Data Wrangling and Data Analysis
Regression, Classification and Evaluation Techniques

Hakim Qahtan

Department of Information and Computing Sciences

Utrecht University

Topics for Today

- Data Analytics Models
- Regression
- Classification

Data Analytics Models

- Prediction vs Inference
- Regression vs Classification
- Supervised vs Unsupervised
- Typologies of analytical tasks

Prediction vs. Inference

$$\hat{y} = \hat{f}(x)$$

- Prediction
  - Main interest is in the outcome value $\hat{y}$
  - Use the model to predict the outcomes for new data points.
- Inference
  - Main interest is in $\hat{f}$
  - Use training data to learn a model
  - Association of predictors
  - Types of relations

Regression vs. Classification

- Regression
  - Algorithms attempt to estimate the mapping function f from the input variables x to numerical or continuous output variables y
  - Given a dataset about house prices – predict the price of a given house
- Classification
  - Algorithms attempt to estimate the mapping function f from the input variables x to discrete or categorical output variables y
  - Houses dataset – predict if the selling price is more or less the recommended price

Supervised vs. Unsupervised

- Supervised models
  - Training a model using data which is well "labeled"
  - Supervised learning algorithm learns from labeled training data to predict outcomes for unforeseen data
- Unsupervised models
  - The model works on its own to discover information
  - Deals with unlabeled data and looks for unknown patterns in data
- There are also Semi-Supervised Models
  - Small amount of labeled data and large amount of unlabeled data
  - E.g. A teacher provides set of solved and unsolved problem for exam preparation

Regression

Regression

- Given the values of inputs X and the corresponding output Y belongs to the set of real values R, predict output accurately for new input.
- Formally:
  - Given:
    - A set of N observations $\{x_n\}_{n=1\ldots N}$ with their corresponding target values $\{y_n\}_{n=1\ldots N}$
  - Goal:
    - Predict the value of y_{n+1} for a give x_{n+1}
- Predictive technique where the target variable to be estimated is continuous

Regression (Cont.)

- Let D denote a dataset containing N observations,

$$D = \{(x_i, y_i) | i = 1,2, \dots, N\}$$

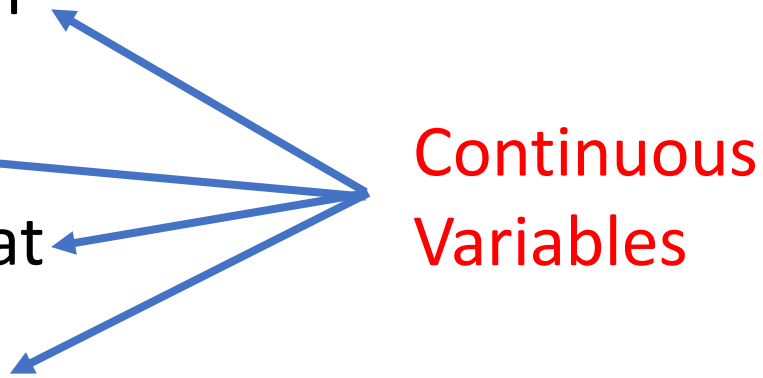
  - x_i corresponds to the values of attributes of the i-th observation.
    - These are called **explanatory variables** and can be discrete or continuous.
  - y_i corresponds to the target variable.
- Target: find a function that can minimize the error between the predicted and the actual values
  - The error can be measured as the sum of absolute or squared error

$$sum\ absolute\ error(SAE) = \sum_i |y_i - f(x_i)|$$

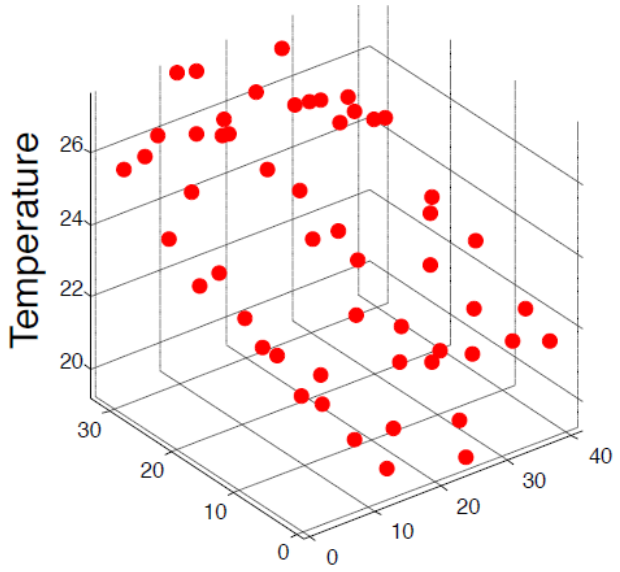
$$sum\ squared\ error\ (SSE) = \sum_i \left(y_i - f(x_i)\right)^2$$

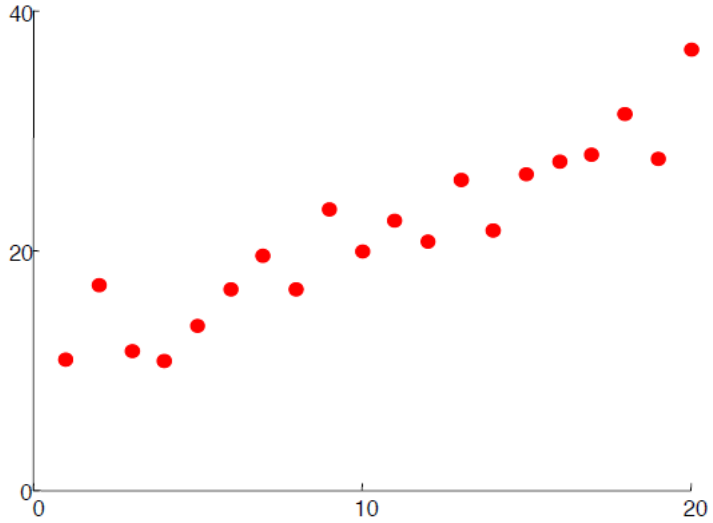
Regression – Examples

- Processes, memory → Power consumption
- Protein structure → Energy
- Heart-beat rate, age, speed, duration → Fat
- Oil supply, consumption, etc. → Oil price
- …



- Definition: Regression is the task of learning a target function f that maps each attribute set x into a continuous-valued output y.

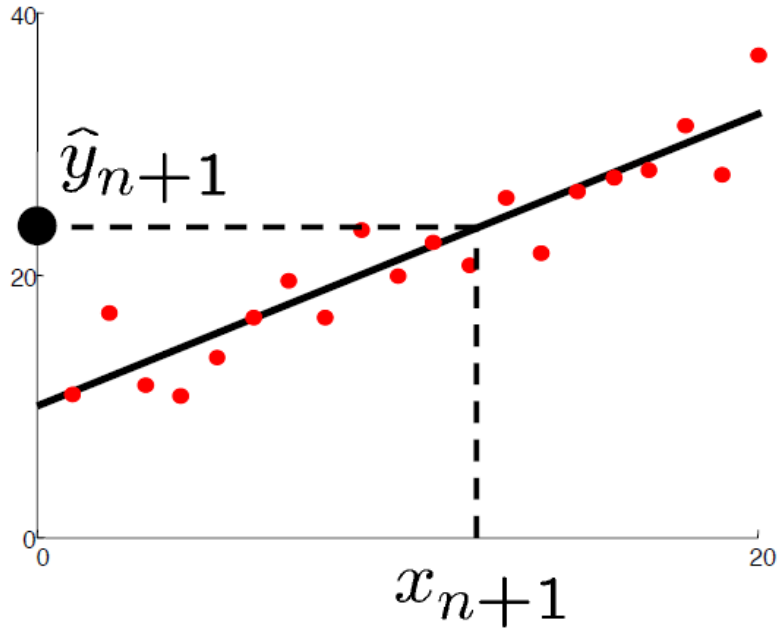


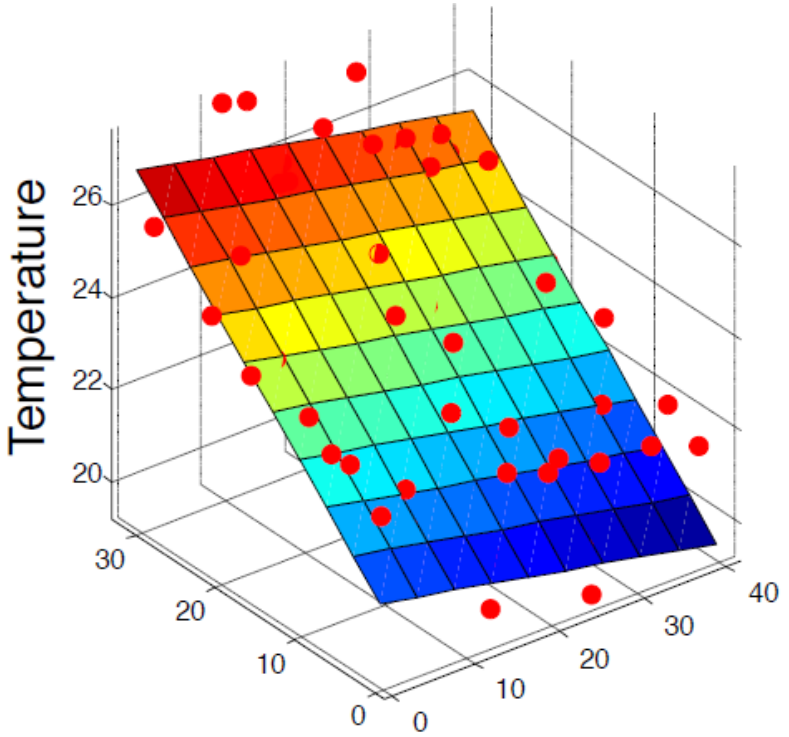


Regression – Examples

- Given $\{x_n, y_n\}_{n=1\ldots N}$
- Predict the value of y_{n+1} for a give x_{n+1}

Regression – Examples





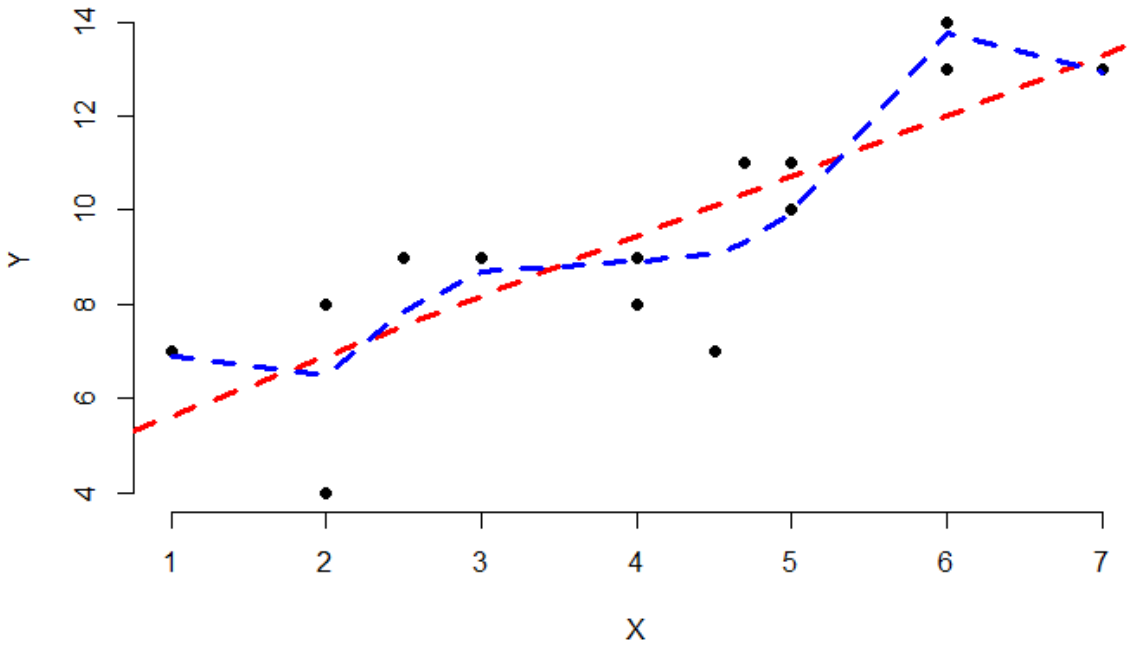
- Prediction $\hat{y}_i = w_0 + w_1 x_i$

- Prediction $\hat{y}_i = w_0 + w_1 x_{i,1} + w_2 x_{i,2}$

$$= (1 \quad x_{i,1} \quad x_{i,2}) \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}$$

- This is called linear regression as the function is linear in the parameters $w_0, w_1 \; and \; w_2$.

Simple Linear Regression



Given a set of points (x_i, y_i) such as the points in the scatterplot, find the best fitting line

$$f(x_i) = \omega_0 + \omega_1 x_i$$

such that:

$$SSE = \sum_i \big(y_i - f(x_i)\big)^2$$

$$= \sum_i (y_i - \omega_0 - \omega_1 x_i)^2$$

is minimized

Simple Linear Regression (Cont.)

- The above optimization problem can be solved by:
  1. Taking the partial derivatives of SSE with respect to ω_0 and ω_1
  2. Setting $\frac{\partial SSE}{\partial \omega_0}$ and $\frac{\partial SSE}{\partial \omega_1}$ to 0
  3. Solving the system of linear equations

     Since: $SSE = \sum_i (y_i - \omega_0 - \omega_1 x_i)^2$

     Then $\frac{\partial SSE}{\partial \omega_0} = -2 \sum_i (y_i - \omega_0 - \omega_1 x_i) = 0$

     And $\frac{\partial SSE}{\partial \omega_1} = -2 \sum_i x_i (y_i - \omega_0 - \omega_1 x_i) = 0$

Simple Linear Regression (Cont.)

- The equations can be summarized by the normal equation:

$$\begin{pmatrix} N & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{pmatrix} \begin{pmatrix} \omega_0 \\ \omega_1 \end{pmatrix} = \begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \end{pmatrix}$$

Example

- Consider the following dataset

x	1	2	2.5	2	4	4	4	4.5	4.7	5	5	6	6	7
y	7	4	9	8	9	8	9	7	11	11	10	13	14	13

$$\sum_i x_i = 56.7 \qquad \sum_i x_i^2 = 269.59$$

$$\sum_i y_i = 133 \qquad \sum_i x_i y_i = 589.7$$

Example (Cont.)

x	1	2	2.5	2	4	4	4	4.5	4.7	5	5	6	6	7
y	7	4	9	8	9	8	9	7	11	11	10	13	14	13

$$\sum_i x_i = 56.7 \qquad \sum_i x_i^2 = 269.59 \qquad \sum_i y_i = 133 \qquad \sum_i x_i y_i = 589.7$$

$$\begin{pmatrix} 14 & 56.7 \\ 56.7 & 269.59 \end{pmatrix} \begin{pmatrix} \omega_0 \\ \omega_1 \end{pmatrix} = \begin{pmatrix} 133 \\ 589.7 \end{pmatrix}$$

By solving the equations we get:

$\omega_0 = 4.3254$ and $\omega_1 = 1.278$

Hence:

$$f(x_i) = 1.278\, x_i + 4.3254$$

Simple Linear Regression (Cont.)

