Classification: Basic Concepts

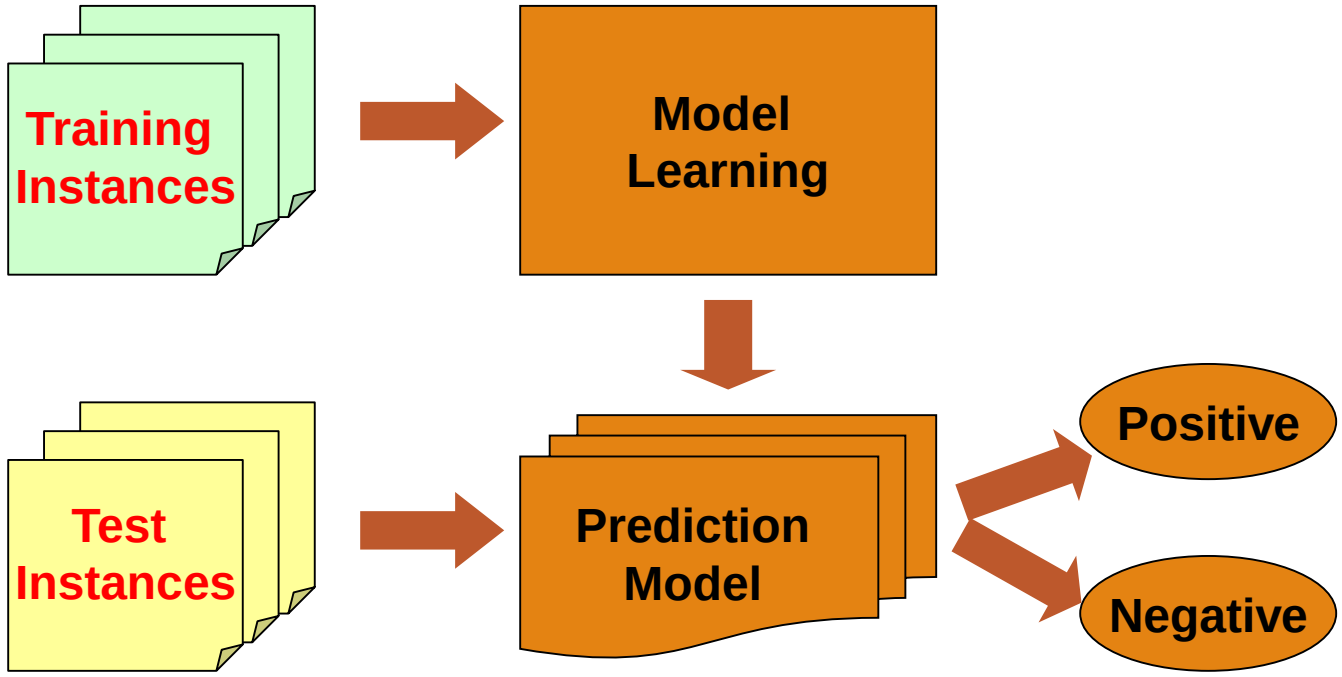
- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Lazy Learners (or learning from your neighbors)
- Linear Classifiers
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy
- Summary

Supervised vs. Unsupervised Learning (1)

- Supervised learning (classification)
  - Supervision: The training data such as observations or measurements are accompanied by **labels** indicating the classes which they belong to
  - New data is classified based on the models built from the training set

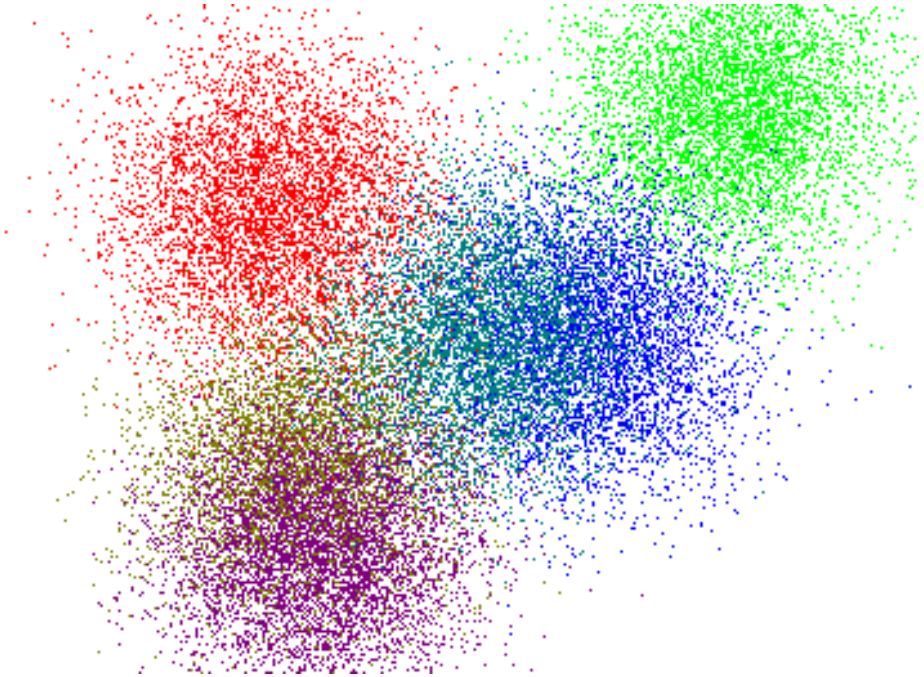
Training Data with class label:

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No



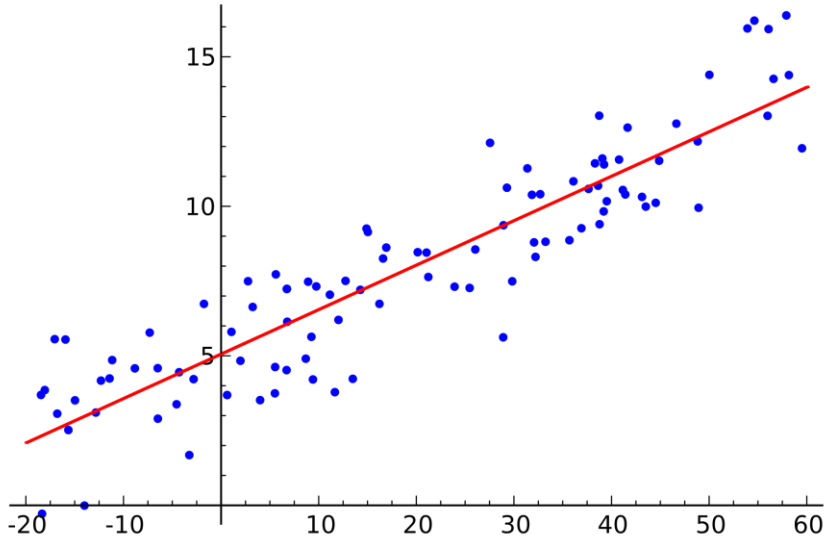
Supervised vs. Unsupervised Learning (2)

- Unsupervised learning (clustering)
  - The class labels of training data are **unknown**
  - Given a set of observations or measurements, establish the possible existence of classes or clusters in the data



Prediction Problems: Classification vs. Numeric Prediction

- Classification
  - Predict **categorical class labels** (discrete or nominal)
  - Construct a model based on the training set and the **class labels** (the values in a classifying attribute) and use it in classifying new data
- Numeric prediction
  - Model **continuous-valued functions** (i.e., predict unknown or missing values)



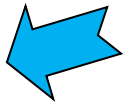
Classification—Model Construction, Validation and Testing

- **Model Construction and Training**
  - Model: Represented as decision trees, rules, mathematical formulas, or other forms
  - Assumption: Each sample belongs to a predefined class /**class label**
  - Training Set: The set of samples used for model construction

Classification—Model Construction, Validation and Testing

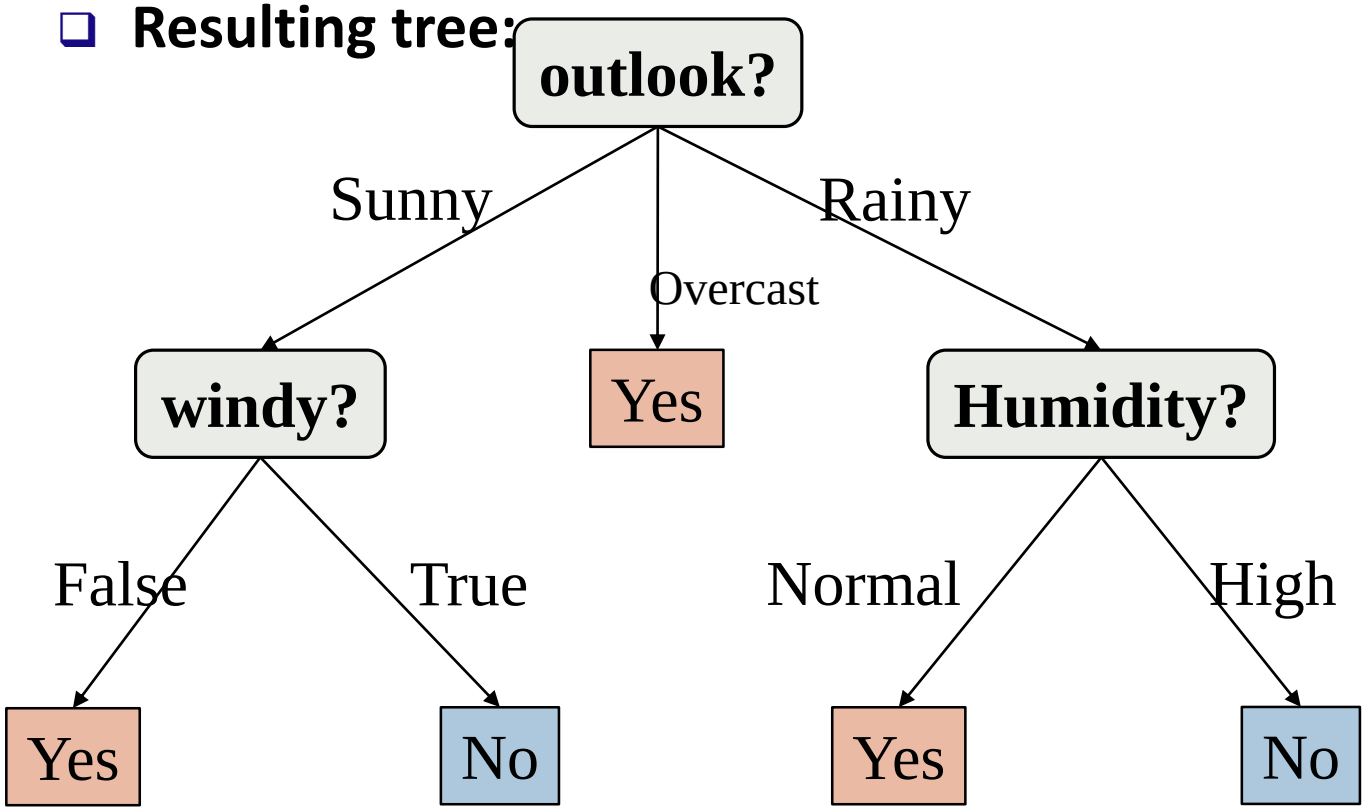
- **Model Validation and Testing**:
  - **Test:** Estimate accuracy of the model
    - The known label of test sample VS. the classified result from the model
    - *Accuracy:* % of test set samples that are correctly classified by the model
    - Test set is independent of training set
  - **Validation**: If *the test set* is used to select or refine models, it is called **validation** (or development) **(test) set**
- **Model Deployment:** If the accuracy is acceptable, use the model to classify new data

Chapter 6. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction 
- Bayes Classification Methods
- Lazy Learners (or learning from your neighbors)
- Linear Classifiers
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy
- Summary

Decision Tree Induction: An Example

- **Decision tree construction**:
  - A top-down, recursive, divide-and-conquer process
- **Resulting tree:**



Training data set: Play Golf?

