# CS 412 Intro. to Data Mining

## Chapter 8. Classification: Basic Concepts

Qi Li, Computer Science, Univ. Illinois at Urbana-Champaign, 2018

# Chapter 8. Classification: Basic Concepts

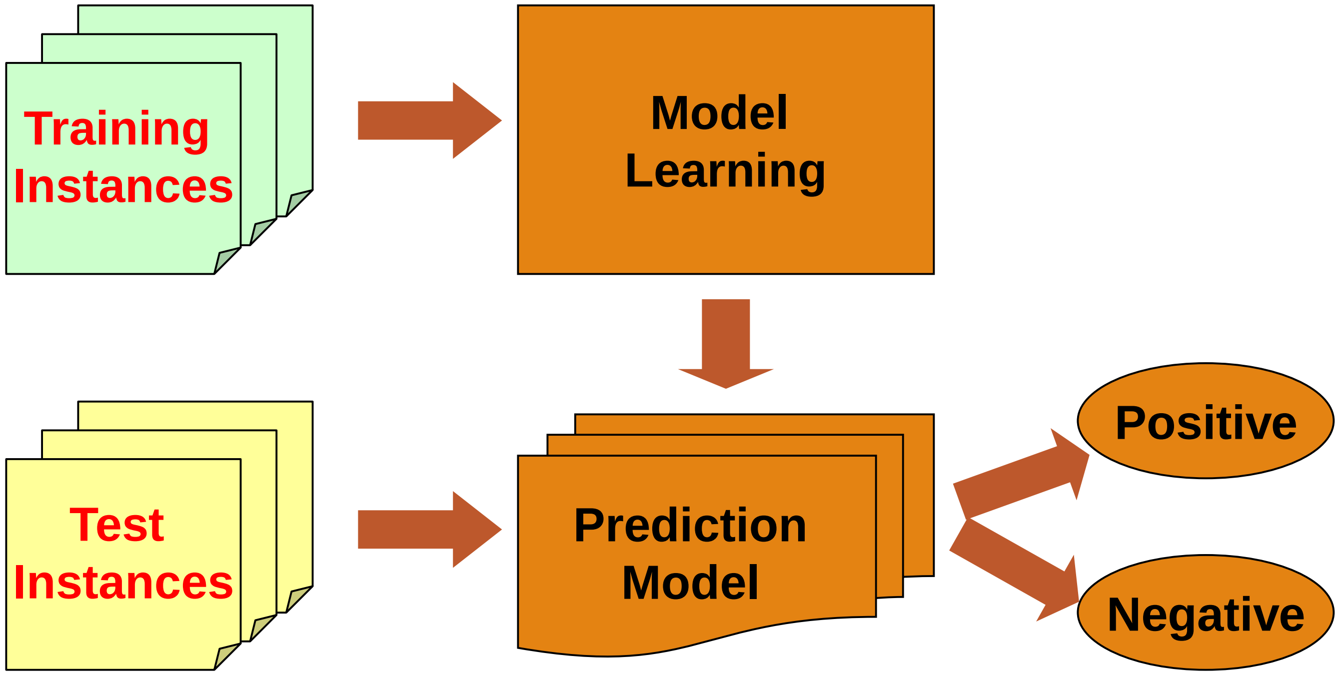
- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Linear Classifier
- Model Evaluation and Selection
- Imbalanced Data Sets
- Additional Concepts on Classification
- Summary

