CSCD 429: Data Mining

— Classification —

Dr. Dan Li

Homework

- Reading assignments
  - Chapter 8 from the book
- HW 2: Titanic project
  - implement a classification algorithm from scratch
- Lab 4: Titanic project
  - use RapidMiner to do the work

Outline

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Nearest Neighbor Classification
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy
- Summary

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data are accompanied by **labels** indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

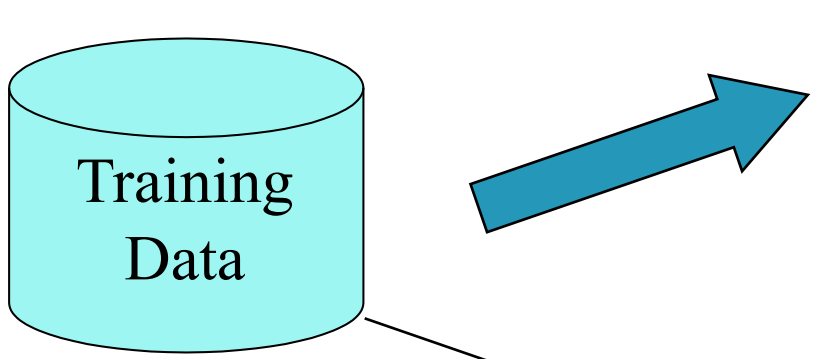
Prediction Problems: Classification vs. Numeric Prediction

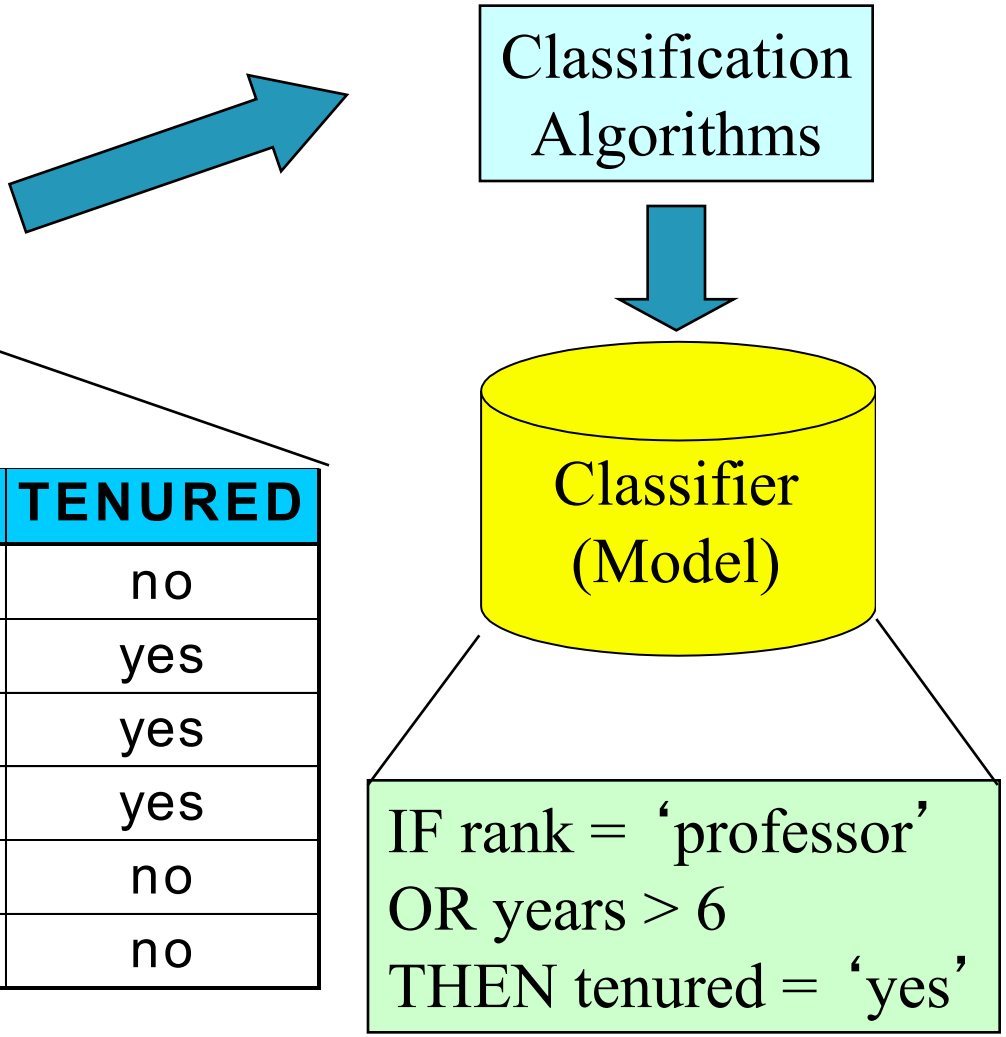
- Classification
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Regression
  - models continuous-valued functions to predict unknown 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, mathematical formulae, neural networks, support vector machines, etc.
- 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 data tuples whose class labels are not known

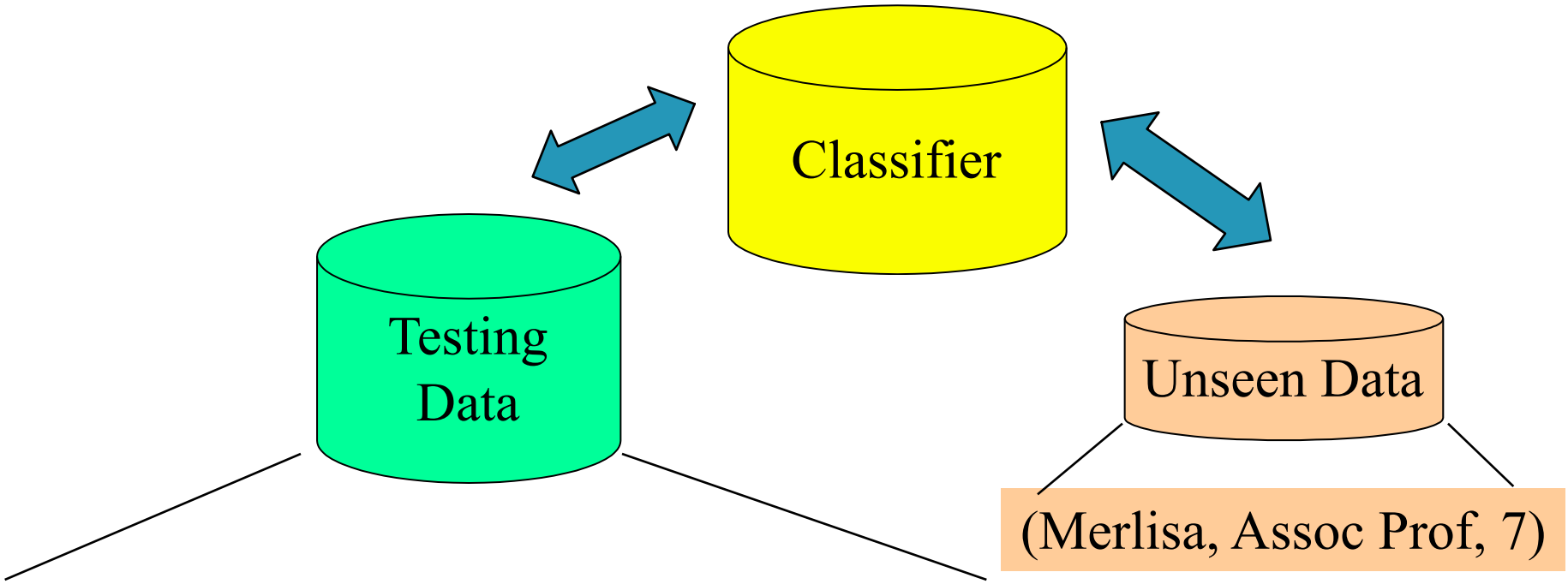
Process (1): Model Construction





NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Process (2): Using the Model in Prediction



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Tenured?