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

https://www.saedsayad.com/decision_tree.htm

Decision Tree Induction: Algorithm

- Basic algorithm
  - Tree is constructed in a **top-down, recursive, divide-and-conquer manner**
  - At start, all the training examples are at the root
  - Examples are partitioned recursively based on selected attributes
  - On each node, attributes are selected based on the training examples on that node, and a heuristic or statistical measure (e.g., **information gain, Gini index**)

Decision Tree Induction: Algorithm

- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning
  - There are no samples left
- Prediction
  - **Majority voting** is employed for classifying the leaf

How to Handle Continuous-Valued Attributes?

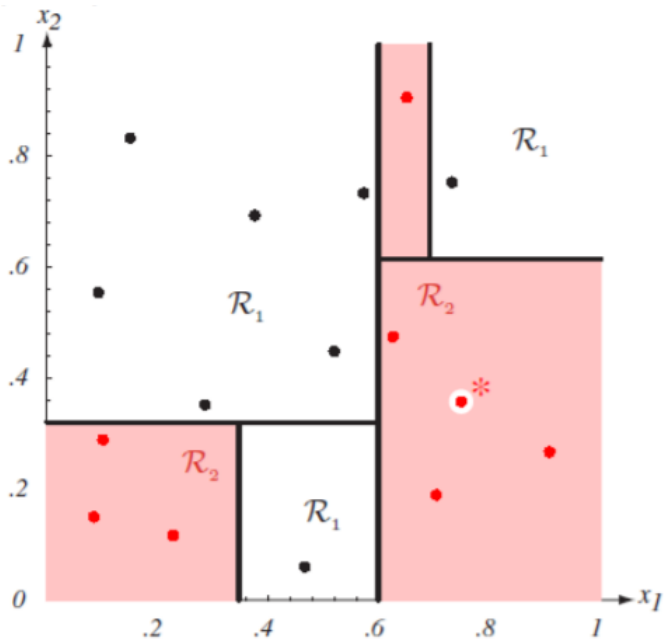
- Method 1: Discretize continuous values and treat them as categorical values
  - E.g., age: < 20, 20..30, 30..40, 40..50, > 50
- Method 2: Determine the ***best split point*** for continuous-valued attribute A
  - Sort:, e.g. 15, 18, 21, 22, 24, 25, 29, 31, …
  - *Possible split point:* $(a_i+a_{i+1})/2$
    - e.g., (15+18)/2 = 16.5, 19.5, 21.5, 23, 24.5, 27, 30, …
  - The point with the *maximum information gain* for A is selected as the **split-point** for A
- Split: Based on split point P
  - The set of tuples in D satisfying A ≤ P vs. those with A > P

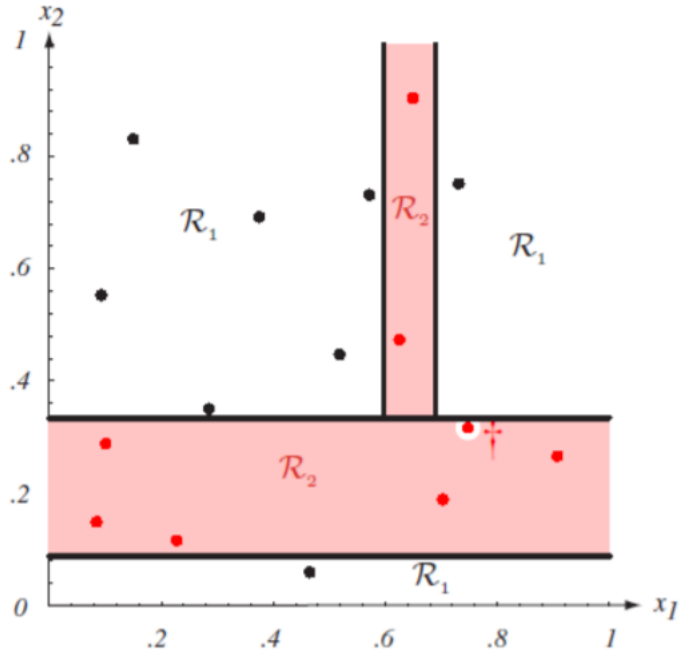
Pro's and Con's

- Pro's
  - Easy to explain (even for non-expert)
  - Easy to implement (many software)
  - Efficient
  - Can tolerant missing data
  - White box
  - No need to normalize data
  - Non-parametric: No assumption on data distribution, no assumption on attribute independency
  - Can work on various attribute types

Con's

- Con's
  - Unstable. Sensitive to noise
  - Accuracy may be not good enough (depending on your data)
  - The optimal splitting is NP. Greedy algorithms are used
  - Overfitting





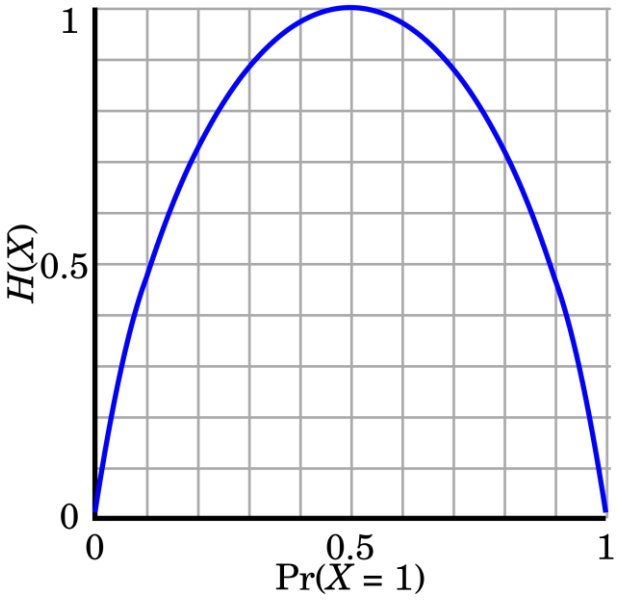
Splitting Measures: Information Gain

- Entropy (Information Theory)
  - A measure of uncertainty associated with a random number
  - Calculation: For a discrete random variable Y taking 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 → higher uncertainty
    - Lower entropy → lower uncertainty
- Conditional entropy

$$H(Y|X) = \sum_{x} p(x) H(Y|X = x)$$



m = 2

Information Gain: An Attribute Selection Measure

- Select the attribute with the highest information gain (used in typical decision tree induction algorithm: ID3/C4.5)
- Let p_i be the 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:

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

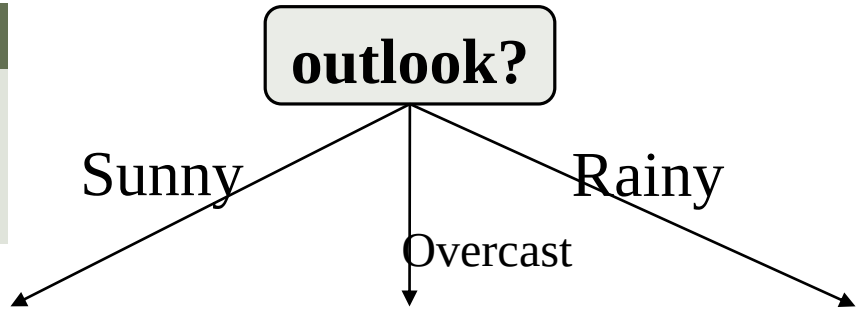
- Information gained by branching on attribute A

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

Example: Attribute Selection with Information Gain

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

outlook	yes	no	I(yes, no)
rainy	2	3	0.971
overcast	4	0	0
sunny	3	2	0.971



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

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

$\frac{5}{14}I(2,3)$ means "outlook=rainy" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence $Gain(outlook) = Info(D) - Info_{outlook}(D) = 0.246$

Example: Attribute Selection with Information Gain

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Temp	Yes	No	I(Yes, No)
Hot	2	2	?
Mild	4	2	?
Cool	3	1	?

Windy	Yes	No	I(Yes, No)
True	?	?	?
False	?	?	?

Humidity	Yes	No	I(Yes, No)
Normal	6	1	?
High	3	4	?

Similarly, we can get

$$Gain(Temp) = 0.029,$$
$$Gain(humidity) = 0.151,$$
$$Gain(Windy) = 0.048$$

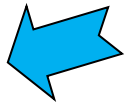
Gain Ratio: A Refined Measure for Attribute Selection

- Information gain measure is biased towards attributes with a large number of values (e.g. ID)
- Gain ratio: Overcomes the problem (as a 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)
  - The attribute with the maximum gain ratio is selected as the splitting attribute
  - Gain ratio is used in a popular algorithm C4.5 (a successor of ID3) by R. Quinlan
- Example
  - $\text{SplitInfo}_{\text{temp}}(\text{D}) = -\frac{4}{14}\log_2\frac{4}{14} - \frac{6}{14}\log_2\frac{6}{14} - \frac{4}{14}\log_2\frac{4}{14} = 1.557$
  - GainRatio(temp) = 0.029/1.557 = 0.019

Chapter 6. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods 
- Lazy Learners (or learning from your neighbors)
- Linear Classifiers
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy
- Summary

Bayes' Theorem: Basics

- Total probability Theorem:

$$p(B) = \sum_i p(B|A_i)p(A_i)$$

- Bayes' Theorem:

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

posteriori probability — What we should choose

likelihood — What we just see

prior probability — What we knew previously

- **X:** a data sample ("*evidence*")
- H: X belongs to class C

Prediction can be done based on Bayes' Theorem:

Classification is to derive the maximum posteriori

Bayes' Theorem Example: Picnic Day

- The morning is cloudy ☹
- What is the chance of rain? P(Rain | Cloud) = ?
- 50% of all rainy days start off cloudy. P(Cloud | Rain) = 50%
- Cloudy mornings are common (40% of days start cloudy) P(Cloud) = 40%
- This is usually a dry month (only 3 of 30 days tend to be rainy) P(Rain) = 10%

P(Rain | Cloud) = P(Rain) P(Cloud | Rain) / P(Cloud) = 10% * 50% / 40% = 12.5%

- The chance of rain is probably not as high as expected ☺
- Bayes' Theorem allows us to tell back and forth between posterior and likelihood

(*e.g.*, P(Rain | Cloud) and P(Cloud | Rain)), tests the reality, which is the most important trick in Bayesian Inference

Naïve Bayes Classifier: Making a Naïve Bayes Assumption

- Based on the Bayes' Theorem, we can derive a Bayes Classifier to compute the posterior probability of classifying an object X to a class C
  - $P(C|X) \propto P(X|C)P(C) = P(x1|C)P(x2|x1,C)...P(xn|x1,...,C)P(C)$
- A naïve bayes assumption to simplify the complex dependencies: *features are conditionally independent, given the class label!*
  - $P(C|X) \propto P(X|C)P(C) \approx P(x1|C)P(x2|C)...P(xn|C)P(C)$
- Super efficient: each feature only conditions on the class (boils down to sample counting)
- Achieves surprisingly comparable performance

Naïve Bayes Classifier: Categorical vs. Continuous Valued Features

- If feature x_k is categorical, $p(x_k = v_k|C_i)$ is the # of tuples in C_i with $x_k = v_k$, divided by $|C_{i,D}|$ (# of tuples of C_i in D)

$$p(X|C_i) = \prod_k p(x_k|C_i) = p(x_1|C_i) \cdot p(x_2|C_i) \cdots\cdots p(x_n|C_i)$$

