Classification

Classification: Basic Concepts and Methods

- Classification: Basic Concepts
- Decision Tree
- Bayes Classification Methods
- Model Evaluation and Selection
- Ensemble Methods

Motivating Example – Fruit Identification

Skin	Color	Size	Flesh	Conclusion
Hairy	Brown	Large	Hard	safe
Hairy	Green	Large	Hard	Safe
Smooth	Red	Large	Soft	Dangerous
Hairy	Green	Large	Soft	Safe
Smooth	Red	Small	Hard	Dangerous
…				

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) 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

Machine Learning

- **Supervised:** Given input/output samples (X, y), we learn a function f such that y = f(X), which can be used on new data.
  - **Classification:** y is discrete (class labels).
  - **Regression:** y is continuous, e.g. linear regression.

- **Unsupervised:** Given only samples X, we compute a function f such that y = f(X) is "simpler".
  - **Clustering:** y is discrete
  - **Dimension reduction:** y is continuous, e.g. matrix factorization

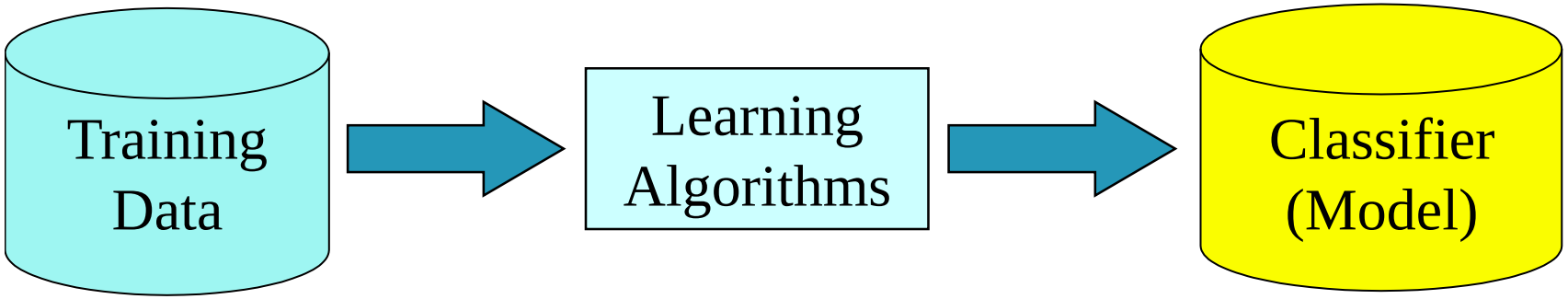
Machine Learning Problems

	Supervised Learning	*Unsupervised Learning*
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

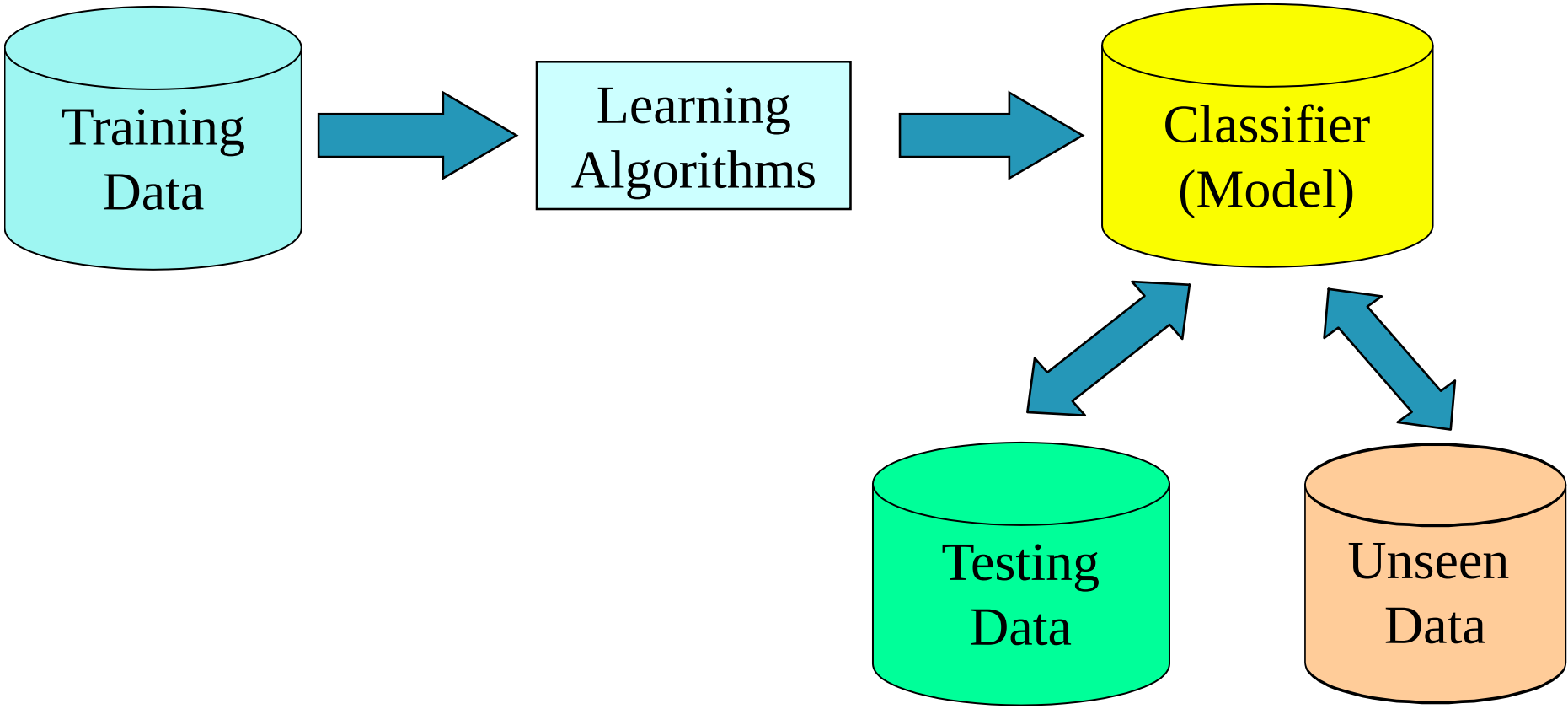
Classification—A Two-Step Process

- Model construction:
  - The set of tuples used for model construction is training set
  - Each tuple/sample has a class label attribute
  - The model can be represented as classification rules, decision trees, mathematical function, neural networks, …
- Model evaluation and usage:
  - Estimate accuracy of the model on test set that is independent of training set (otherwise overfitting)
  - If the accuracy is acceptable, use the model on new data

