



CS5824: Advanced Machine Learning

Dawei Zhou
CS, Virginia Tech

Please keep your face covering on!

Classification: Basic Concepts

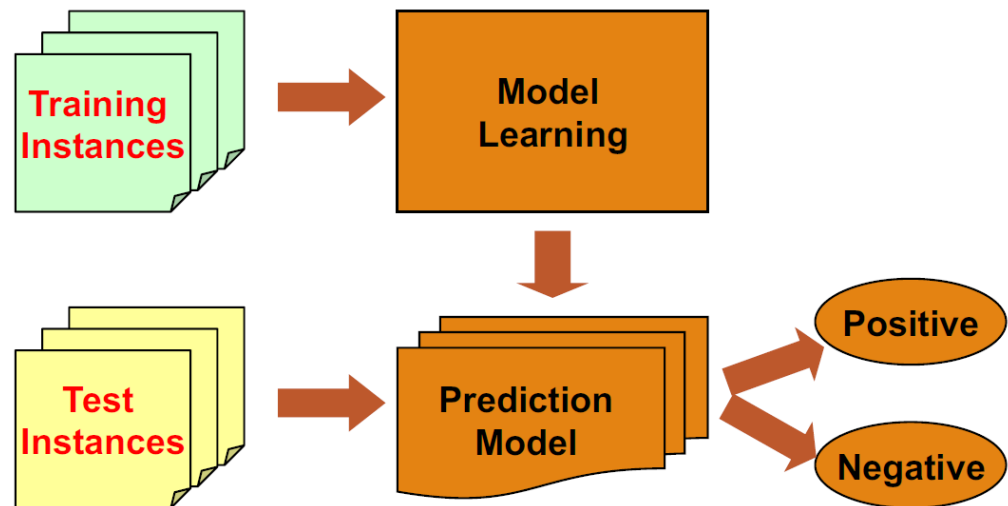
Supervised vs. Unsupervised Learning

- **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

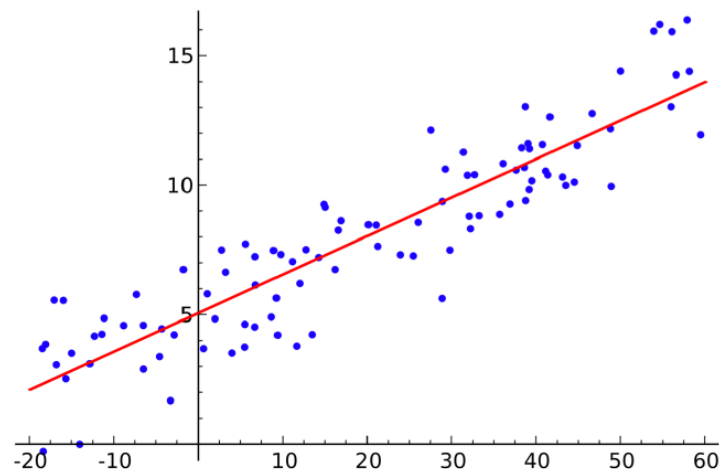
Training Data with class label:

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



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

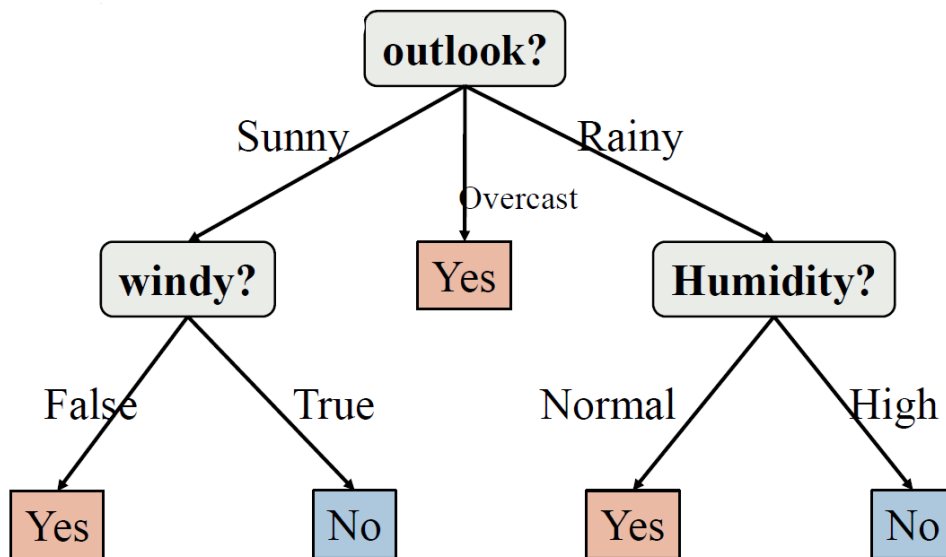
Classification—Model Construction, Validation and Testing