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

$$p(x_k = v_k|C_i) = N\left(x_k|\mu_{C_i}, \sigma_{C_i}\right) = \frac{1}{\sqrt{2\pi}\sigma_{C_i}} e^{-\frac{\left(x-\mu_{C_i}\right)^2}{2\sigma^2}}$$

Naïve Bayes Classifier Example 1: Training Dataset

Class:
play golf= ‘yes’
play golf = ‘no’

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Naïve Bayes Classifier Example *P(Yes / Sunny)*

$$P(x \mid c) = P(Sunny \mid Yes) = 3/9 = 0.33$$

Frequency Table		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3

Likelihood Table		Play Golf		
		Yes	No	
Outlook	Sunny	3/9	2/5	5/14
	Overcast	4/9	0/5	4/14
	Rainy	2/9	3/5	5/14
		9/14	5/14	

$$P(x) = P(Sunny) = 5/14 = 0.36$$

$$P(c) = P(Yes) = 9/14 = 0.64$$

Posterior Probability:

$$P(c \mid x) = P(Yes \mid Sunny) = 0.33 \times 0.64 \div 0.36 = 0.60$$

Naïve Bayes Classifier Example: P(No | Sunny)

Frequency Table		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
		9	5	14

$$P(x \mid c) = P(Sunny \mid No) = 2/5 = 0.4$$

$$P(x) = P(Sunny) = 5/14 = 0.36$$

$$P(c) = P(No) = 5/14 = 0.36$$

Posterior Probability:

$$P(c \mid x) = P(No \mid Sunny) = 0.40 \times 0.36 \div 0.36 = 0.40$$

Naïve Bayes Classifier Example: Likelihood Tables

Frequency Table

		Play Golf	
		Yes	No
Outlook	Sunny	**3**	**2**
	Overcast	**4**	**0**
	Rainy	**2**	**3**

		Play Golf	
		Yes	No
Humidity	High	**3**	**4**
	Normal	**6**	**1**

		Play Golf	
		Yes	No
Temp.	Hot	**2**	**2**
	Mild	**4**	**2**
	Cool	**3**	**1**

		Play Golf	
		Yes	No
Windy	False	**6**	**2**
	True	**3**	**3**

Likelihood Table

		Play Golf	
		Yes	No
Outlook	Sunny	**3/9**	**2/5**
	Overcast	**4/9**	**0/5**
	Rainy	**2/9**	**3/5**

		Play Golf	
		Yes	No
Humidity	High	**3/9**	**4/5**
	Normal	**6/9**	**1/5**

		Play Golf	
		Yes	No
Temp.	Hot	**2/9**	**2/5**
	Mild	**4/9**	**2/5**
	Cool	**3/9**	**1/5**

		Play Golf	
		Yes	No
Windy	False	**6/9**	**2/5**
	True	**3/9**	**3/5**

Naïve Bayes Classifier Example: Likelihood Tables

Outlook	Temp	Humidity	Windy	Play
Rainy	Cool	High	True	?

$$P(Yes \mid X) = P(Rainy \mid Yes) \times P(Cool \mid Yes) \times P(High \mid Yes) \times P(True \mid Yes) \times P(Yes) \;/\; P(x)$$

$$P(Yes \mid X) = \underbrace{2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14}_{P(x)} = 0.00529 \rightarrow 0.2 = \frac{0.00529}{0.02057 + 0.00529}$$

$$P(No \mid X) = P(Rainy \mid No) \times P(Cool \mid No) \times P(High \mid No) \times P(True \mid No) \times P(No) \;/\; P(x)$$

$$P(No \mid X) = \underbrace{3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14}_{P(x)} = 0.02057 \rightarrow 0.8 = \frac{0.02057}{0.02057 + 0.00529}$$

$P(x) = P(x \mid yes) * P(yes) + P(x \mid no) * P(no)$

Avoiding the Zero-Probability Problem

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

$$\mathrm{p}(\mathrm{X}|C_i) = \prod_k p(x_k|C_i) = p(x_1|C_i) \cdot p(x_2|C_i) \cdots\cdots p(x_n|C_i)$$

- Example. Suppose a dataset with 1,000 tuples:

  income = low (0), income= medium (990), and income = high (10)

- Use **Laplacian correction** (or Laplacian estimator)
  - *Adding 1 (or a small integer) to each case*

    Prob(income = low) = 1/(1000 + 3)

    Prob(income = medium) = (990 + 1)/(1000 + 3)

    Prob(income = high) = (10 + 1)/(1000 + 3)

$$\hat{P}(w_i \mid c) = \frac{count(w_i,c)+1}{\sum_{w \in V}\left(count(w,c)+1\right)}$$

$$= \frac{count(w_i,c)+1}{\left(\sum_{w \in V} count(w,c)\right) + |V|}$$

  - The "corrected" probability estimates are close to their "uncorrected" counterparts

Naïve Bayes Classifier: Strength vs. Weakness

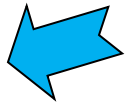
- Strength
  - Performance: A *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
  - Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct—prior knowledge can be combined with observed data

Naïve Bayes Classifier: Strength vs. Weakness

- Weakness
  - Assumption: attributes conditional independence, therefore loss of accuracy
    - E.g., Patient's Profile: (age, family history),
    - Patient's Symptoms: (fever, cough),
    - Patient's Disease: (lung cancer, diabetes).
    - Dependencies among these cannot be modeled by Naïve Bayes Classifier
  - How to deal with these dependencies?

    Use Bayesian Belief Networks (chapter 7)

Chapter 6. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Lazy Learners (or learning from your neighbors) 
- Linear Classifiers
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy
- Summary

Lazy vs. Eager Learning

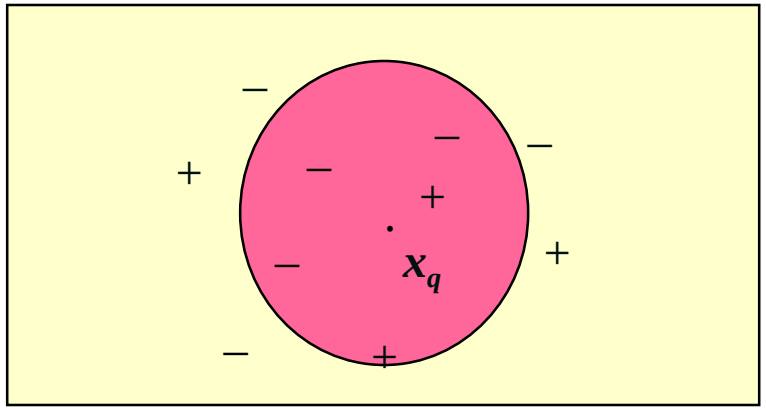
- Lazy vs. eager learning
  - **Lazy learning** (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - **Eager learning** (the above discussed methods): Given a set of training tuples, constructs a classification model before receiving new (e.g., test) data to classify
- Lazy: less time in training but more time in predicting
- Accuracy
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form an implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space

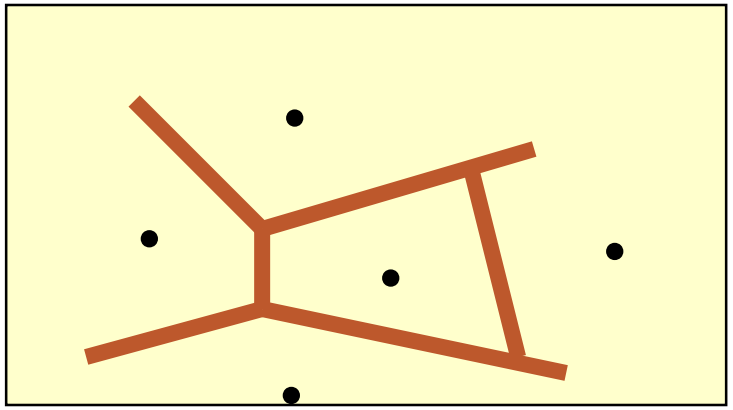
Lazy Learner: Instance-Based Methods

- Instance-based learning:
  - Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified
- Typical approaches
  - *k*-nearest neighbor approach
    - Instances represented as points in a Euclidean space.
  - Locally weighted regression
    - Constructs local approximation
  - Case-based reasoning
    - Uses symbolic representations and knowledge-based inference

The *k*-Nearest Neighbor Algorithm

- All instances correspond to points in the n-D space
- The nearest neighbor are defined in terms of Euclidean distance, dist($\mathbf{X}_1$, $\mathbf{X}_2$)
- Target function could be discrete- or real- valued
- For discrete-valued, *k*-NN returns the most common value among the *k* training examples nearest to x_q
- Vonoroi diagram: the decision surface induced by 1-NN for a typical set of training examples





Discussion on the *k*-NN Algorithm

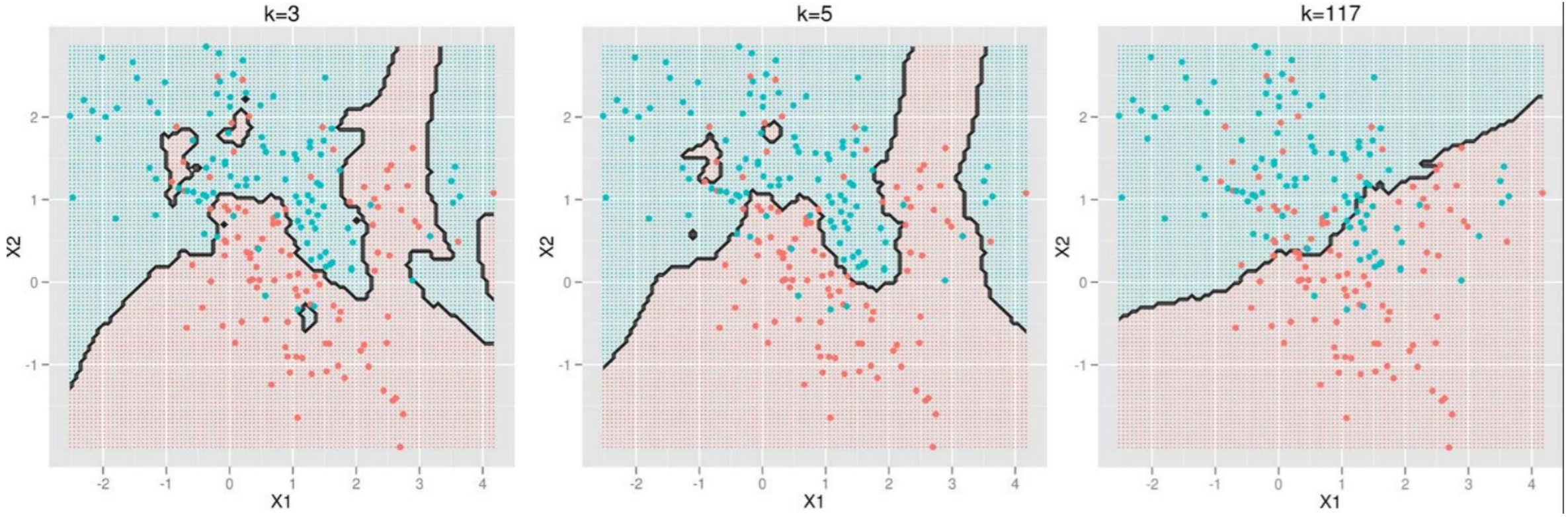
- *k*-NN for real-valued prediction for a given unknown tuple
  - Returns the mean values of the *k* nearest neighbors
- Distance-weighted nearest neighbor algorithm
  - Weight the contribution of each of the *k* neighbors according to their distance to the query x_q
    - Give greater weight to closer neighbors

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

- Robust to noisy data by averaging *k*-nearest neighbors
- Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes
  - To overcome it, axes stretch or elimination of the least relevant attributes

Selection of k for kNN

- The number of neighbors k
  - Small k: overfitting (high var., low bias)
  - Big k: bringing too many irrelevant points (high bias, low var.)