# 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

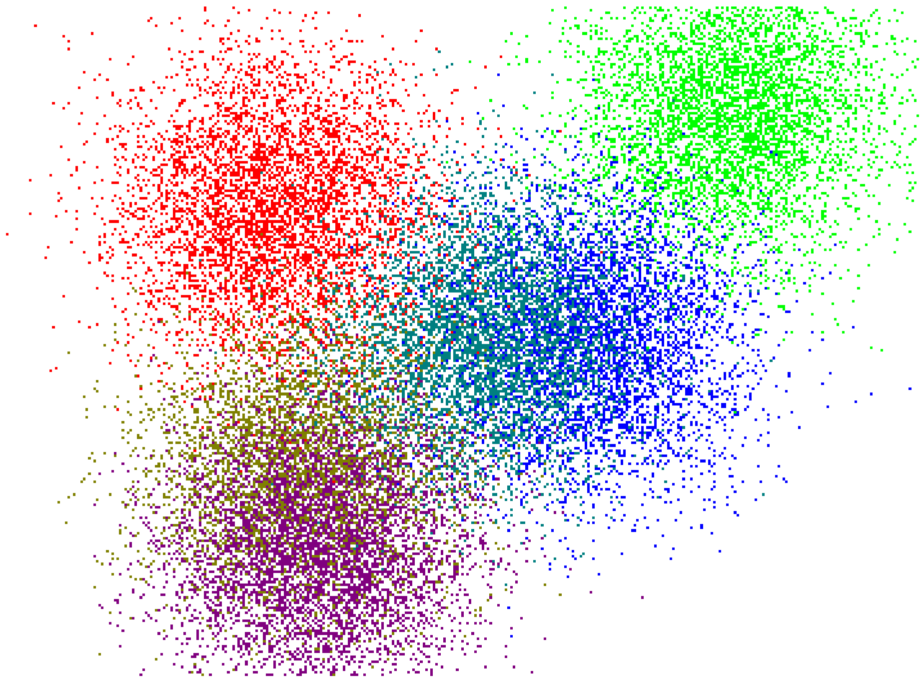
Training Data with class label:

| 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 |



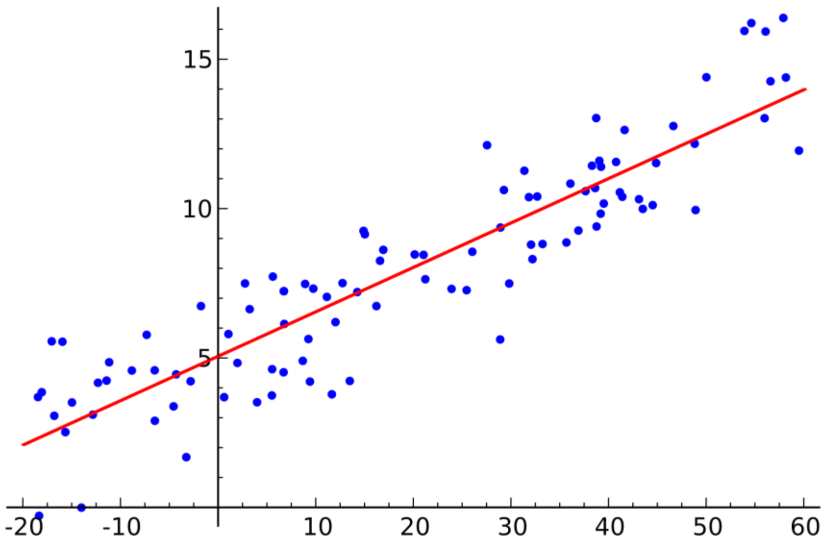
# Supervised vs. Unsupervised Learning (2)

- 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)



# Prediction Problems: Classification vs. Numeric Prediction

- Typical applications of classification
  - Credit/loan approval
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is