Yes

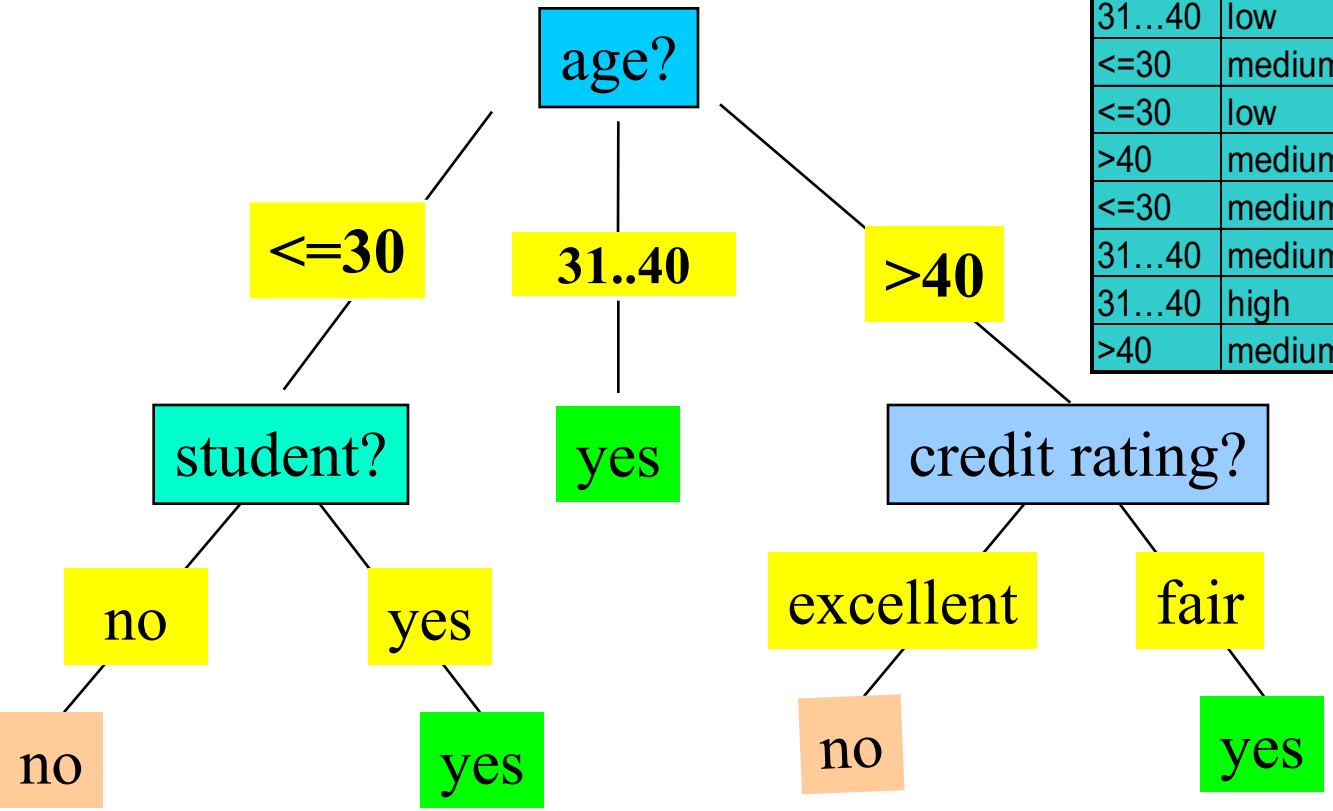
Outline

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Nearest Neighbor Classification
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy
- Summary

Decision Tree Induction: An Example

- Training data set: Buys_computer
- 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
  - Partitioning 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

Attribute Selection Measure: Information Gain (ID3)

- Select the attribute with the highest information gain
- Let p_i be the non-zero probability that an arbitrary tuple in 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

$$Gain(A) = Info(D) - Info_A(D)$$

Attribute Selection: Information Gain

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

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

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

$$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

Computing 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 ≤ split-point, and D2 is the set of tuples in D satisfying A > split-point

Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of distinct 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)
- e.g.

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

  - gain_ratio(income) = 0.029/1.557 = 0.019
- The attribute with the maximum gain ratio is selected as the splitting attribute

Gini Index (CART)

- If a data set *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 data set *D* is split on A into two subsets D_1 and D_2, the *gini* index *gini*(*D*) is defined as $gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$
- Reduction in Impurity: $\Delta gini(A) = gini(D) - gini_A(D)$
- The attribute provides the largest reduction in impurity is chosen to split the node

Gini Index: Example

Past Trend	Open Interest	Trading Volume	Return
Positive	Low	High	Up
Negative	High	Low	Down
Positive	Low	High	Up
Positive	High	High	Up
Negative	Low	High	Down
Positive	Low	Low	Down
Negative	High	High	Down
Negative	Low	High	Down
Positive	Low	Low	Down
Positive	High	High	Up

Calculating the Gini Index for 'Past Trend':

P(Past Trend=Positive): 6/10
P(Past Trend=Negative): 4/10
If (Past Trend = Positive & Return = Up), probability = 4/6
If (Past Trend = Positive & Return = Down), probability = 2/6
Gini index = $1 - ((4/6)^2 + (2/6)^2) = 0.45$
If (Past Trend = Negative & Return = Up), probability = 0
If (Past Trend = Negative & Return = Down), probability = 4/4
Gini index = $1 - ((0)^2 + (4/4)^2) = 0$
Weighted sum of the Gini Indices can be calculated as follows:
Gini Index for Past Trend = $(6/10)\mathit{0.45 + (4/10)}0 = 0.27$

Gini Index: Example

Past Trend	Open Interest	Trading Volume	Return
Positive	Low	High	Up
Negative	High	Low	Down
Positive	Low	High	Up
Positive	High	High	Up
Negative	Low	High	Down
Positive	Low	Low	Down
Negative	High	High	Down
Negative	Low	High	Down
Positive	Low	Low	Down
Positive	High	High	Up

Calculating the Gini Index for 'Open Interest':

P(Open Interest=High): 4/10
P(Open Interest=Low): 6/10
If (Open Interest = High & Return = Up), probability = 2/4
If (Open Interest = High & Return = Down), probability = 2/4
Gini index = $1 - ((2/4)^2 + (2/4)^2) = 0.5$
If (Open Interest = Low & Return = Up), probability = 2/6
If (Open Interest = Low & Return = Down), probability = 4/6
Gini index = $1 - ((2/6)^2 + (4/6)^2) = 0.45$
Weighted sum of the Gini Indices can be calculated as follows:
Gini Index for Open Interest = $(4/10)\mathit{0.5} + \mathit{(6/10)}0.45 = 0.47$

Gini Index: Example

Past Trend	Open Interest	Trading Volume	Return
Positive	Low	High	Up
Negative	High	Low	Down
Positive	Low	High	Up
Positive	High	High	Up
Negative	Low	High	Down
Positive	Low	Low	Down
Negative	High	High	Down
Negative	Low	High	Down
Positive	Low	Low	Down
Positive	High	High	Up

Calculating the Gini Index for 'Trading Volume':