- **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

Decision Tree Induction

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

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., $(15+18)/2 = 16.5, 19.5, 21.5, 23, 24.5, 27, 30, \dots$
 - 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 $A \leq P$ vs. those with $A > P$

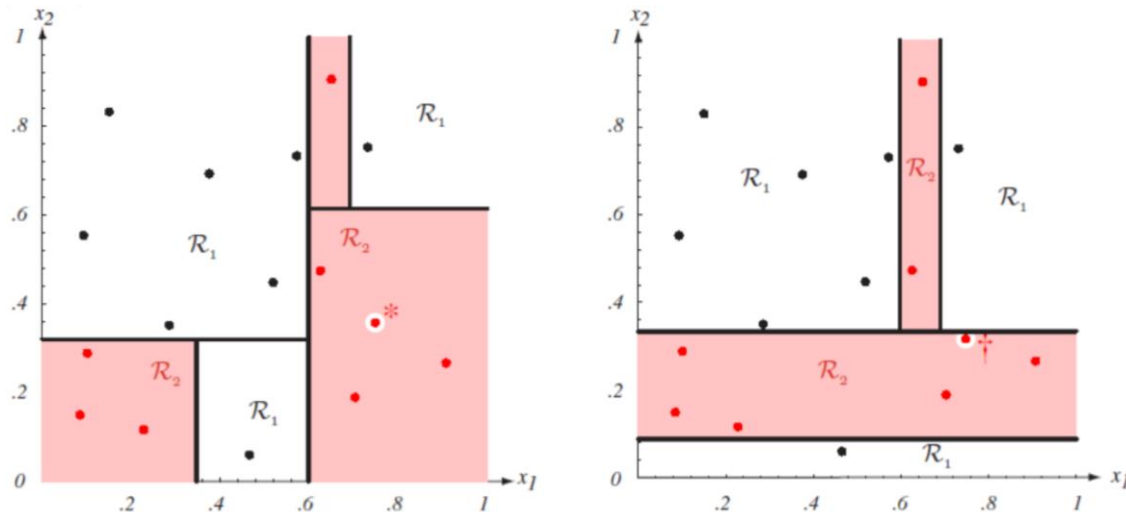
Pro's and Con's

- **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)**

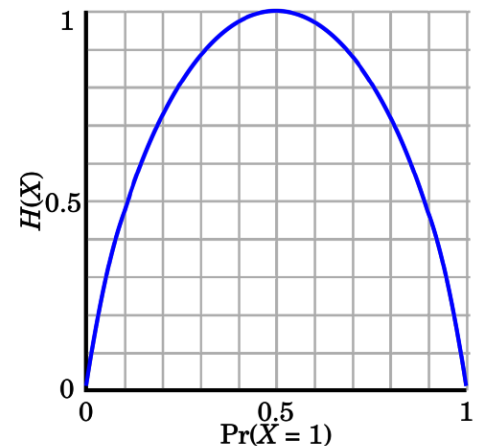
- A measure of uncertainty associated with a random number
- Calculation: For a discrete random variable Y taking m distinct values $\{y_1, y_2, \dots, y_m\}$

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

- Interpretation
 - Higher entropy \rightarrow higher uncertainty
 - Lower entropy \rightarrow 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 :

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

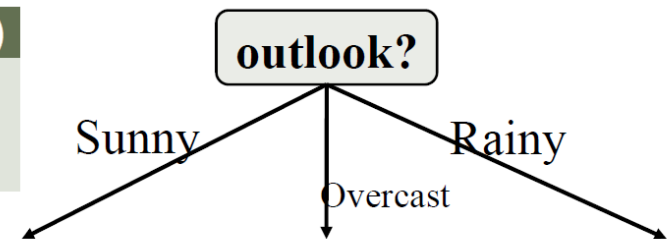
- Information gained by branching on attribute A

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

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



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

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

$\frac{5}{14}I(2,3)$ 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

$$\begin{aligned}
 \text{Gain}(\text{Temp}) &= 0.029, \\
 \text{Gain}(\text{humidity}) &= 0.151, \\
 \text{Gain}(\text{Windy}) &= 0.048
 \end{aligned}$$

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
 - SplitInfo:

$$SplitInfo_{temp}(D) = -\frac{4}{14} \log_2 \frac{4}{14} - \frac{6}{14} \log_2 \frac{6}{14} - \frac{4}{14} \log_2 \frac{4}{14} = 1.557$$