# Classification—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

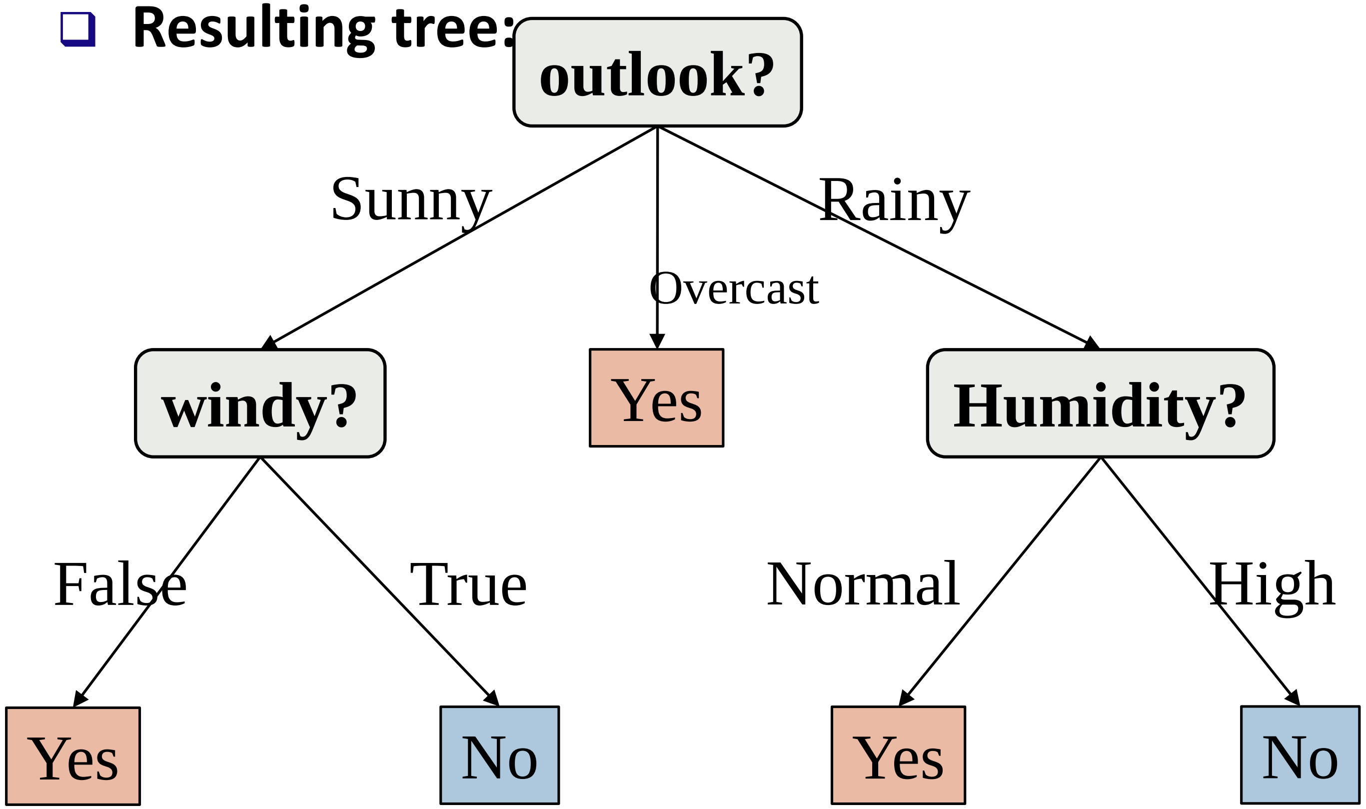
- **Model Validation and Testing**:
  - **Test:** Estimate accuracy of the model
    - The known label of test sample VS. the classified result from the model
    - *Accuracy:* % of test set samples that are correctly classified by the model
    - Test set is independent of training set
  - **Validation**: 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 8. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Linear Classifier
- Model Evaluation and Selection
- Imbalanced Data Sets
- Additional Concepts on Classification
- 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?