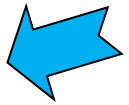


http://scott.fortmann-roe.com/docs/BiasVariance.html

Case-Based Reasoning (CBR)

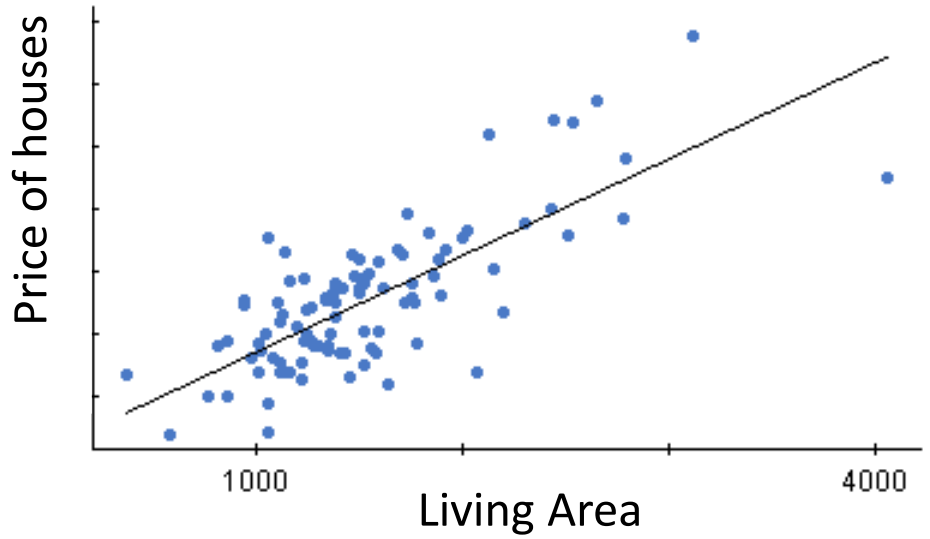
- **CBR**: Uses a database of problem solutions to solve new problems
- Store symbolic description (tuples or cases)—not points in a Euclidean space
- Applications: Customer-service (product-related diagnosis), legal ruling
- Methodology
  - Instances represented by rich symbolic descriptions (e.g., function graphs)
  - Search for similar cases, multiple retrieved cases may be combined
  - Tight coupling between case retrieval, knowledge-based reasoning, and problem solving
- Challenges
  - Find a good similarity metric
  - Indexing based on syntactic similarity measure, and when failure, backtracking, and adapting to additional cases

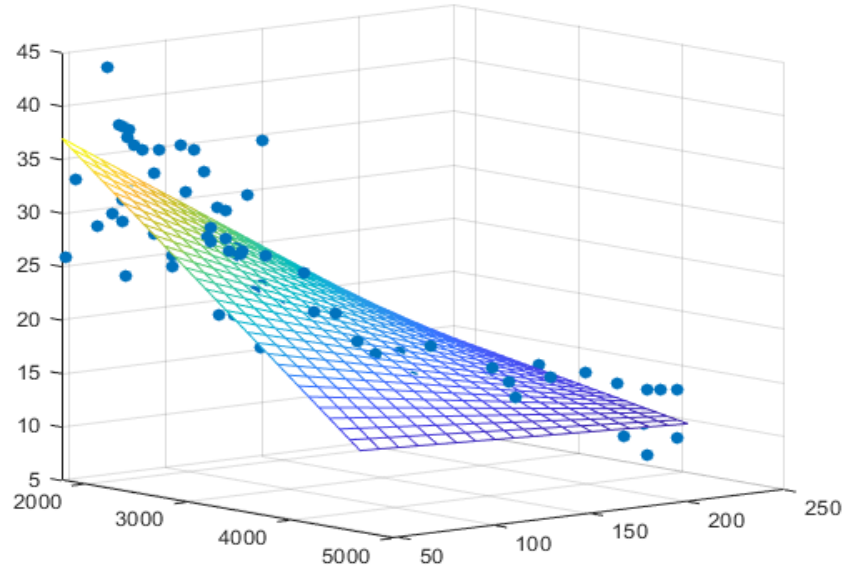
Chapter 6. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Lazy Learners (or learning from your neighbors)
- Linear Classifiers 
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy
- Summary

Linear Regression Problem: Example

- Mapping from independent attributes to **continuous value**: x => y
- {living area} => Price of the house
- {college; major; GPA} => Future Income





Linear Regression Problem: Model

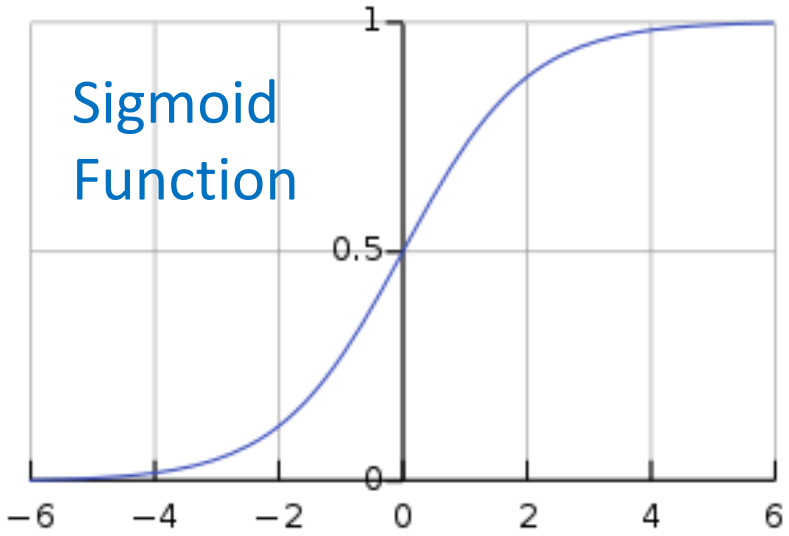
- Linear regression
  - Data: n independent objects
    - Observed Value: $y_i, i = 1,2,3, \cdots, n$
    - p-dimensional attributes: $x_i = \left(x_{i1}, x_{i2}, \cdots, x_{ip}\right)^T,\ i = 1,2,3 \cdots, n$
  - Model:
    - Weight vector: $w = \left(w_1, w_2, \cdots, w_p\right)$
    - $y_i = w^T x_i + b$
  - The weight vector w and bias b are the model parameter learnt by data

Linear Regression Model: Solution

- Least Square Method
  - Cost / Loss Function: $\mathrm{L}(w,b) = \Sigma_{i=1}^{n}(y_i - wx_i - b)^2$
  - Optimization Goal: $\underset{(w,b)}{\operatorname{argmin}}\ \mathrm{L}(w,b) = \Sigma_{i=1}^{n}(y_i - wx_i - b)^2$
  - Closed-form solution:
    - $w = \frac{\Sigma_{i=1}^{n} x_i(y_i - \bar{y})}{\Sigma_{i=1}^{n} x_i^2 - n(\Sigma_{i=1}^{n} x_i)^2}$ $\qquad b = \frac{1}{n}\Sigma_{i=1}^{n}(y_i - wx_i)$

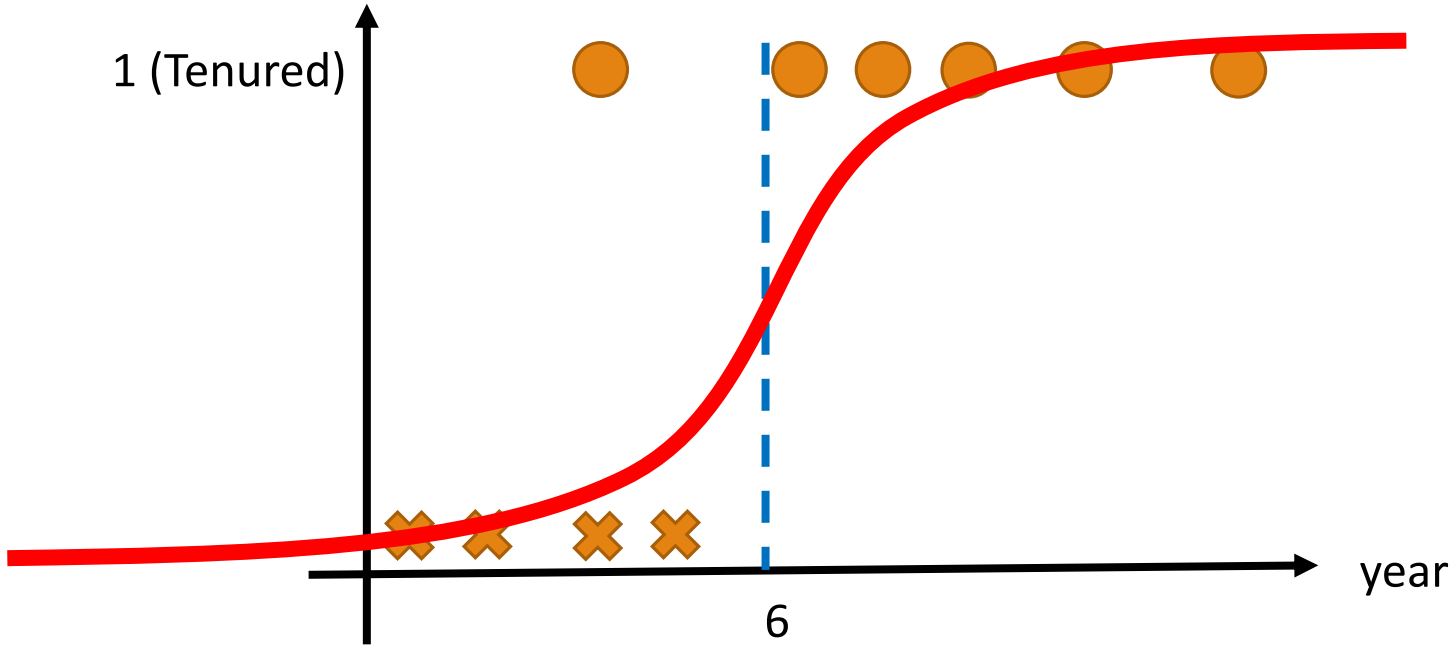
Logistic Regression: General Ideas

- How to solve “classification” problems by regression?
- Key idea of Logistic Regression
  - We need to transform the real value Y into a probability value ∈ [0,1]
- Sigmoid function (differentiable function) :
  - $\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{e^z+1}$
  - Projects $(-\infty, +\infty)$ to $[0, 1]$
  - Not only LR uses this function, but also neural network, deep learning
- The projected value changes sharply around zero point
- Notice that $\ln \frac{y}{1-y} = w^T x + b$

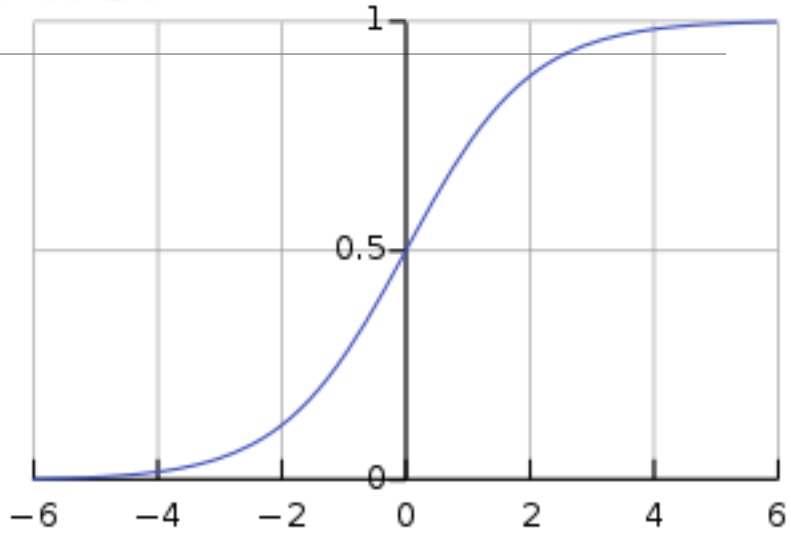


Logistic Regression: An Example

- Suppose we only consider the year as feature
  - Data points are converted by sigmoid function ("activation" function)