Process (1): Model Construction



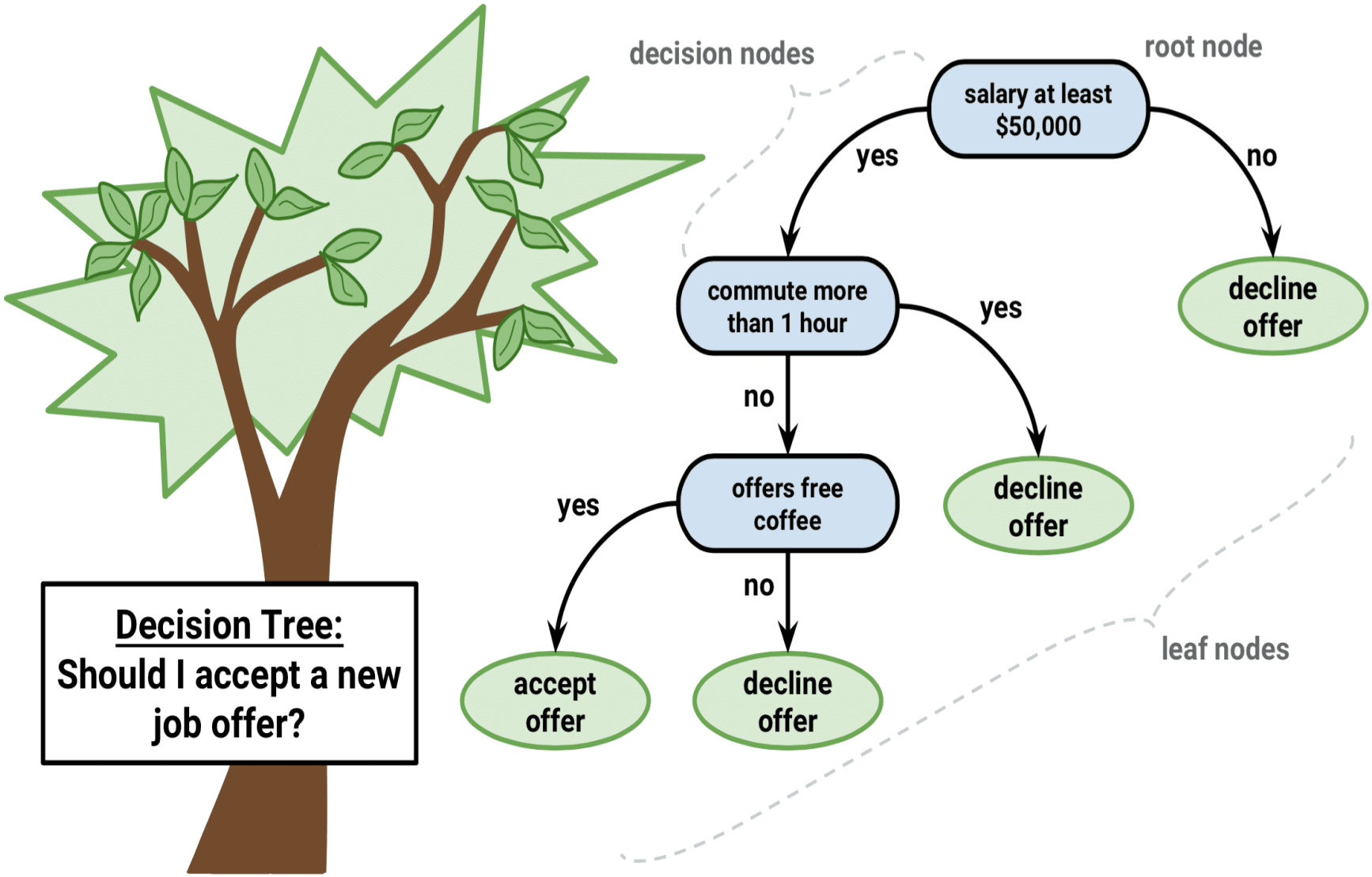
Process (2): Model Evaluation and Using Model



Classification: Basic Concepts and Methods

- Classification: Basic Concepts
- Decision Tree
- Bayes Classification Methods
- Model Evaluation and Selection
- Ensemble Methods