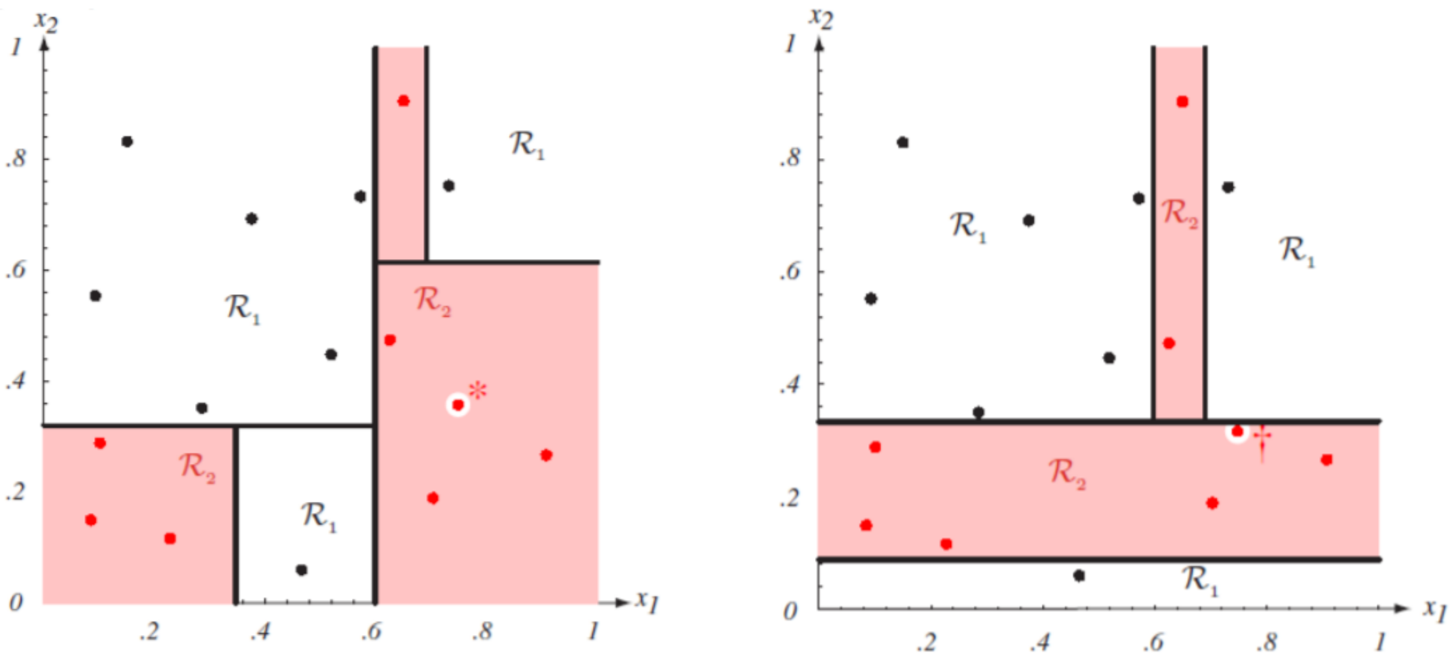
- Method 1: Discretize continuous values and treat them as categorical values
  - E.g., age: < 20, 20..30, 30..40, 40..50, > 50
- Method 2: Determine the ***best split point*** for continuous-valued attribute A
  - Sort:, e.g. 15, 18, 21, 22, 24, 25, 29, 31, …
  - *Possible split point:* $(a_i+a_{i+1})/2$
    - 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

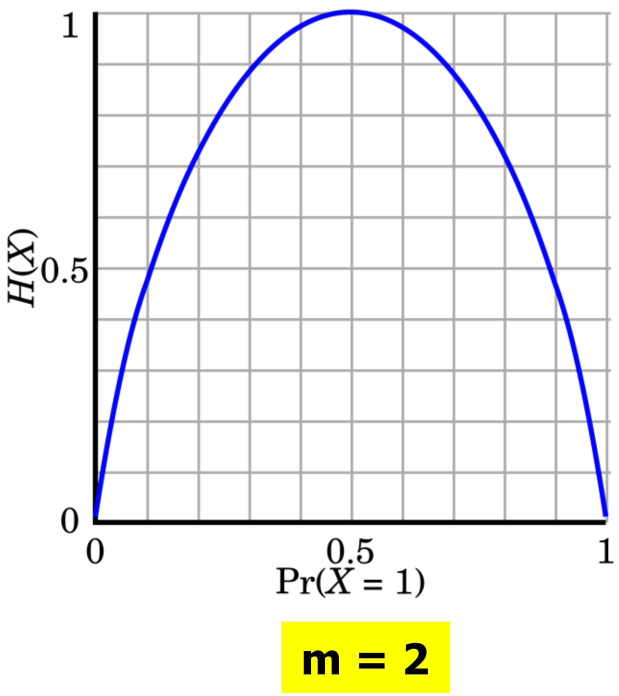
- Con's
  - Unstable. Sensitive to noise
  - Accuracy may be not good enough (depending on your data)
  - The optimal splitting is NP. Greedy algorithms are used
  - Overfitting



# 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

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

# 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:

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

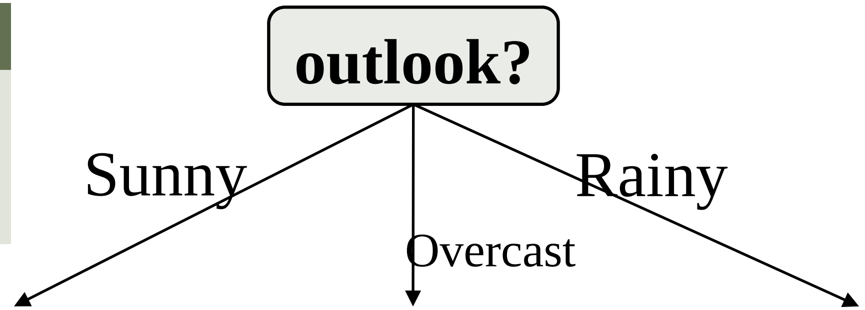
- Information gained by branching on attribute A