- A general solution for the normal equation can be found as follows:

$$\omega_o = \bar{y} - \omega_1 \bar{x}$$

and

$$\omega_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

where

$\bar{x}, \bar{y}$ are the mean (average) values for the vectors x, y

Evaluating the Goodness of a Fit

- To evaluate how well data points fit to a line, we use the R^2 which is defined as:

$$R^2 = \frac{\sum_i (f(x_i) - \bar{y})^2}{\sum_i (y_i - \bar{y})^2}$$

- R^2 takes values in the interval $[0,1]$ where the values close to 1 means that the data fits well to the regression line
- When adding more explanatory variables, the value of R^2 increases so it is adjusted using the formula

$$Adjusted\ R^2 = 1 - \left(\frac{N-1}{N-d}\right)(1 - R^2)$$

where N is the number of data points and $d + 1$ is the number of parameters of the regression model

Classification

Classification

- Given an input vector of the data $\boldsymbol{x}$, the goal is to assign $\boldsymbol{x}$ to one of the K discrete classes.
  - Classes are taken to be disjoint.
- Classification techniques:
  - Logistic Regression
  - Bayesian classification
  - Learning from the neighbors
  - Decision tree based methods
  - Rule-Based classification
  - Neural networks
  - Support vector machines

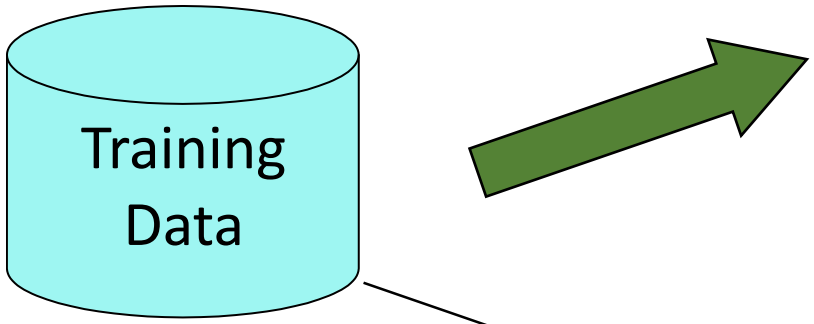
Classification vs. Prediction

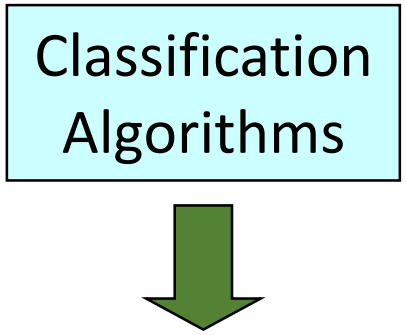
- Classification
  - Predicts categorical class labels (discrete or nominal)
  - Classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Numeric Prediction
  - Models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - Credit/loan approval
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is

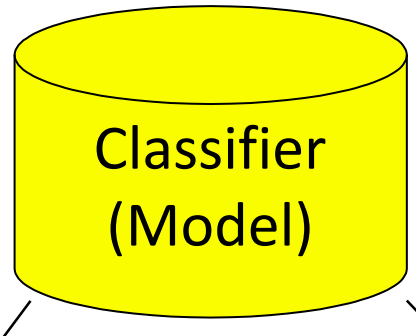
Classification – A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - 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 overfitting)
  - If the accuracy is acceptable, use the model to classify new data
- Note: If the test set is used to select models, it is called validation (test) set

Step (1): Model Construction



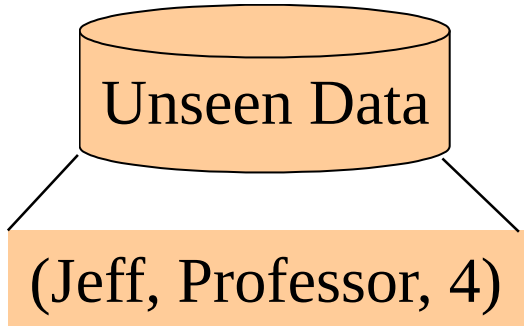


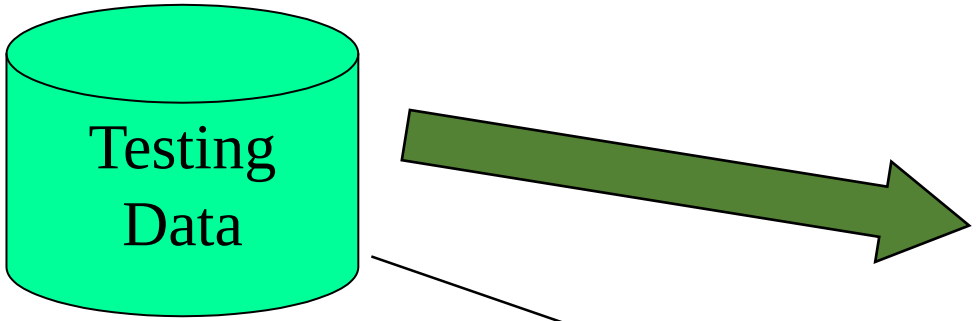


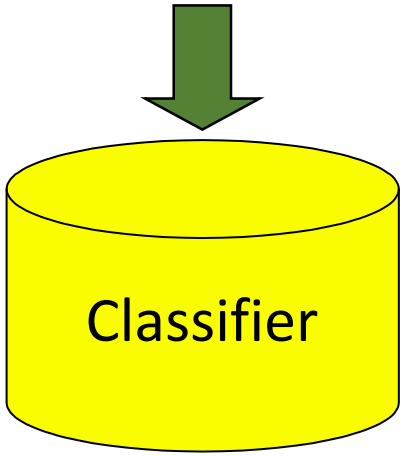
Name	Rank	Years	Tenured
Mike	Assistant Professor	3	No
Mary	Assistant Professor	7	Yes
Bill	Professor	2	Yes
Jim	Associate Professor	7	Yes
Anne	Associate Professor	6	No
Dave	Assistant Professor	3	No

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Step (2): Using the Model in Prediction







Name	Rank	Years	Tenured
Tom	Assistant Professor	2	No
Joane	Assistant Professor	7	No
Greg	Professor	5	Yes
Sally	Assistant Professor	7	Yes

Tenured?

Yes

Classification

Logistic Regression

Classification – Logistic Regression

- Useful when the target is binary
- Logistic regression is a type of probabilistic statistical classification model
- It measures the relationship between the dependent (target) binary variable and the independent explanatory variables
- We have a binary target variable Y,and we want to model the conditional probability $P(Y = 1 \mid X = x)$ as a function $p(x)$ of the explanatory variables x.
- Any unknown parameters (recall ω_0 and ω_1) are estimated by maximum likelihood.
- Can we use linear regression?

Logistic Regression (Cont.)

- Let $p(x)$ be a linear function
  - We are estimating a probability, which must be between 0 and 1
  - Linear functions are unbounded, so this approach doesn't work
- Better idea: Set the odds ratio to a linear function:

$$log(odds) = logit(p) = \ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x$$

Solving for p:

$$p(x) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}} = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}}$$

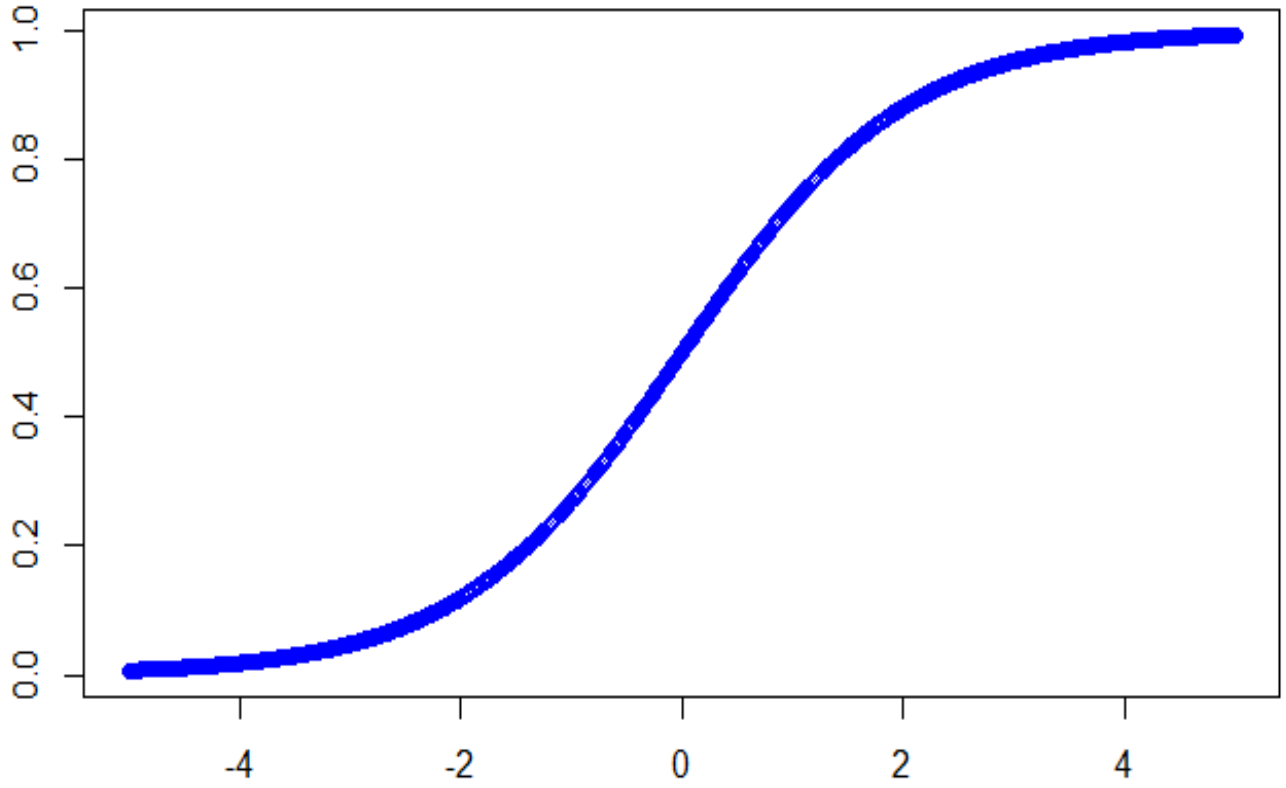
- This is called the logistic (logit) function and it takes values in the interval $[0,1]$

Logistic Curve

- A sigmoid function that assumes values in the range [0,1]

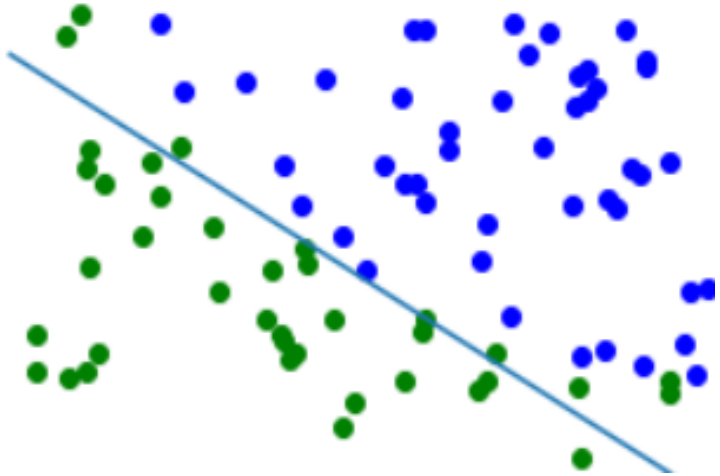
$$p(x) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}}$$

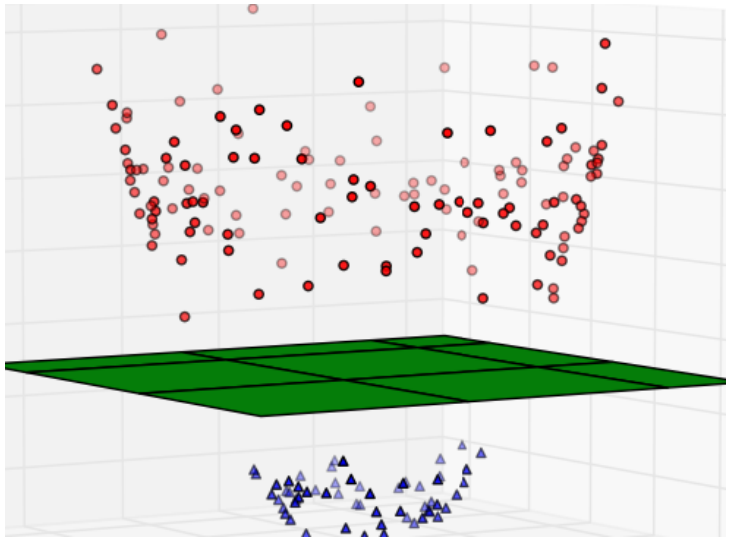
$$= \frac{1}{1 + e^{-(\beta_0+\beta_1 x)}}$$

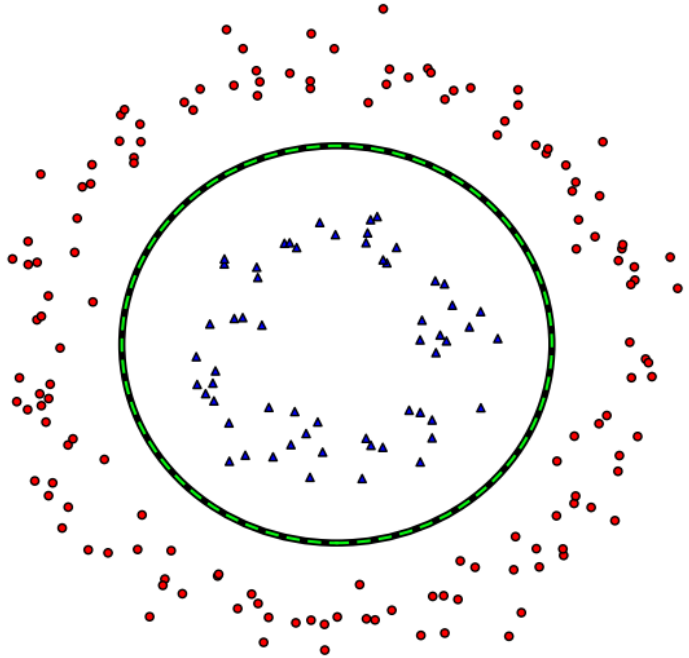


Logistic Regression

- To minimize misclassification rates, we predict:
  - $Y = 1$ when $p(x) \geq 0.5$ and $Y = 0$ when $p(x) < 0.5$
  - So $Y = 1$ when $\beta_0 + \beta_1 x$ is non-negative and 0 otherwise
- Logistic regression gives us a linear classifier where the decision boundary separating the two classes is the solution of $\beta_0 + \beta_1 x = 0$
  - A point if we have 1 explanatory variable
  - A line if we have 2
  - A plane if we have 3
  - A disaster if we have more than that







Decision Boundaries

Logistic Regression

- The parameters $\beta_0, \beta_1, \ldots$ are estimated using a technique called Maximum Likelihood Estimation (MLE)
  - Unlike the least squares methods used for Linear regression, finding a closed form for the coefficients using MLE is not possible. Instead, an iterative process (e.g., Newton's method) is used.
  - This process begins with a tentative solution, revises it slightly to see if it can be improved, and repeats this revision until improvement is very small, at which point the process is said to have converged.

