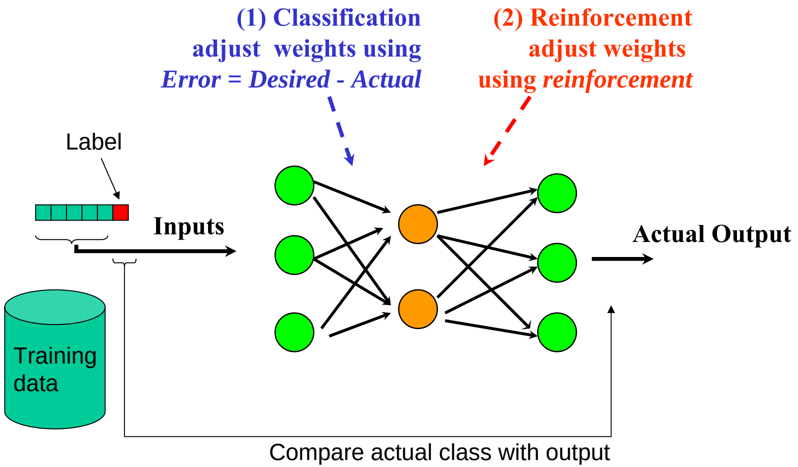
Constructing the Network

- The number of input nodes: Generally corresponds to the dimensionality of the input tuples. Input is converted into binary and concatenated to form a bitstream.
  - Eg. age 20-80: 6 intervals
    - [20, 30) → 000001, [30, 40) → 000010, …., [70, 80) → 100000
    - [20, 30) → 001, [30, 40) → 010, …., [70, 80) → 110
- Number of hidden nodes: Determined by expert, or in some cases, adjusted during training.
- Number of output nodes: Generally number of classes
  - Eg. 10 classes
    - 0000000001 → C1, 0000000010 → C2, …., 1000000000 → C10
    - 0001 → C1, 0010 → C2, …., 1010 → C10

Neural Networks - Pros and Cons

- Advantages
  - prediction accuracy is generally high.
  - robust, works when training examples contain errors.
  - output may be discrete, real-valued, or a vector of several discrete or real-valued attributes.
  - fast evaluation of the learned target function.
- Criticism
  - long training time.
  - difficult to understand the learned function (weights).
  - Typically for numerical data
  - not easy to incorporate domain knowledge.
  - Design can be tedious and error prone (Too small: slow learning - Too big: instability or poor performance)

Learning Paradigms



(1) Classification
adjust weights using
Error = Desired - Actual

(2) Reinforcement
adjust weights
using *reinforcement*

Learning Algorithms

- **Back propagation for classification**
- Kohonen feature maps for clustering
- Recurrent back propagation for classification
- Radial basis function for classification
- Adaptive resonance theory
- Probabilistic neural networks

Major Steps for Back Propagation Network

- Constructing a network
  - input data representation
  - selection of number of layers, number of nodes in each layer.
- Training the network using training data
- Pruning the network
- Interpret the results

Network Training

- The ultimate objective of training
  - obtain a set of weights that makes almost all the tuples in the training data classified correctly.
- Steps:
  - Initial weights are set randomly.
  - Input tuples are fed into the network one by one.
  - Activation values for the hidden nodes are computed.
  - Output vector can be computed after the activation values of all hidden node are available.
  - Weights are adjusted using error (desired output - actual output) and propagated backwards.

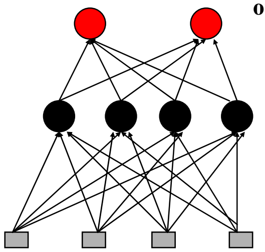
Network Pruning

- Fully connected network will be hard to articulate
- *n* input nodes, *h* hidden nodes and *m* output nodes lead to *h(m+n)* links (weights)
- Pruning: Remove some of the links without affecting classification accuracy of the network.

Backpropagation Network - Architecture

- 1) A network with 1 or more hidden layers

Outlook	Tempreature	Humidity	Windy	Play
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No



output neurons

1 output neuron for each class

hidden neurons (1 hidden layer)

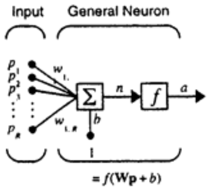
inputs

1 input neuron for each attribute

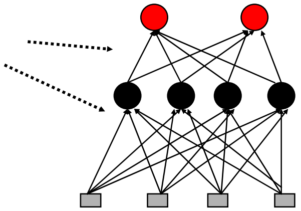
- 2) Feedforward network - each neuron receives input only from the neurons in the previous layer
- 3) Typically fully connected - all neurons in a layer are connected with all neurons in the next layer
- 4) Weights initialization – small random values, e.g. [-1,1]

Backpropagation Network – Architecture 2

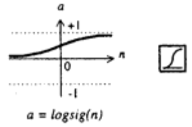
- 5) Neuron model - weighed sum of input signals + differentiable transfer function



$a = f(\mathbf{wp}+b)$

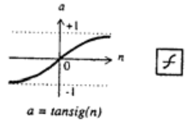


- any *differentiable* transfer function *f* can be used; most frequently the sigmoid and tan-sigmoid (hyperbolic tangent sigmoid) functions are used:



$$a = \frac{1}{1+e^{-n}}$$

Log-Sigmoid Transfer Function



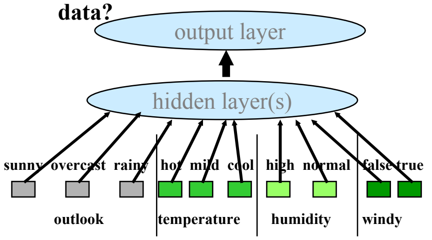
$$a = \frac{e^{n} - e^{-n}}{e^{n} + e^{-n}}$$

Tan-Sigmoid Transfer Function

Architecture – Number of Input Units

Outlook	Tempreature	Humidity	Windy	Play
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

- **Numerical data - typically 1 input unit for each attribute**
- **Categorical data – 1 input unit for each attribute value)**
  - **How many input units for the weather data?**



- **Encoding of the input examples – typically binary depending on the value of the attribute (on and off)**
  e.g.: 100 100 10 01

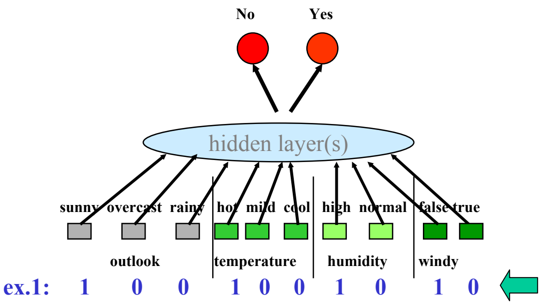
Other possibilities are also acceptable. For example “Windy” could be coded with only one unit: *true* or *false* (*1 or 0*).

Number of Output Units

Outlook	Tempreature	Humidity	Windy	Play
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

- **Typically 1 neuron for each class**

target class ex1: 1 0



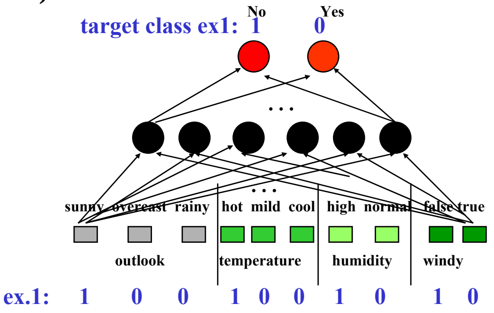
sunny	hot	high	false	No

- **Encoding of the targets (classes) – typically binary**
  e.g. class1 (no): 1 0, class2 (yes): 0 1

Another possibility is to code the target class with only one unit: *Yes* or No (*1 or 0*).

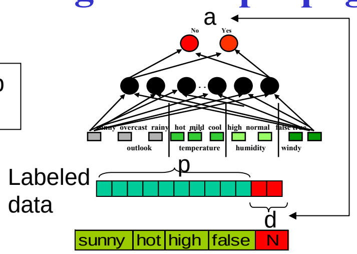
Number of Hidden Layers and Units in Them

- **An art! Typically - by trial and error**
- **The task constrains the number of inputs and output units but not the number of hidden layers and neurons in them**
  - **Too many hidden layers and units (i.e. too many weights) – overfitting**
  - **Too few – underfitting, i.e. the NN is not able to learn the input-output mapping**
  - **A heuristic to start with: 1 hidden layer with *n* hidden neurons, *n=(inputs+output_neurons)/2***



Learning in Backpropagation NNs

1 Propagate p forward



Compare and calculate error

2 Propagate error adjustments backward

Idea of backpropagation learning

For each training example p

- Propagate p through the network and calculate the output *a* . Compare the desired *d* with the actual output *a* and calculate the error;
- Update weights of the network to reduce the error;

Until error over all examples < threshold

- Why “backpropagation”? Adjusts the weights *backwards* (from the output to the input units) by propagating the *weight change Δw*

$$w_{pq}^{new} = w_{pq}^{old} + \Delta w_{pq}$$

How to calculate the weight change?