Logistic Regression: Model



- The prediction function to learn
- Probability that Y=1:
  - $p(Y = 1 \,|X = x; \boldsymbol{w}) \;= \; Sigmoid(w_0 + \sum_{i=1}^{n} w_i \cdot x_i)$
  - $\boldsymbol{w} = (w_0, w_1, w_2, \ldots, w_n)$ are the parameters
- A single data object with attributes x_i and class label y_i
  - Suppose the probability of $p(\hat{y}_i = 1|x_i, w) = p_i$, then $p(\hat{y}_i = 0|x_i, w) = 1 - p_i$
  - $p(\hat{y}_i = y_i) = p_i^{y_i}(1 - p_i)^{1-y_i}$
- Maximum Likelihood Estimation
  - $L = \Pi_i p_i^{y_i}(1 - p_i)^{1-y_i} = \Pi_i \left(\frac{\exp(w^T x_i)}{1+\exp(w^T x_i)}\right)^{y_i} \left(\frac{1}{1+\exp(w^T x_i)}\right)^{1-y_i}$

Logistic Regression: Optimization

- Maximum Likelihood Estimation
  - $L = \Pi_i p_i^{y_i}(1-p_i)^{1-y_i} = \Pi_i \left(\frac{\exp(w^T x_i)}{1+\exp(w^T x_i)}\right)^{y_i} \left(\frac{1}{1+\exp(w^T x_i)}\right)^{1-y_i}$
- Log likelihood:

$$l(w) = \sum_{i=1}^{N} y_i \log p(Y=1|X=x_i;\boldsymbol{w}) + (1-y_i)\log\big(1-p(Y=1|X=x_i;\boldsymbol{w})\big)$$

$$= \sum_{i=1}^{N} y_i x_i^T \boldsymbol{w} - \log(1+\exp(\boldsymbol{w}^T x_i))$$

- There's no closed form solution
  - Gradient Descent/Ascent

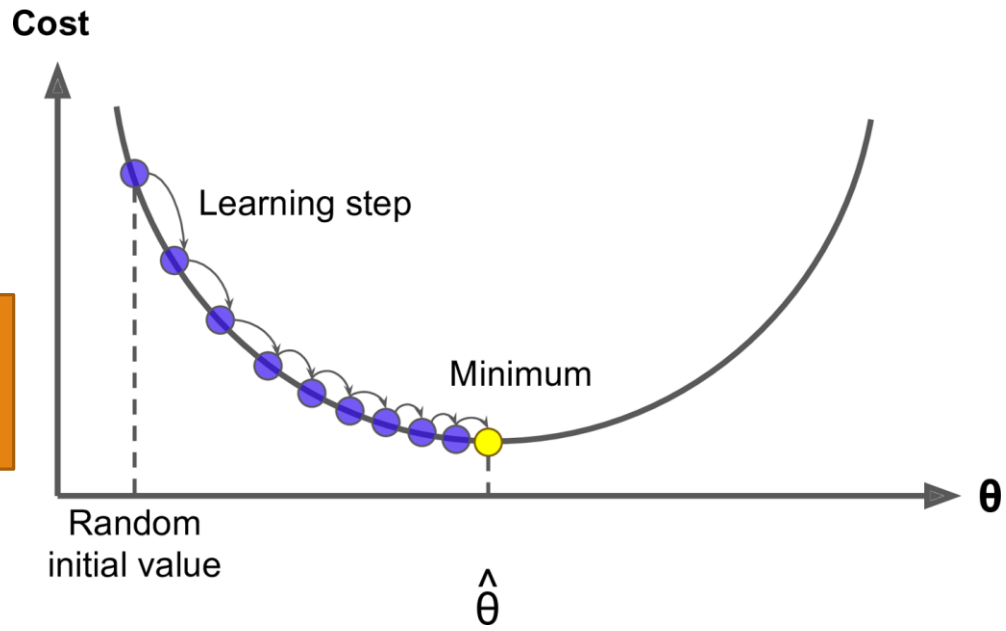
Gradient Descent

- Gradient Descent is an iterative optimization algorithm for finding the minimum of a function (e.g., the negative log likelihood)
- For a function F(x) at a point **a**, F(x) *decreases fastest* if we go in the direction of the negative gradient of **a**

Step size

$$\mathbf{a}_{n+1} = \mathbf{a}_n - \gamma \nabla F(\mathbf{a}_n)$$

When the gradient is zero, we arrive at the local minimum



Gradient Descent

[footnote: It is gradient ascent for maximizing log likelihood]

$$l(w) = \ln \prod_l P(Y^l \mid X^l, w)$$

$$= \sum_l (Y^l - 1)(w_0 + \sum_{i=1}^{n} w_i X_i^l) - \ln(1 + \exp(-(w_0 + \sum_{i=1}^{n} w_i X_i^l)))$$

$$\frac{\partial l(w)}{\partial w_i} = \sum_l X_i^l (Y^l - P(Y^l = 1 \mid X^l, w))$$

Iterate until change < threshold

For all i,

$$w_i \leftarrow w_i + \eta \sum_l X_i^l (Y^l - P(Y^l = 1 \mid X^l, w))$$

What is the intuition?

Note we have switched the index *i* in the previous slides to *l, and use i to refer to different components of weight vector w*

Gradient Descent

[footnote: It is gradient ascent for maximizing log likelihood]

Iterate until change < threshold

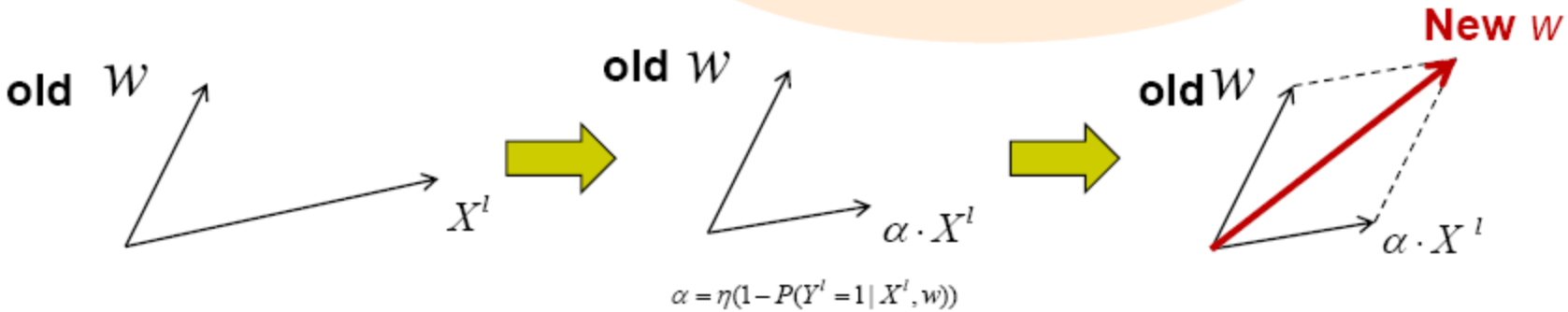
For all i,

$$w_i \leftarrow w_i + \eta \sum_l X_i^l (Y^l - P(Y^l = 1 \mid X^l, w))$$

Feature, true label, current prediction

If $Y^l = 1$

$$w \leftarrow w + \eta \times X^l (1 - P(Y^l = 1 \mid X^l, w))$$



Gradient Descent

[footnote: It is gradient ascent for maximizing log likelihood]

Iterate until change < threshold

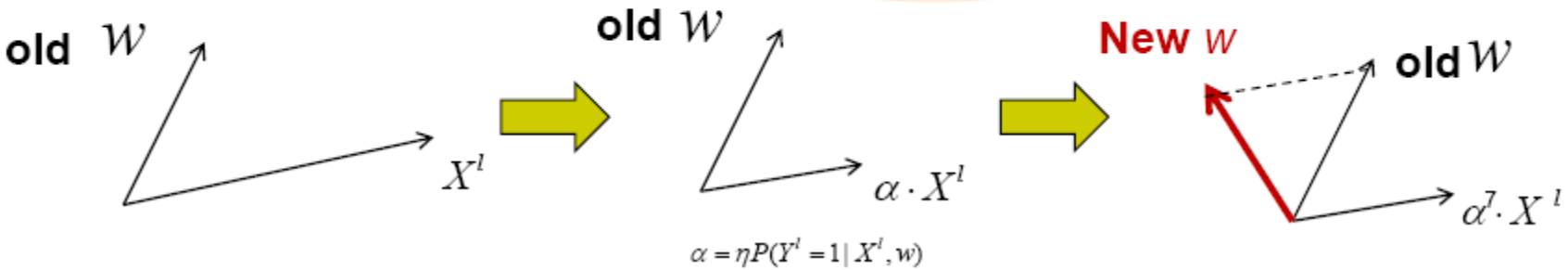
For all i,

$$w_i \leftarrow w_i + \eta \sum_l X_i^l (Y^l - P(Y^l = 1 \mid X^l, w))$$

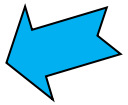
Feature, true label, current prediction

If $Y^l = 0$

$$w \leftarrow w - \eta \times X^l P(Y^l = 1 \mid X^l, w)$$



Chapter 6. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Lazy Learners (or learning from your neighbors)
- Linear Classifiers
- 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 **validation test set** of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating a classifier's accuracy
  - Holdout method
  - Cross-validation
  - Bootstrap (not covered)
- Comparing classifiers:
  - ROC Curves

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)**

- In a confusion matrix w. *m* classes, $CM_{i,j}$ indicates # of tuples in class *i* that were labeled by the classifier as class *j*
  - May have extra rows/columns to provide totals
- **Example of Confusion Matrix:**

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

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

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

Real-world truth

Predictions

- **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

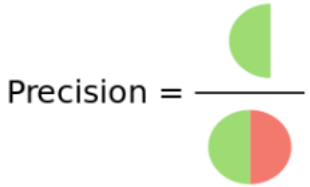
- **Class imbalance problem**
  - One class may be *rare*
    - E.g., fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class
  - Measures handle the class imbalance problem
    - **Sensitivity** (recall): True positive recognition rate
      - **Sensitivity = TP/P**
    - **Specificity**: True negative recognition rate
      - **Specificity = TN/N**

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

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

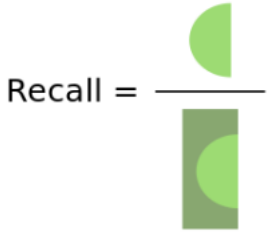
- **Precision**: Exactness: what % of tuples that the classifier labeled as positive are actually positive?