Classification

Bayesian Classification

Bayesian Classification

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

Bayes' Theorem

- Total probability Theorem: $p(B) = \sum_{i=1}^{M} p(B|A_i)p(A_i)$
- Bayes' Theorem: $p(H|\boldsymbol{X}) = p(\boldsymbol{X}|H) \times p(H)/p(\boldsymbol{X})$
  - Let $\boldsymbol{X}$ be a data sample ("evidence"): class label is unknown
  - Let H be a hypothesis that $\boldsymbol{X}$ belongs to class C
  - Classification is to determine P(H| $\boldsymbol{X}$), (i.e., posteriori probability): the probability that the hypothesis holds given the observed data sample $\boldsymbol{X}$
  - $p(H)$ (prior probability): the initial probability
    - E.g., $\boldsymbol{X}$ will buy computer, regardless of age, income, …
  - $p(\boldsymbol{X})$: probability that sample data is observed
  - $p(\boldsymbol{X}|H)$ (likelihood): the probability of observing the sample $\boldsymbol{X}$, given that the hypothesis holds
    - E.g., Given that $\boldsymbol{X}$ will buy computer, the prob. that $\boldsymbol{X}$ is 31..40, medium income

Prediction Based on Bayes' Theorem

- Given training data $\boldsymbol{X}$, posteriori probability of a hypothesis H, $p(H|\boldsymbol{X})$, follows the Bayes' theorem

$$p(H|\boldsymbol{X}) = p(\boldsymbol{X}|H) \times p(H)/p(\boldsymbol{X})$$

- Informally, this can be viewed as

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

- Predicts $\boldsymbol{X}$ belongs to C_i iff. the probability $p(C_i|\boldsymbol{X})$ is the highest among all the $p(C_k|\boldsymbol{X})$ for all the k classes
- Practical difficulty: it requires initial knowledge of many probabilities, involving significant computational cost

Derive the Maximum Posteriori

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n attribute vector $\boldsymbol{X} = (x_1, x_2, \ldots, x_n)$
- Suppose there are k classes $C_1, C_2, \ldots, C_k$.
- Classification is to derive the maximum posteriori, i.e., the maximal $p(C_i|\boldsymbol{X})$
- This can be derived from Bayes' theorem

$$p(C_i|\boldsymbol{X}) = p(\boldsymbol{X}|C_i) \times p(C_i)/p(\boldsymbol{X})$$

- Since $p(\boldsymbol{X})$ is constant for all classes, only

$$p(C_i|\boldsymbol{X}) = p(\boldsymbol{X}|C_i) \times p(C_i)/p(\boldsymbol{X})$$

needs to be maximized

Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$p(\boldsymbol{X}|C_i) = \prod_{j=1}^{n} p(x_j|C_i) = p(x_1|C_i) \times p(x_2|C_i) \times \cdots \times p(x_n|C_i)$$

- This greatly reduces the computation cost: only counts the class distribution
- If A_m is categorical, $p(x_m|C_i)$ is the # of tuples in C_i having value x_m for A_m divided by $|C_{i,D}|$ (# of tuples of C_i in D)
- If A_m is continuous-valued, $p(x_m|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ: $g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$ and $p(x_m|C_i)$ is $p(\boldsymbol{X}|C_i) = g(x_m, \mu_{C_i}, \sigma_{C_i})$

Naïve Bayes Classifier: Example

$p(C_i)$: p(buys_computer = “yes”) = 9/14 = 0.643

p(buys_computer = “no”) = 5/14= 0.357

Compute $p(\boldsymbol{X}|C_i)$ for each class

p(age = “<=30” | buys_computer = “yes”) = 2/9 = 0.222

p(age = “<= 30” | buys_computer = “no”) = 0.6

p(income = “medium” | buys_computer = “yes”) = 0.444

p(income = “medium” | buys_computer = “no”) = 0.4

p(student = “yes” | buys_computer = “yes) = 0.667

p(student = “yes” | buys_computer = “no”) = 0.2

p(credit_rating = “fair” | buys_computer = “yes”) = 0.667

p(credit_rating = “fair” | buys_computer = “no”) = 0.4

- $\boldsymbol{X}$ = (age <= 30 , income = medium, student = yes, credit_rating = fair)
- $p(\boldsymbol{X}|C_i)$: $p(\boldsymbol{X}|$buys_computer = “yes”) = 0.222 x 0.444 x 0.667 x 0.667 = 0.044

  $p(\boldsymbol{X}|$ buys_computer = “no”) = 0.6 x 0.4 x 0.2 x 0.4 = 0.019
- $p(\boldsymbol{X}|C_i) \times p(C_i)$: $p(\boldsymbol{X}|$ buys_computer = “yes”) × p(buys_computer = “yes”) = 0.028

  $p(\boldsymbol{X}|$ buys_computer = “no”) × p(buys_computer = “no”) = 0.007
- Therefore, $\boldsymbol{X}$ belongs to class (“buys_computer = yes”)

Avoiding the Zero-Probability Problem

- Naïve Bayesian prediction requires each conditional probability be non-zero. Otherwise, the predicted probability will be zero since:
  $p(\boldsymbol{X}|C_i) = \prod_{j=1}^{n} p(x_j|C_i)$
- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10)
- Use Laplacian correction (or Laplacian estimator)
  - Add 1 to each case

    p(income = low) = 1/1003, p(income = medium) = 991/1003, p(income = high) = 11/1003
- The "corrected" prob. estimates are close to their "uncorrected" counterparts

Naïve Bayes Classifier: Final Remarks

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., Hospitals: patients: Profile: age, family history, etc.

      Symptoms: fever, cough etc.,

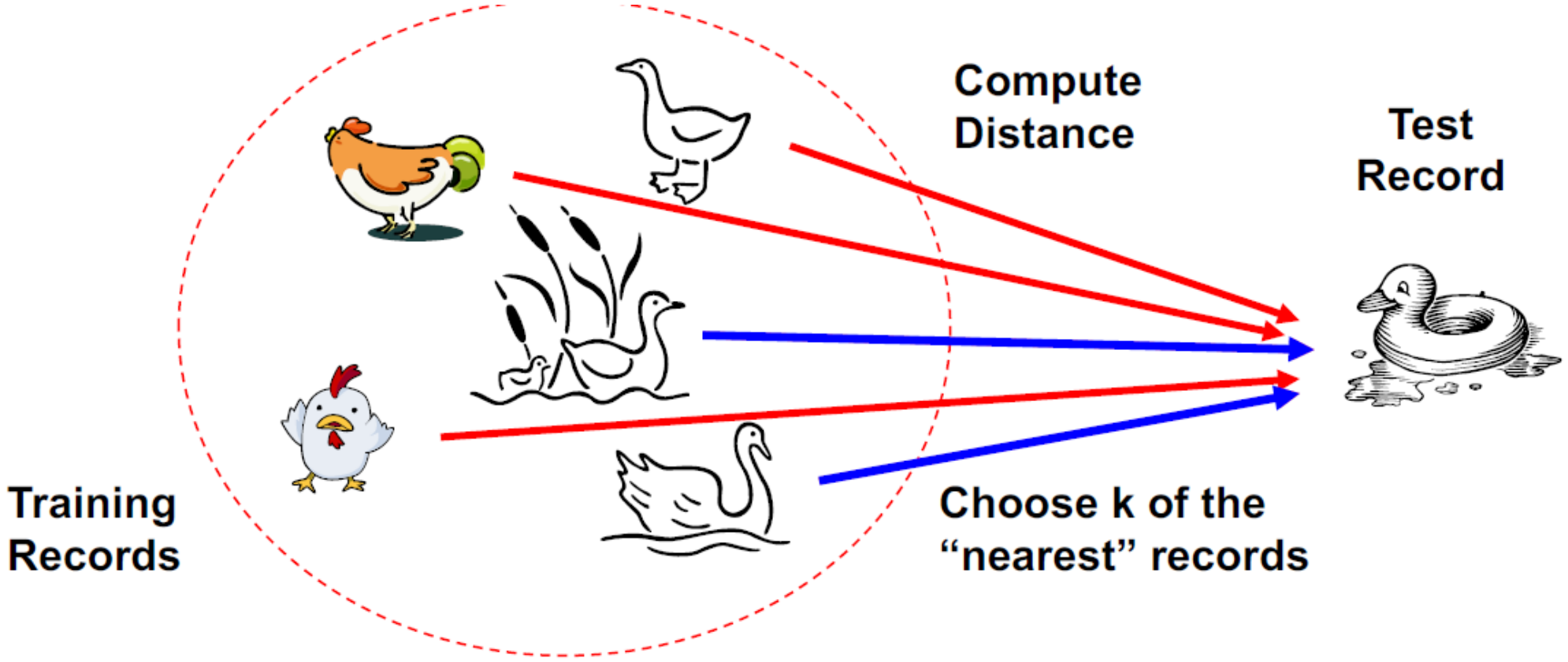
      Disease: lung cancer, diabetes, etc.
  - Dependencies among these cannot be modeled by Naïve Bayes Classifier

Classification

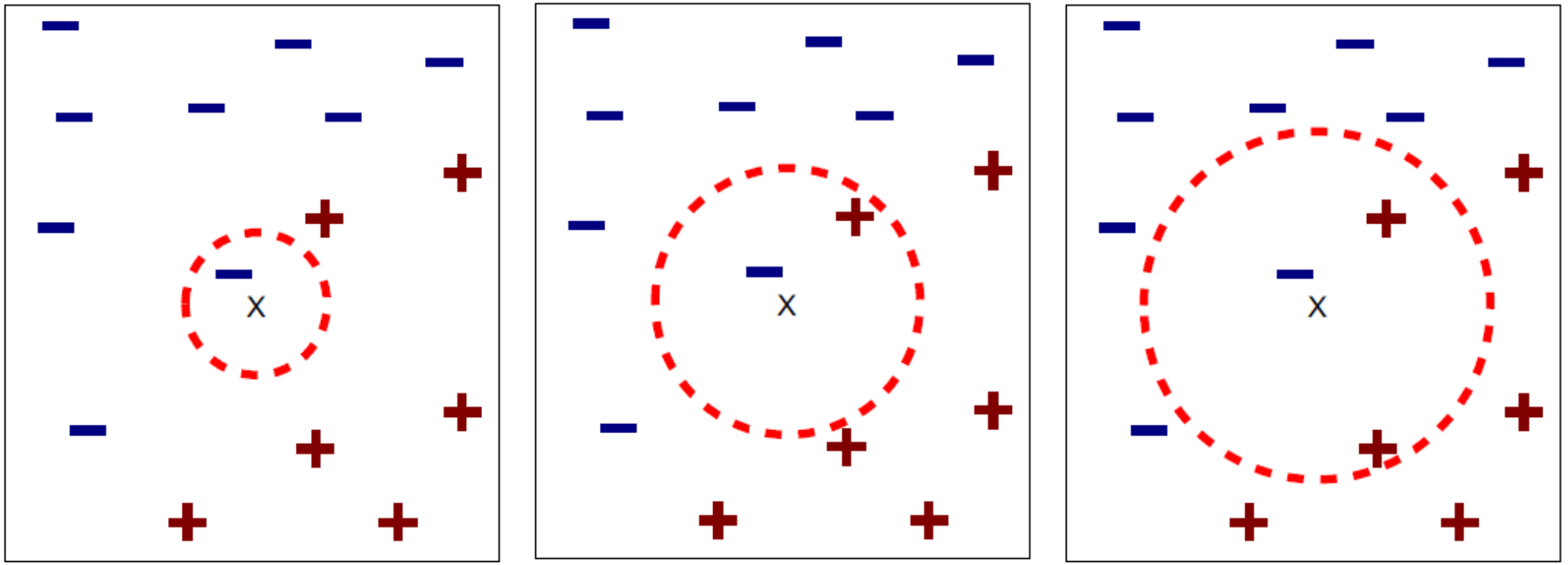
Nearest Neighbors Classifiers

Nearest Neighbors Classifiers

Basic idea: if it walks like a duck, quacks like a duck, then it is probably a duck



Nearest Neighbors Classifiers (Cont.)



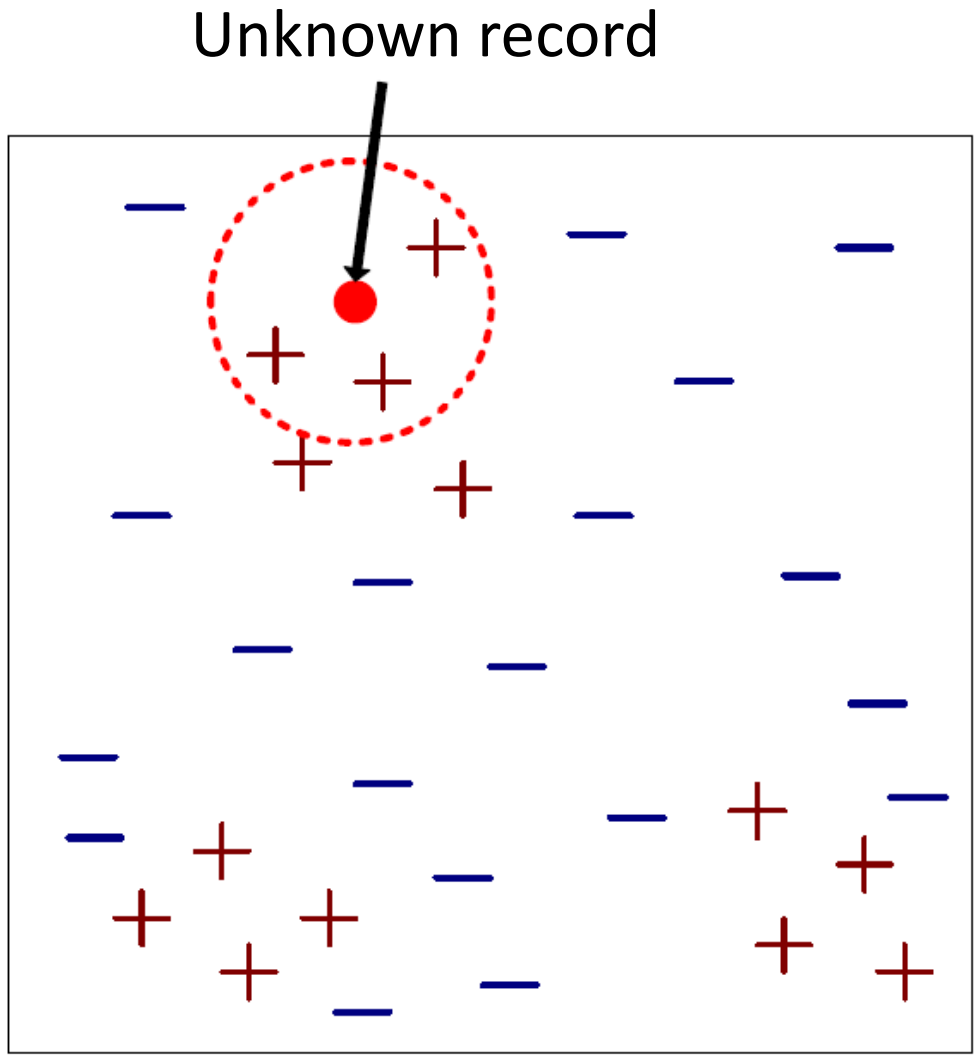
1- nearest neighbor

2- nearest neighbors

3- nearest neighbors

Nearest Neighbors Classifiers (Cont.)

- Three requirements:
  - Set of records (training set)
  - Distance metric
  - The number of neighbors to be considered k
- Classifying unknown record x:
  - Compute the distance from x to the other training records
  - Identify the k-Nearest Neighbors (kNN(x))
  - Use class labels of the kNN records to determine the class of x .. How?