Backpropagation Learning - 2

- **Sum of Squared Errors (E) is a classical measure of error**
  - **E for a single training example over all output neurons**
  - **d_i :desired, a_i :actual network output for <u>output neuron</u> *i***

$$E = \frac{1}{2}\sum_i e_i^2 = \frac{1}{2}\sum_i (d_i - a_i)^2$$

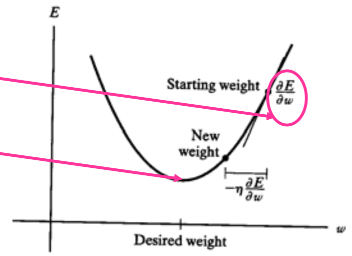
- **Thus, backpropagation learning can be viewed as an optimization <u>search in the weight space</u>**
  - **Goal state – the set of weights for which the performance index (error) is minimum**
  - **Search method – hill climbing** [reduce error for each training example]

Steepest Gradient Descent

- **The direction of the *steepest descent* is called *gradient* and can be computed (∂E/∂w)**
- **A function *decreases* most rapidly when the direction of movement is *in the direction of the negative of the gradient***
- **Hence, we want to adjust the weights so that the change moves the system down the error surface in the direction of the locally steepest descent, given by the negative of the gradient**
- **η- learning rate, defines the step; typically in the range (0,1)**

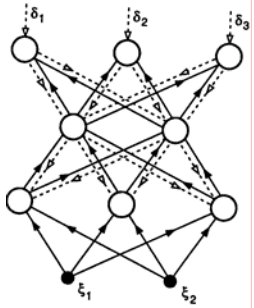
➢ Gives the slope (gradient) of the error function for one weight

➢ We want to find the weight where the slope (gradient) is 0



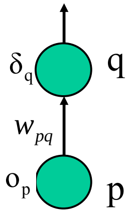
Backpropagation Algorithm - Idea

- **The backpropagation algorithm adjust weights by working backward from the output layer to the input layer**
- **Calculate the error and propagate this error from layer to layer**
- **2 approaches**
  - **Incremental – the weights are adjusted after each training example is applied**
    - **Called also an approximate steepest descent**
    - **Preferred as it requires less space**
  - **Batch – weights are adjusted once after all training examples are applied and a total error was calculated**



- **Solid lines - forward propagation of signals**
- **Dashed lines – backward propagation of error**

Backpropagation Rule – Delta change



$w_{pq}(t)$ **: weight from node *p* to node *q* at time *t***

$w_{pq}(t+1) = w_{pq}(t) + \Delta w_{pq}$

$\Delta w_{pq} = \eta \cdot \delta_q \cdot o_p$ **- weight change**

• The weight change is proportional to the output activation of neuron p (ie. O_p) **and the error δ of neuron q** (ie. δ_p)

- **δ is calculated in 2 different ways:**
  - **q is an output neuron** $\delta_q = (d_q - o_q)\cdot f'(net_q)$
  - **q is a hidden neuron** $\delta_q = f'(net_q)\sum_i w_{qi}\,\delta_i$ **(*i* is over the nodes in the layer above *q*)**

Derivative of the activation function at neuron q with respect to the input of q (netq)

Derivative of Sigmoid Activation Function

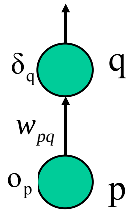
- From the formulas for δ, we must be able to calculate the derivatives for *f*. For a sigmoid transfer function:

$$f(net_m) = o_m = \frac{1}{1+e^{-net_m}}$$

$$f'(net_m) = \frac{\partial o_m}{\partial net_m} = \frac{\partial\left(\frac{1}{1+e^{-net_m}}\right)}{\partial net_m} =$$

$$= \frac{e^{-net_m}}{\left(1+e^{-net_m}\right)^2} = \boxed{o_m \cdot \left(1-o_m\right)}$$

- Thus, backpropagation errors for a network with sigmoid transfer function:
  - q is an output neuron $\delta_q = \left(d_q - o_q\right)\boxed{o_q\left(1-o_q\right)}$
  - q is a hidden neuron $\delta_q = \boxed{o_q\left(1-o_q\right)}\sum_i w_{qi}\,\delta_i$



Backpropagation Algorithm - Summary

1. Determine the architecture of the network
   - how many input and output neurons; what output encoding
   - hidden neurons and layers
2. Initialize all weights (biases incl.) to small random values, typically ∈[-1,1]
3. Repeat until termination criterion satisfied:
   - *(forward pass)* Present a training example and propagate it through the network to calculate the actual output
   - *(backward pass)* Compute the error (the δ values for the output neurons). Starting with output layer, repeat for each layer in the network:
     - propagate the δ values back to the previous layer
     - update the weights between the two layers

- The stopping criteria is checked at the end of each epoch:
  - The error (mean absolute or mean square) is below a threshold
    - All training examples are propagated and the total error is calculated
    - The threshold is determined heuristically – e.g. 0.3
  - Maximum number of epochs is reached
  - Early stopping using a validation set
- It typically takes hundreds or thousands of epochs for an NN to converge

- epoch - 1 pass through the training set

Some Interesting NN Applications

- There are many examples of applications using NNs
- Network design is typically the result of several months trial and error experimentation
- Moral: NNs are widely applicable but they cannot magically solve problems; wrong choices lead to poor performance
- *“NNs are the second best way of doing just about anything”* John Denker
  - NN provide passable performance on many tasks that would be difficult to solve explicitly with other techniques

Lecture Outline

Part I: Artificial Neural Networks (ANN) *(1 hour)*

- **Introduction to Neural Networks**
  - Biological Neural System
  - What is an artificial neural network?
  - Neuron model and activation function
  - Construction of a neural network
- **Learning: Backpropagation Algorithm**
  - Forward propagation of signal
  - Backward propagation of error
  - Example

Part II: Bayesian Classifiers (Statistical-based) *(1 hour)*

- **What is Bayesian Classification**
- **Bayes theorem**
- **Naïve Bayes Algorithm**
  - Using Laplace Estimate
  - Handling Missing Values and Numerical Data
- **Belief Networks**

What is Bayesian Learning (Classification)?

- **Baysian classifiers are statistical classifiers**
- **They can predict the class membership probability, i.e. the probability that a given example belongs to a particular class.**
- **They are based on the Bayes Theorem, presented in the *Essay Towards Solving a Problem in the Doctrine of Chances* published posthumously by his friend Richard Price in the *Philosophical Transactions of the Royal Society of London* in 1763.**

Thomas Bayes [1702-1761]

More on Bayesian Classifiers

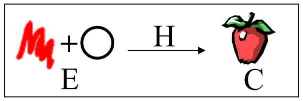
- It uses probabilistic learning by calculating explicit probabilities for hypothesis.
- A naïve Bayesian classifier, that assumes total independence between attributes, is commonly used for data classification and learning problems. It performs well with large data sets and exhibits high accuracy.
- The model is incremental in the sense that each training example can incrementally increase or decrease the probability that a hypothesis is correct. Prior knowledge can be combined with observed data.

Bayes Theorem

- Given a data sample *E (also called Evidence)* with an unknown class label, *H* is the hypothesis that *E* belongs to a specific class *C*.
- The *probability* of a hypothesis *H, P(H|E), probability of E conditioned on H, also called Posteriori Probability,* follows the Bayes theorem:

$$P(H|E)=\frac{P(E|H)P(H)}{P(E)}$$

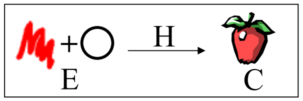
- Example: Instances of fruits, described by their colour and shape. Let *E* is red and round, *H* is the hypothesis that *E* is an apple.