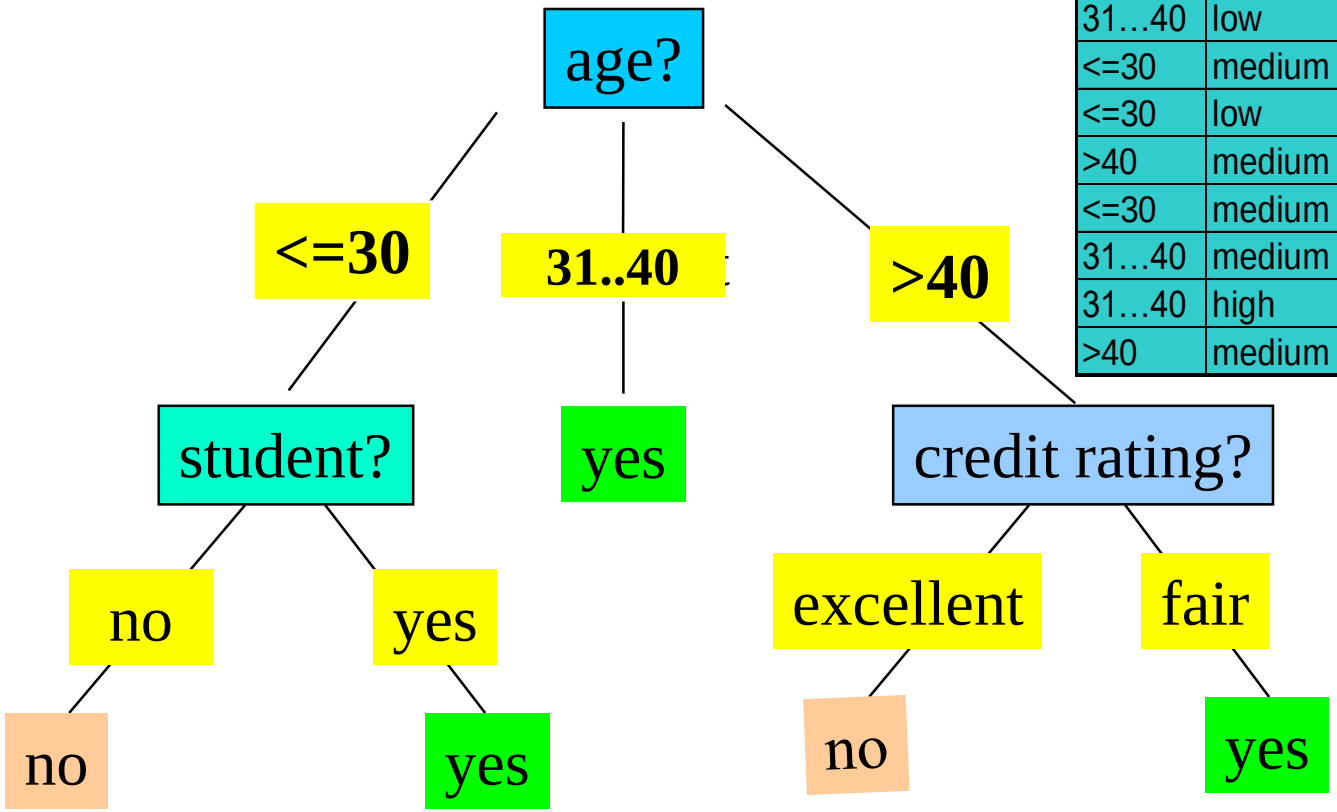
Decision tree



Decision Tree: An Example

- Training data set:
- 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 Learning the Decision Tree

- ID3 (Iterative Dichotomiser), C4.5, by Quinlan
- CART (Classification and Regression Trees)

- 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
  - Split 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 Measures

- Idea: select attribute that partition samples into homogeneous groups
- Measures
  - Information gain (ID3)
  - Gain ratio (C4.5)
  - Gini index (CART)
  - Variance reduction for continuous target variable (CART)

Brief Review of Entropy

- Entropy (Information Theory)
  - A measure of uncertainty associated with a random variable
  - Calculation: For a discrete random variable *Y* taking *m* distinct values $\{y_1, \ldots, y_m\}$,
    - $H(Y) = -\sum_{i=1}^{m} p_i \log(p_i)$, where $p_i = P(Y = y_i)$
  - Interpretation:
    - Higher entropy => higher uncertainty
    - Lower entropy => lower uncertainty
- Conditional Entropy
  - $H(Y|X) = \sum_x p(x) H(Y|X = x)$

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Information entropy of the classes in D:

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

- Information entropy after using A to split D into v partitions D_j:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

- Information gain by branching on attribute A

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

Attribute Selection: Information Gain

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

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

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

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

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

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

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

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

Continuous-Valued Attributes

- To determine the *best split point* for a continuous-valued attribute A
  - Sort the values of 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}
  - Select split point with highest info gain
- 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 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) – smaller *splitinfo* preferred

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

  - GainRatio(A) = Gain(A)/SplitInfo(A)
- 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 (CART)

- If a data set *D* contains examples from *n* classes, gini index (impurity), *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 smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node
- Continuous attributes: use variance reduction

Computation of 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 D_1: {low, medium} and 4 in D_2

$$gini_{income \in \{low,medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right)$$

$$= 0.443$$

$$= Gini_{income \in \{high\}}(D).$$

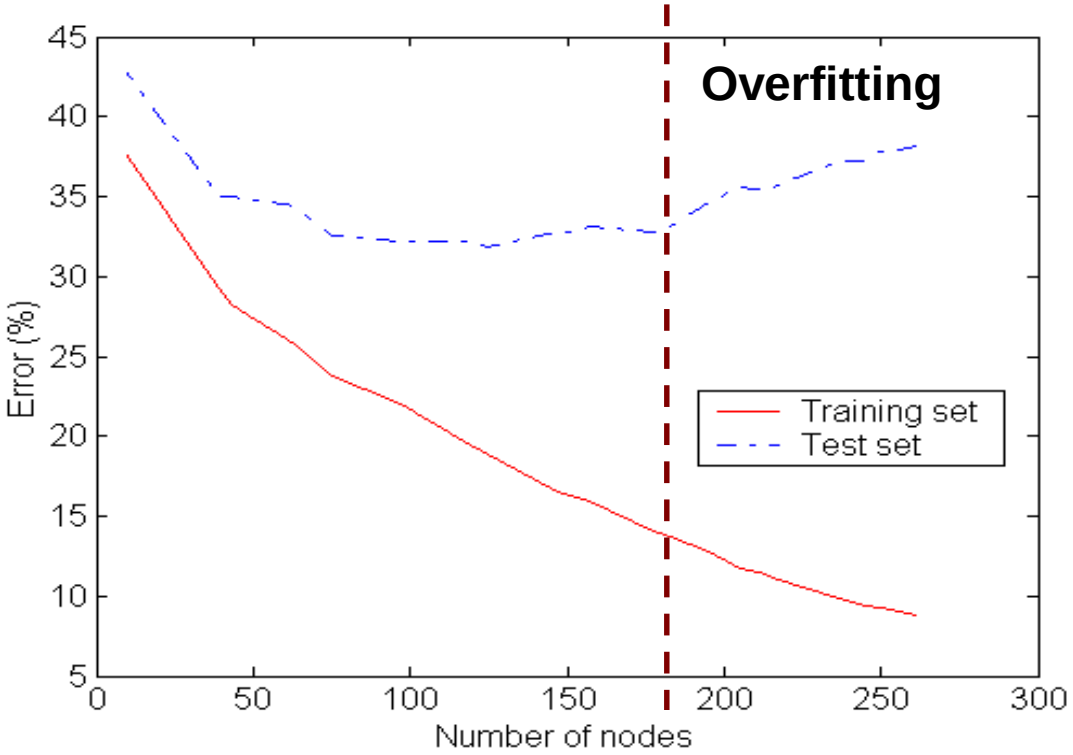
$Gini_{\{low,high\}}$ is 0.458; $Gini_{\{medium,high\}}$ is 0.450. Thus, split on the {low,medium} (and {high}) since it has the lowest Gini index