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

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

# 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
  - $\mathrm{SplitInfo}_{\mathrm{temp}}(\mathrm{D}) = -\frac{4}{14}\log_2\frac{4}{14} - \frac{6}{14}\log_2\frac{6}{14} - \frac{4}{14}\log_2\frac{4}{14} = 1.557$
  - GainRatio(temp) = 0.029/1.557 = 0.019

# Another Measure: Gini Index

- Gini index (or Gini impurity): Used in CART, and also in IBM IntelligentMiner
- CART is a binary tree
- If a data set $D$ contains examples from $n$ classes, gini index, $gini(D)$ is defined as
  - $gini(D) = 1 - \sum_{j=1}^{n} p_j^2$
    - $p_j$ is the relative frequency of class $j$ in $D$
- What is the range of Gini index?
  - The minimum= 0, meaning pure
  - The maximum=? What is the case that Gini index reach the maximum?

# Another Measure: Gini Index

- 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 (*need to enumerate all the possible splitting points for each attribute*)

# Computation of Gini Index

- Example: D has 9 tuples in play_golf= "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 temp partitions D into 10 in $D_1$: {cool, mild} and 4 in $D_2$
  - $gini_{temp\in\{cool,mild\}}(D) = \frac{10}{14} gini(D_1) + \frac{4}{14} gini(D_2)$
    $= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right)$
    $= 0.443$
  - $Gini_{\{cool,mild\}}$ is 0.458; $Gini_{\{mild,hot\}}$ is 0.450
  - Thus, split on the {cool,mild} (and {hot}) since it has the lowest Gini index

| 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 |

# Comparing Three Attribute Selection Measures

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

# Comparing Three Attribute Selection Measures

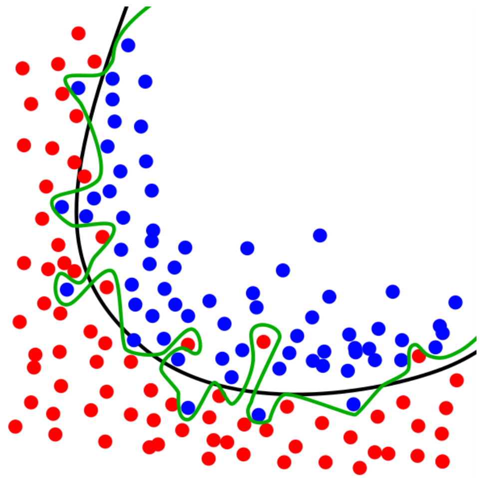
- In reality
- [Theoretical comparison between the gini index and information gain criteria]
- It only matters in 2% of the cases.
- Entropy might be a little slower to compute (because of the logarithm).