P(Trading Volume=High): 7/10
P(Trading Volume=Low): 3/10
If (Trading Volume = High & Return = Up), probability = 4/7
If (Trading Volume = High & Return = Down), probability = 3/7
Gini index = $1 - ((4/7)^2 + (3/7)^2) = 0.49$
If (Trading Volume = Low & Return = Up), probability = 0
If (Trading Volume = Low & Return = Down), probability = 3/3
Gini index = $1 - ((0)^2 + (1)^2) = 0$
Weighted sum of the Gini Indices can be calculated as follows:
Gini Index for Trading Volume = $(7/10)\mathit{0.49 + (3/10)}0 = 0.34$

Gini Index: Example

Past Trend	Open Interest	Trading Volume	Return
Positive	Low	High	Up
Negative	High	Low	Down
Positive	Low	High	Up
Positive	High	High	Up
Negative	Low	High	Down
Positive	Low	Low	Down
Negative	High	High	Down
Negative	Low	High	Down
Positive	Low	Low	Down
Positive	High	High	Up

Attributes/Features	Gini Index
Past Trend	0.27
Open Interest	0.47
Trading Volume	0.34

'Past Trend' has the lowest Gini Index and hence it will be chosen as the root node for how decision tree works.

Comparing Attribute Selection Measures

- The three measures, in general, return good results but
  - **Information gain**
    - biased towards multivalued attributes
  - **Gain ratio**
    - tends to favor unbalanced splits in which one partition is much smaller than the others
  - **Gini index**
    - biased to multivalued attributes
    - tends to favor tests that result in equal-sized partitions and purity in both partitions

Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
  - Too many branches, good accuracy for labeled samples
  - 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"

Decision Tree: Summary

- 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
- Things to consider
  - Overfitting

Outline

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Nearest Neighbor Classification
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy
- Summary

Bayesian Classification: Introduction

- 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

Bayesian Theorem: Basics

- Let **X** be a data sample ("*evidence*"): class label is unknown
- Let H be a *hypothesis* that X belongs to class C
- Classification is to determine P(H|**X**), (*posteriori probability),* the probability that the hypothesis holds given the observed data sample **X**
- P(H) (*prior probability*), the initial probability
  - E.g., **X** will buy computer, regardless of age, income, …
- P(**X**): probability that sample data is observed
- P(**X**|H) (likelyhood), the probability of observing the sample **X**, given that the hypothesis holds
  - E.g., Given that **X** will buy computer, the prob. that X is 31..40, medium income

Bayesian Theorem

- Given data **X**, *posteriori probability of a hypothesis* H, P(H|**X**), follows the **Bayes theorem**

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H)/P(\mathbf{X})$$

- Informally, this can be written as

  posteriori = likelihood x prior/evidence

- Predicts **X** belongs to C_i iff the probability $P(C_i|\mathbf{X})$ is the highest among all the $P(C_k|X)$ for all the *k* classes
- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

Towards Naïve Bayesian Classifier

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

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- Since P(X) is constant for all classes, only

needs to be maximized

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

Derivation of Naïve Bayes Classifier

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

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i)$$

- This greatly reduces the computation cost: Only counts the class distribution
- If A_k is categorical, $P(x_k|C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_{i,D}|$ (# of tuples of C_i in D)
- If A_k is continous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

and $P(x_k|C_i)$ is

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

$$P(\mathbf{X}|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

Naïve Bayesian Classifier: Training Dataset

Class:

C1:buys_computer = 'yes'

C2:buys_computer = 'no'

Data sample:

X = (age <=30,

Income = medium,

Student = yes

Credit_rating = Fair)

age	income	student	credit_rating	comp
<=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

Naïve Bayesian Classifier: An Example

age	income	student	credit_rating	com
<=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

- Calculate *prior probability* $P(C_i)$
- $P(C_i)$: P(buys_computer = "yes") = 9/14 = 0.643

  P(buys_computer = "no") = 5/14= 0.357

Naïve Bayesian Classifier: An Example

- Calculate *likelihood* $P(X|C_i)$
- Assume hypothesis is C_1, i.e., **buy_computer = yes**
- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**
- Using naïve assumption, compute $P(X|C_i)$ for each attribute

P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222

P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444

P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667

P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667

- **$P(X|C_i)$:** P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044

age	income	student	credit_rating	comp
<=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

Naïve Bayesian Classifier: An Example

- Calculate *likelihood* $P(X|C_i)$
- Assume hypothesis is C_2, i.e., **buy_computer = no**
- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**
- Using naïve assumption, compute $P(X|C_i)$ for each attribute

  P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6

  P(income = "medium" | buys_computer = "no") = 2/5 = 0.4

  P(student = "yes" | buys_computer = "no") = 1/5 = 0.2

  P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4
- **$P(X|C_i)$:** P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

age	income	student	credit_rating	comp
<=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

Naïve Bayesian Classifier: An Example

- Putting all together
- **$P(C_i)$:**
  - P(buys_computer = "yes") = 9/14 = 0.643
  - P(buys_computer = "no") = 5/14= 0.357
- X = (age <= 30 , income = medium, student = yes, credit_rating = fair)
- **$P(X|C_i)$:**
  - P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
  - P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019
- **$P(X|C_i)*P(C_i)$:**
  - P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028
  - P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007
- **Therefore, X belongs to class ("buys_computer = yes")**

Avoiding the Zero-Probability

- Naïve Bayesian prediction requires each conditional prob. be **non-zero**. Otherwise, the predicted prob. will be zero

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10)
- Use **Laplacian correction** (or Laplacian estimator)
  - *Adding 1 to each case*

    Prob(income = low) = 1/1003

    Prob(income = medium) = 991/1003

    Prob(income = high) = 11/1003
  - The "corrected" prob. estimates are close to their "uncorrected" counterparts

Naïve Bayesian Classifier: Comments

- 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 Bayesian Classifier
- How to deal with these dependencies? Bayesian Belief Networks (Chapter 9)

Outline

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Nearest Neighbor Classification
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy
- Summary

Using IF-THEN Rules for Classification

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

  R: IF *age* <= 30 AND *student* = no THEN *buys_computer* = no

  – LHS: Rule antecedent/precondition

  – RHS: Rule consequent

- Assessment of a rule: *coverage* and *accuracy*

  – n_{covers} = # of tuples covered by R (i.e., satisfying rule antecedent)

  – $n_{correct}$ = # of tuples correctly classified by R

age	income	student	credit_rating	com
<=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

coverage(R) = n_{covers} /|D| /* D: training data set */

accuracy(R) = $n_{correct}$ / n_{covers}

Using IF-THEN Rules for Classification

- 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

  Example:

  Rule 1: IF age <= 30 and income = "high" THEN buys-computer = yes

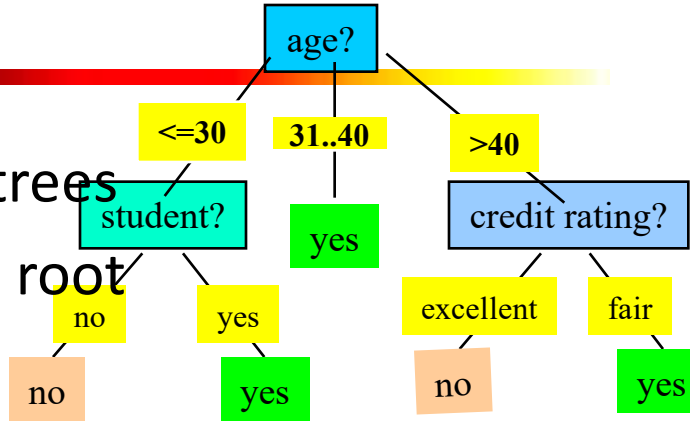
  Rule 2: IF age <= 30 and credit_rating = "fair" and student = "no" THEN buys-computer = no