K Nearest Neighbors Classification

- To determine the class of unknown record x from the classes of its neighbors:
  - Use the majority vote – is this enough?
  - Weight the vote according to the distance
  - Examples of weight factors, $w = d^{-2}, w = e^{-d^2}$, etc.
- How to choose k:
  - k too small, sensitive to noise
  - k too large, neighborhood may include points from other classes

K Nearest Neighbors Classification (Cont.)

- Scaling issues
  - Attributes may have to be scaled to prevent distance
  - measures from being dominated by one of the attributes
- Example:
  - Height of a person may vary from 1.5m to 1.8m
  - Weight of a person may vary from 90lb to 300lb
  - Income of a person may vary from €10K to €1M
- Solution: Normalize the vectors to unit length

K Nearest Neighbors Classification (Cont.)

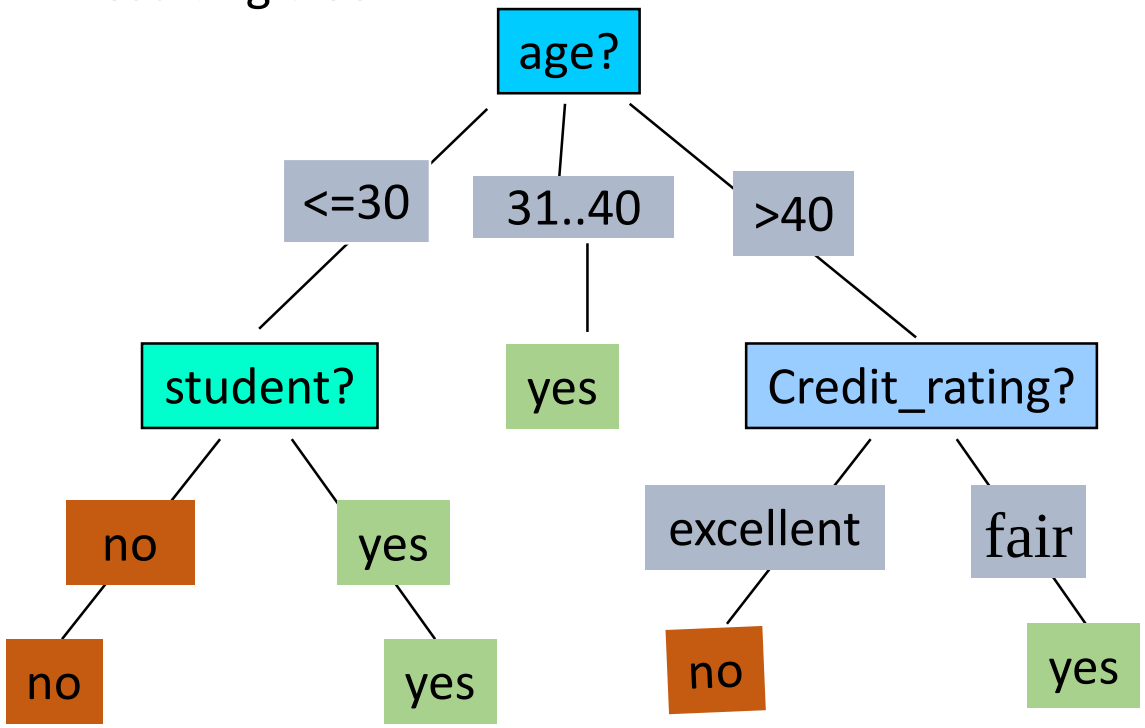
- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Robust to noisy data by averaging k-nearest neighbors unlike eager learners such as decision tree induction and rule-based systems
  - Classifying unknown records are relatively expensive

Classification

Decision Tree Induction

Decision Tree Induction

- Training dataset: Buys_computer
- The dataset follows an example of Quinlan's ID3 (Playing Tennis)
- Resulting tree:



age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Algorithm for Decision Tree Induction

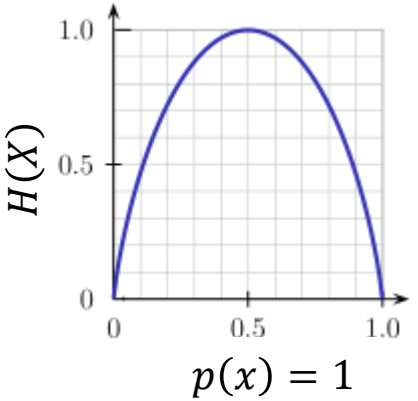
- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based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

Review of Entropy

- Entropy (from information theory):
  - A measure of uncertainty associated with a random variable
  - Computing the entropy: for a discrete random variable Y that takes m distinct values $\{y_1, y_2, \ldots, y_m\}$:

$$H(Y) = -\sum_{i=1}^{m} p_i \log(p_i), \quad where, p_i = p(Y = y_i)$$

  - Interpretation:
    - Higher entropy implies higher uncertainty and vice versa
  - Conditional entropy: $H(Y|X) = \sum_x p(x) H(Y|X = x)$



m = 2

Attribute Selection Measure: Information Gain

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in a dataset D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

$$info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed (after using A to split D into v partitions) to classify D:

$$info_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times info(D_j)$$

- Information gained by branching on attribute A

$$gain(A) = info(D) - info_A(D)$$

Information Gain – Example

- Class C_1: buys_computer = “yes”
- Class C_2: buys_computer = “no”

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

$$info(D) = I(9,5) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.94$$

$$info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2)$$

$$= \frac{5}{14}(0.971) + 0 + \frac{5}{14}(0.971) = 0.694$$

$\frac{5}{14}I(2,3)$ means that “$age \leq 30$” has 5 out of 14 samples with 2 from class C_1 and 3 from class C_2

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Information Gain – Example

$$gain(age) = info(D) - info(D_{age}) = 0.94 - 0.694 = 0.246$$

- Similarly:

$$gain(income) = 0.029$$
$$gain(student) = 0.151$$
$$gain(credit\_rating) = 0.048$$

Information-Gain for Continuous-Valued Attributes

- Let attribute A be a continuous-valued attribute
  - Must determine the best split point for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible split point
    - $(a_i + a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
  - The point with the minimum expected information requirement for A is selected as the split-point for A
- Split:
  - $D1$ is the set of tuples in D satisfying $A \leq$ split-point, and $D2$ is the set of tuples in D satisfying $A >$ split-point

Gain Ratio for Attribute Selection

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

$$GainRatio(A) = Gain(A)/SplitInfo_A(D)$$

- Ex.

$$SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 1.557$$

$$gain\_ratio(income) = 0.029/1.557 = 0.019$$

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

Gini Index

- If a dataset D contains examples from n classes, gini index, $gini(D)$ is defined as

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

where p_j is the relative frequency of class j in D

- If a dataset D is split on A into two subsets $D1$ and $D2$, the gini index $gini(D)$ is defined as

$$gini_A(D) = \frac{|D1|}{|D|} gini(D1) + \frac{|D2|}{|D|} gini(D2)$$

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute A_i that provides the smallest $gini_{A_i}(D)$ (or the largest reduction in impurity) is chosen to split the node
  - need to enumerate all the possible splitting points for each attribute

Computing Gini Index

- Ex. D has 9 tuples in buys_computer = “yes” and 5 in “no”

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Suppose the attribute income partitions D into 10 in $D1$: {low, medium} and 4 in $D2$

$$gini_{income \in \{low,medium\}}(D) = \left(\frac{10}{14}\right) gini(D1) + \left(\frac{4}{14}\right) gini(D2)$$

$$gini_{income \in \{low,medium\}}(D) = \left(\frac{10}{14}\right)\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \left(\frac{4}{14}\right)\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right) = 0.443$$

$$gini_{income \in \{medium,high\}} = 0.450$$

Thus, split on the {low,medium} (and {high}) since it has the lowest Gini index

- All attributes are assumed continuous-valued
- May need other tools, e.g., clustering, to get the possible split values
- Can be modified for categorical attributes

Comparing Attribute Selection Measures

- The three measures, in general, return good results but
  - Information gain:
    - Biased towards multivalued attributes
  - Gain ratio:
    - Tends to prefer unbalanced splits in which one partition is much smaller than the others
  - Gini index:
    - Biased to multivalued attributes
    - Has difficulty when # of classes is large
    - Tends to favor tests that result in equal-sized partitions and purity in both partitions

Other Attribute Selection Measures

- CHAID: a popular decision tree algorithm, measure based on χ^2 test for independence
- C-SEP: performs better than information gain and gini index in certain cases
- G-statistic: has a close approximation to χ2 distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear combination of attributes.
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others

Overfitting and Tree Pruning

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

Enhancements to Basic Decision Tree Induction

- Allow for continuous-valued attributes
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle missing attribute values
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- Attribute construction
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication

Classification in Large Databases

- Classification—a classical problem extensively studied by statisticians and machine learning researchers
- Scalability: Classifying data sets with millions of examples and hundreds of attributes with reasonable speed
- Why is decision tree induction popular?
  - Relatively faster learning speed (than other classification methods)
  - Convertible to simple and easy to understand classification rules
  - Can use SQL queries for accessing databases
  - Comparable classification accuracy with other methods

Classification

Rule-Based Classification

Using IF-THEN Rules for Classification

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

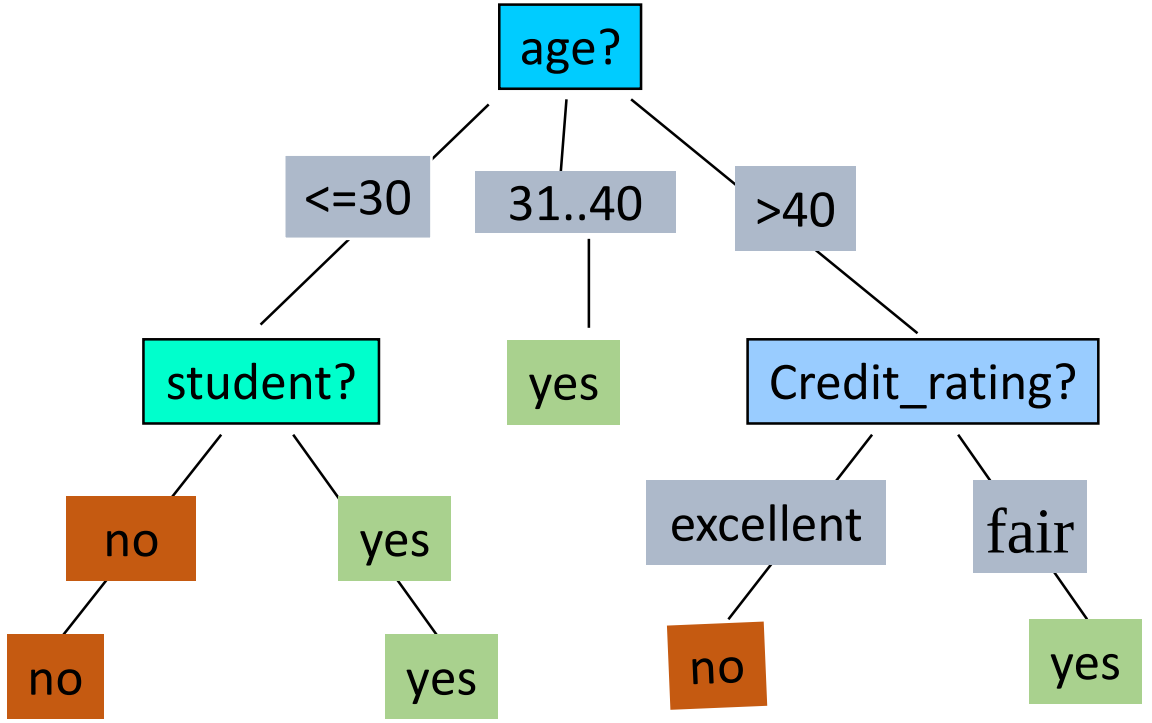
  Rule: IF age = youth AND student = yes THEN buys_computer = yes
  - Rule antecedent/precondition vs. rule consequent
- Assessment of a rule: coverage and accuracy
  - ncovers = # of tuples covered by R
  - ncorrect = # of tuples correctly classified by R

  $coverage(R) = ncovers\ /|D|\ \ /*\ D\text{: } training\ dataset\ */$

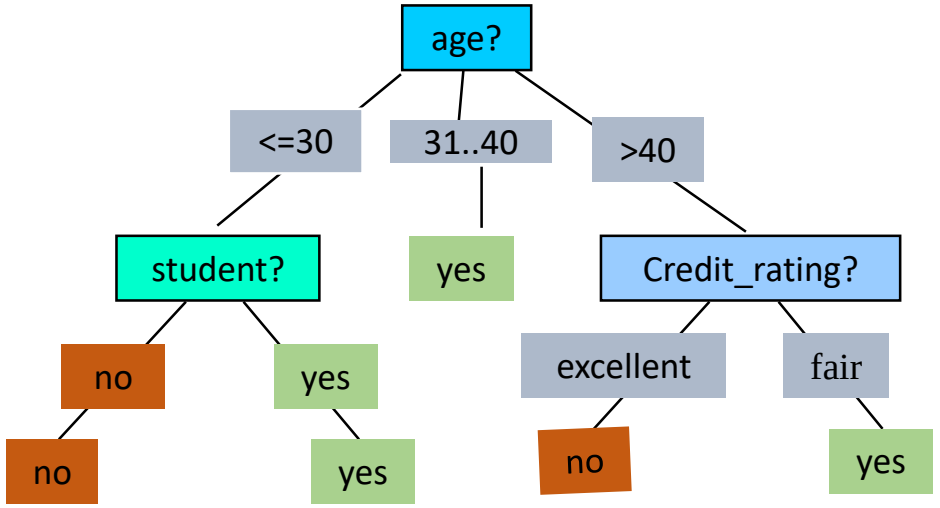
  $accuracy(R) = ncorrect\ /\ ncovers$
- If more than one rule are triggered, need conflict resolution
  - Size ordering: assign the highest priority to the triggering rules that has the “toughest” requirement (i.e., with the most attribute tests)
  - Class-based ordering: decreasing order of prevalence or misclassification cost per class
  - Rule-based ordering (decision list): rules are organized into one long priority list, according to some measure of rule quality or by experts

Rule Extraction from a Decision Tree

- Rules are easier to understand than large trees
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction: the leaf holds the class prediction
- Rules are mutually exclusive and exhaustive



Rule Extraction from a Decision Tree (Cont.)