- GainRatio(temp) = 0.029/1.557 = 0.019

Bayes Classification Methods

Bayes' Theorem: Basics

- **Total probability Theorem:**

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

- **Bayes' Theorem:**

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

posteriori probability likelihood prior probability

What we should choose What we just see 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 L
- What is the chance of rain? $P(\text{Rain} \mid \text{Cloud}) = ?$
- 50% of all rainy days start off cloudy. $P(\text{Cloud} \mid \text{Rain}) = 50\%$
- Cloudy mornings are common (40% of days start cloudy) $P(\text{Cloud}) = 40\%$
- This is usually a dry month (only 3 of 30 days tend to be rainy) $P(\text{Rain}) = 10\%$

$$P(\text{Rain} \mid \text{Cloud}) = P(\text{Rain}) P(\text{Cloud} \mid \text{Rain}) / P(\text{Cloud}) = 10\% * 50\% / 40\% = 12.5\%$$

- 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., $P(\text{Rain} \mid \text{Cloud})$ and $P(\text{Cloud} \mid \text{Rain})$), 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

$$P(C|X) \propto P(X|C)P(C) = P(x_1|C)P(x_2|x_1,C)\dots P(x_n|x_1,\dots,C)P(C)$$

- A naïve bayes assumption to simplify the complex dependencies: features are conditionally independent, given the class label!

$$P(C|X) \propto P(X|C)P(C) \approx P(x_1|C)P(x_2|C)\dots P(x_n|C)P(C)$$

- 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)

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

- 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

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

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

$P(\text{Yes} \mid \text{Sunny})$

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

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	

$$P(x) = P(\text{Sunny}) = 5 / 14 = 0.36$$

$$P(c) = P(\text{Yes}) = 9 / 14 = 0.64$$

Posterior Probability:

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



Naïve Bayes Classifier Example

$P(\text{No} \mid \text{Sunny})$

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

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



$$P(x) = P(\text{Sunny}) = 5 / 14 = 0.36$$



$$P(c) = P(\text{No}) = 5 / 14 = 0.36$$



Posterior Probability:

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



Naïve Bayes Classifier Example: Likelihood Tables

Frequency Table

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



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	4
	Normal	6	1



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

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



		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	2
	True	3	3



		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	?

$$P(\text{Yes} | X) = P(\text{Rainy} | \text{Yes}) \times P(\text{Cool} | \text{Yes}) \times P(\text{High} | \text{Yes}) \times P(\text{True} | \text{Yes}) \times P(\text{Yes}) / P(x)$$

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

$$P(\text{No} | X) = P(\text{Rainy} | \text{No}) \times P(\text{Cool} | \text{No}) \times P(\text{High} | \text{No}) \times P(\text{True} | \text{No}) \times P(\text{No}) / P(x)$$

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

$$P(x) = P(x | \text{yes}) * P(\text{yes}) + P(x | \text{no}) * P(\text{no})$$

Avoiding the Zero-Probability Problem

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

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

- 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) = $(10 + 1)/(1000 + 3)$

$$\begin{aligned}\hat{P}(w_i|c) &= \frac{\text{count}(w_i, c) + 1}{\sum_{w \in V} (\text{count}(w, c) + 1)} \\ &= \frac{\text{count}(w_i, c) + 1}{\left(\sum_{w \in V} \text{count}(w, c) \right) + |V|}\end{aligned}$$