- P(H)=P(🍎)   P(E) = P(🟥+○) P(E|H)=P(🟥+○if 🍎)

Bayes Theorem The Fruit Example

$$P(H|E)=\frac{P(E|H)P(H)}{P(E)}$$



- ***P(H|E)* reflects our confidence that *E* is an apple given that we have seen that *E* is red and round** P(H|E)=P(🍎 if 🟥 +○)
  - **Called *posterior*, or *posteriori probability*, of *H* conditioned on *E***
- ***P(H)* is the probability that any given example is an apple, regardless of how it looks** P(H)=P(🍎)
  - **Called *prior*, or *apriori probability*, of *H***
- **The posteriori probability is based on more information that the apriori probability which is independent of *E***
- **What is *P(E|H)* ?** P(E|H)=P(🟥 +○ if 🍎)
  - **the posterior probability of *E* conditioned on *H:* the probability that *E* is red and round given that we know that *E* is an apple.**
- **What is *P(E)*** P(E)=P(🟥 +○)
  - **the prior probability of *E: the probability that an example from the fruit data set is red and round***

Bayes Theorem – How to use it for classification?

$$P(H|E)=\frac{P(E|H)P(H)}{P(E)}$$

- **In classification tasks we would like to predict the class of a new example *E*. We can do this by:**
  - **Calculating *P(H|E)* for each *H* (class) – the probability that the hypothesis *H* is true given the example *E***
  - **Comparing these probabilities and assigning *E* to the class with the highest probability.**
  - **How to estimate *P(E)*, *P(H)* and *P(E|H)*?**
  - **From the given data (this is the training phase of the classifier)**

Naïve Bayes Classifier

- Suppose we have n classes $C_1, C_2,\ldots,C_n$. Given an unknown sample X, the classifier will predict that $X=(x_1,x_2,\ldots,x_m)$ belongs to the class with the highest posteriori probability:

$$X \in C_i \text{ if } P(C_i|X) > P(C_j|X) \text{ for } 1\leq j \leq n, j\neq i$$

Maximize $\frac{P(X|C_i)P(C_i)}{P(X)}$ ➔ maximize $P(X|C_i)P(C_i)$

- $P(C_i) = s_i/s$
- $P(X|C_i)= \prod_{k=1}^{n} P(x_k|C_i)$ where $P(x_k|C_i) = s_{ik}/s_i$
- Greatly reduces the computation cost, only count the class distribution.
- Naïve: class conditional independence

Naïve Bayes Algorithm - Basic Assumption

- **Naïve Bayes uses all attributes to make a decision and allows them to make contributions to the decision that are *equally important & independent* of one another**
  - **Independence assumption – attributes are conditionally independent of each other given the class**
  - **Equally importance assumption – attributes are equally important**
  - **Unrealistic assumptions! ➔ it is called *Naïve* Bayes**
    - **Are dependent of one another**
    - **Attributes are not equally important**
  - **But these assumptions lead to a simple method which works surprisingly well in practice!**

Naïve Bayes (NB) for the Tennis Example

- **Consider the tennis data**
- **Suppose we encounter a new example which has to be classified:**

Outlook	Tempreature	Humidity	Windy	Play
sunny	cool	high	true	??

Outlook	Tempreature	Humidity	Windy	Play
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

- **Recall the Bayes theorem:**

$$P(H|E)=\frac{P(E|H)P(H)}{P(E)}$$

- **What are H & E for our example?**
  - **the hypothesis H is that the class is Play=Yes (and there is another hypothesis: that the class is Play=No)**
  - **the evidence E is the new example (i.e. a particular combination of observed attribute values for the new day)**

Naïve Bayes for the Tennis Example - 2

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}$$

- We need to calculate P(*yes*|E) and P(*no*|E)

  where E is

Outlook	Tempreature	Humidity	Windy	Play
sunny	cool	high	true	??

  compare them
- If we denote the 4 pieces of evidence
  - outlook=sunny with with E_1
  - temperature=cool with E_2
  - humidity=high with E_3
  - windy=true with E_4

  and assume that they are independent given the class, than their combined probability is obtained by multiplication:

$$P(E \mid yes) = P(E_1 \mid yes)\,P(E_2 \mid yes)\,P(E_3 \mid yes)\,P(E_4 \mid yes)$$

Naïve Bayes for the Tennis Example - 3

- Hence

$$P(yes \mid E) = \frac{P(E_1 \mid yes)\,P(E_2 \mid yes)\,P(E_3 \mid yes)\,P(E_4 \mid yes)\,P(yes)}{P(E)}$$

- Probabilities in the numerator will be estimated from the data.
- There is no need to estimate P(E) as it will appear also in the denominators of the other hypotheses, i.e. it will disappear when we compare them.

$$P(no \mid E) = \frac{P(E_1 \mid no)\,P(E_2 \mid no)\,P(E_3 \mid no)\,P(E_4 \mid no)\,P(no)}{P(E)}$$

Naïve Bayes for the Tennis Example – cont.1

- Tennis data - counts and probabilities:

	outlook		temperature			humidity			windy			play	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2	3	hot	2	2	high	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								

Outlook	Tempreature	Humidity	Windy	Play
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

proportions of days when play is yes

proportions of days when humidity is normal and play is yes i.e. the probability of humidity to be normal given that play is yes

Naïve Bayes for the Tennis Example – cont.2

$$P(yes \mid E) = ?$$

$$P(yes \mid E) = \frac{P(E_1 \mid yes)\,P(E_2 \mid yes)\,P(E_3 \mid yes)\,P(E_4 \mid yes)\,P(yes)}{P(E)}$$

	outlook		temperature			humidity			windy			play	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2	3	hot	2	2	high	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								

⇒ P(E_1|*yes*)=P(outlook=sunny|*yes*)=2/9

P(E_2|*yes*)=P(temperature=cool|*yes*)=3/9

P(E_3|*yes*)=P(humidity=high|*yes*)=3/9

P(E_4|*yes*)=P(windy=true|*yes*)=3/9

- P(*yes*) =? - the probability of a *Play=yes* without knowing any E, i.e. anything about the particular day; the prior probability of yes; P(Play=yes) = 9/14

Naïve Bayes for the Tennis Example – cont.3

- **By substituting the respective evidence probabilities:**

$$P(yes\,|\,E)=\frac{\frac{2}{9}\frac{3}{9}\frac{3}{9}\frac{3}{9}\frac{9}{14}}{P(E)}=\frac{0.0053}{P(E)}$$

Outlook	Yes	No		Humidity	Yes	No
sunny	2/9	3/5		high	3/9	4/5
overcast	4/9	0		normal	6/9	1/5
rain	3/9	2/5		Windy		
Tempreature				true	3/9	3/5
hot	2/9	2/5		false	6/9	2/5
mild	4/9	2/5		**Play=yes**	9/14	
cool	3/9	1/5		**Play=No**	5/14	

- **Similarly calculating:** $P(no\,|\,E)$

$$P(no\,|\,E)=\frac{\frac{3}{5}\frac{1}{5}\frac{4}{5}\frac{3}{5}\frac{5}{14}}{P(E)}=\frac{0.0206}{P(E)}$$

- **=>** $P(no|\,E)>P(yes|\,E)$
- **=> for the new day play = no is more likely than play = yes (4 times more likely)**

A Problem with Naïve Bayes

- **Suppose that the training data for the tennis example was different:**
  - **outlook=sunny had been always associated with play=no (i.e. outlook=sunny had never occurred together with play=yes)**
- **Then**
  - **P(yes|outlook=sunny)=0 and P(no|outlook=sunny)=1**

$$P(yes\,|\,E)=\frac{\underbrace{P(E_1\,|\,yes)}_{=0}\,P(E_2\,|\,yes)\,P(E_3\,|\,yes)\,P(E_4\,|\,yes)\,P(yes)}{P(E)}$$

  - **=> final probability P(yes|E)=0 no matter of the other probabilities, i.e. zero probabilities hold a veto over the other probabilities**
- **This is a problem!**
  - **If it happens in the training set ➔ poor prediction on new data**
- **Solution: use Laplace estimator (correction) to calculate probabilities**
  - **Adds 1 to the numerator and *k* to the denominator, where *k* is the number of attribute values for a given attribute**

Laplace Correction – Modified Tennis Example

	outlook		
	yes	no	…
sunny	0	5	…
overcast	4	0	…
rainy	3	2	…
			…
sunny	0/7	5/7	…
overcast	4/7	0/7	…
rainy	3/7	2/7	…

P(sunny|yes)=0/7

P(overcast|yes)=4/7

P(rainy|yes)=3/7

- **Laplace correction adds 1 to the numerator and 3 to the denominator**

$$P(sunny\,|\,yes)=\frac{0+1}{7+3}=\frac{1}{10}=0.1$$

$$P(overcast\,|\,yes)=\frac{4+1}{7+3}=\frac{5}{10}=0.5$$

$$P(rainy\,|\,yes)=\frac{3+1}{7+3}=\frac{4}{10}=0.4$$

Ensures that an attribute value which occurs 0 times will receive a nonzero (although small) probability.