- Example:

Rule extraction from our buys_computer decision-tree

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

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

IF age = mid-age THEN buys_computer = yes

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

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

Rule Induction: Sequential Covering Method

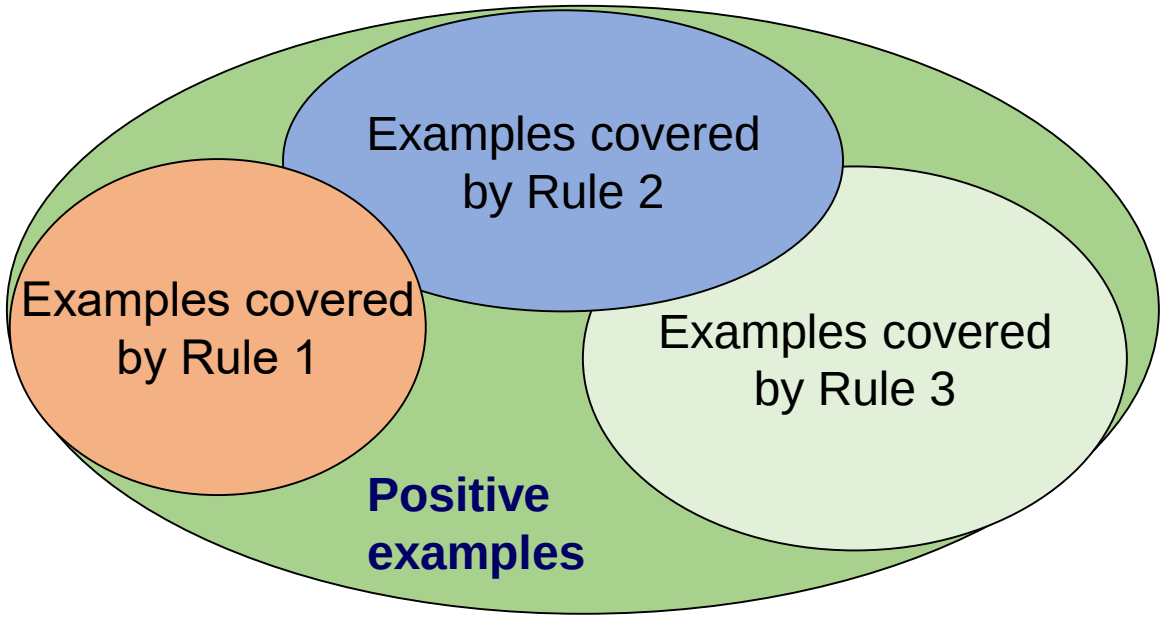
- Sequential covering algorithm: Extracts rules directly from training data
- Typical sequential covering algorithms: FOIL, AQ, CN2, RIPPER
- Rules are learned sequentially, each for a given class C_i will cover many tuples of C_i but none (or a few) of the tuples of other classes
- Steps:
  - Rules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

Sequential Covering Algorithm

WHILE (enough target tuples left)

generate a rule

remove positive tuples that satisfy the rule

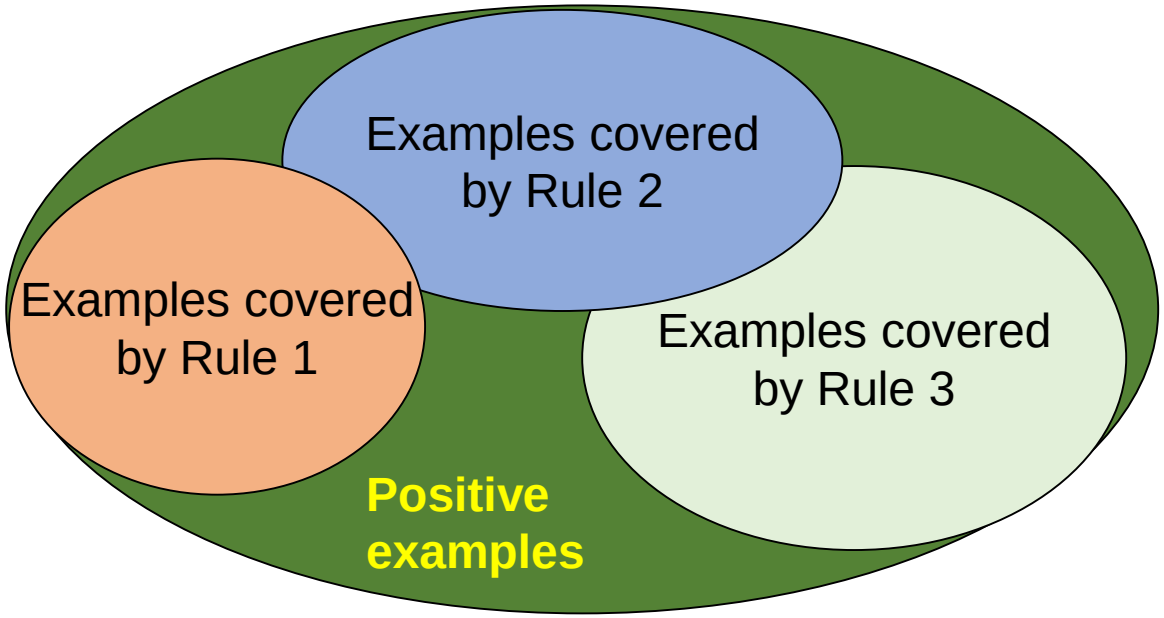


Sequential Covering Algorithm

WHILE (enough target tuples left)

generate a rule

remove positive tuples that satisfy the rule

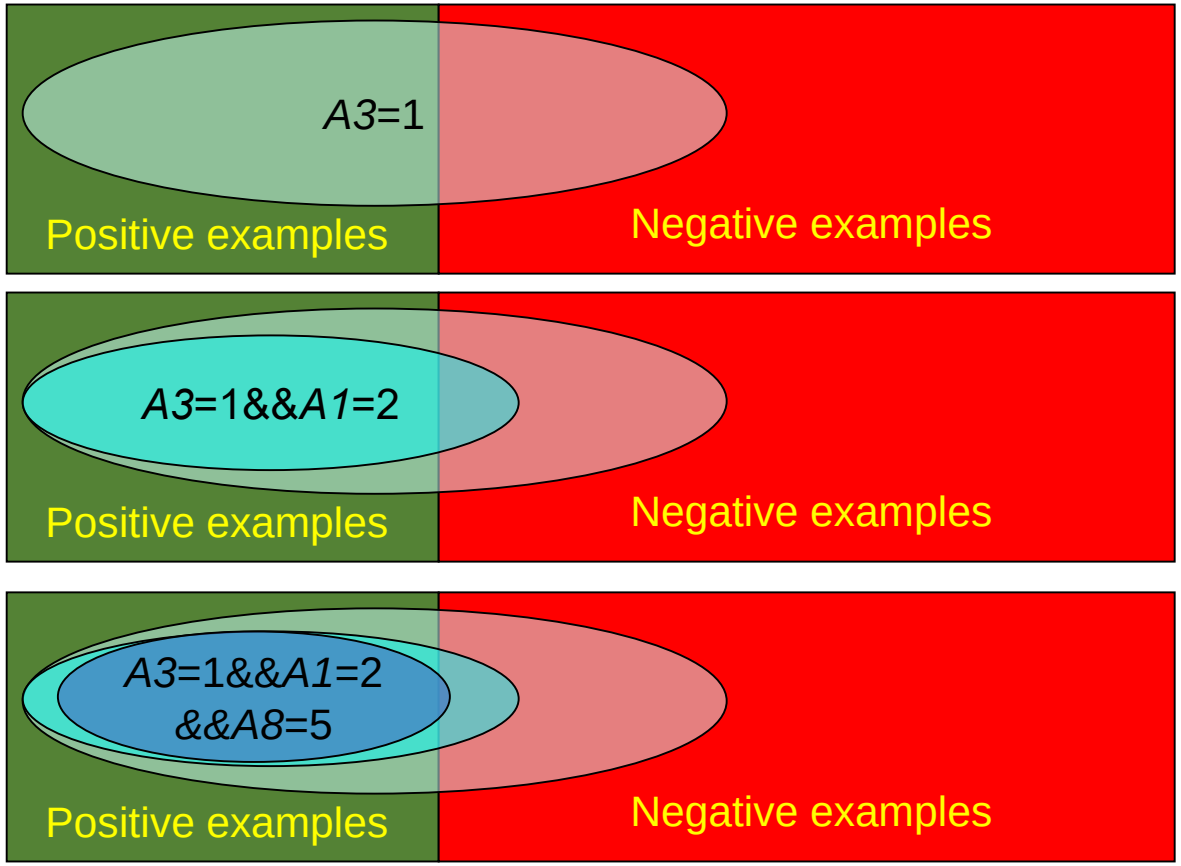


WHILE(TRUE)

find the best predicate p

if foil-gain(p) > threshold then add p to current rule

else break



Learning a Rule

- Start with the most general rule possible: condition = empty
- Adding new attributes by adopting a greedy depth-first strategy
  - Picks the one that most improves the rule quality
- Rule-Quality measures: consider both coverage and accuracy
  - Foil-gain (in FOIL & RIPPER): assesses info_gain by extending condition

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

    - favors rules that have high accuracy and cover many positive tuples
- Rule pruning based on an independent set of test tuples

$$FOIL\_PRUNE(R) = \frac{pos - neg}{pos + neg}$$

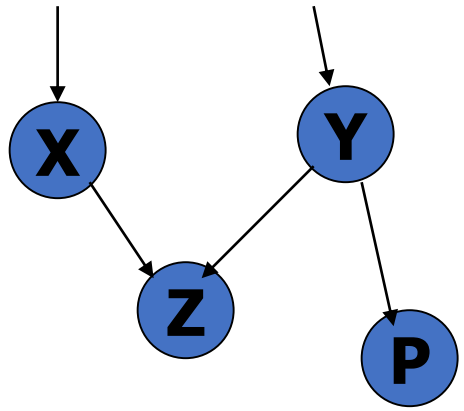
pos/neg and pos'/neg' are # of positive/negative tuples covered by R and R'.
If FOIL_Prune is higher for the pruned version of R, prune R

Classification

Neural Networks

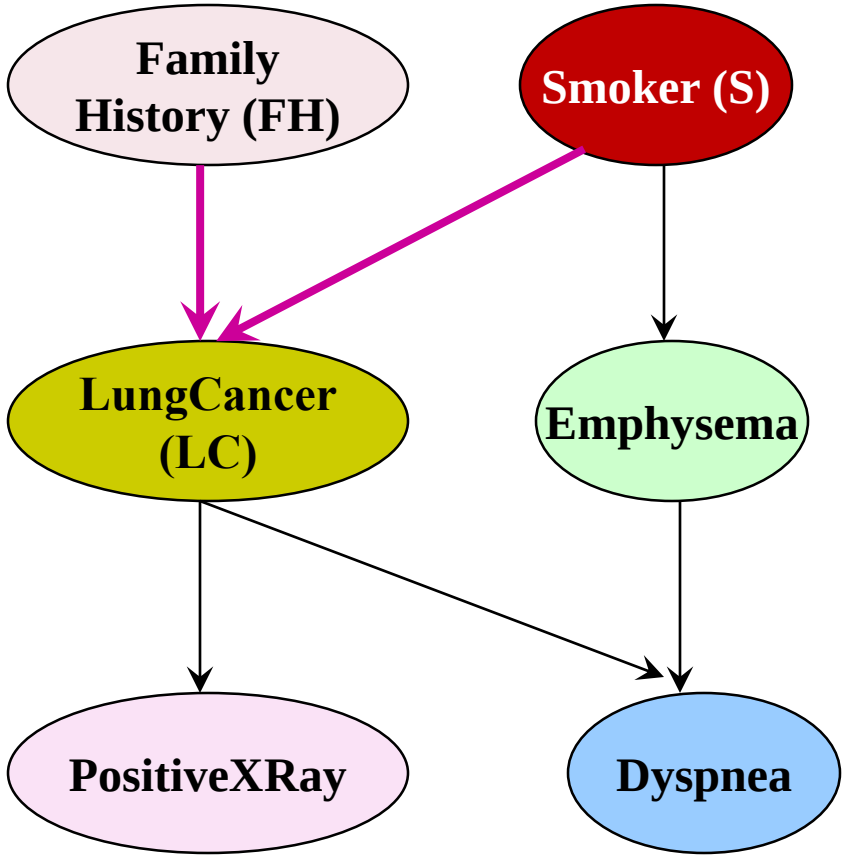
Bayesian Belief Networks

- Bayesian belief networks (also known as Bayesian networks, probabilistic networks): allow class conditional independencies between subsets of variables
- A (directed acyclic) graphical model of causal relationships
  - Represents dependency among the variables
  - Gives a specification of joint probability distribution



- Nodes: random variables
- Links: dependency
- X and Y are the parents of Z, and Y is the parent of P
- No dependency between Z and P
- Has no loops/cycles

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

Conditional Probability Table (CPT)

- shows the conditional probability for each possible combination of its parents
- Derivation of the probability of a particular combination of values of $\boldsymbol{X}$, from CPT:

$$p(x_1, x_2, \ldots x_n) = \prod_{j=1}^{n} p(x_j | Parents(Y_i))$$

Training Bayesian Networks

- Scenario 1: Given both the network structure and all variables observable: *compute only the CPT entries*
- Scenario 2: Network structure known, some variables hidden: *gradient descent* (greedy hill-climbing) method, i.e., search for a solution along the steepest descent of a criterion function
  - Weights are initialized to random probability values
  - At each iteration, it moves towards what appears to be the best solution at the moment, without backtracking
  - Weights are updated at each iteration & converge to local optimum
- Scenario 3: Network structure unknown, all variables observable: search through the model space to *reconstruct network topology*
- Scenario 4: Unknown structure, all hidden variables: No good algorithms known for this purpose

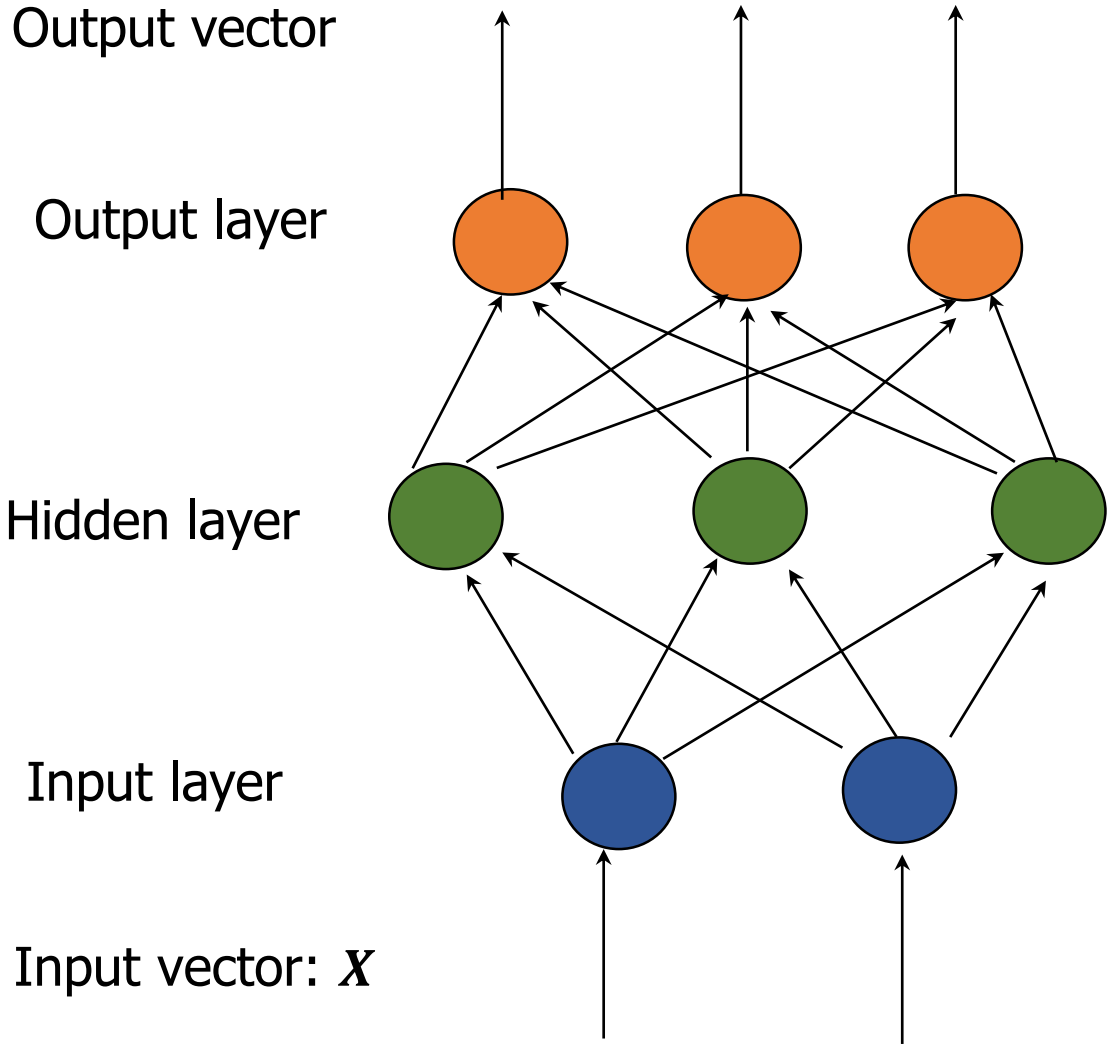
Classification by Backpropagation

- Backpropagation: A **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
- 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

Neural Network as a Classifier

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

A Multi-Layer Feed-Forward Neural Network



$$w_j^{(k+1)} = w_j^{(k)} + \lambda \left(y_i - \hat{y}_i^{(k)} \right) x_{ij}$$

How a Multi-Layer Neural Network Works?