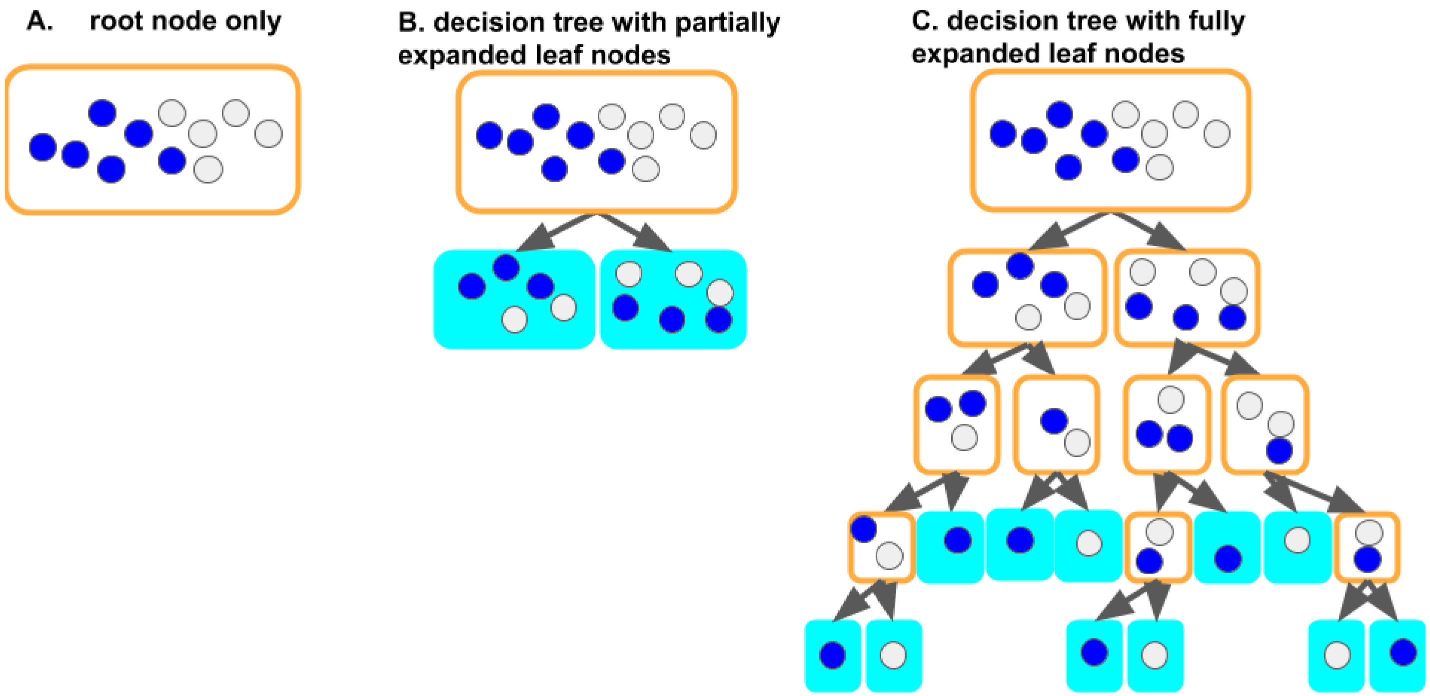
# Other Attribute Selection Measures

- Minimal Description Length (MDL) principle
  - Philosophy: The simplest solution is preferred
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- CHAID: a popular decision tree algorithm, measure based on $\chi^2$ test for independence
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear combination of attributes
- There are many other measures proposed in research and applications
  - E.g., G-statistics, C-SEP
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others

# Overfitting and Tree Pruning

- **Overfitting:**
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples





# Overfitting and Tree Pruning

- Two approaches to avoid overfitting
  - Errors: use a cross-validation to compute
  - Pre-pruning (Early stop): Error does not decrease *significantly* -> stop splitting
    - Efficient but prone to under-fit (stop too early)
  - Post-pruning: After the full tree is constructed, prune back to the point where the cross-validation error is minimum
    - Extra computations but mathematically rigorous
  - Can be used alone, in combination, or not at all
    - For different purposes (accuracy, efficiency, interpretability)

# Classification in Large Databases

- Scalability: Classifying data sets with **millions** of examples and hundreds of attributes with **reasonable speed**
- Why is decision tree induction popular?
  - Relatively fast learning speed
  - Convertible to simple and easy to understand classification rules
  - Easy to be adapted to database system implementations (e.g., using SQL)
  - Comparable classification accuracy with other methods
  - Easy to ensemble, i.e., random forests, xgboost

# RainForest: A Scalable Classification Framework

- The criteria that determine the quality of the tree can be computed separately
  - Builds an AVC-list: **AVC (Attribute, Value, Class_label)**
- **AVC-set** (of an attribute *X* )
  - Projection of training dataset onto the attribute *X* and class label where counts of individual class label are aggregated
- **AVC-group** (of a node *n* )
  - Set of AVC-sets of all predictor attributes at the node *n*

| 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 |

The Training Data

AVC-set on *outlook*

| outlook | Play Golf | |
|---|---|---|
| | yes | no |
| rainy | 2 | 3 |
| overcast | 4 | 0 |
| sunny | 3 | 2 |