- 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

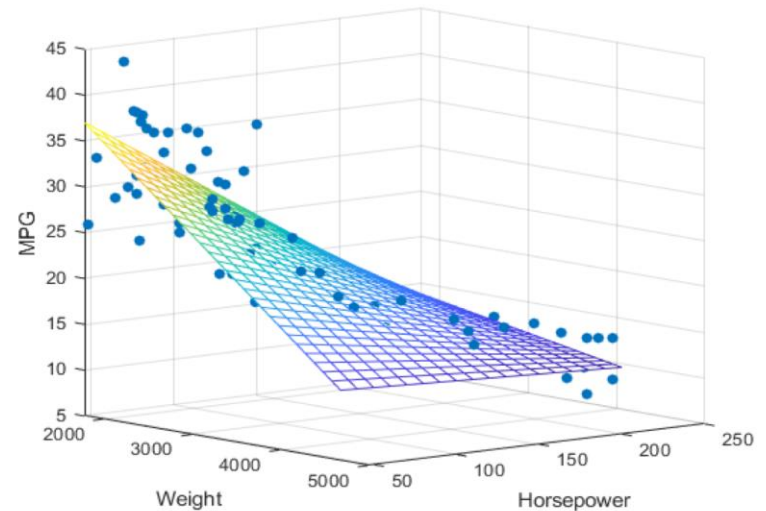
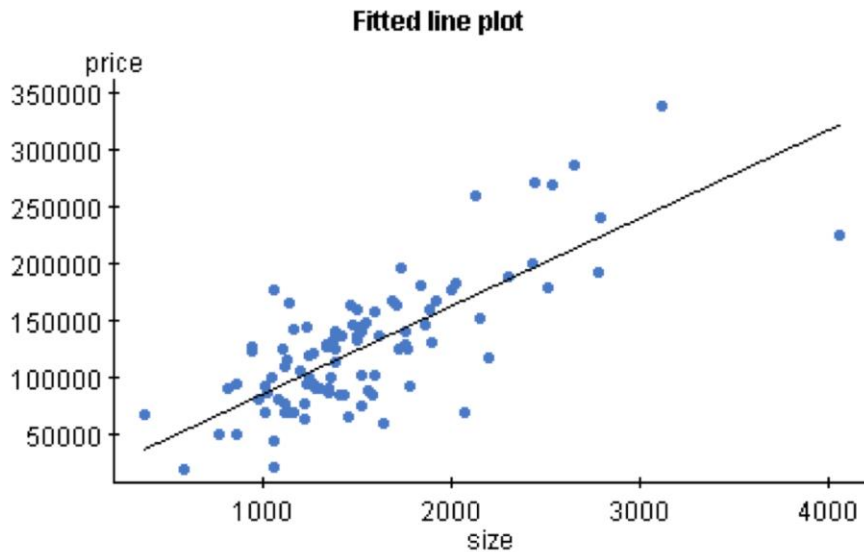
- **Weakness**

- **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

Linear Classifier

Linear Regression Problem: Example

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



Linear Regression Problem: Model

- **Linear regression**

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

Linear Regression Model: Solution

- **Least Square Method**

- Cost / Loss Function: $L(w, b) = \sum_{i=1}^n (y_i - wx_i - b)^2$
- Optimization Goal:

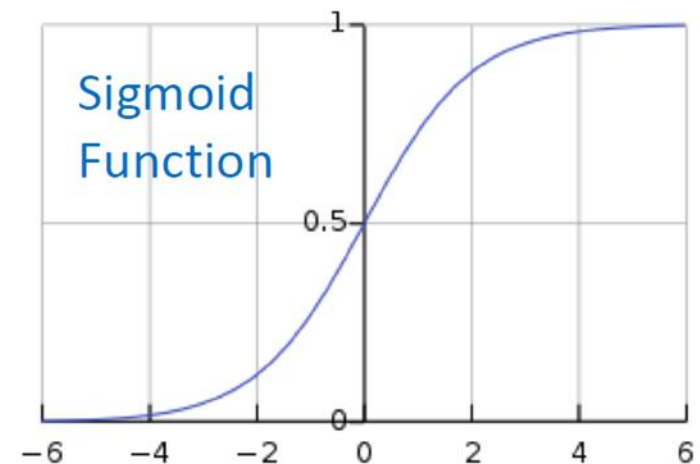
$$\underset{(w,b)}{\operatorname{argmin}} L(w, b) = \sum_{i=1}^n (y_i - wx_i - b)^2$$

- Closed-form solution:

$$w = \frac{\sum_{i=1}^n x_i (y_i - \bar{y})}{\sum_{i=1}^n x_i^2 - n(\sum_{i=1}^n x_i)^2} \quad b = \frac{1}{n} \sum_{i=1}^n (y_i - wx_i)$$