  Test case: age <= 30, income = "high", student = "no', credit_rating = "fair"

Building Classification Rules

- Indirect Method:
  - Extract rules from other classification models (e.g. decision trees, etc).
  - e.g: C4.5 rules
- Direct Method:
  - Extract rules directly from data
  - e.g.: sequential covering algorithm, e.g., RIPPER

Indirect Method: Rule Extraction



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

IF *age* = 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*

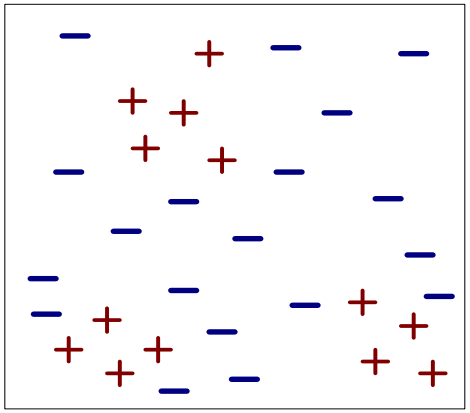
IF *age* = old AND *credit_rating* = *fair* THEN *buys_computer* = *yes*

Direct Method: Sequential Covering Method

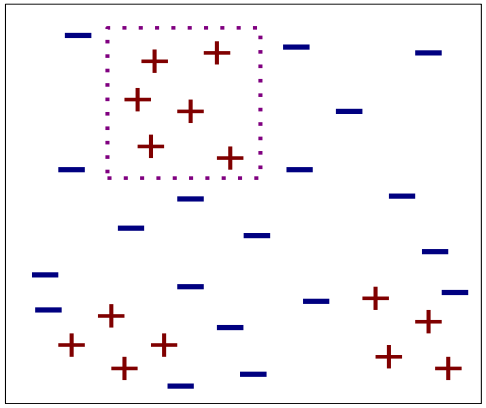
- Sequential covering algorithm: Extracts rules directly from training data
- Rules are learned *sequentially*, each for a given class C_i will cover many tuples of C_i but none (or few) of the tuples of other classes
- Steps:
  - Rules are learned one at a time
  - Each time a rule is learned, the tuples covered by the rules are removed
  - The process repeats on the remaining tuples unless *termination condition*, e.g., when no more training examples or when the quality of a rule returned is below a user-specified threshold

while (enough target tuples left)
    generate a rule
    remove tuples satisfying this rule

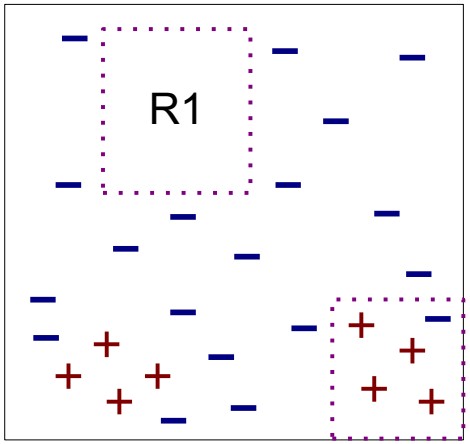
Example of Sequential Covering



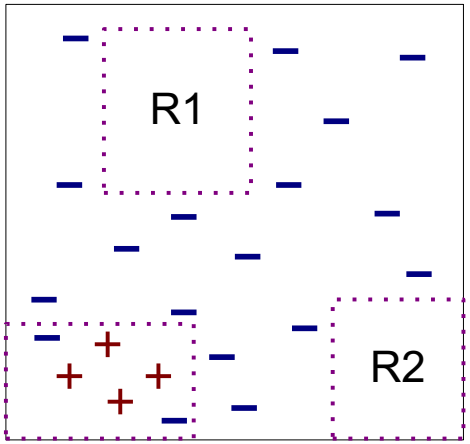
(i) Original Data



(ii) Step 1



(iii) Step 2



(iv) Step 3

Rule Generation

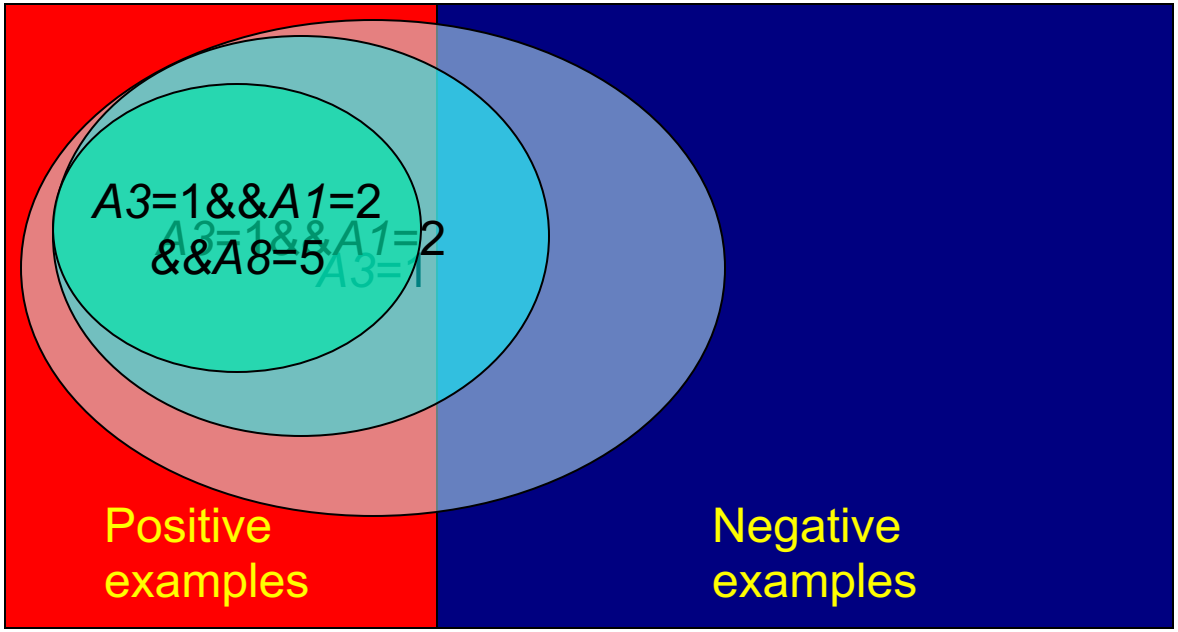
- To generate a rule

  while(true)

  find the best predicate *p*

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

  else break



How to Learn-One-Rule?

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

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

    - favors rules that have high accuracy and cover many positive tuples
    - *pos* and *neg* are # of positive and negative tuples covered by R before adding an additional condition.
    - *pos'* and *neg'* are # of positive and negative tuples covered by R after adding an additional condition.