Comparing Attribute Selection Measures

- The three measures, in general, return good results but
  - **Information gain**:
    - biased towards multivalued attributes
  - **Gain ratio**:
    - Biased towards smaller unbalanced splits in which one partition is much smaller than the others
  - **Gini index**:
    - biased to multivalued attributes
    - tends to favor equal-sized partitions and purity in both partitions
- Decision tree can be considered a feature selection method

Overfitting

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies and noises
  - Poor accuracy for unseen sample
- Underfitting: when model is too simple, both training and test errors are large
- Bias-variance tradeoff (discussed later)



Tree Pruning

- Pruning 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
    - Use a set of data different from the training data to decide which is the "best pruned tree"
- Ensemble methods: random forest (discussed later)

Decision Tree: Comments

- Why is decision tree induction popular?
  - relatively faster learning speed (than other classification methods)
  - convertible to simple and easy to understand classification rules
  - comparable classification accuracy with other methods

Classification: Basic Concepts and Methods

- Classification: Basic Concepts
- Decision Tree
- **Bayes Classification Methods**
- Model Evaluation and Selection
- Ensemble Methods

Bayesian Classification: Why?

- 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

Review of Bayes' theorem

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

- Red jar: 10 chocolate + 30 plain
- Yellow jar: 20 chocolate + 20 plain
- Pick a jar, and then pick a cookie
- If it's a plain cookie, what's the probability the cookie is picked out of red jar?

Bayes' Theorem: Basics

- Bayes' Theorem:

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

- Informally, this can be viewed as: posteriori = likelihood x prior/evidence
  - Let **X** be a data sample ("*evidence*")
  - Let H be a *hypothesis* that X belongs to class C
  - Classification is to determine P(H|**X**) (i.e., *posteriori probability):* the probability that the hypothesis holds given the observed data sample **X**
  - P(H) (*prior probability*): the initial probability regardless of **X**
    - E.g., **X** will buy computer, regardless of age, income, …
  - P(**X**): probability that sample data is observed
  - P(**X**|H) (likelihood): probability of observing the sample **X**, given that the hypothesis holds
    - E.g., Given **X** will buy computer, the prob. that X is age 31..40, medium income

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

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

  needs to be maximized
- Practical difficulty: It requires initial knowledge of many probabilities, involving significant computational cost

Naïve Bayes Classifier

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

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

- 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 continuous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

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

and $P(x_k|C_i)$ is $g(x_k, \mu_{C_i}, \sigma_{C_i})$

Naïve Bayes Classifier: Training Dataset

Class:
C1:buys_computer = ‘yes’
C2:buys_computer = ‘no’

Data to be classified:
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 Bayes Classifier: An Example

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

- $P(C_i)$: P(buys_computer = "yes") = 9/14 = 0.643
  P(buys_computer = "no") = 5/14= 0.357
- Compute $P(X|C_i)$ for each class
  P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222
  P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6
  P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444
  P(income = "medium" | buys_computer = "no") = 2/5 = 0.4
  P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667
  P(student = "yes" | buys_computer = "no") = 1/5 = 0.2
  P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667
  P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4
- **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 Problem

- 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 Bayes 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., patients: age, family history, symptoms, diagnosis
- How to deal with these dependencies? Bayesian Belief Networks (discussed later)

Classification: Basic Concepts and Methods

- Classification: Basic Concepts
- Decision Tree
- Bayes Classification Methods
- **Model Evaluation and Selection**
- Ensemble Methods

Model Evaluation and Selection

- Evaluation metrics
- Evaluation methods
  - Holdout method, random subsampling
  - Cross-validation
  - Bootstrap
- Model selection
  - Bias-Variance tradeoff
  - Cost-benefit analysis and ROC Curves

Classifier Evaluation Metrics: Accuracy, Error Rate

- **Classifier Accuracy**: percentage of test set tuples that are correctly classified
- **Error rate** (*1 – accuracy)*: percentage of test tuples that are incorrectly classified

Classifier Evaluation Metrics: Confusion Matrix

Confusion Matrix:

Actual class\Predicted class	C_1	¬ C_1
C_1	**True Positives (TP)**	**False Negatives (FN)**
¬ 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

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

Classifier Evaluation Metrics: Sensitivity and Specificity

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

- **Sensitivity**: True Positive recognition rate
  - **Sensitivity = TP/P**
- **Specificity**: True Negative recognition rate
  - **Specificity = TN/N**

- **Class Imbalance Problem**:
  - One class may be *rare*, e.g. fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class

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

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

- **Precision**: positive predictive value (exactness)

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

- **Recall (sensitivity):** true positive recognition rate (completeness)

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

- ***F* measure (*F_1* or *F*-score)**: harmonic mean of precision and recall,

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

- ***F_β*:** weighted measure of precision and recall
  - assigns ß times weight to recall as to precision

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

Classifier Evaluation Metrics: Example

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	**90**	**210**	300	30.00 (*sensitivity)*
cancer = no	**140**	**9560**	9700	98.56 (*specificity)*
Total	230	9770	10000	96.40 (*accuracy*)

- *Precision* = 90/230 = 39.13% *Recall* = 90/300 = 30.00%

Model Evaluation and Selection

- Evaluation metrics
- Evaluation methods
  - Holdout method, random subsampling
  - Cross-validation
  - Bootstrap
- Model selection
  - Bias-Variance tradeoff
  - Cost-benefit analysis and ROC Curves

Evaluating Classifier

- **Holdout method**
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
- Random sampling: a variation of holdout
  - Repeat holdout k times, accuracy = avg. of the accuracies obtained

Evaluating Classifier

- **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

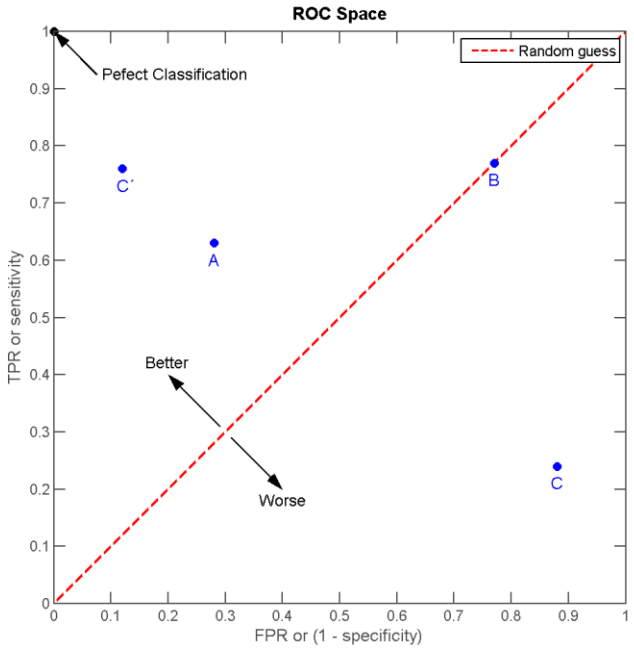
- **Bootstrap**
  - A resampling technique, works well with small data sets
  - Samples given data for training tuples uniformly *with replacement*
- **.632 boostrap**
  - A data set with *d* tuples is sampled *d* times with replacement, resulting in a training set of *d* samples.
  - About 63.2% of original data end up in the bootstrap, and remaining 36.8% form the test set ($(1 – 1/d)^d \approx e^{-1} = 0.368$)