Laplace Correction – Original Tennis Example

	outlook		
	yes	no	…
sunny	2	3	…
overcast	4	0	…
rainy	3	2	…
			…
sunny	2/9	3/5	…
overcast	4/9	0/5	…
rainy	3/9	2/5	…

P(sunny|yes)=2/9

P(overcast|yes)=4/9

P(rainy|yes)=3/9

$$P(sunny\,|\,yes)=\frac{2+1}{9+3}=\frac{3}{12}=0.25$$

$$P(overcast\,|\,yes)=\frac{4+1}{9+3}=\frac{5}{12}=0.416$$

$$P(rainy\,|\,yes)=\frac{3+1}{9+3}=\frac{4}{12}=0.33$$

Handling Missing Values

- **Easy:**
  - **Missing value in the evidence E (the new example) - omit this attribute**

    e.g. E: outlook=?, temperature=cool, humidity=high, windy=true

    then

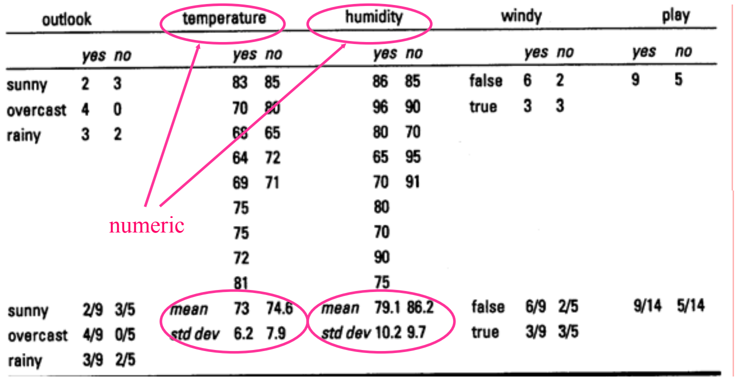
$$P(yes\,|\,E)=\frac{\frac{3}{9}\frac{3}{9}\frac{3}{9}\frac{9}{14}}{P(E)}=\frac{0.0238}{P(E)}\qquad P(no\,|\,E)=\frac{\frac{1}{5}\frac{4}{5}\frac{3}{5}\frac{5}{14}}{P(E)}=\frac{0.0343}{P(E)}$$

- **Compare these results with the previous!**

  - as one of the fractions is missing, the probabilities are higher then before, but this is not a problem as there is a a missing fraction in both cases

  - **Missing value in the training example:**
    - **do not include them in the frequency counts and calculate the probabilities based on the number of values that actually occur and not on the total number of training examples**

Handling Numeric Attributes



outlook			temperature			humidity			windy			play	
	yes	*no*		*yes*	*no*		*yes*	*no*		*yes*	*no*	*yes*	*no*
sunny	2	3		83	85		86	85	false	6	2	9	5
overcast	4	0		70	80		96	90	true	3	3		
rainy	3	2		68	65		80	70					
				64	72		65	95					
				69	71		70	91					
				75			80						
				75			70						
				72			90						
				81			75						
sunny	2/9	3/5	*mean*	73	74.6	*mean*	79.1	86.2	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	*std dev*	6.2	7.9	*std dev*	10.2	9.7	true	3/9	3/5		
rainy	3/9	2/5											

- **We would like to classify the following new example:**

outlook=sunny, temperature=66, humidity=90, windy=true

- **Q. How to calculate P(temperature=66|yes), P(humidity=90|yes), P(temperature=66|no), P(humidity=90|no) ?**

Using Probability Density Function

- **A. By assuming that numerical values have a *normal* (Gaussian) probability distribution and using <u>probability density function</u>**
- **For a *normal* distribution with mean μ and standard deviation σ, the probability density function is:**

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- **What is the meaning of the probability density function of a continuous random variable?**
  - **Closely related to probability but is not exactly the probability (e.g. the probability that *x is <u>exactly</u> 66 is 0)***
  - **The probability that a given value *x* takes a value in a small region (between *x*- ε/2 and *x* + ε/2) is ε f(x) (e.g. that probability that *x* is between 64 and 68 is *f(x)*)**

Calculating Probabilities Using Probability Density Function

$$f(temperature=66\,|\,yes)=\frac{1}{6.2\sqrt{2\pi}}e^{-\frac{(66-73)^2}{2*6.2^2}}=0.034$$

$$f(humidity=90\,|\,yes)=0.0221$$

$$P(yes\,|\,E)=\frac{\frac{2}{9}\,0.034\;0.0221\frac{3}{9}\frac{9}{14}}{P(E)}=\frac{0.000036}{P(E)}$$

$$P(no\,|\,E)=\frac{\frac{3}{5}\,0.0291\;0.038\frac{3}{5}\frac{5}{14}}{P(E)}=\frac{0.000136}{P(E)}$$

=>P(no|E) > P(yes|E)

=> no play

- **Compare with the categorical tennis data!**

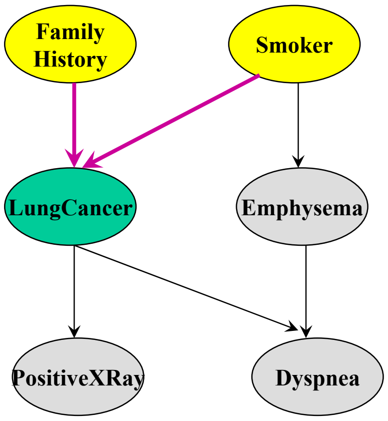
Naïve Bayes – Advantages & Disadvantages

- **Advantages:**
  - **simple approach**
  - **clear semantics for representing, using and learning probabilistic knowledge**
  - **requires 1 scan of the training data**
  - **in many cases outperforms more sophisticated learning methods ➔ always try the simple method first!**
- **Disadvantages:**
  - **While there is only 1 scan, it is still computationally expensive**
  - **since attributes are treated as though they were completely independent, the existence of dependencies between attributes skews the learning process!**
  - **Normal distribution assumption when dealing with numeric attributes – (minor) restriction ➔ discretize the data or follow other distributions**

Belief Network

- Allows class conditional dependencies to be expressed.
- It has a directed acyclic graph (DAG) and a set of conditional probability tables (CPT).
- Nodes in the graph represent variables and arcs represent probabilistic dependencies. (child dependent on parent)
- There is one table for each variable X. The table contains the conditional distribution P(X|Parents(X)).

Bayesian Belief Networks Example



Bayesian Belief Networks

	(FH, S)	(FH, ~S)	(~FH, S)	(~FH, ~S)
LC	0.8	0.5	0.7	0.1
~LC	0.2	0.5	0.3	0.9

The conditional probability table for the variable LungCancer

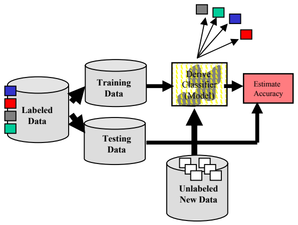
Bayesian Belief Networks

Several cases of learning Bayesian belief networks:

- When both network structure and all the variables are given then the learning is simply computing the CPT.
- When network structure is given but some variables are not known or observable, then iterative learning is necessary (compute gradient *ln*P(S|H), take steps toward gradient and normalize).
- Many algorithms for learning the network structure exist.

Classification Methods

- ❖ Neural Networks
- ❖ Bayesian Classification
- ❖ K-Nearest Neighbour
- ❖ Decision Tree Induction
- ❖ Associative Classifiers
- ❖ Support Vector Machines
- ❖ Case-Based Reasoning
- ❖ Genetic Algorithms
- ❖ Rough Set Theory
- ❖ Fuzzy Sets
- ❖ Etc.