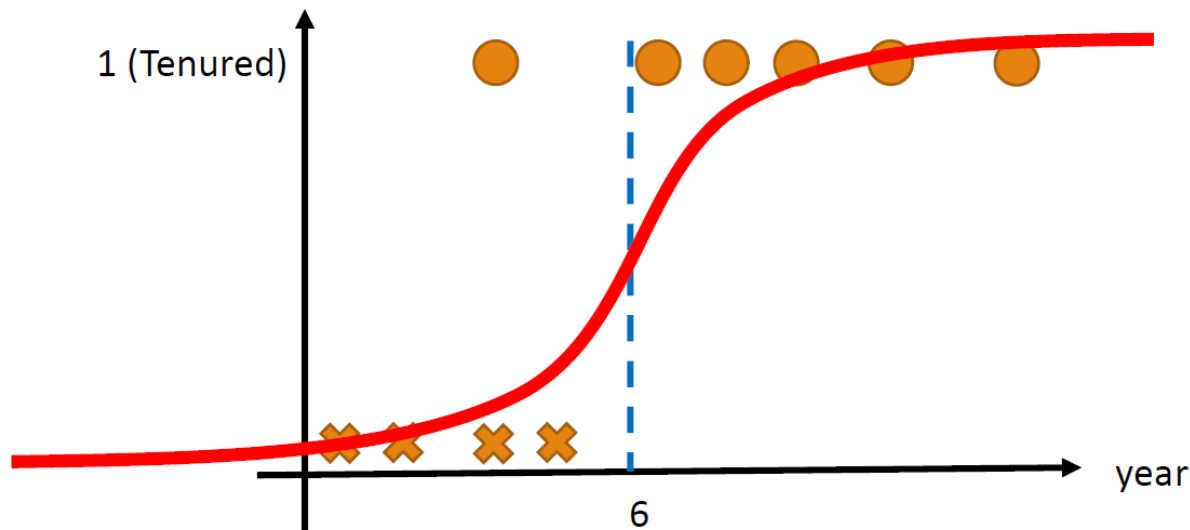
Logistic Regression: General Ideas

- How to solve “classification” problems by regression?
- **Key idea of Logistic Regression**
 - We need to transform the real value Y into a probability value $\in [0,1]$
- **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 $\ln \frac{y}{1-y} = w^T x + b$

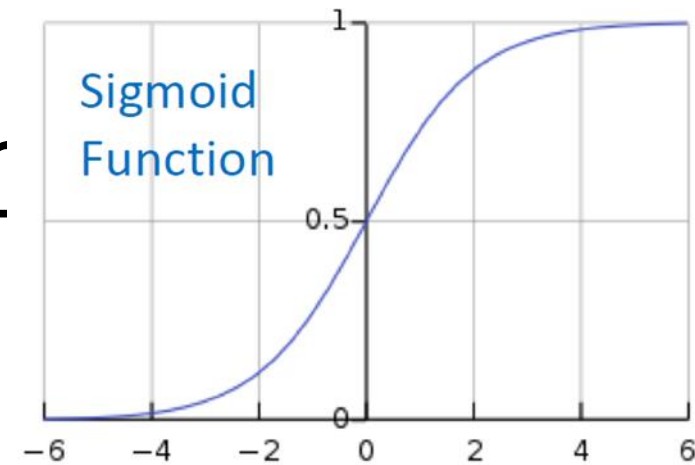


Logistic Regression: An Example

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



Logistic Regressior



- The prediction function to learn
- Probability that $Y=1$:
 - $p(Y = 1 | X = x; \mathbf{w}) = \text{Sigmoid}(w_0 + \sum_{i=1}^n w_i \cdot x_i)$
 - $\mathbf{w} = (w_0, w_1, w_2, \dots, w_n)$ 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

$$L = \prod_i p_i^{y_i} (1 - p_i)^{1-y_i} = \prod_i \left(\frac{\exp(w^T x_i)}{1 + \exp(w^T x_i)} \right)^{y_i} \left(\frac{1}{1 + \exp(w^T x_i)} \right)^{1-y_i}$$

Logistic Regression: Optimization

- **Logistic Regression: Optimization**

$$L = \prod_i p_i^{y_i} (1 - p_i)^{1-y_i} = \prod_i \left(\frac{\exp(w^T x_i)}{1 + \exp(w^T x_i)} \right)^{y_i} \left(\frac{1}{1 + \exp(w^T x_i)} \right)^{1-y_i}$$

- **Log likelihood:**

$$\begin{aligned} l(w) &= \sum_{i=1}^N y_i \log p(Y = 1|X = x_i; \mathbf{w}) + (1 - y_i) \log(1 - p(Y = 1|X = x_i; \mathbf{w})) \\ &= \sum_{i=1}^N y_i x_i^T \mathbf{w} - \log(1 + \exp(\mathbf{w}^T x_i)) \end{aligned}$$