Rule-base Classification: Summary

- As highly expressive as decision trees
- Easy to interpret, commonly used for medical disease diagnosis
- Easy to generate
- Can classify new instances rapidly
- Performance comparable to decision trees

Outline

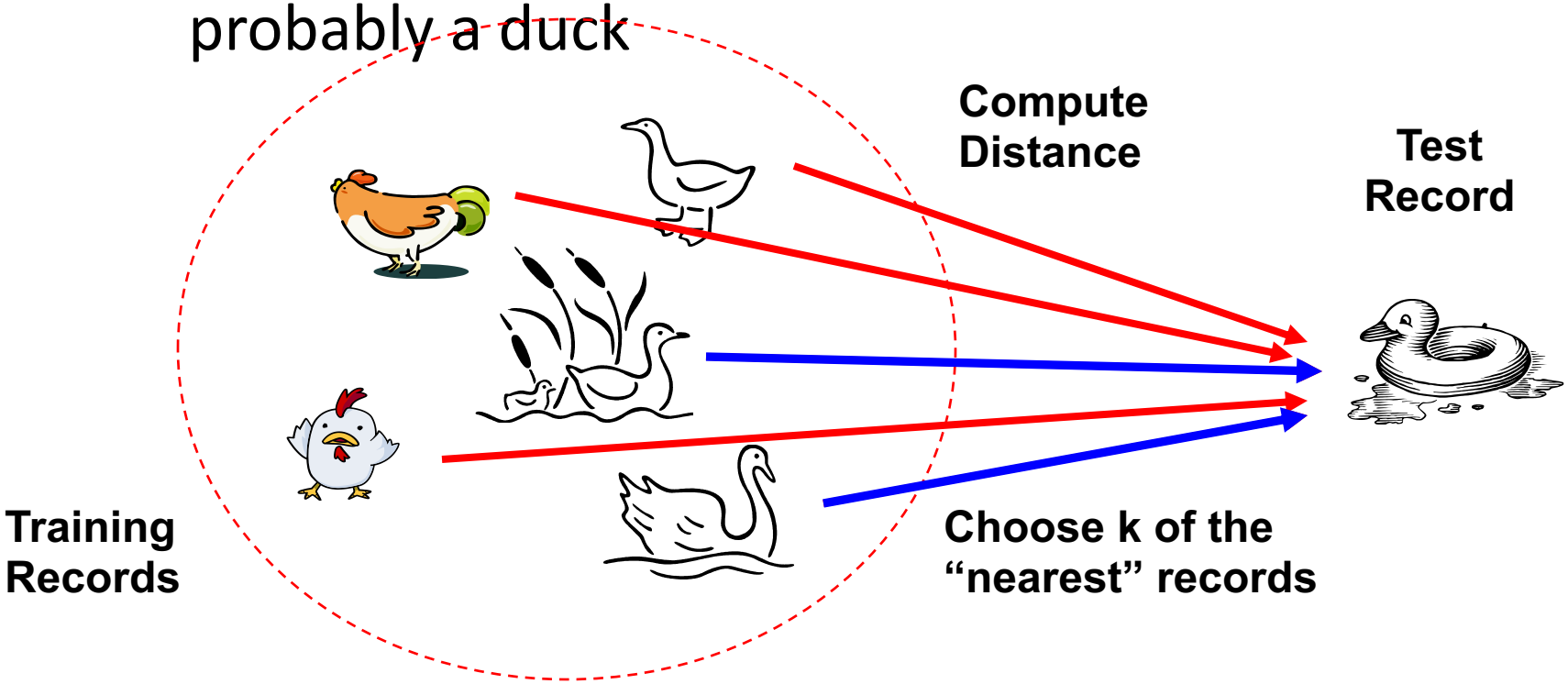
- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Nearest Neighbor Classification
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy
- Summary

Eager vs. Lazy Learning

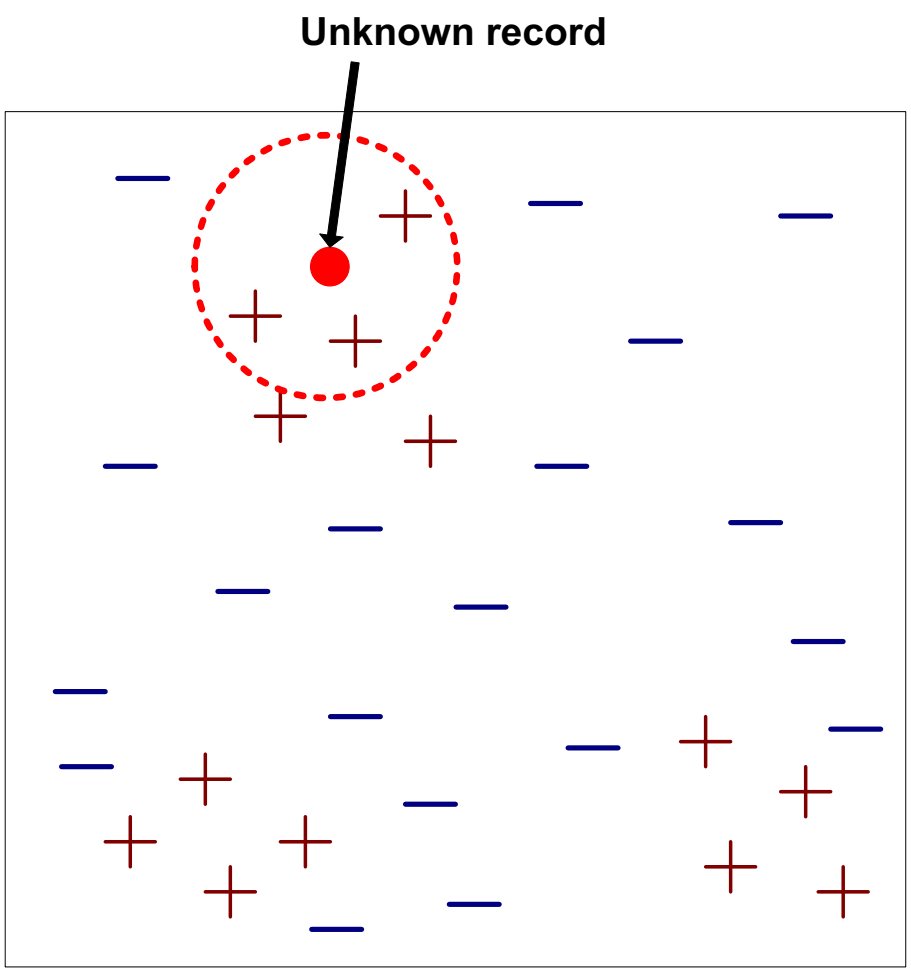
- Eager vs. lazy learning
  - **Eager learning** (the above discussed methods): Given a set of training tuples, constructs a classification model before receiving new (e.g., test) data to classify
  - **Lazy learning** (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
- Lazy: less time in training but more time in predicting
- Typical approach
  - *k*-nearest neighbor approach