Lecture Outline

Part III: k-Nearest Neighbour *(30 minutes)*

- **Lazy Learning**
  - **Nearest Neighbour**
  - **K-Nearest neighbours**
- **Agglomerative Nearest Neighbours**

Part IV: Decision Trees *(1 hour)*

- **What is a Decision Tree?**
- **Building a tree**
- **Pruning a tree**

Part V: Associative Classifiers *(1 hour)*

- **Rule Generation**
- **Rule Pruning**
- **Rule Selection**
- **Rule Combination**

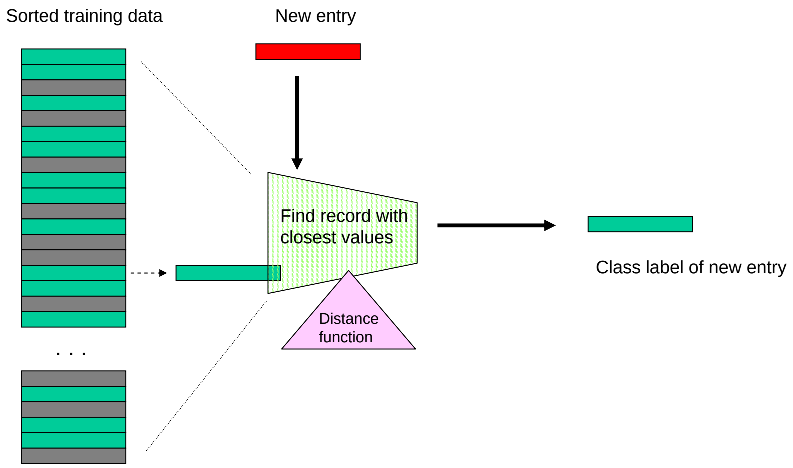
k-Nearest Neighbours (k-NN) Classification

- In k-nearest-neighbour classification, the training dataset is used to classify each member of a "target" dataset.
- There is no model created during a learning phase but the training set itself.
- It is called a *lazy-learning* method.
- Rather than building a model and referring to it during the classification. K-NN directly refers to the training set for classification.

The Simple Nearest Neighbour Approach

- Nearest Neighbour is very simple. The training is nothing more than sorting the training data and storing it in a list.
- To classify a new entry, this entry is compared to the list to find the closest record, with value as similar as possible to the entry to classify (i.e. nearest neighbour). The class of this record is simply assigned to the new entry.
- Different measures of similarity or distance can be used.

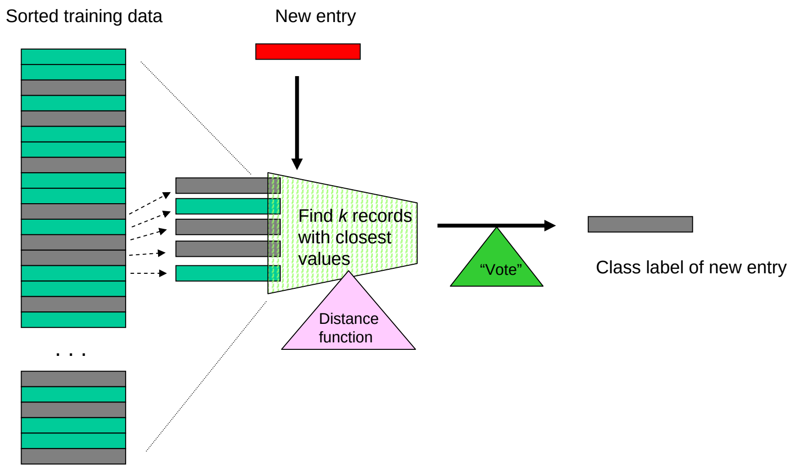
The Nearest Neighbour



The k-Nearest Neighbour Approach

- The k-Nearest Neighbour is a variation of Nearest Neighbour. Instead of looking for only the closest record to the entry to classify, we look for the k records closest to it.
- To assign a class label to the new entry, from all the labels of the k nearest records we take the majority class label.
- Nearest Neighbour is a case of k-Nearest Neighbours with k=1.

k Nearest Neighbours



Agglomerative Nearest Neighbours

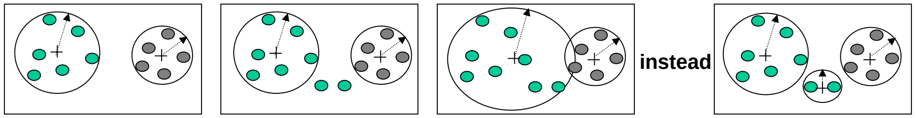
- Training records are put together in groups as the learning process goes on. The approach is named agglomerative because groups or clusters are merged during the learning.
- The training is relatively simple:
  - Each cluster has a center *c* and a radius *r* and the class label of its records. Initially each record in the training set forms a cluster on its own.
  - Two clusters that are close together (within some *epsilon* distance of each other) and classify the same category are combined to build a new aggregate cluster: a hypersphere of a larger radius and a new center.
  - If a cluster is not close to any other clusters given *epsilon*, it remains separate.

Agglomerative NN Classification

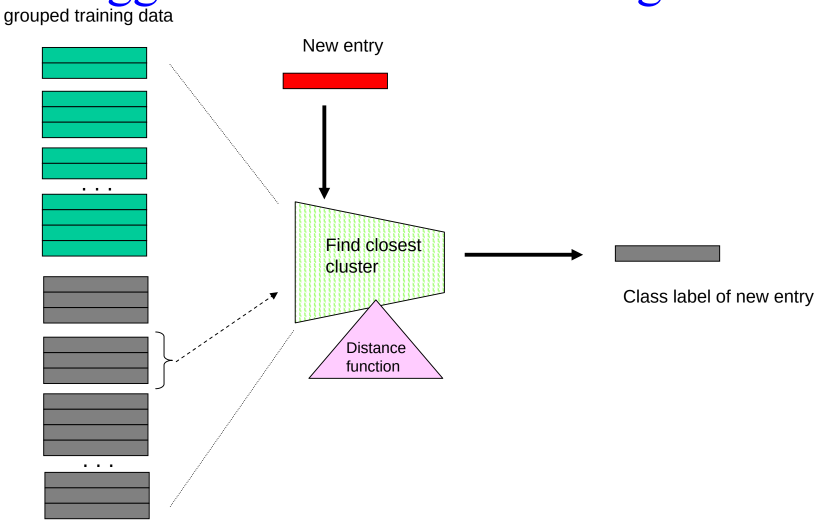
- The classification of a new entry consists of finding the closest cluster to it and assign it the label attached to that cluster.
- Agglomerative NN has a slower training than NN or k-NN but has the advantage of using less memory. There is no need to store all the training set but only the center and radius of each cluster.
- In the two extremes: if all records are far from each other they remain separate clusters ➔ Nearest Neighbour. If all points are close to each other we end-up with as many clusters as we have classes.

Cluster Overlap Problem

- Since clusters are hyperspheres, overlap of clusters of different labels are bound to happen.
- Clusters that grow and overlap with nearby clusters that classify differently can reduce accuracy. A new entry that falls in an overlap area between two clusters can easily be misclassified.
- One solution is to inhibit clusters from growing if enlarging a hypersphere would generate overlap with a different class label cluster, and simply create new a small cluster in between.



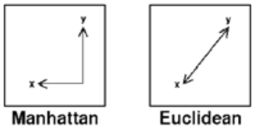
Agglomerative Nearest Neighbours



Distance Measures

- The most used distance function is the Euclidian distance: $d(X,Y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$
- However, other measures are possible
  - the Manhattan distance: $d(X,Y)=\sum_{i=1}^{n}(x_i-y_i)$
  - the Chebychev: $d(X,Y)=\max_{i=1}^{n}(x_i-y_i)$
  - the cosine measure: $d(X,Y)=\frac{\sum_{i=1}^{n}(x_i.y_i)}{\sqrt{\sum_{i=1}^{n}x_i}.\sqrt{\sum_{i=1}^{n}y_i}}$
  - Pearson's correlation:

$$d(X,Y)=\frac{\sum_{i=1}^{n}(x_i-\bar{x}).(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}.\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}}$$



Similarity of Categorical Data

- Similarity measure is the inverse of distance measure
- For categorical Data or Binary data the distance is:
  - Simple match: $d(X,Y)=\sum_{i=1}^{n}\partial(x_i,y_i)$

    where $\partial(x_i,y_i)=0$ *if* $x_i=y_i$; 1 *otherwise*
  - Jaccard Coefficient: $Sim(X,Y)=\frac{\bigcap_{i=1}^{n}(x_i,y_i)}{\bigcup_{i=1}^{n}(x_i,y_i)}$

Attribute Weights

- Not all attributes have the same importance in measuring similarity (or distance). Attributes could be weighted.

$$d(X,Y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

Non-weighted versus weighted Euclidian distance

$$d(X,Y)=\sqrt{\sum_{i=1}^{n}w_i*(x_i-y_i)^2}$$

Lecture Outline

Part III: k-Nearest Neighbour *(30 minutes)*

- **Lazy Learning**
  - **Nearest Neighbour**
  - **K-Nearest neighbours**
- **Agglomerative Nearest Neighbours**

Part IV: Decision Trees *(1 hour)*

- **What is a Decision Tree?**
- **Building a tree**
- **Pruning a tree**

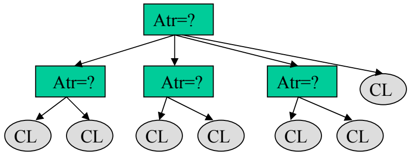
Part V: Associative Classifiers *(1 hour)*

- **Rule Generation**
- **Rule Pruning**
- **Rule Selection**
- **Rule Combination**

What is a Decision Tree?