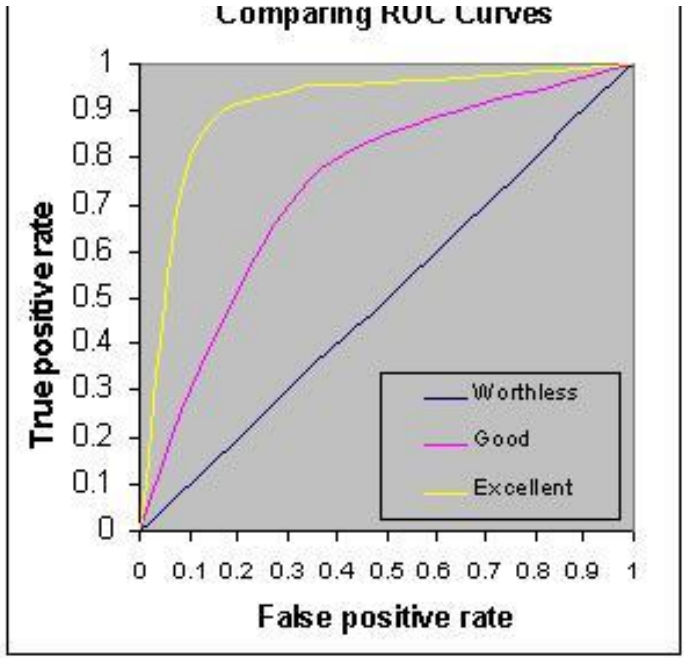
Classification of Class-Imbalanced Data Sets

- Class-imbalance problem: Rare positive example but numerous negative ones, e.g., medical diagnosis, fraud, oil-spill, fault, etc.
- Typical methods for imbalance data in 2-class classification:
  - **Oversampling**: re-sampling of data from positive class
  - **Under-sampling**: randomly eliminate tuples from negative class
  - **Threshold-moving**: moves the decision threshold so that the rare class tuples are easier to classify

Model Selection: ROC Curves

- **ROC** (Receiver Operating Characteristics) curves: for visual comparison of binary classification models
- Shows the trade-off between the true positive rate and the false positive rat
  - Y: true positive rate
  - X: false positive rate
- Perfect classification and line of no-discrimination
- The area under the ROC curve (Area Under Curve, AUC) is a measure of the accuracy of the model





Predicting from Samples

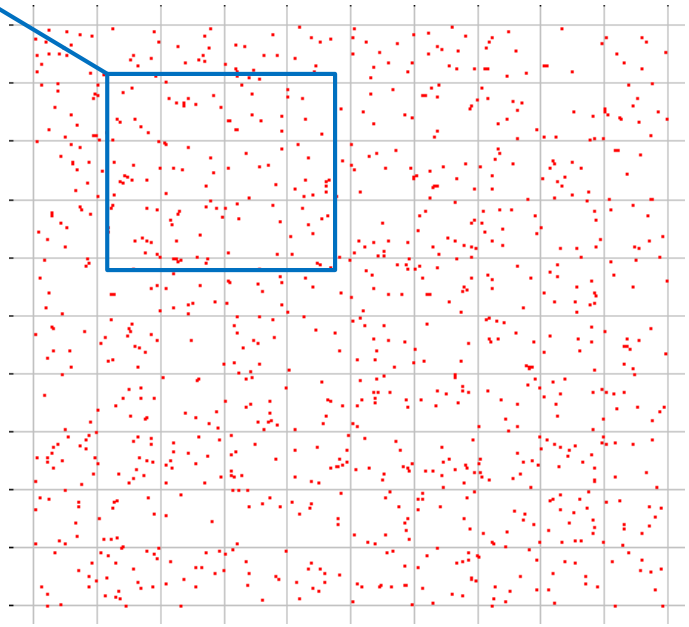
- Most datasets are **samples** from an **infinite population**.
- We are most interested in **models of the population**, but we have access only to a **sample** of it.

For datasets consisting of (X,y)

- features X + label y

a model is a prediction y = f(X)

We train on a training sample D
and we denote the model as $f_D(X)$



Bias and Variance

Our data-generated model $f_D(X)$ is a **statistical estimate** of the true function $f(X)$.

Because of this, its subject to bias and variance:

Bias: if we train models $f_D(X)$ on many training sets D, bias is the expected difference between their predictions and the true y's.

i.e. $$Bias = \mathrm{E}[f_D(X) - y]$$

E[] is taken over points X and datasets D

Variance: if we train models $f_D(X)$ on many training sets D, variance is the variance of the estimates:

$$Variance = \mathrm{E}\left[\left(f_D(X) - \bar{f}(X)\right)^2\right]$$

Where $\bar{f}(X) = \mathrm{E}[f_D(X)]$ is the average prediction on X.