- The **inputs** to the network correspond to the attributes measured for each training tuple
- 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**: 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

- Decide the **network topology:** Specify # of units in the *input layer*, # of *hidden layers* (if > 1), # of units in *each hidden layer*, and # of units in the *output layer*
- Normalize the input values for each attribute measured in the training tuples to [0.0—1.0]
- One **input** unit per domain value, each initialized to 0
- **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*

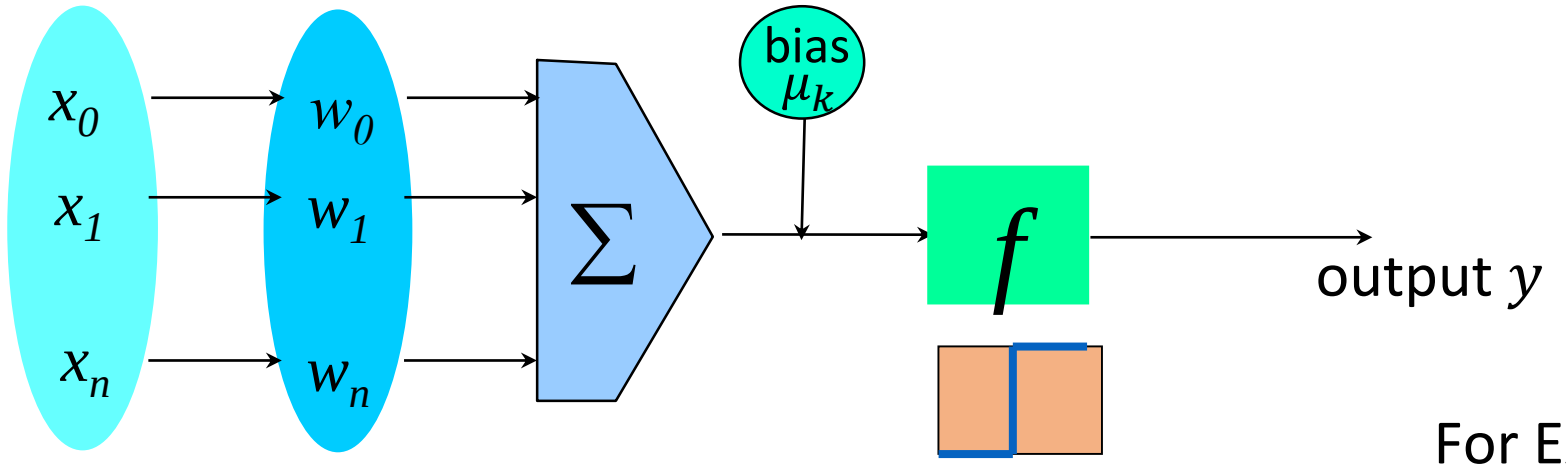
Backpropagation

- Iteratively process a set of training tuples & compare the network's prediction with the actual known target value
- For each training tuple, the weights are modified to **minimize 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
  - Initialize weights to small random numbers, associated with biases
  - Propagate the inputs forward (by applying activation function)
  - Backpropagate the error (by updating weights and biases)
  - Terminating condition (when error is very small, etc.)

For Example

$$y = \text{sign}\left(\sum_{i=0}^{n} w_i x_i - \mu_k\right)$$

Neuron: A Hidden/Output Layer Unit



Input vector x | weight vector w | weighted sum | Activation function

For Example:

$y = sin(\sum_{i=0}^{n} w_i x_i - \mu_k)$

or the sigmoid function

- An *n*-dimensional input vector **x** is mapped into variable y by means of the scalar product and a nonlinear function mapping
- The inputs to unit are outputs from the previous layer. They are multiplied by their corresponding weights to form a weighted sum, which is added to the bias associated with unit. Then a nonlinear activation function is applied to it

Efficiency and Interpretability

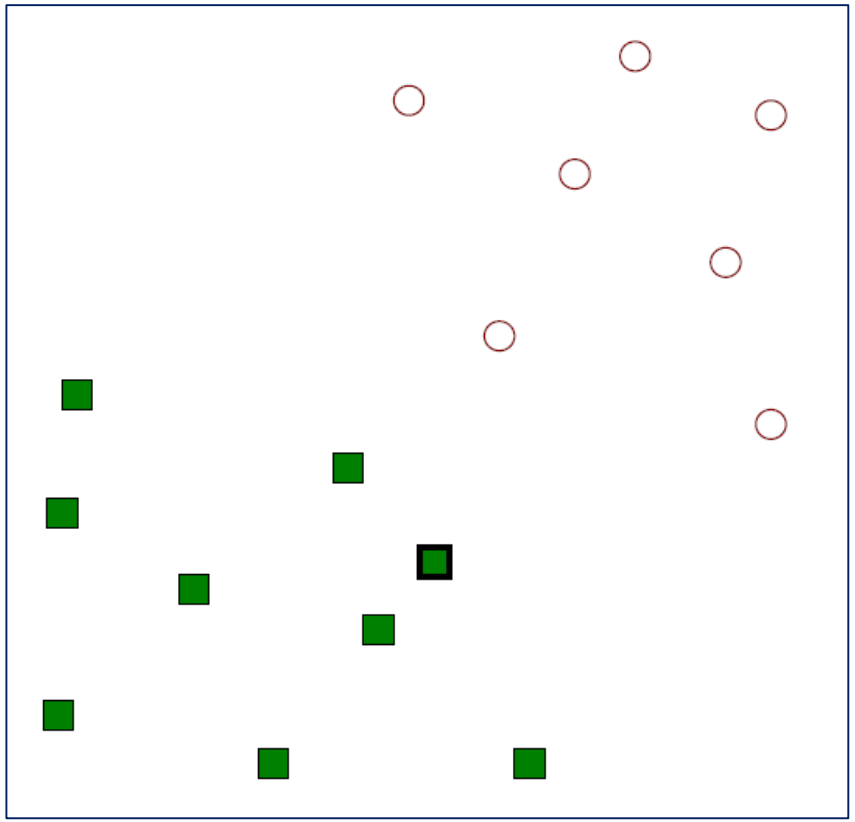
- **Efficiency** of backpropagation: Each epoch (one iteration through the training set) takes $O(|D| \times w)$, with $|D|$ tuples and w weights, but # of epochs can be exponential to n, the number of inputs, in worst case
- For easier comprehension: **Rule extraction** by network pruning
  - Simplify 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

Classification

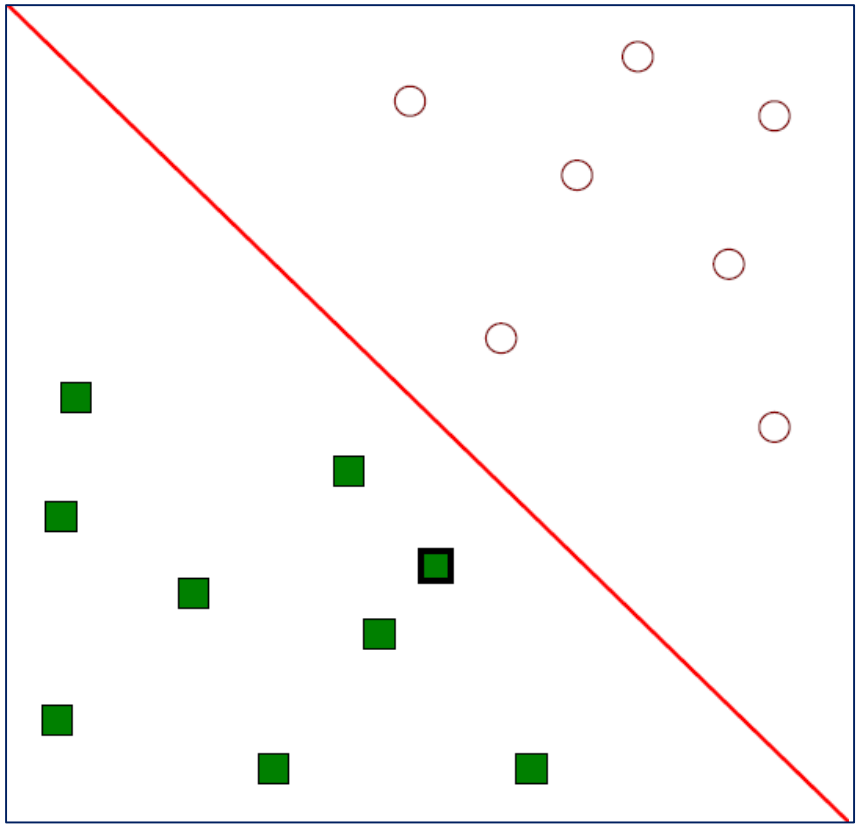
Support Vector Machines

Support Vector Machines (SVM)

Find linear hyperplane (decision boundary) that will separate the data

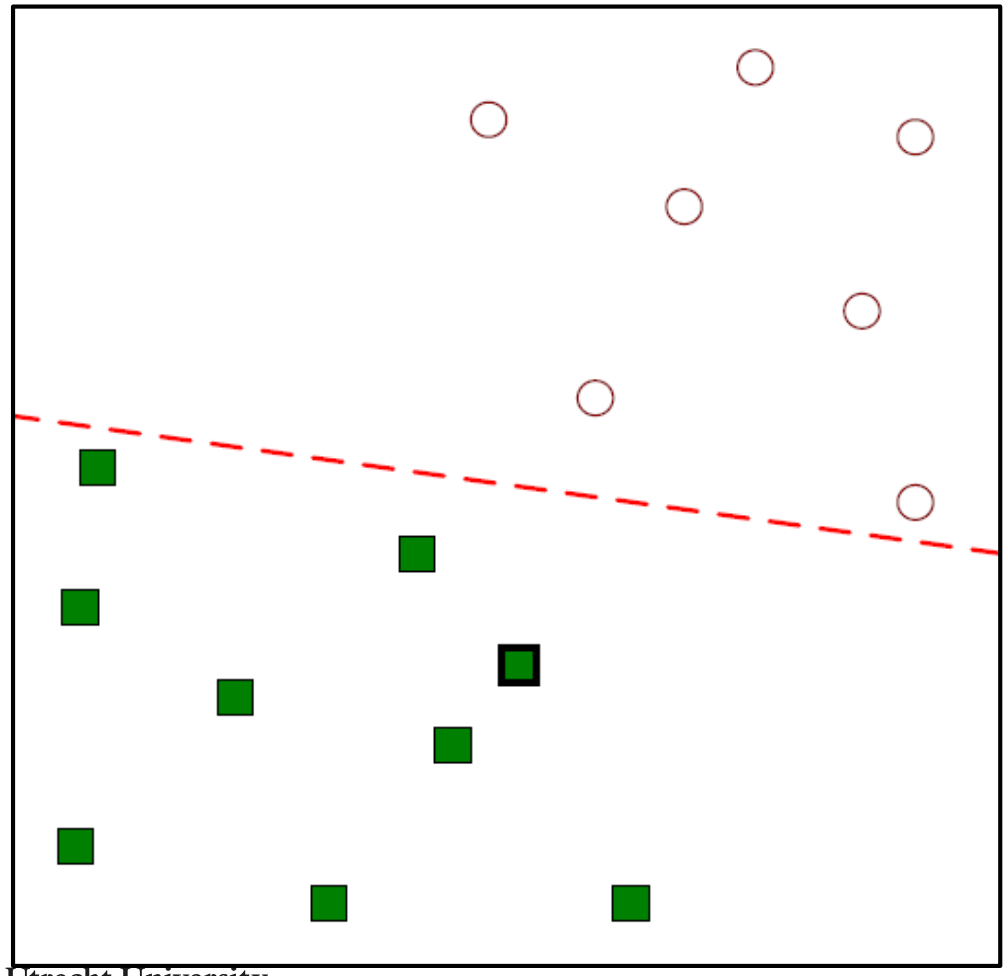


One possible separators

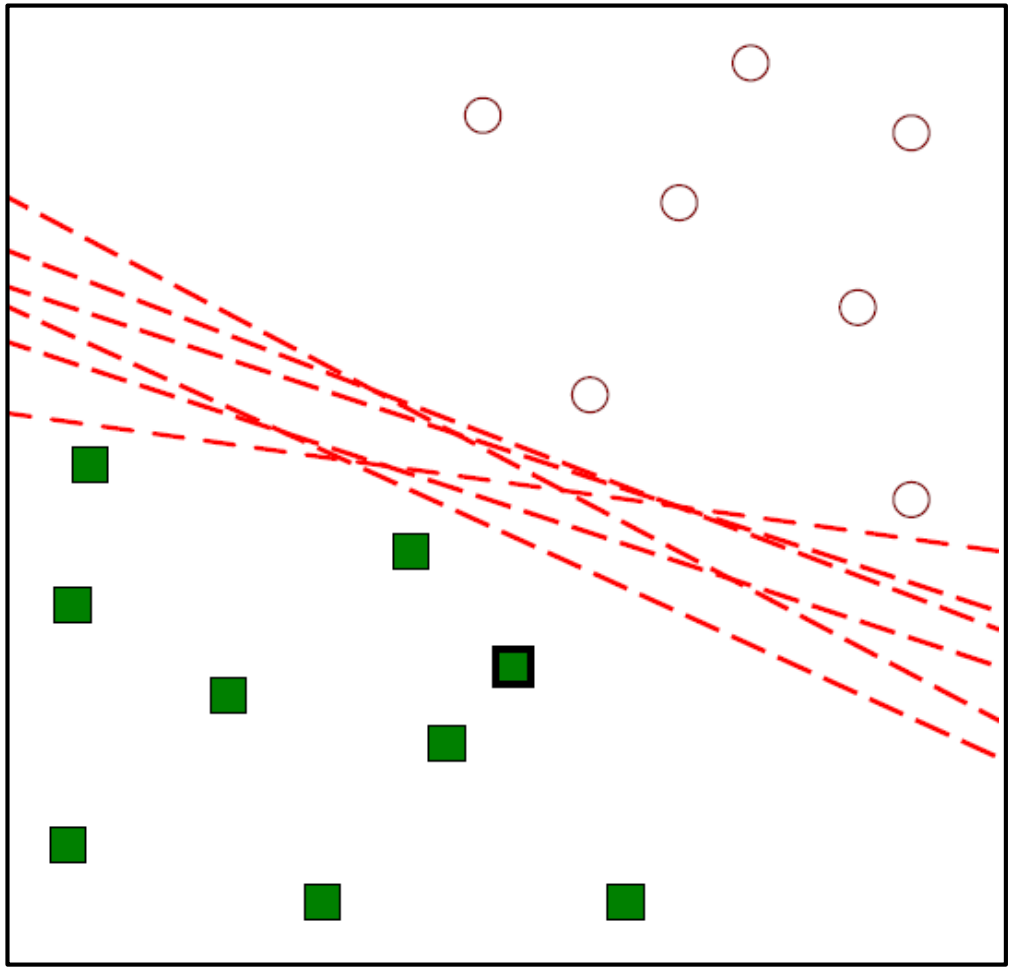


Support Vector Machines (SVM) (Cont.)