$$\text{P} = \text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

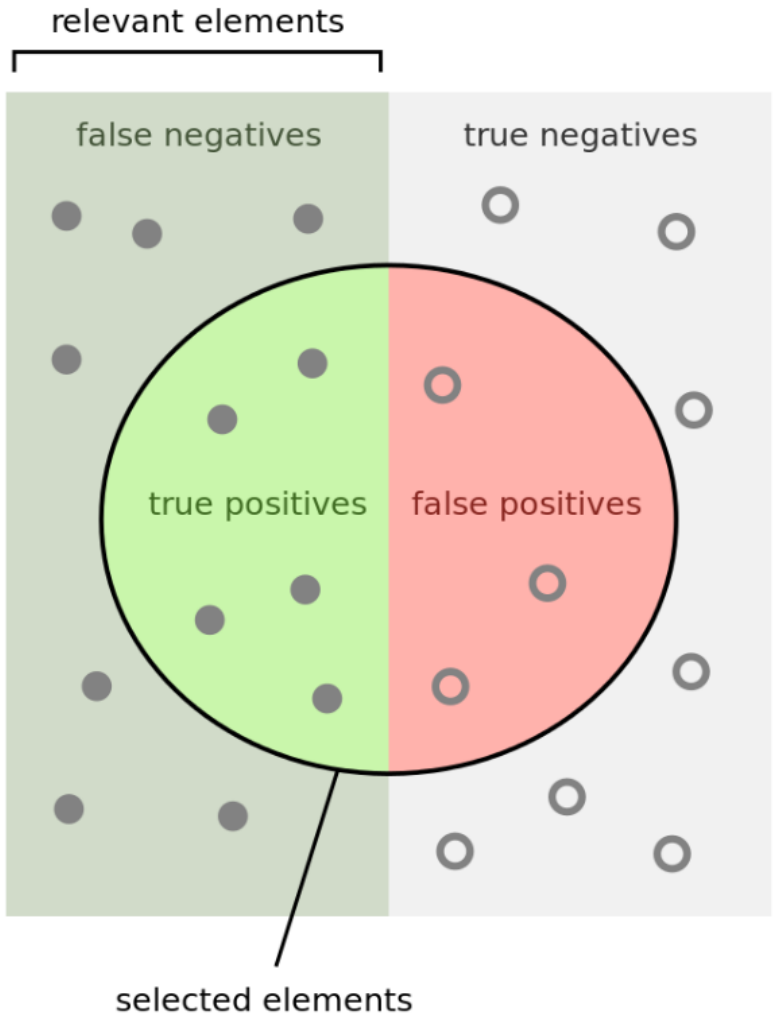


- **Recall:** Completeness: what % of positive tuples did the classifier label as positive?

$$\text{R} = \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$



  - Range: [0, 1]



https://en.wikipedia.org/wiki/Precision_and_recall

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

- The "inverse" relationship between precision & recall
- ***We want one number to say if a classifier is good or not***
- ***F* measure (or *F*-score):** harmonic mean of precision and recall
  - In general, it is the weighted measure of precision & recall

$$F_\beta = \frac{1}{\alpha \cdot \frac{1}{P} + (1 - \alpha) \cdot \frac{1}{R}} = \frac{(\beta^2 + 1)P * R}{\beta^2 P + R}$$

Assigning β times as much weight to recall as to precision

  - ***F1-measure (balanced F-measure)***
    - That is, when β = 1,

$$F_1 = \frac{2P * R}{P + R}$$

Classifier Evaluation Metrics: Example

- Use the same confusion matrix, calculate the measure just introduced

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

  - Sensitivity =
  - Specificity =
  - Accuracy =
  - Error rate =
  - Precision =
  - Recall =
  - F1 =

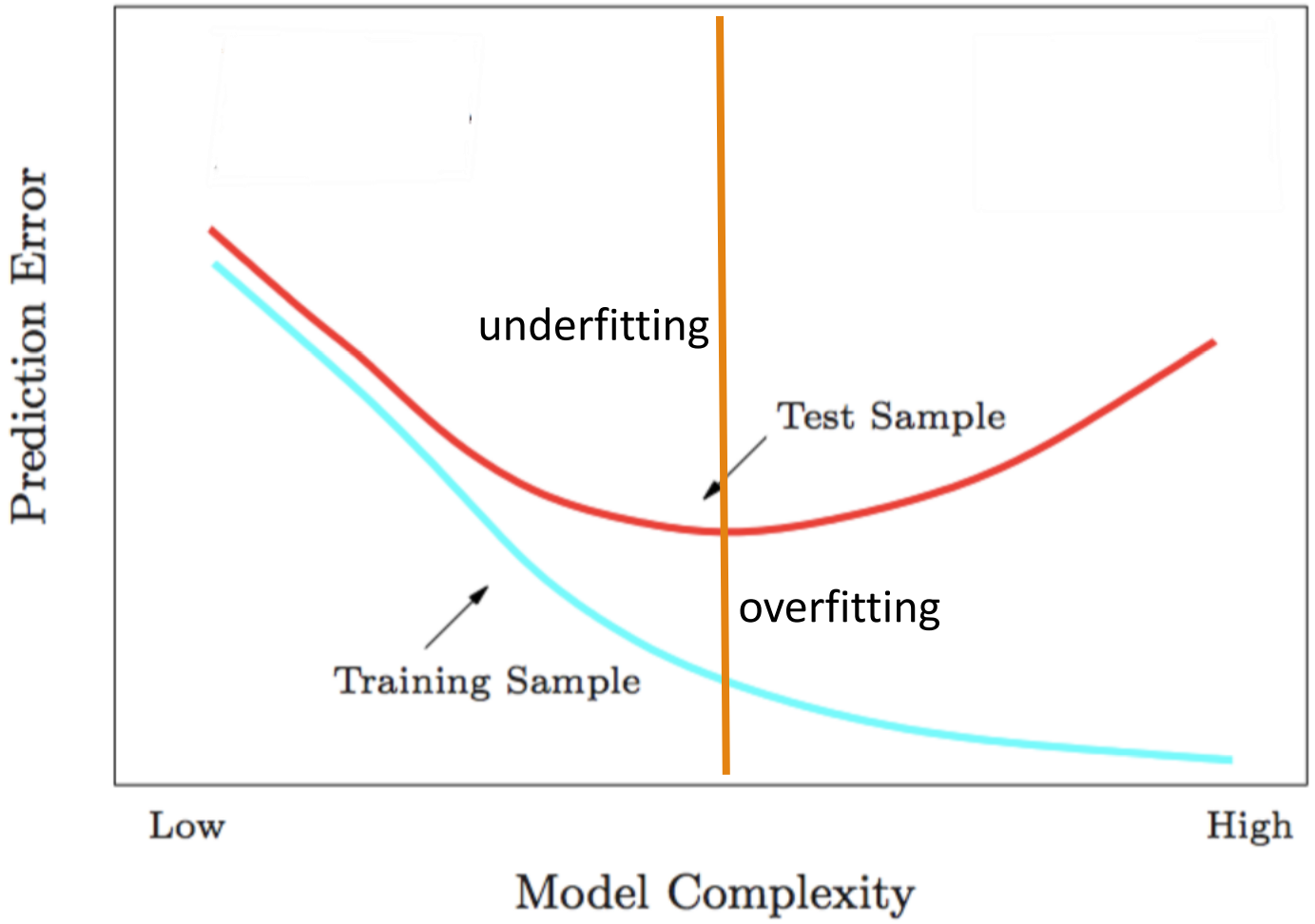
Classifier Evaluation Metrics: Example

- Use the same confusion matrix, calculate the measure just introduced

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

- Sensitivity = TP/P = 90/300 = 30%
- Specificity = TN/N = 9560/9700 = 98.56%
- Accuracy = (TP + TN)/All = (90+9560)/10000 = 96.50%
- Error rate = (FP + FN)/All = (140 + 210)/10000 = 3.50%
- Precision = TP/(TP + FP) = 90/(90 + 140) = 90/230 = 39.13%
- Recall = TP/ (TP + FN) = 90/(90 + 210) = 90/300 = 30.00%
- F1 = 2 P × R /(P + R) = 2 × 39.13% × 30.00%/(39.13% + 30%) = 33.96%

Training Error VS Testing Error



59

Classifier Evaluation: Holdout

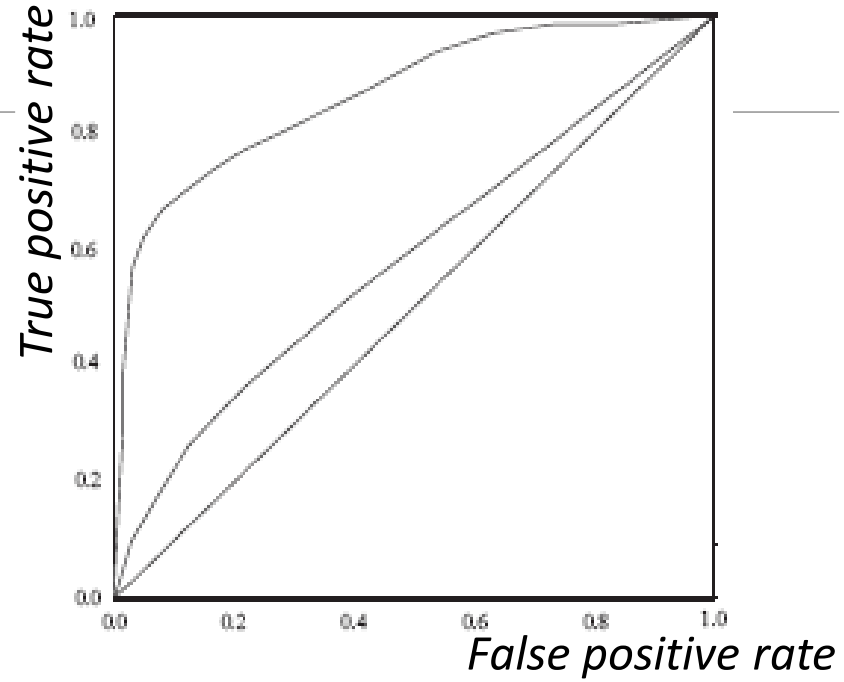
- **Holdout method**
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Repeated random sub-sampling validation: a variation of holdout
    - Repeat holdout k times, accuracy = avg.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

Classifier Evaluation: Cross-Validation

- **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 distribution, in each fold is approximately the same as that in the initial data

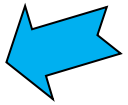
Model Selection: ROC Curves

- **ROC** (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Originated from signal detection theory
- Shows the trade-off between the true positive rate and the false positive rate
- The area under the ROC curve (**AUC**: Area Under 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



- Vertical axis represents the true positive rate (TP/P)
- Horizontal axis rep. the false positive rate (FP/N)
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0

Chapter 6. Classification: Basic Concepts

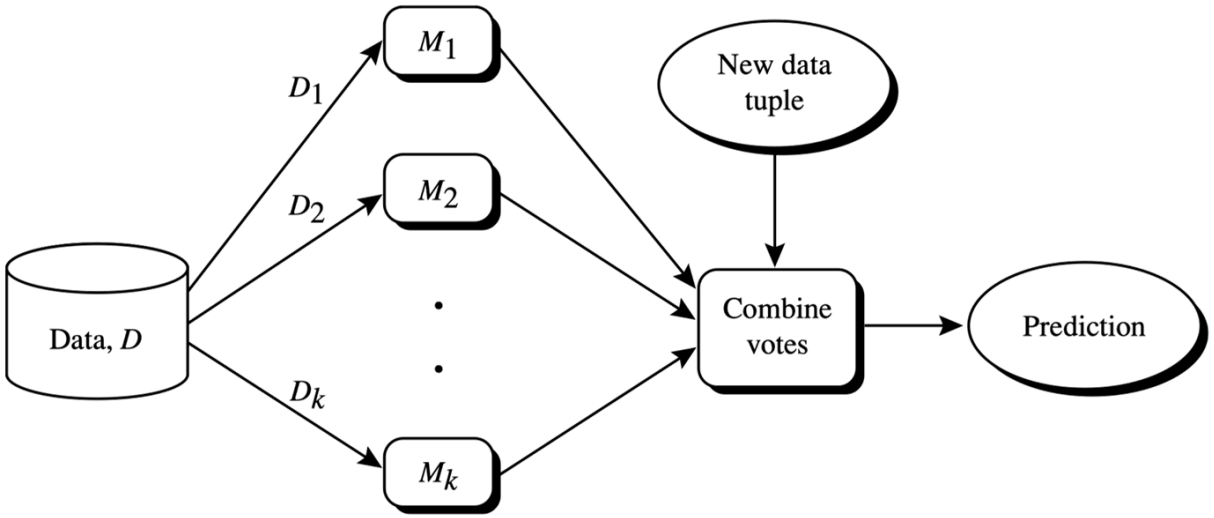
- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Lazy Learners (or learning from your neighbors)
- Linear Classifiers
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy 
- Summary