Dart Example

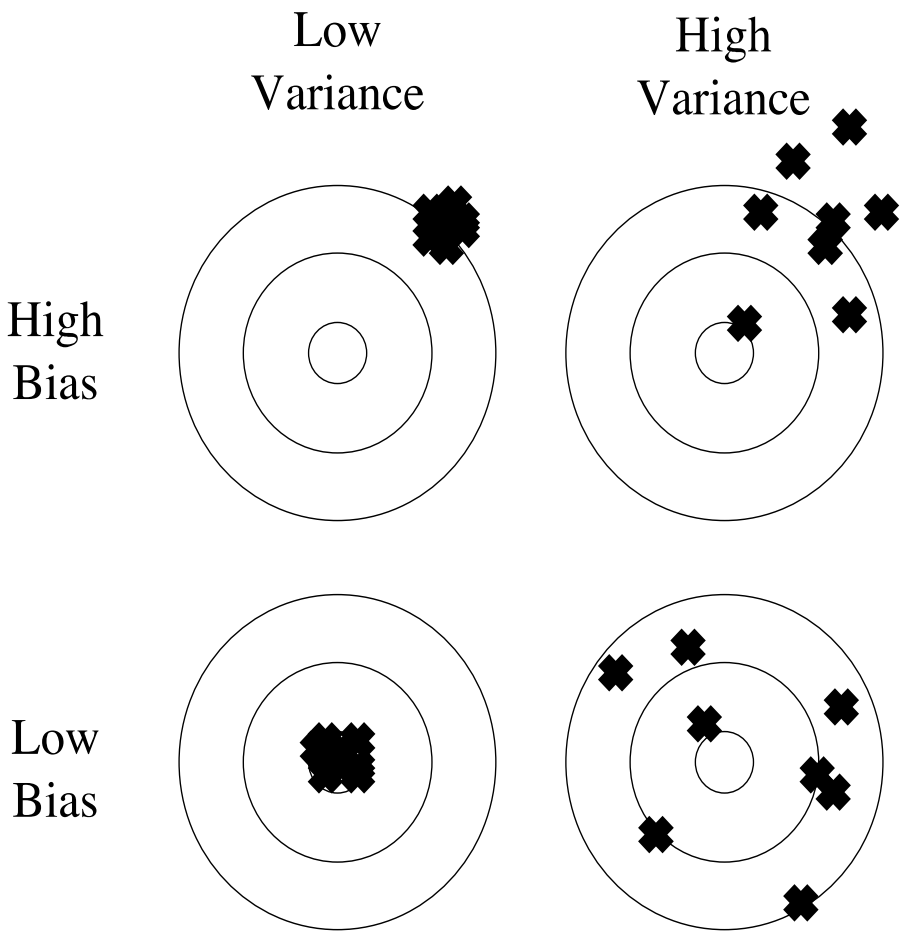


Figure 1: Bias and variance in dart-throwing.

Bias and Variance Tradeoff

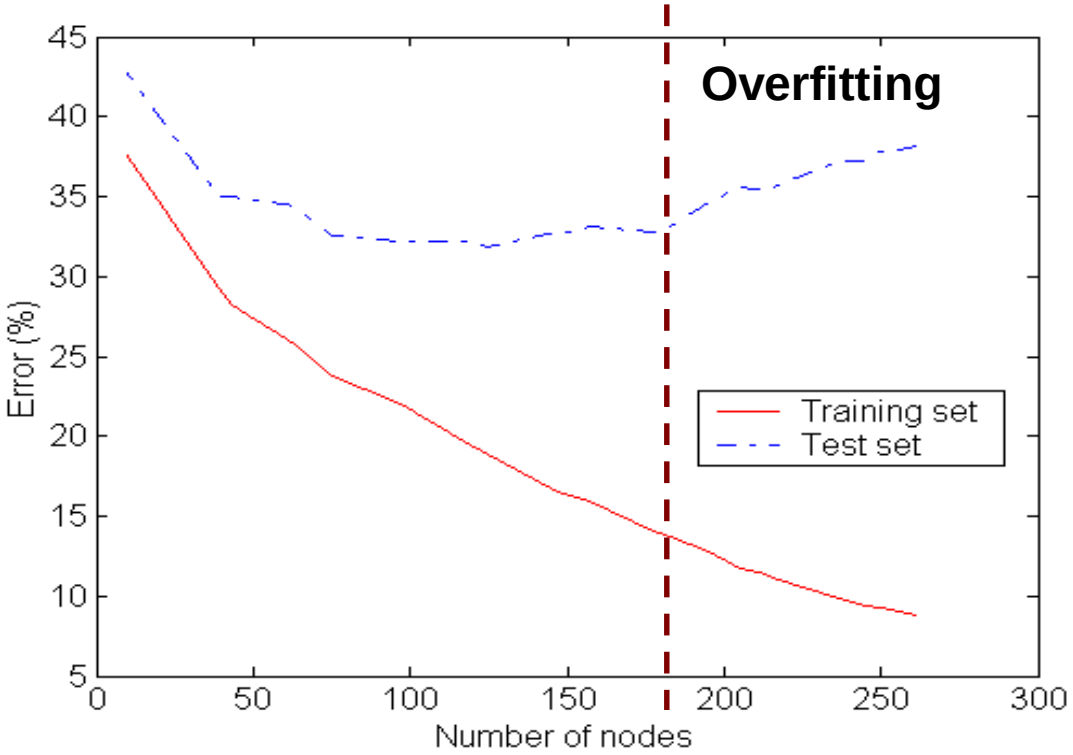
There is usually a bias-variance tradeoff caused by model complexity.

Complex models (many parameters) usually have lower bias, but higher variance.