A decision tree is a flow-chart-like tree structure.

- Atr=? Internal node denotes a test on an attribute
- → Branch represents an outcome of the test
  - All tuples in branch have the same value for the tested attribute.
- CL Leaf node represents class label or class label distribution.

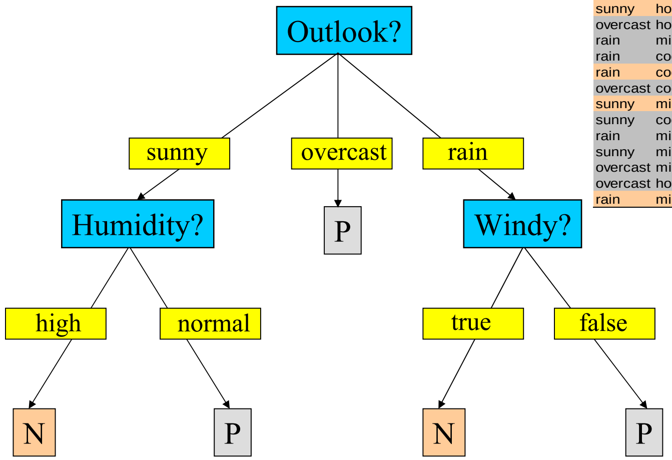


Training Dataset

- An Example from Quinlan's ID3

Outlook	Tempreature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

A Sample Decision Tree



Outlook	Tempreature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Decision-Tree Classification Methods

- The basic top-down decision tree generation approach usually consists of two phases:

1. **Tree construction**
   - At the start, all the training examples are at the root.
   - Partition examples are recursively based on selected attributes.
2. **Tree pruning**
   - Aiming at removing tree branches that may reflect noise in the training data and lead to errors when classifying test data ➔ improve classification accuracy.

Decision Tree Construction

Recursive process:

- Tree starts a single node representing all data.
- If sample are all same class then node becomes a leaf labeled with class label.
- Otherwise, ***select attribute*** that best separates sample into individual classes.

How?

- Recursion stops when:
  - Sample in node belong to the same class (majority);
  - There are no remaining attributes on which to split;
  - There are no samples with attribute value.

Choosing the Attribute to Split Data Set

- The measure is also called ***Goodness function***
- Different algorithms may use different goodness functions:
  - **information gain** (ID3/C4.5)
    - assume all attributes to be categorical.
    - can be modified for continuous-valued attributes.
  - **gini index**
    - assume all attributes are continuous-valued.
    - assume there exist several possible split values for each attribute.
    - may need other tools, such as clustering, to get the possible split values.
    - can be modified for categorical attributes.

Information Gain (ID3/C4.5)

- Assume that there are two classes, *P* and *N*.
  - Let the set of examples *S* contain *x* elements of class *P* and *y* elements of class *N*.
  - The amount of information, needed to decide if an arbitrary example in *S* belong to *P* or *N* is defined as:

$$I(S_P,S_N) = -\frac{x}{x+y}\log_2\frac{x}{x+y} - \frac{y}{x+y}\log_2\frac{y}{x+y}$$

In general

p_i is estimated by s_i/s

$$I(s_1,s_2,\ldots,s_n) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

- Assume that using attribute A as the root in the tree will partition *S* in sets $\{S_1, S_2, \ldots, S_v\}$.
  - If S_i contains x_i examples of *P* and y_i examples of *N*, the information needed to classify objects in all subtrees S_i :

$$E(A) = \sum_{i=1}^{v}\frac{x_i+y_i}{x+y} I(S_{Pi},S_{Ni})$$

In general

$$E(A) = \sum_{i=1}^{v}\frac{s_{1i}+s_{2i}+\ldots+s_{ni}}{s} I(s_{1i},s_{2i},\ldots,s_{ni})$$

Information Gain -- Example

- The attribute *A* is selected such that the ***information gain***

$$\text{gain}(A) = I(S_P,S_N) - E(A)$$

is maximal, that is, *E(A)* is minimal since $I(S_P,S_N)$ is the same to all attributes at a node.

- In the given sample data, attribute *outlook* is chosen to split at the root :

gain(outlook) = 0.246

gain(temperature) = 0.029

gain(humidity) = 0.151

gain(windy) = 0.048

Information gain measure tends to favor attributes with many values. Other possibilities: Gini Index, χ^2, etc.

***Gini* Index**

- If a data set *S* contains examples from *n* classes, gini index, *gini*(*S*) is defined as

$$gini(S) = 1 - \sum_{j=1}^{n} p_j^2$$

where p_j is the relative frequency of class *j* in *S*.

- If a data set *S* is split into two subsets S_1 and S_2 with sizes N_1 and N_2 respectively, the *gini* index of the split data contains examples from *n* classes, the *gini* index *gini*(*S*) is defined as

$$gini_{split}(S) = \frac{N_1}{N} gini(S_1) + \frac{N_2}{N} gini(S_2)$$

- The attribute that provides the smallest $gini_{split}(S)$ is chosen to split the node (***need to enumerate all possible splitting points for each attribute***).

Example for gini Index

- Suppose there two attributes: *age* and *income*, and the class label is buy and not buy.
- There are three possible split values for age: 30, 40, 50.
- There are two possible split values for income: 30K, 40K
- We need to calculate the following gini index
  - gini *age = 30* (S),
  - gini *age = 40* (S),
  - gini *age = 50* (S),
  - gini *income = 30k* (S),
  - gini *income = 40k* (S)
- Choose the minimal one as the split attribute

Primary Issues in Tree Construction

- **Split criterion**:
  - Used to select the attribute to be split at a tree node during the tree generation phase.
  - Different algorithms may use different goodness functions: information gain, gini index, etc.
- **Branching scheme**:
  - Determining the tree branch to which a sample belongs.
  - binary splitting (gini index) versus many splitting (information gain).
- **Stopping decision**: When to stop the further splitting of a node, e.g. impurity measure.
- **Labeling rule**: a node is labeled as the class to which most samples at the node belong.

How to construct a tree?

- Algorithm
  - greedy algorithm
    - make optimal choice at each step: select the best attribute for each tree node.
  - top-down recursive divide-and-conquer manner
    - from root to leaf
    - split node to several branches
    - for each branch, recursively run the algorithm

Example for Algorithm (ID3)

- All attributes are categorical
- Create a node N;
  - if samples are all of the same class C, then return N as a leaf node labeled with C.
  - if attribute-list is empty then return N as a left node labeled with the most common class.
- Select split-attribute with highest information gain
  - label N with the split-attribute
  - for each value A_i of split-attribute, grow a branch from Node N
  - let S_i be the branch in which all tuples have the value A_i for split- attribute
    - if S_i is empty then attach a leaf labeled with the most common class.
    - Else recursively run the algorithm at Node S_i
- Until all branches reach leaf nodes

How to use a tree?

- Directly
  - test the attribute value of unknown sample against the tree.
  - A path is traced from root to a leaf which holds the label.
- Indirectly
  - decision tree is converted to classification rules.
  - one rule is created for each path from the root to a leaf.
  - IF-THEN rules are easier for humans to understand.

Avoid Over-fitting in Classification

- A tree generated may over-fit the training examples due to noise or too small a set of training data.

- Two approaches to avoid over-fitting:
  - (Stop earlier): Stop growing the tree earlier.
  - (Post-prune): Allow over-fit and then post-prune the tree.

- Approaches to determine the correct final tree size:
  - Separate training and testing sets or use cross-validation.
  - Use all the data for training, but apply a statistical test (e.g., chi-square) to estimate whether expanding or pruning a node may improve over entire distribution.
  - Use Minimum Description Length (MDL) principle: halting growth of the tree when the encoding is minimized.
- Rule post-pruning (C4.5): converting to rules before pruning.

Continuous and Missing Values in Decision-Tree Induction

- Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals.

Temperature	40	48	60	72	80	90
play tennis	No	No	Yes	Yes	Yes	No

- Sort the examples according to the continuous attribute *A*, then identify adjacent examples that differ in their target classification, generate a set of candidate thresholds midway, and select the one with the maximum gain.
- Extensible to split continuous attributes into multiple intervals.
- Assign missing attribute values either
  - Assign the most common value of *A(x)*.
  - Assign probability to each of the possible values of *A*.