Techniques to Improve Classification Accuracy

- Introducing Ensemble Methods
- Bagging
- Boosting
- Random Forests
- Imbalanced Classification

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*



Ensemble Methods: Increasing the Accuracy

- What are the requirements to generate an improved model?
  - Example: majority voting

Base model performance →

Ensemble performance →

	x_1	x_2	x_3
M_1	✓	✓	✗
M_2	✗	✓	✓
M_3	✓	✗	✓
Voting Ensemble	✓	✓	✓

Case 1:
Ensemble has positive effect

	x_1	x_2	x_3
M_1	✓	✓	✗
M_2	✓	✓	✗
M_3	✓	✓	✗
Voting Ensemble	✓	✓	✗

Case 2:
Ensemble has no effect

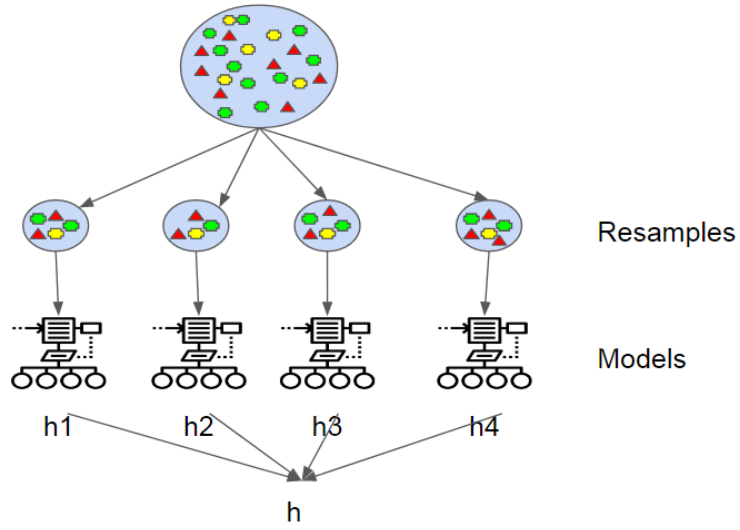
	x_1	x_2	x_3
M_1	✓	✗	✗
M_2	✗	✓	✗
M_3	✗	✗	✓
Voting Ensemble	✗	✗	✗

Case 3:
Ensemble has negative effect

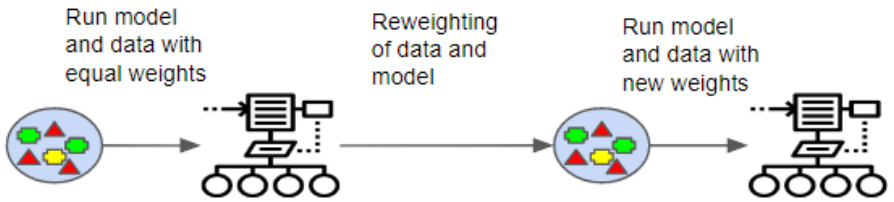
- Base models should be
  - Accurate
  - Diverse

Ensemble Methods: Increasing the Accuracy

- Popular ensemble methods
  - Bagging: Trains each model using a subset of the training set, and models learned in parallel
  - Boosting: Trains each new model instance to emphasize the training instances that previous models mis-classified, and models learned in order



Bagging



Boosting

Bagging: Bootstrap Aggregation

- Analogy: Diagnosis based on multiple doctors' majority vote

- Training
  - For i = 1 to k
    - create bootstrap sample, D_i, by sampling D with replacement;
    - use D_i and the learning scheme to derive a model, M_i ;

- Classification: classify an unknown sample **X**
  - let each of the k models classify X and return the majority vote

- Prediction:
  - To predict continuous variables, use average prediction instead of vote

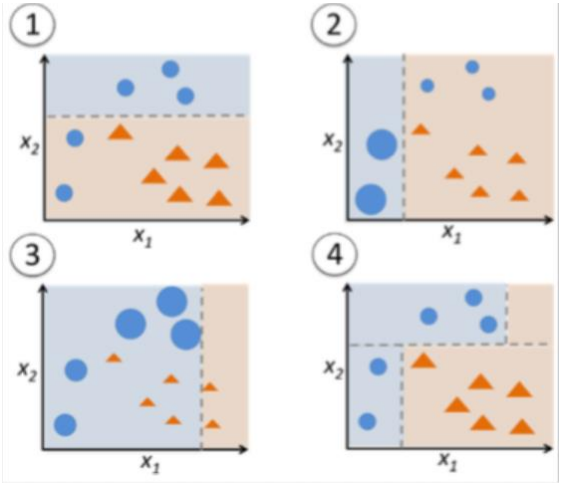
Boosting

- Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy
- How boosting works?
  - A series of k classifiers are iteratively learned
  - After a classifier M_i is learned, set 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 algorithm can be extended for numeric prediction

Adaboost (Freund and Schapire, 1997)

1. Assign initial weights to each training tuple

3. Update weights based on current model



$\{w_n^{(1)}\}$ $\{w_n^{(2)}\}$ ... $\{w_n^{(k)}\}$

M_1 M_2 ... M_k

2. Train base classifier on weighted dataset

Two 'weighting' strategy:

1. Assign weights to each training example
2. Sample dataset based on weight distribution

$$M^*(x) = \text{sign}\left(\sum_{i=1}^{k} \alpha_i M_i(x)\right)$$

4. After base classifiers are trained, they are combined to give the final classifier

Adaboost (Freund and Schapire, 1997)

- **Input**: Training set $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
- Initialize all weights $\{w_n^{(1)}\}$ to 1/N
- For round i = 1 to k,
  - Fit a classifier M_i based on weighted error function

$$J_m = \sum_{n=1}^{N} w_n^{(i)} I(M_i(x_n) \neq y_n)$$

  - Evaluate error rate $\epsilon_i = J_m / \sum w_n^{(i)}$ (stop iteration if ϵ_i < threshold)

    and the base model M_i's vote $\alpha_i = \frac{1}{2} \ln\left(\frac{1-\epsilon_i}{\epsilon_i}\right)$

  - Update weights

$$w_n^{(i+1)} = w_n^{(i)} \exp\{\alpha_i \cdot I(M_i(x_n) \neq y_n)\}$$

- The final model is given by voting based on $\{\alpha_n\}$

Gradient Boosting

- Operates on:
  - A differentiable loss function
  - A weak learner to make predictions (usually trees)
  - An additive model to add weak learners to minimize the loss function
- Each time adds an additional weak learner

$$
\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\dots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}
$$

Previous model

New weak learner

- Scalable implementation: XGBoost

Random Forest: Basic Concepts

- Random Forest (first proposed by L. Breiman in 2001)
  - Bagging with **decision trees** as base models
  - *Data bagging*
    - Use a **subset of training data** by sampling with replacement for each tree
  - *Feature bagging* ← Advantage of decision trees – more diversity
    - At each node use a **random selection of attributes** as candidates and split by the best attribute among them
  - During classification, each tree votes and the most popular class is returned

Random Forest

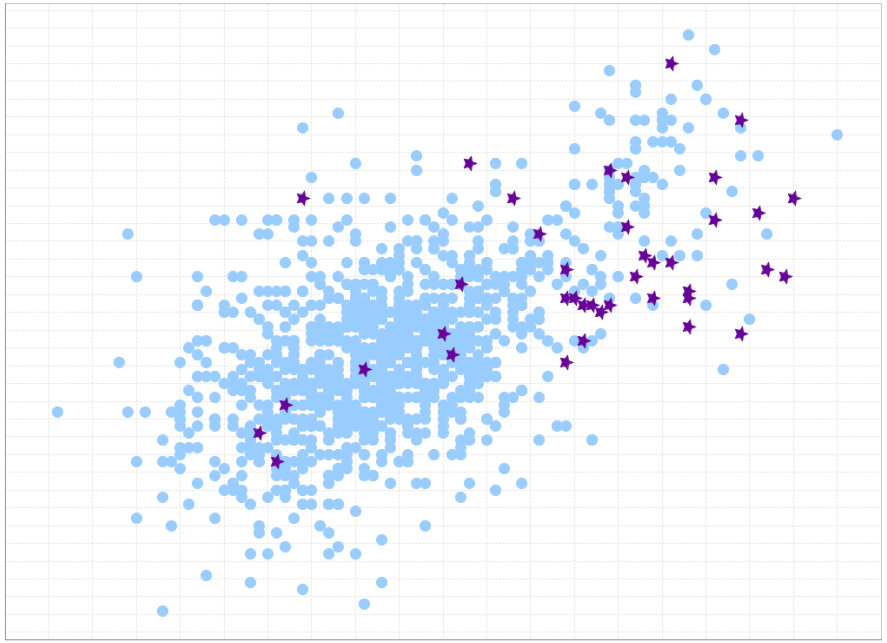
- Two Methods to construct Random Forest:
  - Forest-RI (*random input selection*): Randomly select, at each node, F attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size
  - Forest-RC (*random linear combinations*): Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- Comparable in accuracy to Adaboost, but more robust to errors and outliers
- Insensitive to the number of attributes selected for consideration at each split, and faster than typical bagging or boosting

Ensemble Methods Recap

- Random forest and XGBoost are the most commonly used algorithms for tabular data
- Pros
  - Good performance for tabular data, requires no data scaling
  - Can scale to large datasets
  - Can handle missing data to some extent
- Cons
  - Can overfit to training data if not tuned properly
  - Lack of interpretability (compared to decision trees)

Classification of Class-Imbalanced Data Sets

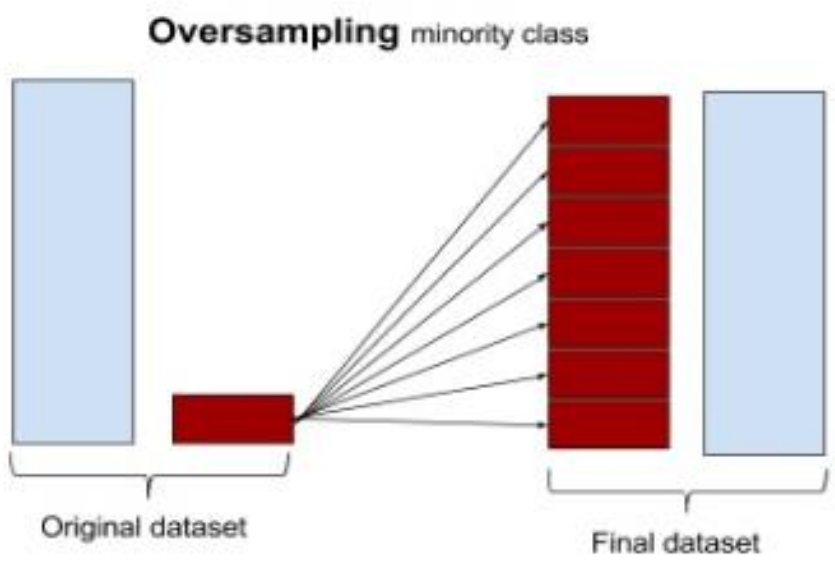
- Traditional methods assume a balanced distribution of classes and equal error costs. But in real world situations, we may face imbalanced data sets, which has rare positive examples but numerous negative ones.