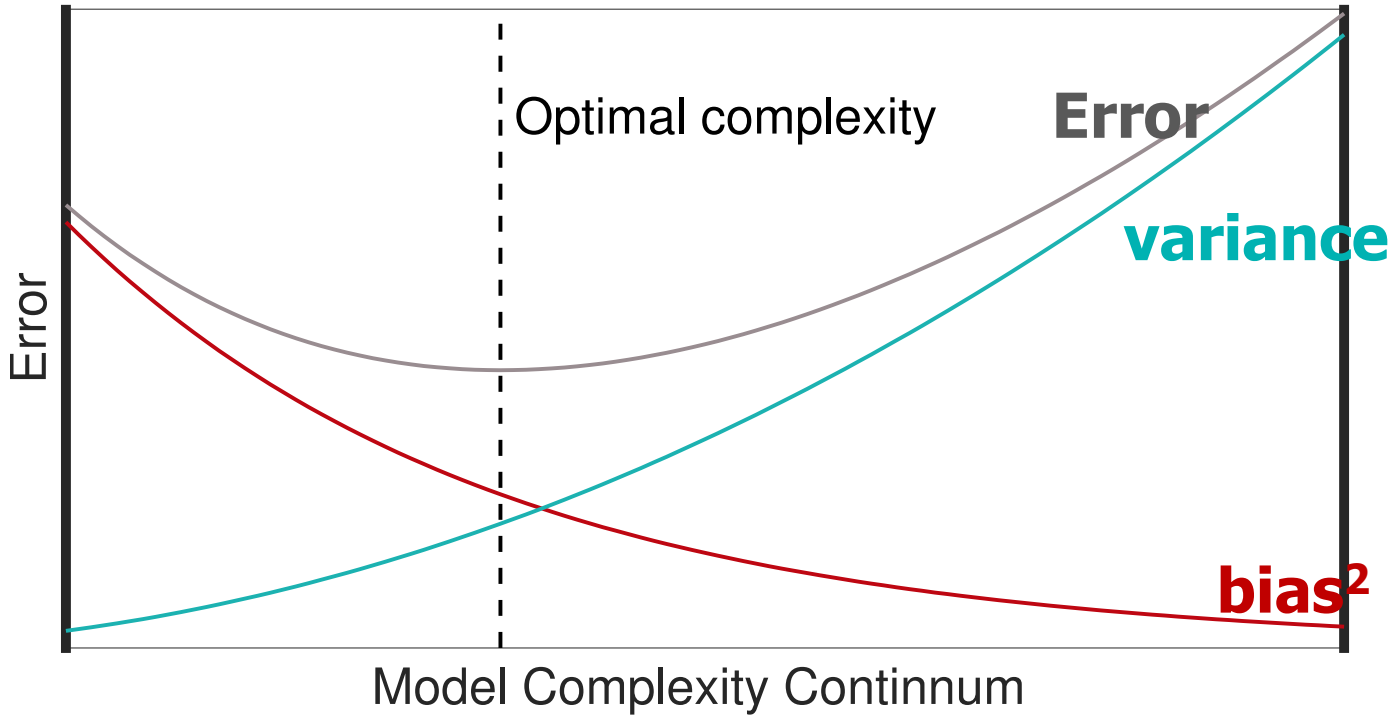
Simple models (few parameters) have higher bias, but lower variance.

Model Complexity

- Overfitting:
  - When model is too complex, good accuracy for training data but poor accuracy for unseen sample
- Underfitting:
  - when model is too simple, both training and test errors are large

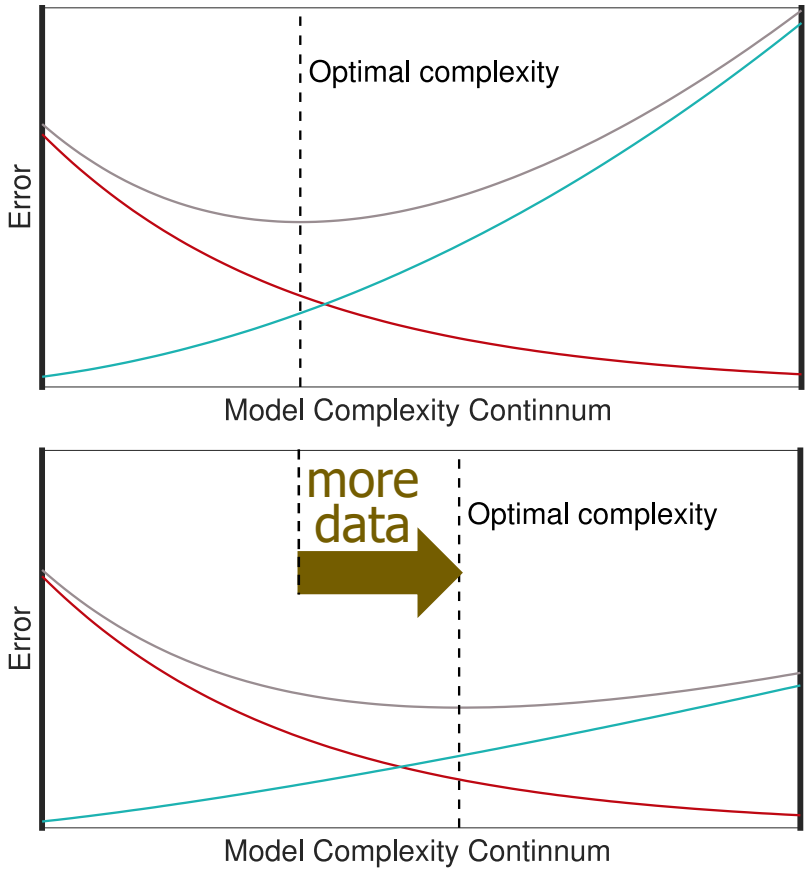


Bias-Variance Trade Off



Current Perspective In Machine Learning

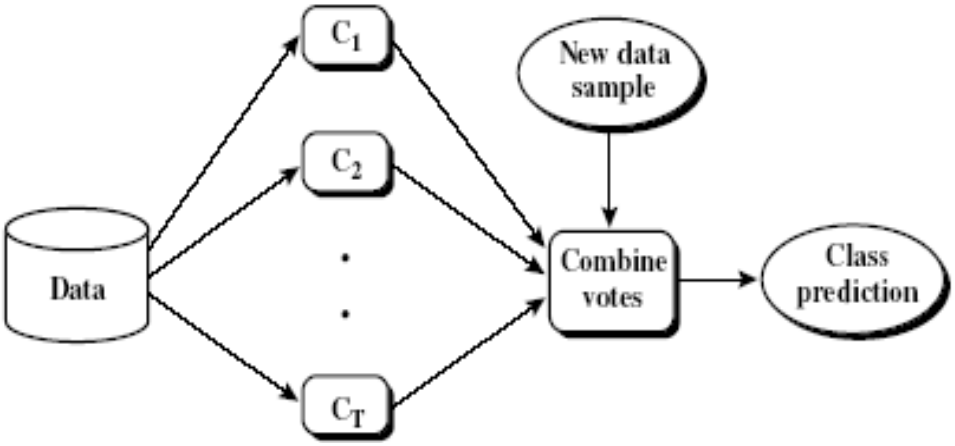
- We can learn complex domains using
  - low bias model (deep net)
  - Using more training data
  - Ensemble methods



Classification: Basic Concepts and Methods

- Classification: Basic Concepts
- Decision Tree
- Bayes Classification Methods
- Model Evaluation and Selection
- **Ensemble Methods**

Ensemble Methods: Increasing the Accuracy



- Ensemble methods
  - Use a combination of models to increase accuracy
  - 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: Bootstrap Aggregation

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
  - Given a set D of *d* tuples, at each iteration *i*, a training set D_i of *d* tuples is sampled with replacement from D (i.e., bootstrap)
  - A classifier model M_i is learned for each 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
- Often improved accuracy