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

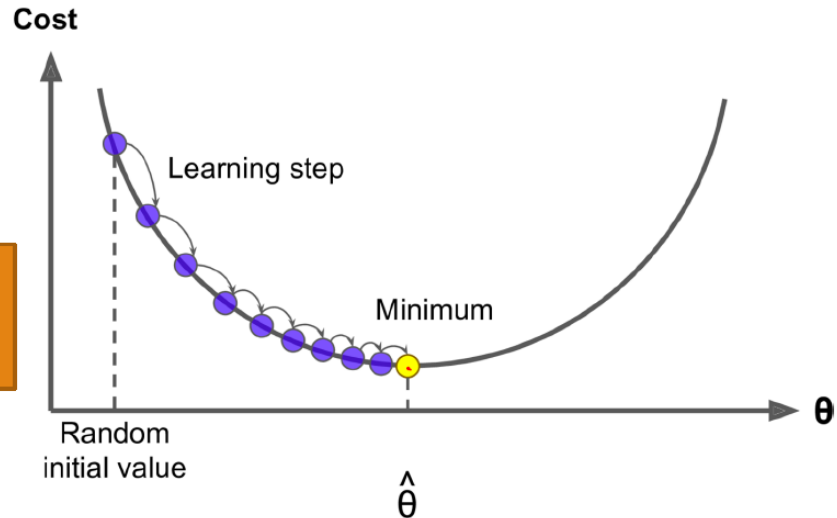
Gradient Descent

- **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

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

Step size

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



Gradient Descent

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

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

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

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

Iterate until change < threshold

For all i ,

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

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

Iterate until change < threshold

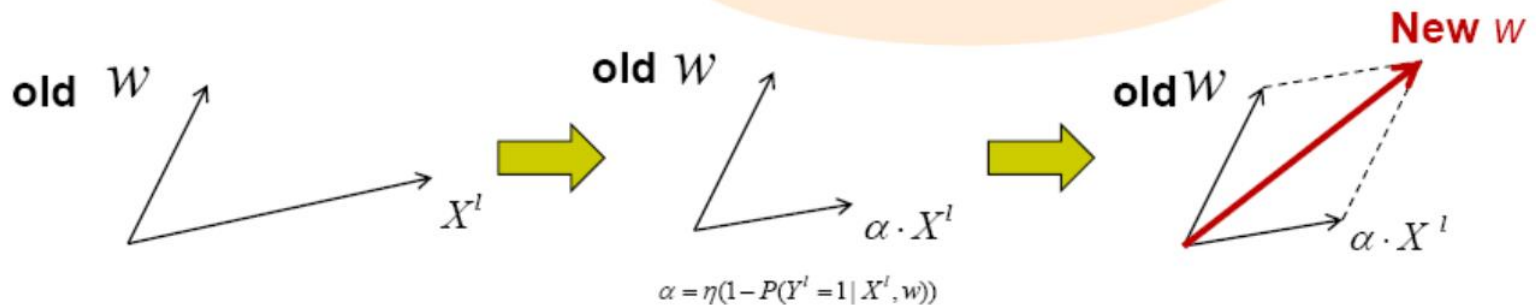
For all i ,

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

Feature, true label, current prediction

If $Y^l = 1$

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



Gradient Descent

Iterate until change < threshold

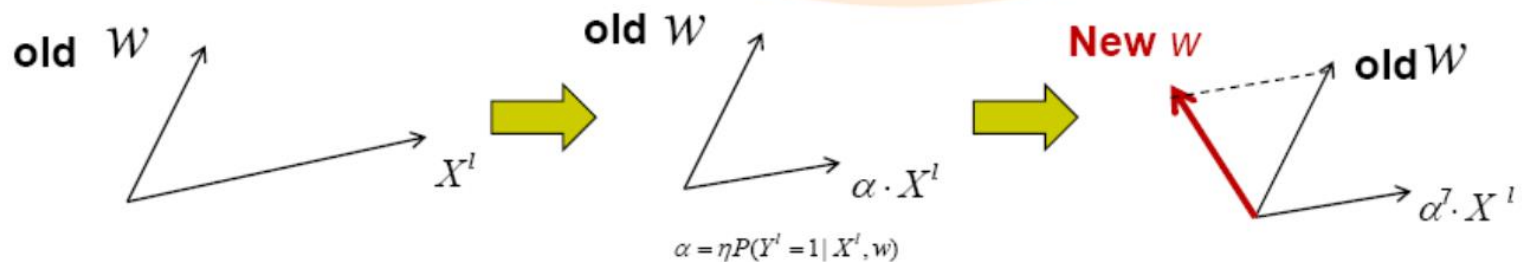
For all i ,

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

Feature, true label, current prediction

If $Y^l = 0$

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



Model Evaluation and Selection

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	$\neg C_1$
C_1	True Positives (TP)	False Negatives (FN)
$\neg C_1$	False Positives (FP)	True Negatives (TN)

- In a confusion matrix with 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, 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**: $1 - \text{accuracy}$, 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?