Alternative Measures for Selecting Attributes

- Info gain naturally favours attributes with many values.
- One alternative measure: gain ratio (Quinlan'86) which is to penalize attribute with many values.

$$SplitInfo\ (S,A) \equiv -\sum \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}.$$

$$GainRatio\ (S,A) = \frac{Gain\ (S,A)}{SplitInfo\ (S,A)}$$

  - Problem: denominator can be 0 or close which makes GainRatio very large.
- Distance-based measure (Lopez de Mantaras'91):
  - define a distance metric between partitions of the data.
  - choose the one closest to the perfect partition.
- There are many other measures. Mingers'91 provides an experimental analysis of effectiveness of several selection measures over a variety of problems.

Tree Pruning

- A decision tree constructed using the training data may have too many branches/leaf nodes.
  - Caused by noise, over-fitting.
  - May result poor accuracy for unseen samples.
- Prune the tree: merge a subtree into a leaf node.
  - Using a set of data different from the training data.
  - At a tree node, if the accuracy without splitting is higher than the accuracy with splitting, replace the subtree with a leaf node, label it using the majority class.
- Issues:
  - Obtaining the testing data.
  - Criteria other than accuracy (e.g. minimum description length).

Pruning Criterion

- Use a separate set of examples to evaluate the utility of post-pruning nodes from the tree.
  - CART uses cost-complexity pruning.
- Apply a statistical test to estimate whether expanding (or pruning) a particular node.
  - C4.5 uses pessimistic pruning.
- Minimum Description Length (no test sample needed).
  - SLIQ and SPRINT use MDL pruning.

Pruning Criterion --- MDL

- Best binary decision tree is the one that can be encoded with the fewest number of bits
  - Selecting a scheme to encode a tree
  - Comparing various subtrees using the cost of encoding
  - The best model minimizes the cost
- Encoding schema
  - One bit to specify whether a node is a leaf (0) or an internal node (1)
  - *loga* bits to specify the splitting attribute
  - Splitting the value for the attribute:
    - categorical --- log(*v*-1) bits
    - numerical --- log 2^{v-2}

Lecture Outline

Part III: k-Nearest Neighbour *(30 minutes)*

- **Lazy Learning**
  - **Nearest Neighbour**
  - **K-Nearest neighbours**
- **Agglomerative Nearest Neighbours**

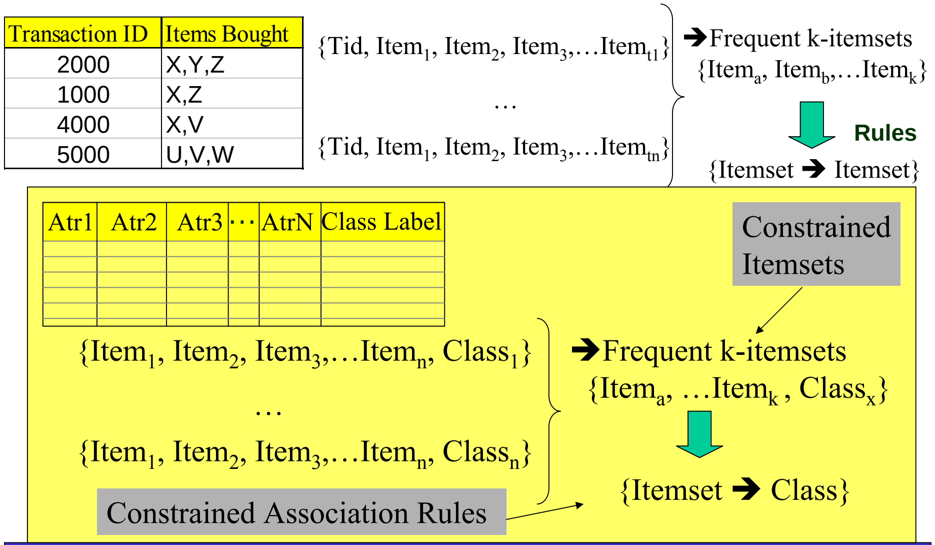
Part IV: Decision Trees *(1 hour)*

- **What is a Decision Tree?**
- **Building a tree**
- **Pruning a tree**

Part V: Associative Classifiers *(1 hour)*

- **Rule Generation**
- **Rule Pruning**
- **Rule Selection**
- **Rule Combination**

How do Associative Classifiers Work?



Transaction ID	Items Bought
2000	X,Y,Z
1000	X,Z
4000	X,V
5000	U,V,W

{Tid, Item_1, Item_2, Item_3,…Item_{t1}}

…

{Tid, Item_1, Item_2, Item_3,…Item_{tn}}

➔Frequent k-itemsets {Item_a, Item_b,…Item_k}

Rules

{Itemset ➔ Itemset}

Atr1	Atr2	Atr3	···	AtrN	Class Label

Constrained Itemsets

{Item_1, Item_2, Item_3,…Item_n, Class_1}

…

{Item_1, Item_2, Item_3,…Item_n, Class_n}

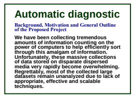
➔Frequent k-itemsets {Item_a, …Item_k , Class_x}

{Itemset ➔ Class}

Constrained Association Rules

Modeling documents

{bread, milk, beer,…} ➡ (Bread, milk) ➡ Bread → milk

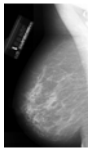


Automatic diagnostic

Background, Motivation and General Outline of the Proposed Project

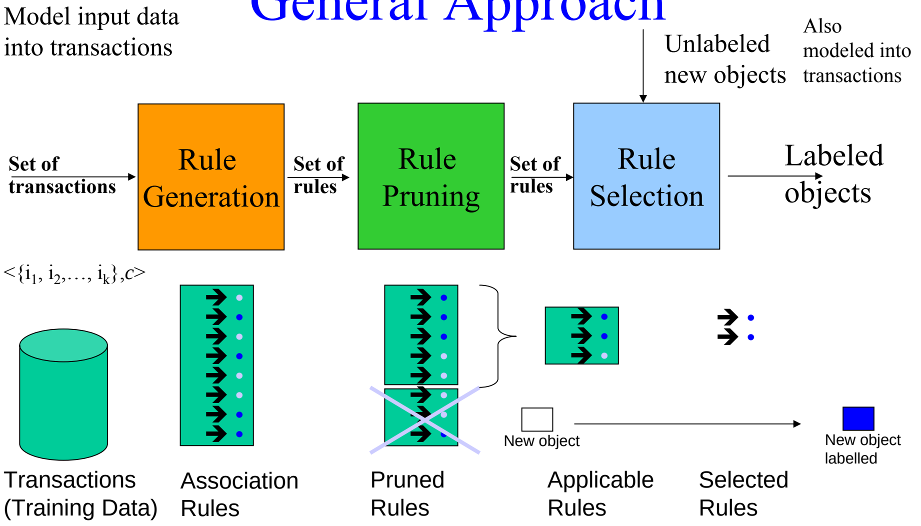
We have been collecting tremendous amounts of information counting on the power of computers to help efficiently sort through this amalgam of information. Unfortunately, these massive collections of data stored on disparate dispersed media very rapidly become overwhelming. Regrettably, most of the collected large datasets remain unanalyzed due to lack of appropriate, effective and scalable techniques.