Boosting

- Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy; improve the classifiers over time
- How boosting works?
  - **Weights** are assigned to each training tuple
  - A series of k classifiers is iteratively 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
- Comparing with bagging: Boosting tends to have greater accuracy, but it also risks overfitting the model to misclassified data

Adaboost (Freund and Schapire, 1997)

- Given a set of *d* class-labeled tuples, ($\mathbf{X_1}$, y_1), …, ($\mathbf{X_d}$, y_d)
- Initially, all the weights of tuples are set the same (1/d)
- Generate k classifiers in k rounds. At round i,
  - Tuples from D are sampled (with replacement) to form a training set D_i of the same size
  - Each tuple's chance of being selected is based on its weight
  - A classification model M_i is derived from D_i
  - Its error rate is calculated using D_i as a test set
  - If a tuple is misclassified, its weight is increased, o.w. it is decreased
- Error rate: err($\mathbf{X_j}$) is the misclassification error of tuple $\mathbf{X_j}$. Classifier M_i error rate is the sum of the weights of the misclassified tuples:

$$error(M_i) = \sum_{j}^{d} w_j \times err(\mathbf{X_j})$$

- The weight of classifier M_i's vote is

$$\log \frac{1 - error(M_i)}{error(M_i)}$$

AdaBoost

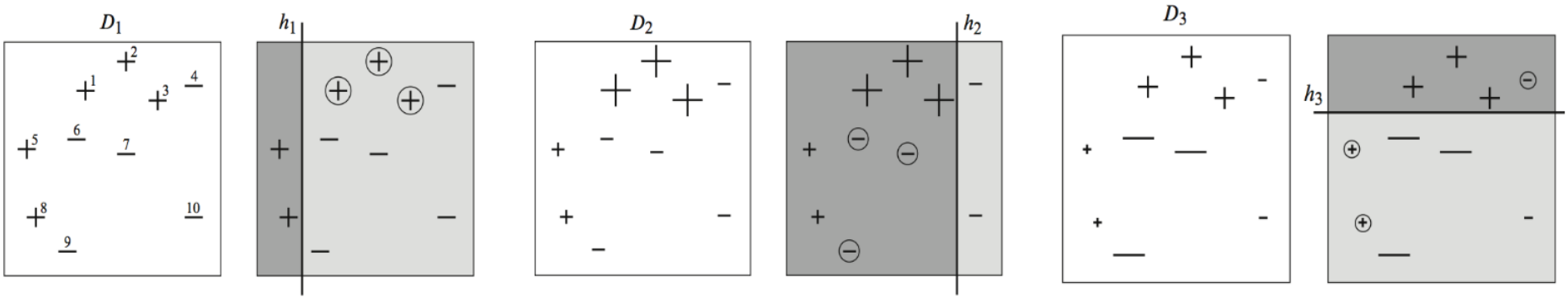
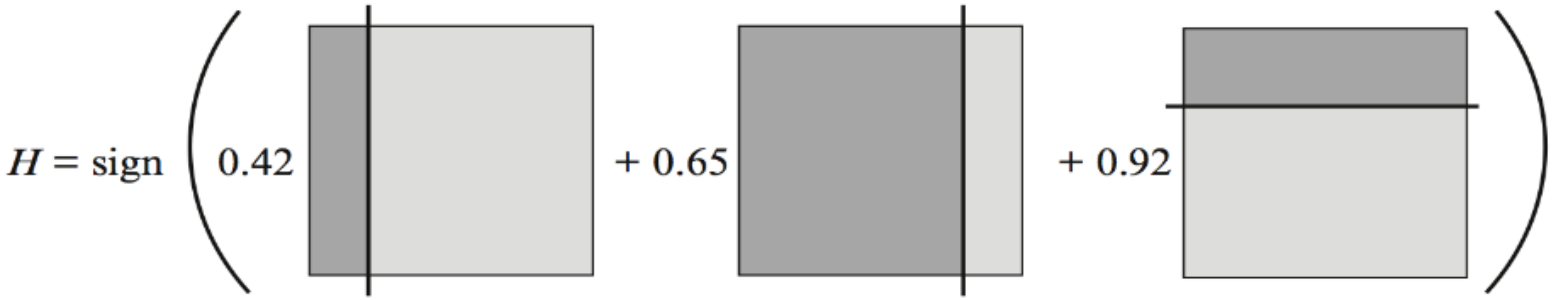
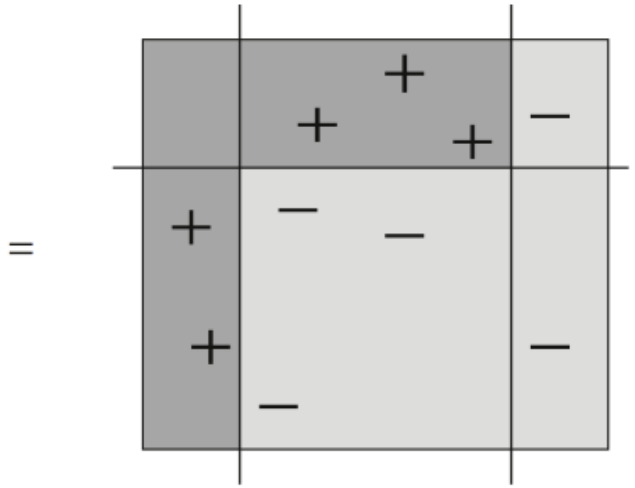


Figure: AdaBoost. Source: Figure 1.1 of [Schapire and Freund, 2012]

AdaBoost





Gradient Boosting

- Gradient Decent + Boosting
- Boosting: in each stage, introduce a classifier to overcome shortcomings of previous one
- AdaBoost: shortcomings are identified by high weight (misclassified) data points
- Gradient Boosting: shortcomings are identified by gradients of the loss function (generalize AdaBoost)

Random Forest (Breiman 2001)

- Bagging + decision tree
  - Each classifier in the ensemble is a *decision tree* classifier
  - During classification, each tree votes and the most popular class is returned
- Two Methods to construct Random Forest:
  - Forest-RI (*random input selection*): Randomly select, at each node, F attributes as candidates for the split at the node.
  - Forest-RC (*random linear combinations*)*:* Creates new attributes (or features) that are a linear combination of the existing attributes

Gradient Boosted Tree

- Gradient boosting + decision tree
- Generally perform better than random forest but requires more parameter tuning