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

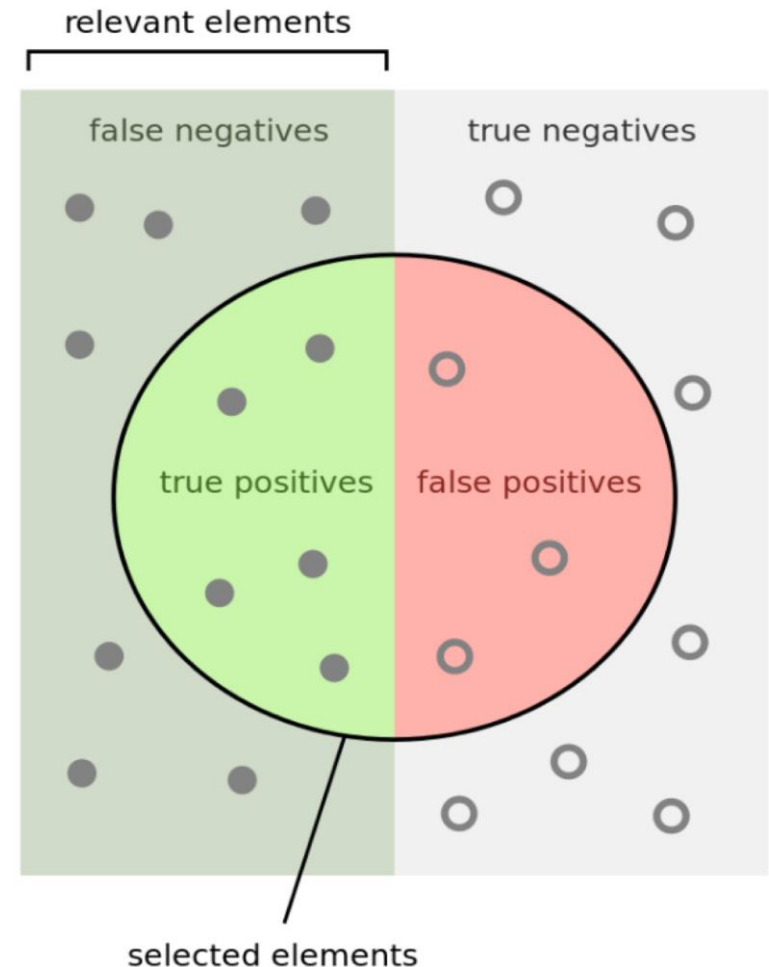
$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$


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

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

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$


Range: [0, 1]