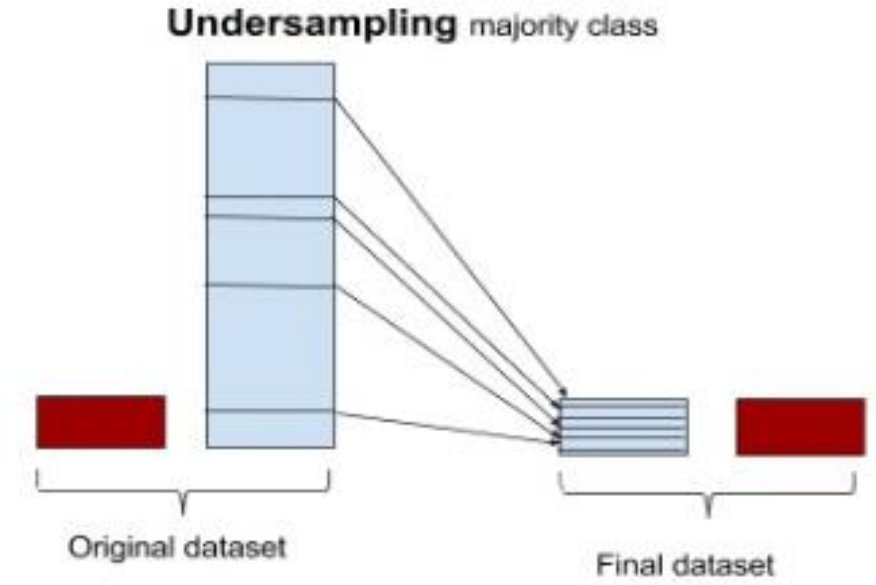


  - Medical diagnosis: Medical screening for a condition is usually performed on a large population of people without the condition, to detect a small minority with it (e.g., HIV prevalence in the USA is ~0.4%)
  - Fraud detection: About 2% of credit card accounts are defrauded per year. (Most fraud detection domains are heavily imbalanced.)
  - Product defect, accident (oil-spill), disk drive failures, etc.

Classification of Class-Imbalanced Data Sets

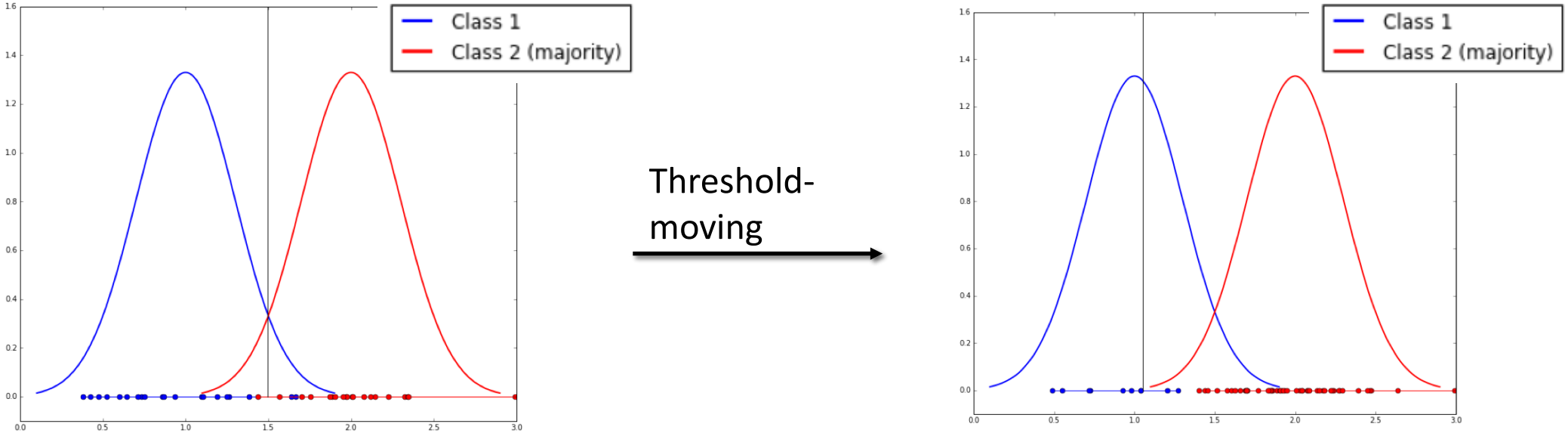
- Typical methods on imbalanced data (Balance the training set)
  - **Oversampling**: Oversample the minority class.
  - **Under-sampling**: Randomly eliminate tuples from majority class
  - **Synthesizing:** Synthesize new minority classes





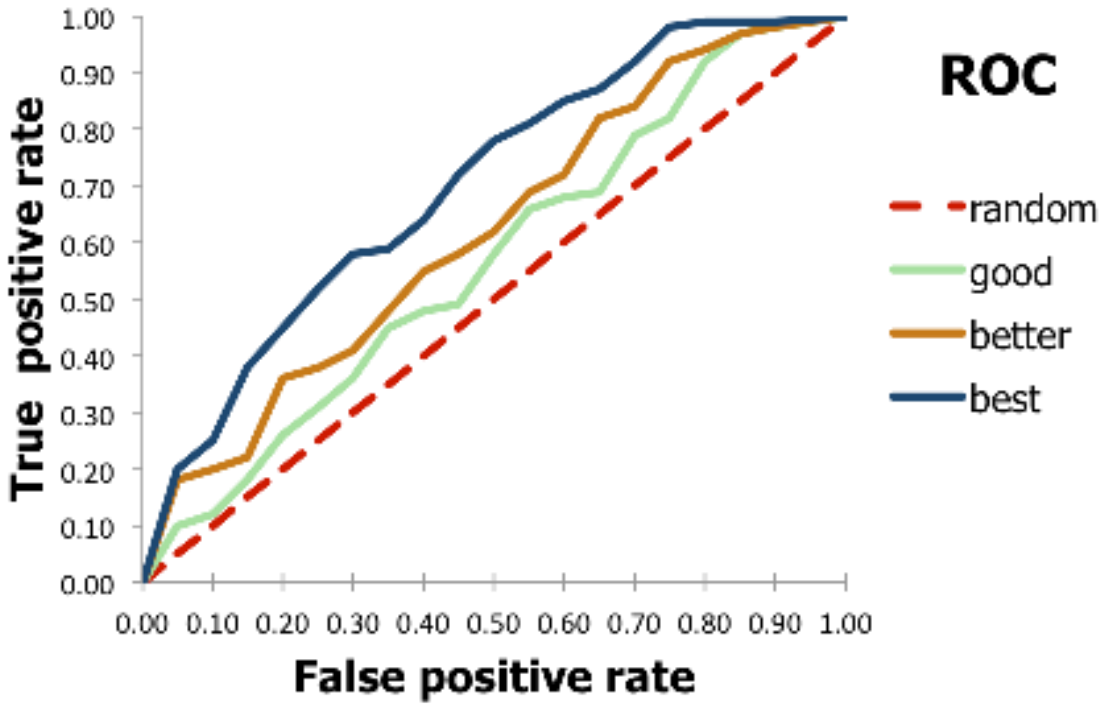
Classification of Class-Imbalanced Data Sets

- Typical methods on imbalanced data (At the algorithm level)
  - **Threshold-moving**: Move the decision threshold, t, so that the rare class tuples are easier to classify, and hence, less chance of costly false negative errors
  - **Class weight adjusting**: Since false negative costs more than false positive, we can give larger weight to false negative
- **Ensemble techniques**: Ensemble multiple classifiers introduced in the following chapter



Evaluate imbalanced data classifier

- Can we use Accuracy to evaluate imbalanced data classifier?
- Accuracy simply counts the number of errors. If a data set has 2% positive labels and 98% negative labels, a classifier that map all inputs to negative class will get an accuracy of 98%!
- ROC Curve



Summary

- Classification: Model construction from a set of training data
- Effective and scalable methods
  - Decision tree induction, Bayes classification methods, lazy learners, linear classifier
  - No single method has been found to be superior over all others for all data sets
- Evaluation metrics: Accuracy, sensitivity, specificity, precision, recall, *F* measure
- Model evaluation: Holdout, cross-validation, bootstrapping, ROC curves (AUC)
- Techniques to improve classification accuracy: Ensemble Methods (bagging, boosting, random forests), imbalanced classification

References (1)

- C. Apte and S. Weiss. **Data mining with decision trees and decision rules**. Future Generation Computer Systems, 13, 1997
- A. J. Dobson. **An Introduction to Generalized Linear Models**. Chapman & Hall, 1990.
- R. O. Duda, P. E. Hart, and D. G. Stork. **Pattern Classification**, 2ed. John Wiley, 2001
- U. M. Fayyad. **Branching on attribute values in decision tree generation**. AAAI'94.
- Y. Freund and R. E. Schapire. **A decision-theoretic generalization of on-line learning and an application to boosting**. J. Computer and System Sciences, 1997.
- J. Gehrke, R. Ramakrishnan, and V. Ganti. **Rainforest: A framework for fast decision tree construction of large datasets**. VLDB'98.
- J. Gehrke, V. Gant, R. Ramakrishnan, and W.-Y. Loh, **BOAT -- Optimistic Decision Tree Construction**. SIGMOD'99.
- T. Hastie, R. Tibshirani, and J. Friedman. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction.** Springer-Verlag, 2001
- J. Pan and D. Manocha. **Bi-level locality sensitive hashing for k-nearest neighbor computation**. IEEE ICDE 2012

References (2)

- T.-S. Lim, W.-Y. Loh, and Y.-S. Shih. **A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms.** Machine Learning, 2000
- J. Magidson. **The Chaid approach to segmentation modeling: Chi-squared automatic interaction detection**. In R. P. Bagozzi, editor, Advanced Methods of Marketing Research, Blackwell Business, 1994
- M. Mehta, R. Agrawal, and J. Rissanen. **SLIQ : A fast scalable classifier for data mining**. EDBT'96
- T. M. Mitchell. **Machine Learning**. McGraw Hill, 1997
- S. K. Murthy, **Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey**, Data Mining and Knowledge Discovery 2(4): 345-389, 1998
- J. R. Quinlan. **Induction of decision trees**. *Machine Learning*, 1:81-106, 1986
- J. R. Quinlan. **C4.5: Programs for Machine Learning**. Morgan Kaufmann, 1993
- J. R. Quinlan. **Bagging, boosting, and c4.5**. AAAI'96
- E. Alpaydin. **Introduction to Machine Learning (2nd ed.)**. MIT Press, 2011

References (3)

- R. Rastogi and K. Shim. **Public: A decision tree classifier that integrates building and pruning**. VLDB'98
- J. Shafer, R. Agrawal, and M. Mehta. **SPRINT : A scalable parallel classifier for data mining**. VLDB'96
- J. W. Shavlik and T. G. Dietterich. **Readings in Machine Learning**. Morgan Kaufmann, 1990
- P. Tan, M. Steinbach, and V. Kumar. **Introduction to Data Mining**. Addison Wesley, 2005
- S. M. Weiss and C. A. Kulikowski. **Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems**. Morgan Kaufman, 1991
- S. M. Weiss and N. Indurkhya. **Predictive Data Mining**. Morgan Kaufmann, 1997
- I. H. Witten and E. Frank. **Data Mining: Practical Machine Learning Tools and Techniques**, 2ed. Morgan Kaufmann, 2005
- T. Chen and C. Guestrin. **Xgboost: A Scalable Tree Boosting System**. ACM SIGKDD 2016
- C. Aggarwal. **Data Classication**. Morgan Springer, 2015