K Nearest-Neighbor Classifier

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck

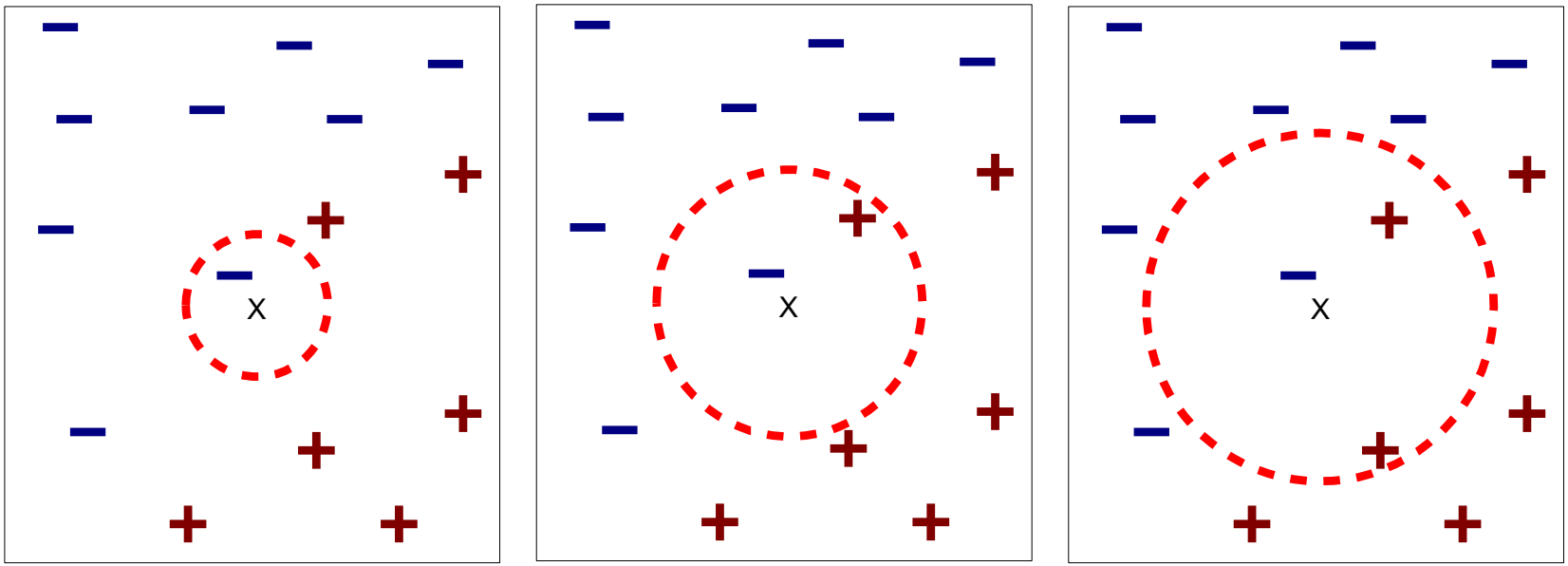


K Nearest-Neighbor Classifier



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve
- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Definition of Nearest Neighbor



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

Nearest Neighbor Classification

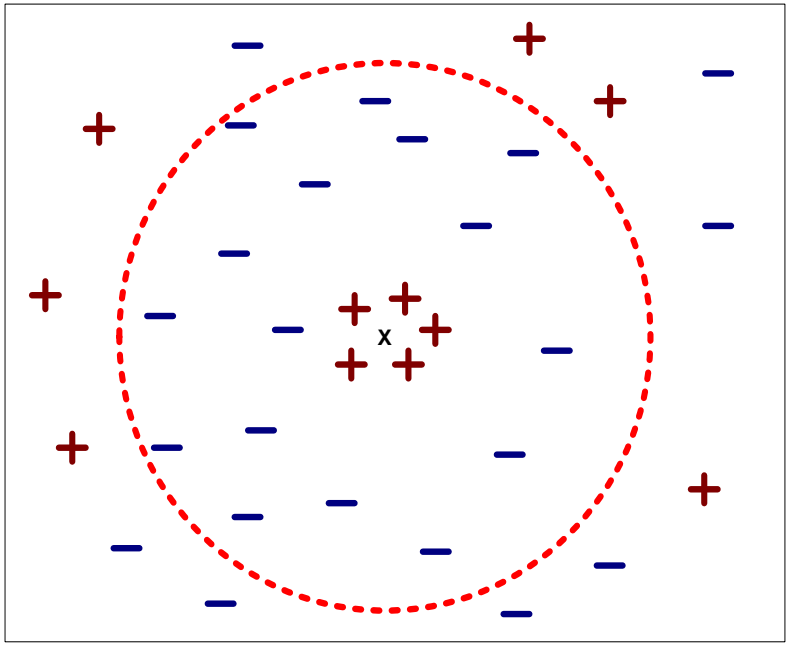
- Compute distance between two points:
  - Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, $w = 1/d^2$

Nearest Neighbor Classification

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes
  - It is best to run through each possible value for k and then decide



Nearest Neighbor Classification

- Scaling issues
  - Attributes may have to be scaled/normalied 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

Nearest Neighbor Classification

- Problem with Euclidean measure:
  - High dimensional data
    - curse of dimensionality
  - Can produce counter-intuitive results

1 1 1 1 1 1 1 1 1 1 1 0		**1 0 0 0 0 0 0 0 0 0 0 0**
	vs	
0 1 1 1 1 1 1 1 1 1 1 1		**0 0 0 0 0 0 0 0 0 0 0 1**
d = 1.4142		**d = 1.4142**

- Solution: Choose different similarity/distance measures

Nearest neighbor Classification: Summary

- k-NN classifier is a lazy learner
  - It does not build models explicitly
  - Unlike eager learners such as decision tree induction and rule-based systems
  - Classifying unknown records are relatively expensive
- Choose various k values
- Choose various similarity measures

Outline

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Nearest Neighbor Classification
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy
- Summary

Model Evaluation and Selection

- Evaluation metrics: How can we measure accuracy? Other metrics to consider?
- Use **test set** of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating a classifier's accuracy:
  - Holdout method, random sampling
  - Cross-validation
  - Bootstrap
- Comparing classifiers:
  - Cost-benefit analysis and ROC Curves

Classifier Evaluation Metrics: Confusion Matrix

Confusion Matrix:

Actual class\Predicted class	C_1	$\neg C_1$
C_1	**True Positives (TP)**	**False Negatives (FN)**
$\neg C_1$	**False Positives (FP)**	**True Negatives (TN)**

Example of Confusion Matrix:

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	**6954**	**46**	7000
buy_computer = no	**412**	**2588**	3000
Total	7366	2634	10000

- Given *m* classes, an entry, $\boldsymbol{CM_{i,j}}$ in a **confusion matrix** indicates # of tuples in class *i* that were labeled by the classifier as class *j*
- May have extra rows/columns to provide totals

Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P'	**N'**	**All**

- **Classifier Accuracy,** or recognition rate: percentage of test set tuples that are correctly classified

  Accuracy = (TP + TN)/All

- **Error rate:** *1 – accuracy*, or

  Error rate = (FP + FN)/All

- **Class Imbalance Problem**:
  - One class may be *rare*, e.g. fraud, or Covid-positive
  - Significant *majority of the negative class* and minority of the positive class
  - **Sensitivity**: True Positive recognition rate
    - **Sensitivity = TP/P**
  - **Specificity**: True Negative recognition rate
    - **Specificity = TN/N**

Classifier Evaluation Metrics: Precision and Recall, and F-measures

- **Precision**: exactness – what % of tuples that the classifier labeled as positive are actually positive

$$precision = \frac{TP}{TP + FP}$$

- **Recall:** completeness – what % of positive tuples did the classifier label as positive?

$$recall = \frac{TP}{TP + FN}$$

- Perfect score is 1.0
- Inverse relationship between precision & recall
- ***F* measure (*F*$_1$ or *F*-score)**: harmonic mean of precision and recall,

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

- ***F*$_\beta$:** weighted measure of precision and recall

$$F_\beta = \frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$

Classifier Evaluation Metrics: Example

Actual Class\Predicted class	cancer = yes	cancer = no	Total
cancer = yes	**90**	**210**	300
cancer = no	**140**	**9560**	9700
Total	230	9770	10000

Calculate:

- *Accuracy*
- *Sensitivity*
- *Specificity*
- *Precision*
- *Recall*

Classifier Evaluation Metrics: Example

Actual Class\Predicted class	cancer = yes	cancer = no	Total
cancer = yes	**90**	**210**	300
cancer = no	**140**	**9560**	9700
Total	230	9770	10000

Calculate:

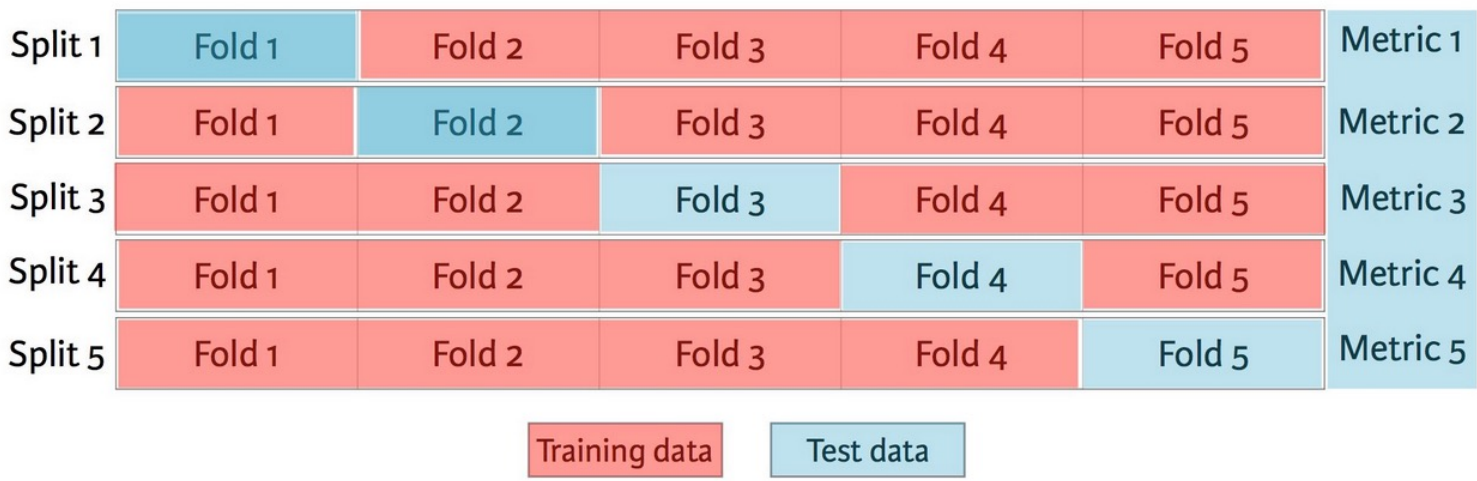
- *Accuracy: 96.4%*
- *Sensitivity: 30%*
- *Specificity:98.56%*
- *Precision: 39.13%*
- *Recall: 30%*

Evaluating Classifier Accuracy: Holdout Methods

- **Holdout method**
  - Given data is randomly partitioned into two independent disjoint sets
    - Training set (e.g., 80%) for model construction
    - Test set (e.g., 20%) for accuracy estimation
  - Random sampling: a variation of holdout
    - Repeat holdout several times
    - Accuracy = avg. of the accuracies obtained
  - The hold-out method is good to use when you have a very large dataset, you're on a time crunch, or you are starting to build an initial model in your project.

Evaluating Classifier Accuracy: Cross-Validation Methods

- **Cross-validation** (*k*-fold, where k = 10 is most popular)
  - Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
  - At *i*-th iteration, use D_i as test set and others as training set
  - Leave-one-out: *k* folds where *k* = # of tuples, for small sized data
  - ***Stratified cross-validation***: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data



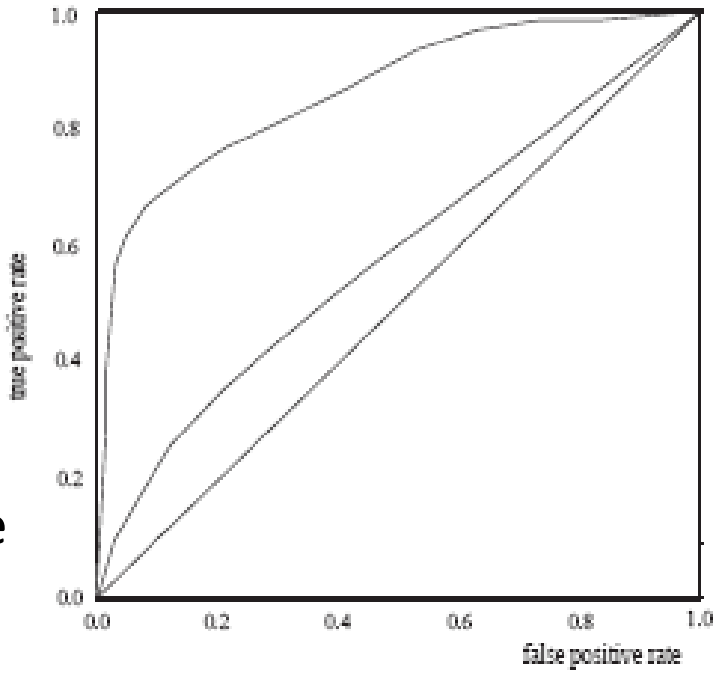
Evaluating Classifier Accuracy: Bootstrap

- Bootstrap
  - Works well with small data sets
  - Iteratively resamples a dataset with replacement
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Several bootstrap methods, and a common one is **.632 boostrap**
  - A data set with *d* tuples is sampled *d* times, with replacement.
  - The data tuples that did not make it into the training set end up forming the test set.
  - About 63.2% of the original data end up in the bootstrap, and the remaining 36.8% form the test set (since $(1 - 1/d)^d \approx e^{-1} = 0.368$)
  - Repeat the sampling procedure *k* times, overall accuracy of the model:

$$Acc(M) = \frac{1}{k}\sum_{i=1}^{k}(0.632 \times Acc(M_i)_{test\_set} + 0.368 \times Acc(M_i)_{train\_set})$$

Model Selection: ROC Curves

- **ROC** (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Shows the trade-off between the true positive rate and the false positive rate
- The area under the ROC curve (AUC) 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 model with perfect accuracy will have an area of 1.0



- Vertical axis rep. the true positive rate
- Horizontal axis rep. the false positive rate

Issues Affecting Model Selection

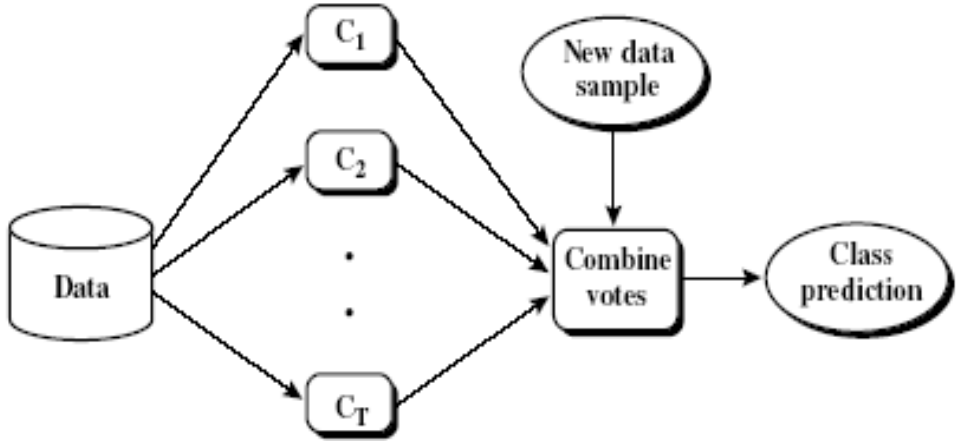
- **Accuracy**
  - classifier accuracy: predicting class label
- **Speed**
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- **Robustness**: handling noise and missing values
- **Scalability**: efficiency in disk-resident databases
- **Interpretability**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

Outline

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Nearest Neighbor Classification
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy
- Summary

Ensemble Methods: Increasing the Accuracy

- Ensemble methods
  - Combine a series of k learned models, M_1, M_2, …, M_k, with the aim of creating an improved model M*
- Popular ensemble methods
  - Bagging: averaging the prediction over a collection of classifiers
  - Boosting: weighted vote with a collection of classifiers