{term1, term2,…,Ca} ➡ (term2, Ca) ➡ term2 → Ca



{f1, f2,…,Ca} ➡ (f3, f5, Ca) ➡ f3^f5 → Ca

General Approach



Model input data into transactions

Unlabeled new objects

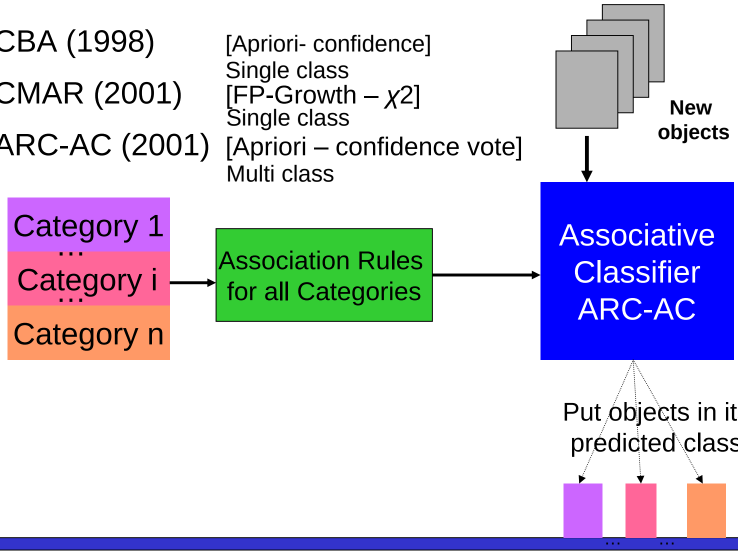
Also modeled into transactions

Association Rules - Classification for all Categories

CBA (1998) [Apriori- confidence]
Single class

CMAR (2001) [FP-Growth – $\chi 2$]
Single class

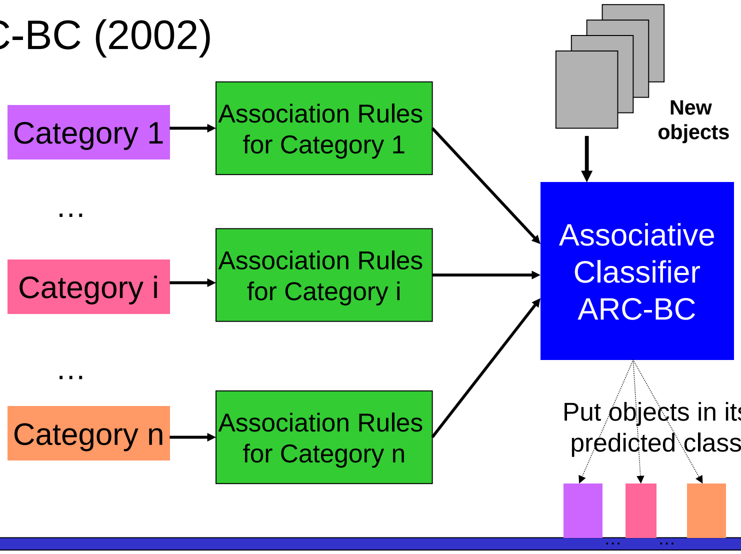
ARC-AC (2001) [Apriori – confidence vote]
Multi class



New objects
Category 1
…
Category i
…
Category n
Association Rules for all Categories
Associative Classifier ARC-AC
Put objects in its predicted class
…
…

Association Rules - Classification by Category

ARC-BC (2002)



New objects
Category 1
Association Rules for Category 1
…
Category i
Association Rules for Category i
…
Category n
Association Rules for Category n
Associative Classifier ARC-BC
Put objects in its predicted class
…
…

Association Rules: Advantages & Issues

- AR are well studied
  - fast
  - scalable
- No independence assumption btw. attributes
- Attributes:
  - large number
  - variable number, can handle missing values
- Transparency

- AC are in an early stage of development
  - use simple rules
  - naïve selection function
- AC models consist of a large number of rules
  - harder selection
  - redundant, uninteresting rules
  - longer classification time
  - difficult to manually revisit rules

Solution: Pruning Techniques

Pruning Rules

Large number of rules ⇨ Noisy information

Large number of rules ⇨ Long classification time

Solution: Pruning Techniques

- Removing low ranked specialized rules;

  $R_1 : F_1 \Rightarrow C$ Confidence 90%

  $R_2 : F_1 \wedge F_2 \Rightarrow C$ Confidence 80%

  $\Rightarrow R_1$

- Eliminate conflicting rules (for single-class classification);

  $F_1 \Rightarrow C_1 \wedge F_1 \Rightarrow C_2$

- Database coverage;

Classification Stage

Let S be the classification system

A new object O <f1; f3; f4; f7; f9 >

f1 ➔ C1 confidence 0.9

f3 & f4 ➔ C2 confidence 0.85

f4 ➔ C2 confidence 0.8

f7 ➔ C1 confidence 0.6

f9 ➔ C3 confidence 0.5

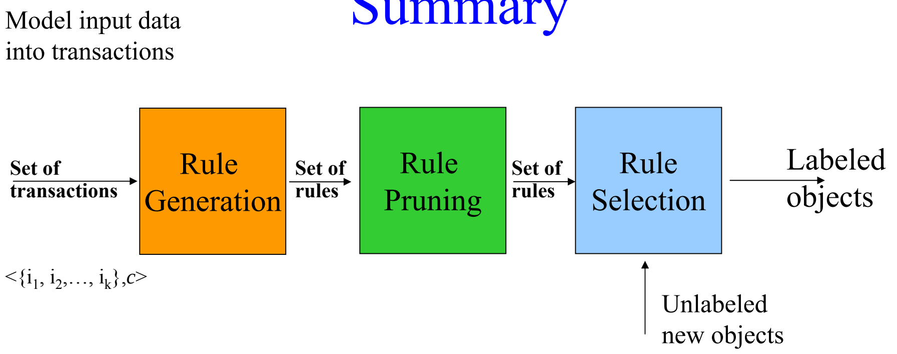
C2 0.825

C1 0.75

C3 0.5

Using the *dominance factor* we chose the winning categories. If δ=100% C2 is winning. If δ=80% O is predicted to fall in C2 and C1.

Summary



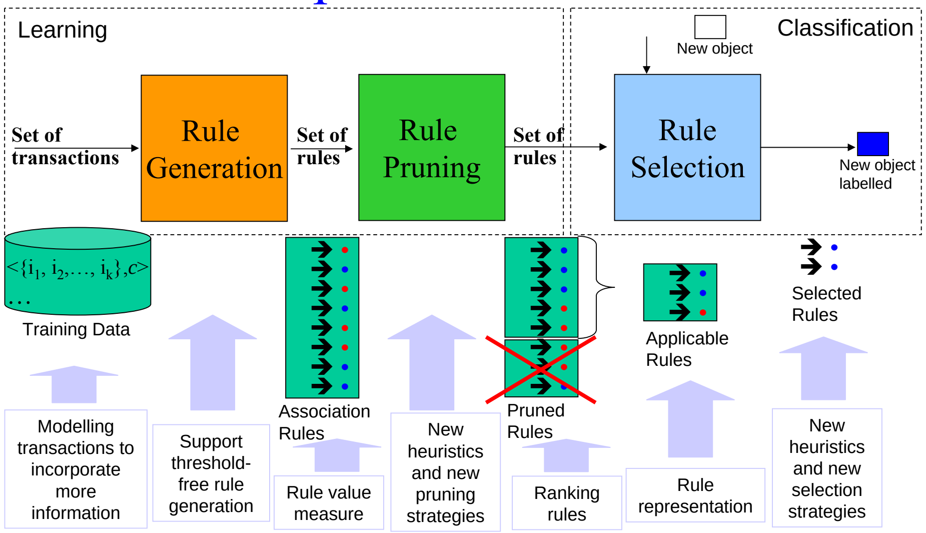
Model input data into transactions

Set of transactions → Rule Generation → Set of rules → Rule Pruning → Set of rules → Rule Selection → Labeled objects

<{i$_1$, i$_2$,…, i$_k$},*c*>

Unlabeled new objects

Open Problems?



Learning

Classification

Set of transactions → Rule Generation → Set of rules → Rule Pruning → Set of rules → Rule Selection → New object labelled

New object

<{i1, i2,…, ik},c> …

Training Data

Association Rules

Pruned Rules

Applicable Rules

Selected Rules

Modelling transactions to incorporate more information

Support threshold-free rule generation

Rule value measure

New heuristics and new pruning strategies

Ranking rules

Rule representation

New heuristics and new selection strategies

What is Prediction?

The goal of prediction is to forecast or deduce the value of an attribute based on values of other attributes.

- A model is first created based on the data distribution.
- The model is then used to predict future or unknown values.

In Data Mining

If forecasting discrete value ➔ **Classification**

If forecasting continuous value ➔ **Prediction**

Prediction

Prediction of continuous values can be modeled by statistical techniques.

- Linear regression
- Multiple regression
- Polynomial regression
- Poisson regression
- Log-linear regression
- Etc.

Linear Regression

- Linear regression:

Approximate data distribution by a line Y = α + βX

Y is the *response variable* and X the *predictor variable*.

α and β are regression coefficients specifying the intercept and the slope of the line. They are calculated by least square method:

$$\beta = \frac{\sum_{i=1}^{s}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{s}(x_i - \bar{x})^2}$$

$$\alpha = \bar{y} - \beta\bar{x}$$

Where $\bar{x}$ and $\bar{y}$ are respectively the average of $x_1, x_2, \ldots, x_s$ and $y_1, y_2, \ldots, y_s$.

- Multiple regression: $Y = \alpha + \beta_1 X_1 + \beta_2 X_2$.
  - Many nonlinear functions can be transformed into the above.