Another possible separator



Other possible separators



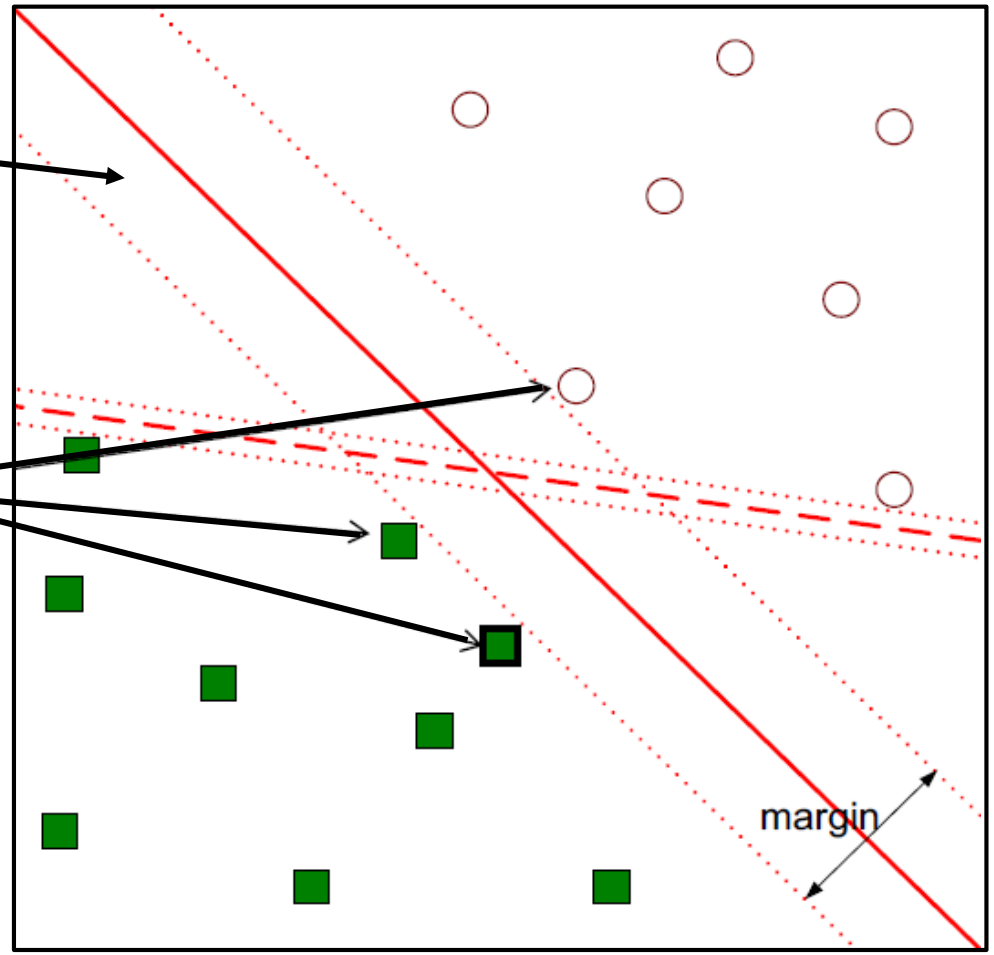
Support Vector Machines (SVM) (Cont.)

Find the hyperplane that maximizes the margin

Better separator

Support Vectors

are the points that the margin is pushed up against



SVM – Linearly Separable

- A separating hyperplane can be written as

$$W \bullet X + b = 0$$

- where $W = \{w_1, w_2, \ldots, w_n\}$ 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:

$$H_1: w_0 + w_1 x_1 + w_2 x_2 \geq 1 \quad \text{for } y_i = +1, \text{ and}$$
$$H_2: w_0 + w_1 x_1 + w_2 x_2 \leq -1 \text{ for } y_i = -1$$

- 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 optimization problem

SVM are Effective on High Dimensional Data

- The complexity of trained classifier is characterized by the # 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
- 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 generalization, even when the dimensionality of the data is high

SVM vs Neural Networks

- SVM
  - Deterministic algorithm
  - Nice generalization properties
  - Hard to learn – learned in batch mode using quadratic programming techniques
  - Using kernels can learn very complex functions

- Neural Network
  - Nondeterministic algorithm
  - Generalizes well but doesn't have strong mathematical foundation
  - Can easily be learned in incremental fashion
  - To learn complex functions—use multilayer perceptron (nontrivial)

Evaluation and Model Selection

Evaluation: the Key to Success

- How predictive is the model we have learned?
- Error on the training data is not a good indicator of performance on future data
  - Otherwise 1-NN would be the optimum classifier!
- Simple solution that can be used if a large amount of (labeled) data is available:
  - Split data into training and test set
- However: (labeled) data is usually limited
  - More sophisticated techniques need to be used

Issues in Evaluation

- Statistical reliability of estimated differences in performance
- Choice of performance measure:
  - Number of correct classifications
  - Accuracy of probability estimates
  - Error in numeric predictions
- Costs assigned to different types of errors
  - Many practical applications involve costs

Training and Testing

- Natural performance measure for classification problems: error rate
  - Success: instance's class is predicted correctly
  - Error: instance's class is predicted incorrectly
  - Error rate: proportion of errors made over the whole set of instances
- Resubstitution error: error rate obtained by evaluating model on training data

Training and Testing (Cont.)

- Test set: independent instances that have played no part in formation of classifier
  - Assumption: both training data and test data are representative samples of the underlying problem
- Test and training data may differ in nature
  - Example: classifiers built using customer data from two different towns A and B
  - To estimate performance of classifier from town A in completely new town, test it on data from B

Parameter Tuning

- It is important that the test data is not used in any way to create the classifier
- Some learning schemes operate in two stages:
  - Stage 1: build the basic structure
  - Stage 2: optimize parameter settings
- The test data cannot be used for parameter tuning!
- Proper procedure uses three sets: training data, validation data, and test data
  - Validation data is used to optimize parameters

Making the Most of the Data

- Once evaluation is complete, all the data can be used to build the final classifier
- Generally, the larger the training data the better the classifier (but returns diminish)
- The larger the test data the more accurate the error estimate
- Holdout procedure: method of splitting original data into training and test set
  - Dilemma: ideally both training set and test set should be large!

Predicting Performance

- Assume the estimated error rate is 25%. How close is this to the true error rate?
- Depends on the amount of test data
- Prediction is just like tossing a (biased!) coin
- “Head” is a “success”, “tail” is an “error”
- In statistics, a succession of independent events like this is called a Bernoulli process
- Statistical theory provides us with confidence intervals for the true underlying proportion

Confidence Intervals

- We can say: p lies within a certain specified interval with a certain specified confidence
- Example: $S = 750$ successes in $N = 1000$ trials
- Estimated success rate: 75%
- How close is p to true success rate?
- Answer: with 80% confidence p is located in $[73.2,76.7]$
- Another example: $S = 75$ and $N = 100$
- Estimated success rate: 75%
- With 80% confidence p in $[69.1,80.1]$

Mean and Variance

- Mean and variance for a Bernoulli trial: $p, p\ (1-p)$
- Expected success rate $X = S/N$
- Mean and variance for X: $p, p\ (1-p)/N$
- For large enough N, X follows a Normal distribution
- $c\%$ confidence interval $[-z \leq X \leq z]$ for a random variable X is determined using:

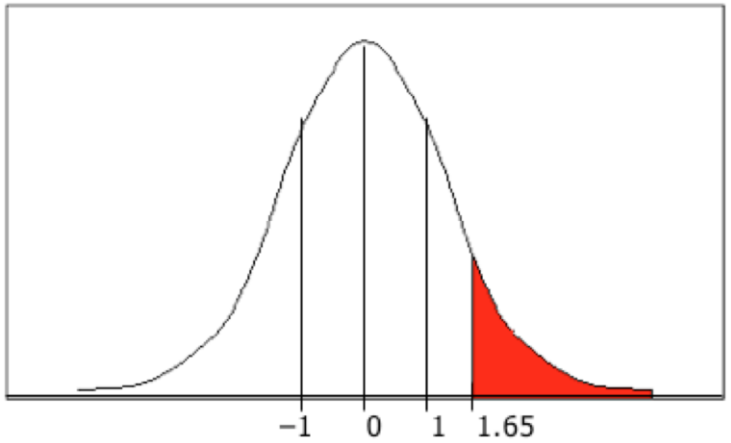
$$p(-z \leq X \leq z) = c$$

- For a symmetric distribution such as the normal distribution we have:

$$p(-z \leq X \leq z) = 1 - 2 \times p(X \geq z)$$

Confidence Limits

- Confidence limits for the normal distribution with 0 mean and a variance of 1:



$p(X > z)$	z
0.1%	3.09
0.5%	2.58
1%	2.33
5%	1.65
10%	1.28
20%	0.84
40%	0.25

- Thus: $p(-1.65 \leq X \leq 1.65) = 90\%$
- We assume the number of trials N is 1000
- To use this we have to transform our random variable X to have 0 mean and unit variance

Transforming X

- Transformed value for X: $\frac{X-p}{\sqrt{p(1-p)/N}}$
  (i.e., subtract the mean and divide by the standard deviation)
- Resulting equation:

$$p\left(-z \leq \frac{X-p}{\sqrt{p(1-p)/N}} \leq z\right) = c$$

- Solving for p yields an expression for the confidence limits:

$$p = \left(X + \frac{z^2}{2N} \pm z\sqrt{\frac{X}{N} - \frac{X^2}{N} + \frac{z^2}{4N^2}}\right) / \left(1 + \frac{z^2}{N}\right)$$

Examples

- $z = 1.28$
  $X = 75\%, N = 1000$ the $p \in [0.732, 0.767]$
  $X = 75\%, N = 100$ the $p \in [0.691, 0.801]$
  $X = 75\%, N = 10$ the $p \in [0.549, 0.881]$
- Note that: the Normal distribution assumption is only valid for large N (i.e., $N > 100$)

Holdout Estimation

- What should we do if we only have a single dataset?
- The holdout method reserves a certain amount for testing and uses the remainder for training, after shuffling
  - Usually: one third for testing, the rest for training
- Problem: the samples might not be representative
  - Example: class might be missing in the test data
- Advanced version uses stratification
  - Ensures that each class is represented with approximately equal proportions in both subsets

Repeated Holdout Method

- Holdout estimate can be made more reliable by repeating the process with different subsamples
  - In each iteration, a certain proportion is randomly selected for training (possibly with stratificiation)
  - The error rates on the different iterations are averaged to yield an overall error rate
- This is called the repeated holdout method
- Still not optimum: the different test sets overlap
  - Can we prevent overlapping?

Cross Validation

- K-fold cross-validation avoids overlapping test sets
  - First step: split data into k subsets of equal size
  - Second step: use each subset in turn for testing, the remainder for training
  - This means the learning algorithm is applied to k different training sets
- Often the subsets are stratified before the cross-validation is performed to yield stratified k-fold cross-validation
- The error estimates are averaged to yield an overall error estimate; also, standard deviation is often computed
- Alternatively, predictions and actual target values from the k folds are pooled to compute one estimate
  - Does not yield an estimate of standard deviation

Cross Validation (Cont.)

- Standard method for evaluation: stratified ten-fold cross-validation
- Why ten?
  - Extensive experiments have shown that this is the best choice to get an accurate estimate
  - There is also some theoretical evidence for this
- Stratification reduces the estimate's variance
- Even better: repeated stratified cross-validation
  - E.g., ten-fold cross-validation is repeated ten times and results are averaged (reduces the variance)

Leave-One-Out Cross Validation

- Leave-one-out: a particular form of k-fold cross-validation:
  - Set number of folds to number of training instances
  - I.e., for n training instances, build classifier n times
- Makes best use of the data
- Involves no random subsampling
- Very computationally expensive (exception: using lazy classifiers such as the nearest-neighbor classifier)

Leave-One-Out CV and Stratification

- Disadvantage of Leave-one-out CV: stratification is not possible
  - It guarantees a non-stratified sample because there is only one instance in the test set!
- Extreme example: random dataset split equally into two classes
  - Best inducer predicts majority class
  - 50% accuracy on fresh data
  - Leave-one-out CV estimate gives 100% error!

The Bootstrap

- CV uses sampling without replacement
  - The same instance, once selected, can not be selected again for a particular training/test set
- The bootstrap uses sampling with replacement to form the training set
  - Sample a dataset of n instances n times with replacement to form a new dataset of n instances
  - Use this data as the training set
  - Use the instances from the original dataset that do not occur in the new training set for testing

The 0.632 Bootstrap

- A particular instance has a probability of $1-1/n$ of not being picked
- Thus its probability of ending up in the test data is:

$$\left(1-\frac{1}{n}\right)^{n} \approx e^{-1} = 0.368$$

- This means the training data will contain approximately 63.2% of the instances

Estimating Classification Error with The 0.632 Bootstrap

- The error estimate on the test data will be quite pessimistic
- Trained on just $\sim 63\%$ of the instances
- Idea: combine it with the resubstitution error:

$$E = 0.632 E_{test\ instances} + 0.368 E_{training\ instances}$$

- The resubstitution error gets less weight than the error on the test data
- Repeat process several times with different samples; average the results

More on Bootstraping

- Probably the best way of estimating performance for very small datasets
- However, it has some problems
- Consider the random dataset from above
- A perfect memorizer will achieve
  0% resubstitution error and $\sim 50\%$ error on test data
- Bootstrap estimate for this classifier:
  $$(0.632 \times 50\% + 0.368 \times 0\%) = 31.6\%$$
- True expected error: 50%

Comparing Machine Learning Schemes

- Frequent question: which of two learning schemes performs better?
- Note: this is domain dependent!
- Obvious way: compare 10-fold cross-validation estimates
- Generally sufficient in applications (we do not loose if the chosen method is not truly better)
- However, what about machine learning research?
- Need to show convincingly that a particular method works better in a particular domain from which data is taken

Comparing Machine Learning Schemes (Cont.)