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 $\beta = 1$,

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

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 =

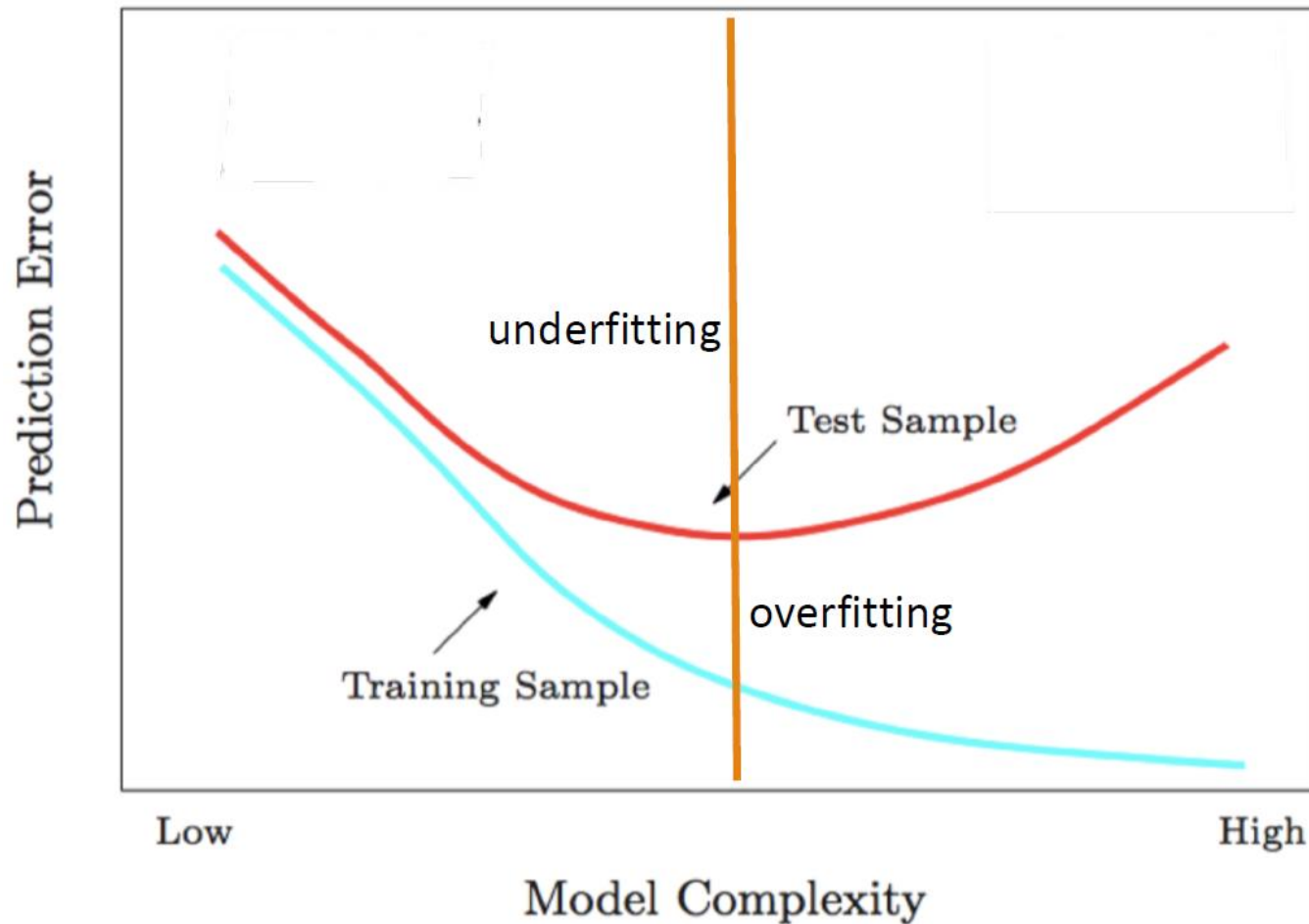
Classifier Evaluation Metrics: Example

- 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 = $TP/(TP + FN) = 90/(90 + 210) = 90/300 = 30.00\%$
- $F1 = 2 P \times R / (P + R) = 2 \times 39.13\% \times 30.00\% / (39.13\% + 30\%) = 33.96\%$

Training Error VS Testing Error



Classifier Evaluation: Holdout

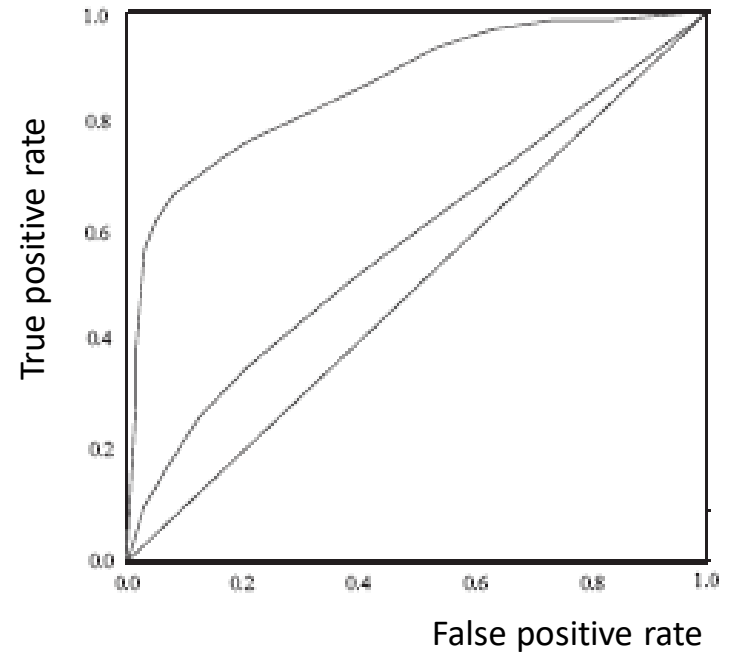
- **Holdout method**
 - Given data is randomly partitioned into two independent sets
 - Training set (e.g., $2/3$) for model construction
 - Test set (e.g., $1/3$) for accuracy estimation
 - Repeated random sub-sampling validation: a variation of holdout
 - Repeat holdout k times, accuracy = avg. of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

Summary

Summary

- Classification: Model construction from a set of training data
- Effective and scalable methods
 - Decision tree induction, Bayes classification methods, 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)

References (1)

- C. Apte and S. Weiss. **Data mining with decision trees and decision rules.** Future Generation Computer Systems, 13, 1997
- P. K. Chan and S. J. Stolfo. **Learning arbiter and combiner trees from partitioned data for scaling machine learning.** KDD'95
- 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.

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.

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. Indurkha. **Predictive Data Mining.** Morgan Kaufmann, 1997
- I. H. Witten and E. Frank. **Data Mining: Practical Machine Learning Tools and Techniques, 2ed.** Morgan Kaufmann, 2005