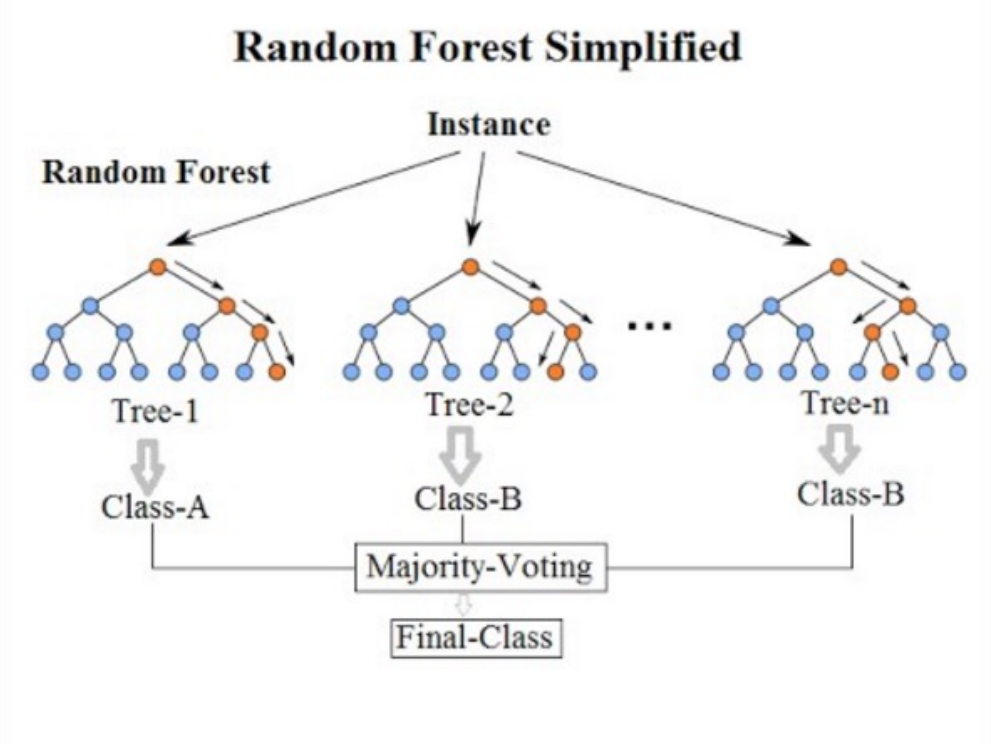


Bagging: Boostrap Aggregation

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
  - Given a set D of *d* tuples, at each iteration *i,* use bootstrap sampling to generate a training set D_i
  - A classifier model M_i is learned using training set D_i
- Classification: classify an unknown sample **X**
  - Each classifier M_i returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the most votes to **X**
- Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy
  - Often significantly better than a single classifier derived from D
  - More robust for noise data
  - Proved improved accuracy in prediction

Bagging Example: Random Forest

- Ensemble learning method
  - define number of trees
  - partition data by bootstrapping
  - on each partition construct trees
  - using a random selection of attributes at each node to determine the split (why?)
  - for classifying a new instance vote over all trees



Boosting

- Analogy
  - Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy
- Basic Idea
  - A series of k classifiers is sequentially learned
  - After a classifier M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to **pay more attention to the training tuples that were misclassified** by M_i
  - The final **M* combines the votes** of each individual classifier, where the weight of each classifier's vote is a function of its accuracy

Boosting Implementation

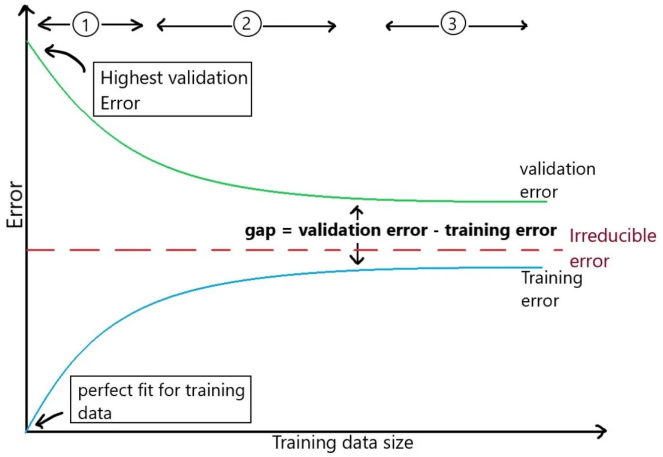
- Boosting Steps
  - Draw a random subset of training samples d1 without replacement from the training set D to train a weak learner C1
  - Draw second random training subset d2 without replacement from the training set and add 50 percent of the samples that were previously falsely classified/misclassified to train a weak learner C2
  - Find the training samples d3 in the training set D on which C1 and C2 disagree to train a third weak learner C3
  - Combine all the weak learners via majority voting.
- Comparing with bagging
  - Boosting tends to have greater accuracy, but it also risks overfitting the model to misclassified data

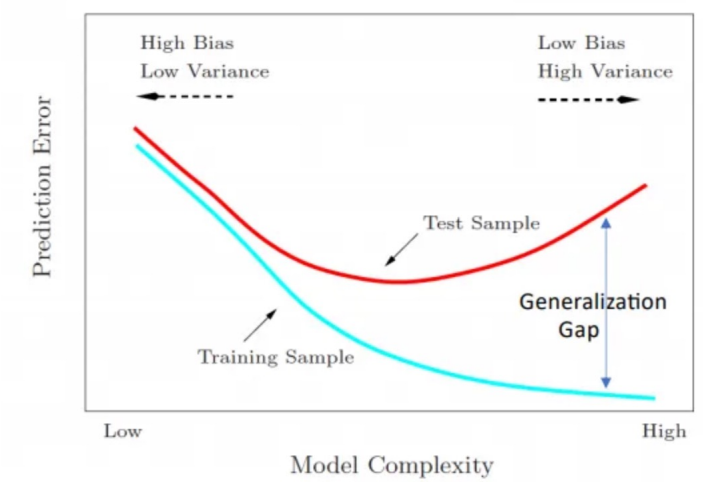
Bias vs. Variance

- Bias: training set error rate
- Variance: the gap between training and validation
- High bias, low variance: underfitting
  - Can’t describe the relationships between input attributes and target attribute
- Low bias, high variance: overfitting
  - Learning model is too complex

Bias vs. Variance

- How to fix?
  - Get more training examples
    - Fixes high variance
  - Try smaller sets of features
    - Fixes high variance
  - Try additional features
    - Fixes high bias
  - Add polynomial features in regression
    - Fixes high bias
  - Add more layers in NNs
    - Fixes high bias





Outline

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Nearest Neighbor Classification
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy
- Summary

Overall Summary (I)

- Classification is a form of data analysis that extracts models describing important data classes.
- Effective and scalable methods have been developed for decision tree induction, Naive Bayesian classification, rule-based classification, and many other classification methods.
- Evaluation metrics include: accuracy, sensitivity, specificity, precision, recall, *F* measure, and F_{β} measure.
- k-fold cross-validation is recommended for accuracy estimation.
- ROC curves are useful for model selection.
- Bagging and boosting can be used to increase overall accuracy by learning and combining a series of individual models.

Overall Summary (II)

- There have been numerous comparisons of the different classification methods; the matter remains a research topic
- No single method has been found to be superior over all others for all data sets
- Issues such as accuracy, training time, robustness, scalability, and interpretability must be considered and can involve trade-offs, further complicating the quest for an overall superior method