- Want to show that scheme A is better than scheme A in a particular domain
  - For a given amount of training data (i.e., data size)
  - On average, across all possible training sets from that domain
- Let's assume we have an infinite amount of data from the domain
- Then, we can simply
  - sample infinitely many dataset of a specified size
  - obtain a cross-validation estimate on each dataset for each scheme
  - check if the mean accuracy for scheme A is better than the mean accuracy for scheme B

Counting the Cost

- In practice, different types of classification errors often incur different costs
- Examples:
- Terrorist profiling: “Not a terrorist” correct 99.99…% of the time
  - Loan decisions
  - Oil-slick detection
  - Fault diagnosis
  - Promotional mailing

Confusion Matrix

- Focus on the predictive capability of a model

  Confusion Matrix:

		Predicted Label	
		Class = Y	Class = N
Actual Label	Class = Y	TP	FN
	Class = N	FP	TN

TP = True Positive
FP = False Positive
FN = False Negative
TN = Ture Negative

- Most widely used metric:

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

Accuracy

- Limitation of accuracy metric:
  - The problem of unbalanced classes
  - Consider a 2-class problem
    - Number of class 1 examples = 9990
    - Number of class 0 examples = 10
  - If the model predict everything as class 1
    - $Accuracy = \frac{9990}{10000} = 99.9\%$
- Accuracy is misleading because the classifier didn't predict any class 0 examples
- $Weighted\ Accuracy = \frac{w_1 TP + w_4 TN}{w_1 TP + w_2 FN + w_3 FP + w_4 TN}$

Classification with Cost

- Two cost Matrices:

		Predicted Label	
		Class = Y	Class = N
(A) Actual Label	Class = Y	0	1
	Class = N	1	0

		Predicted Label		
		a	b	c
(B) Actual Label	a	0	1	1
	b	1	0	1
	c	1	1	0

- In cost-sensitive evaluation of classification methods, success rate is replaced by average cost per prediction
  - Cost is given by appropriate entry in the cost matrix

Cost–Sensitive Classification

- Can take costs into account when making predictions
  - Basic idea: only predict high-cost class when very confident about prediction
- Given: predicted class probabilities
  - Normally, we just predict the most likely class
  - Here, we should make the prediction that minimizes the expected cost
    - Expected cost: dot product of vector of class probabilities and appropriate column in cost matrix
    - Choose column (class) that minimizes expected cost
- This is the minimum-expected cost approach to cost-sensitive classification

Cost–Sensitive Learning

- So far we haven't taken costs into account at training time
- Most learning schemes do not perform cost-sensitive learning
  - They generate the same classifier no matter what costs are assigned to the different classes
  - Example: standard decision tree learner
- Simple methods for cost-sensitive learning:
  - Resampling of instances according to costs
  - Weighting of instances according to costs
- Some schemes can take costs into account by varying a parameter, e.g., naïve Bayes

Lift Charts

- In practice, costs are rarely known
- Decisions are usually made by comparing possible scenarios
- Example: promotional mailout to 1,000,000 households
  - Mail to all; 0.1% respond (1000)
  - Data mining tool identifies subset of 100,000 most promising, 0.4% of these respond (400)
    40% of responses for 10% of cost may pay off
  - Identify subset of 400,000 most promising, 0.2% respond (800)
- A lift chart allows a visual comparison

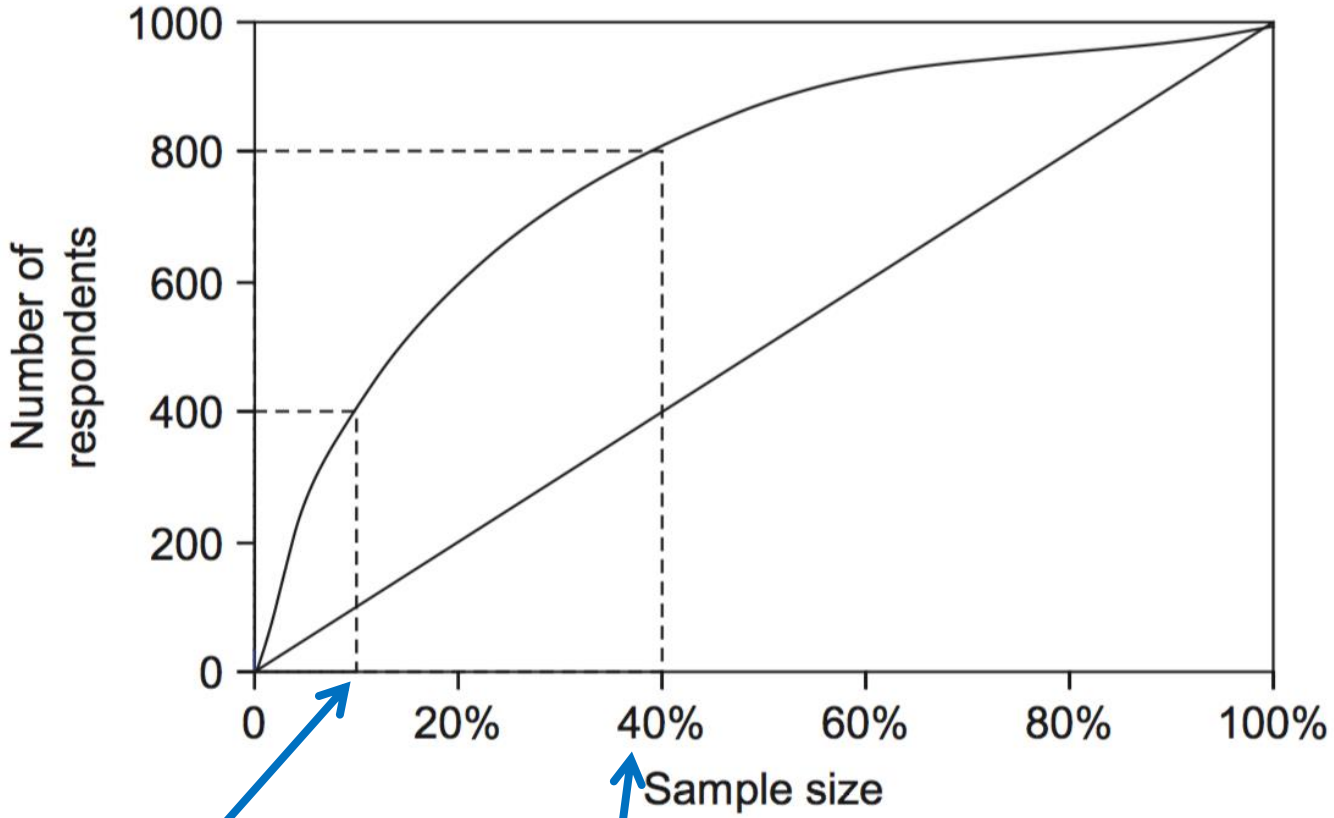
Generating a Lift Chart

- Sort the instances based on predicted probability of being positive

rank	*predicted*	actual	*rank*	*predicted*	actual
1	0.95	Yes	8	0.80	No
2	0.93	Yes	9	0.80	Yes
3	0.93	No	10	0.79	Yes
4	0.88	Yes	11	0.77	No
5	0.86	Yes	12	0.77	Yes
6	0.85	Yes	13	0.76	Yes
7	0.82	Yes	…	…	…

- x-axis in lift charts is sample size for each probability threshold
- y-axis is the number true positives above the threshold

A Hypothetical Lift Chart



40% of responses for 10% of cost

80% of responses for 40% of cost

ROC Curves

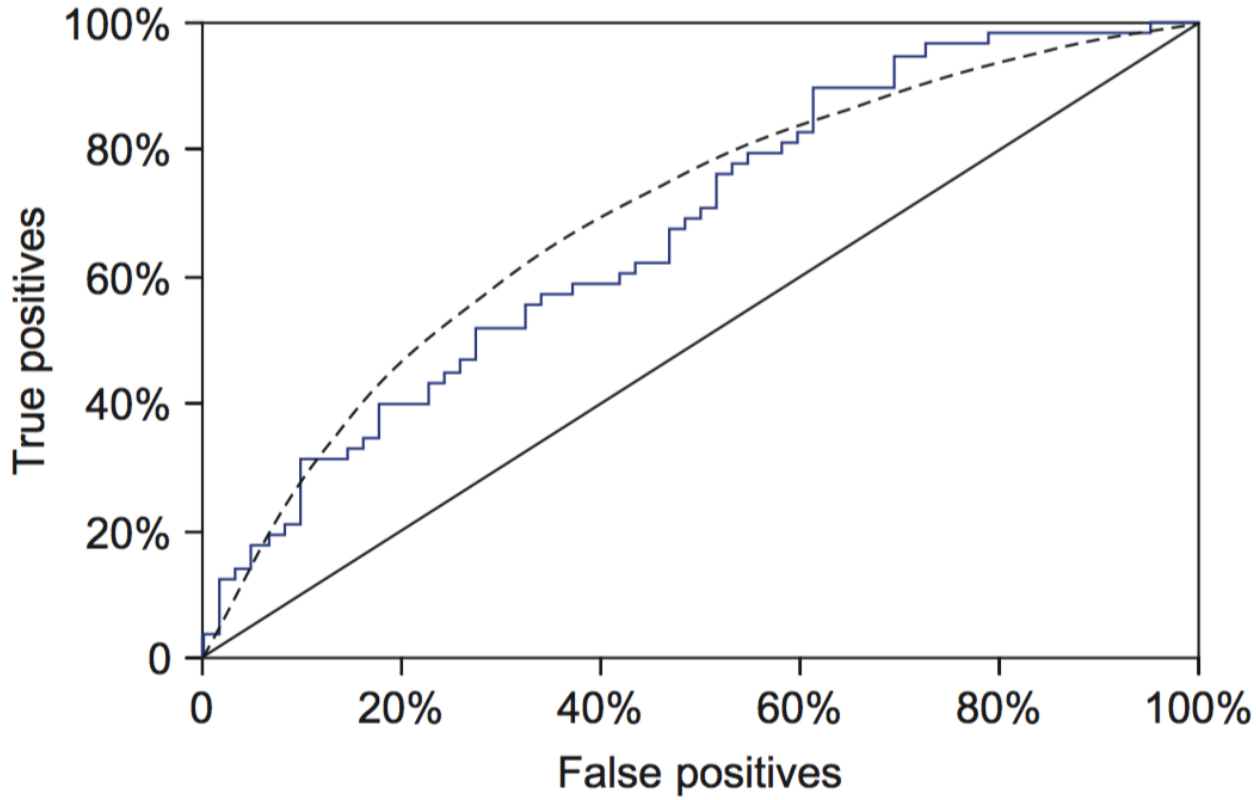
- ROC curves are similar to lift charts
  - Stands for “receiver operating characteristic”
  - Used in signal detection to show tradeoff between hit rate and false alarm rate over noisy channel
- Differences to lift chart:
  - x-axis shows percentage of true positives in sample rather than absolute number
  - x-axis shows percentage of false positives in sample rather than sample size

ROC Curves (Cont.)

- ROC curves: provides visual comparison of classification models
- The area under the ROC curve is a measure of the accuracy of the model
- Rank the test tuples in decreasing order: the one that is most likely to belong to the positive class appears at the top of the list
- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model

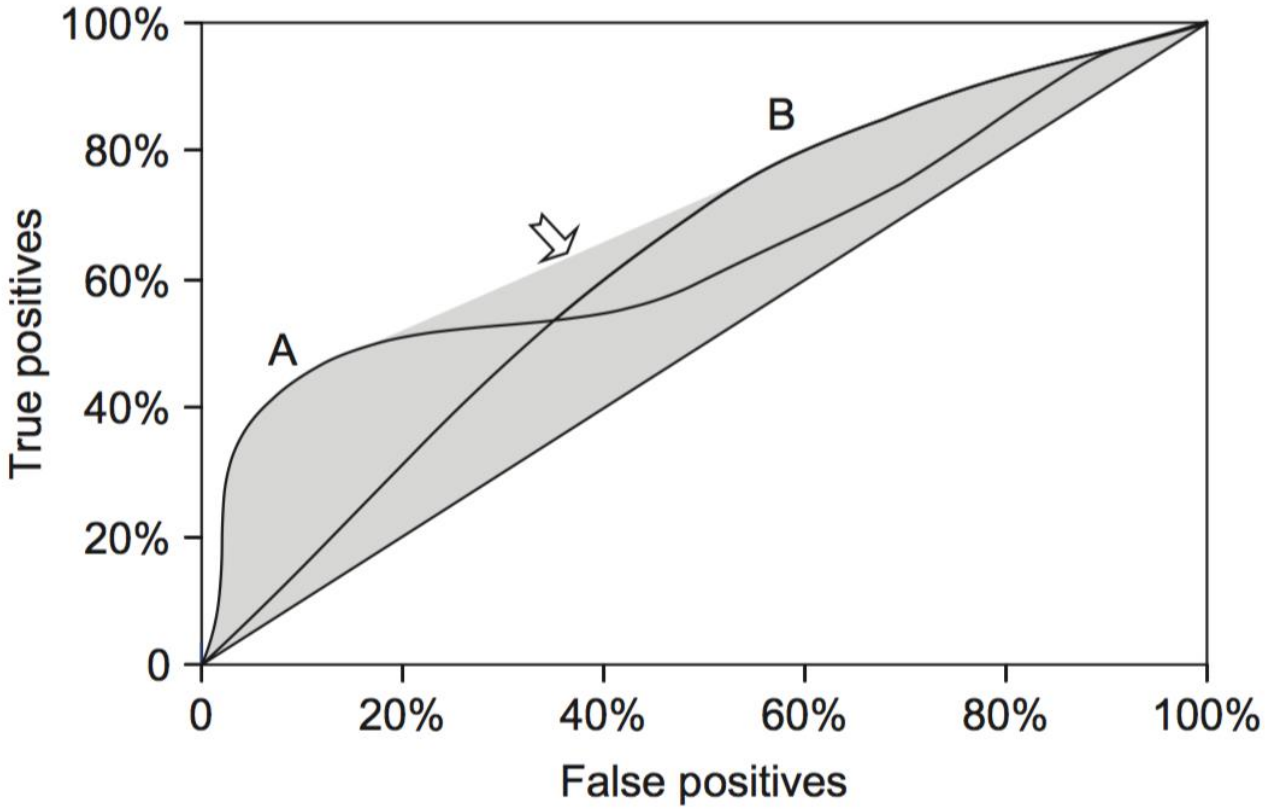
A Sample ROC Curve

- y-axis represents the true positive rate
- x-axis rep. the false positive rate
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0



ROC Curves for Two Schemes

- For a small, focused sample, use method A
- For a larger one, use method B
- In between, choose between A and B with appropriate probabilities



More Evaluation Metrics

- Percentage of retrieved documents that are relevant:

$$precision(P) = TP/(TP + FP)$$

- Percentage of relevant documents that are returned:

$$recall(R) = TP/(TP + FN)$$

- Precision/recall curves have hyperbolic shape
- F-measure: $\mathrm{F}_1 = (2 \times R \times P)/(R + P)$
  - In general: $\mathrm{F}_\beta = (1 + \beta^2)\,(R \times P)/(\beta^2 \times P + R)$
- $sensitivity \times specificity = (TP\,/\,(TP + FN)) \times (TN\,/\,(FP + TN))$
- Area under the ROC curve (AUC): probability that randomly chosen positive instance is ranked above randomly chosen negative one

Summary of Evaluation Metrics

	Domain	Plot	**Explanation**
Lift Chart	Marketing	TP	TP
		Subset Size	(TP+FP)/(TP+FP+TN+FN)
ROC Curve	Communication	TP rate	TP/(TP+FN)
		FP rate	FP/(FP+TN)
Recall-Precision Curve	Information Retrieval	Recall	TP/(TP+FN)
		Precision	TP/(TP+FP)

Issues Affecting Model Selection

- Accuracy: classifier accuracy = correctly predicting class label
- Speed
  - Time to construct the model (training time)
  - Time to use the model (classification/prediction time)
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Interpretability: understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

Reading Material & Exercises

- Chapters 8, 9 of the Data Mining: Concepts and Techniques Book
- Chapter 5 of the Data Mining: Practical Machine Learning Tools and Techniques