AVC-set on Temp

| temp | Play Golf | |
|---|---|---|
| | yes | no |
| hot | 2 | 2 |
| mild | 4 | 2 |
| cool | 3 | 1 |

AVC-set on *Humidity*

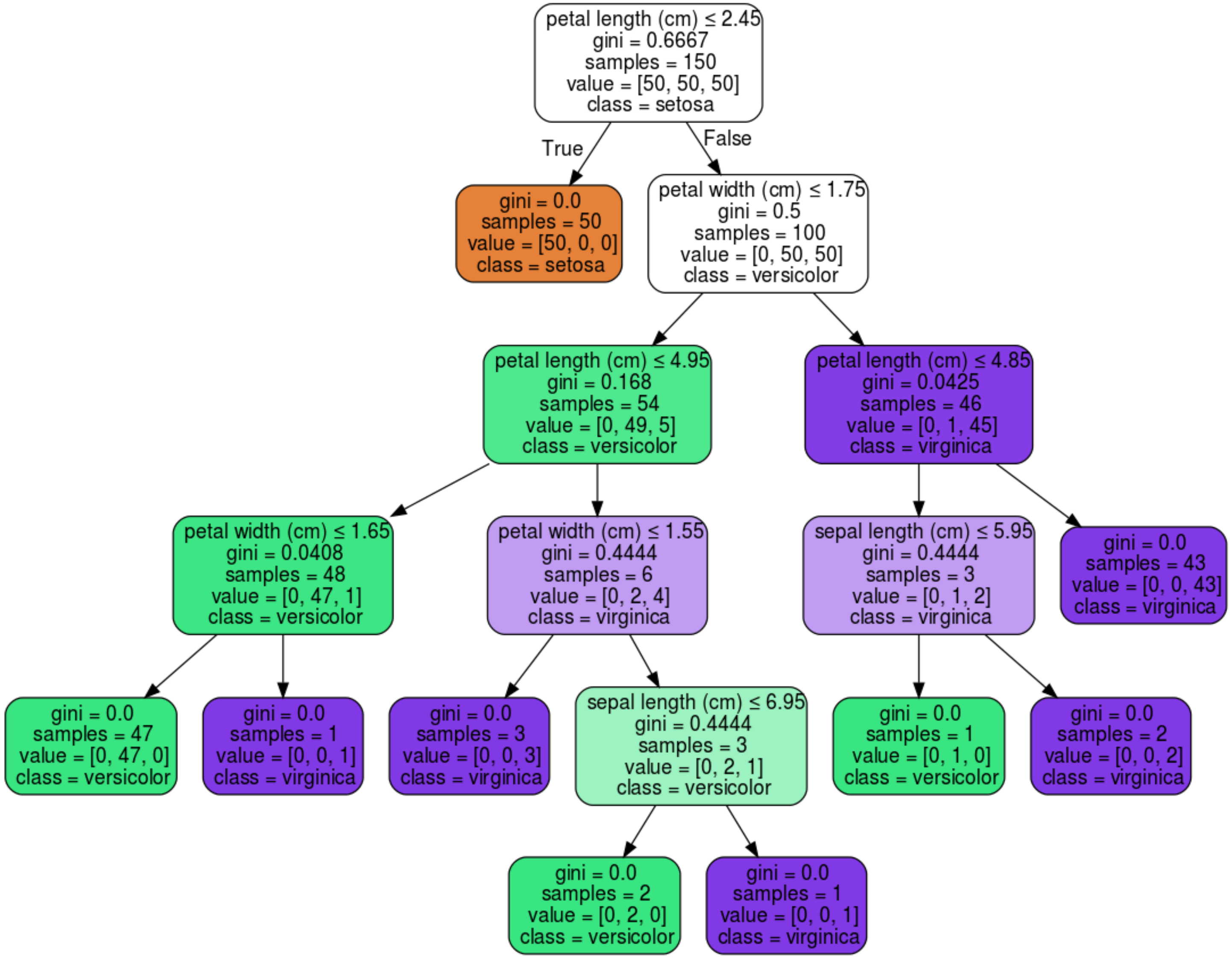
| humidity | Play Golf | |
|---|---|---|
| | yes | no |
| Normal | 6 | 1 |
| High | 3 | 4 |

AVC-set on *Windy*

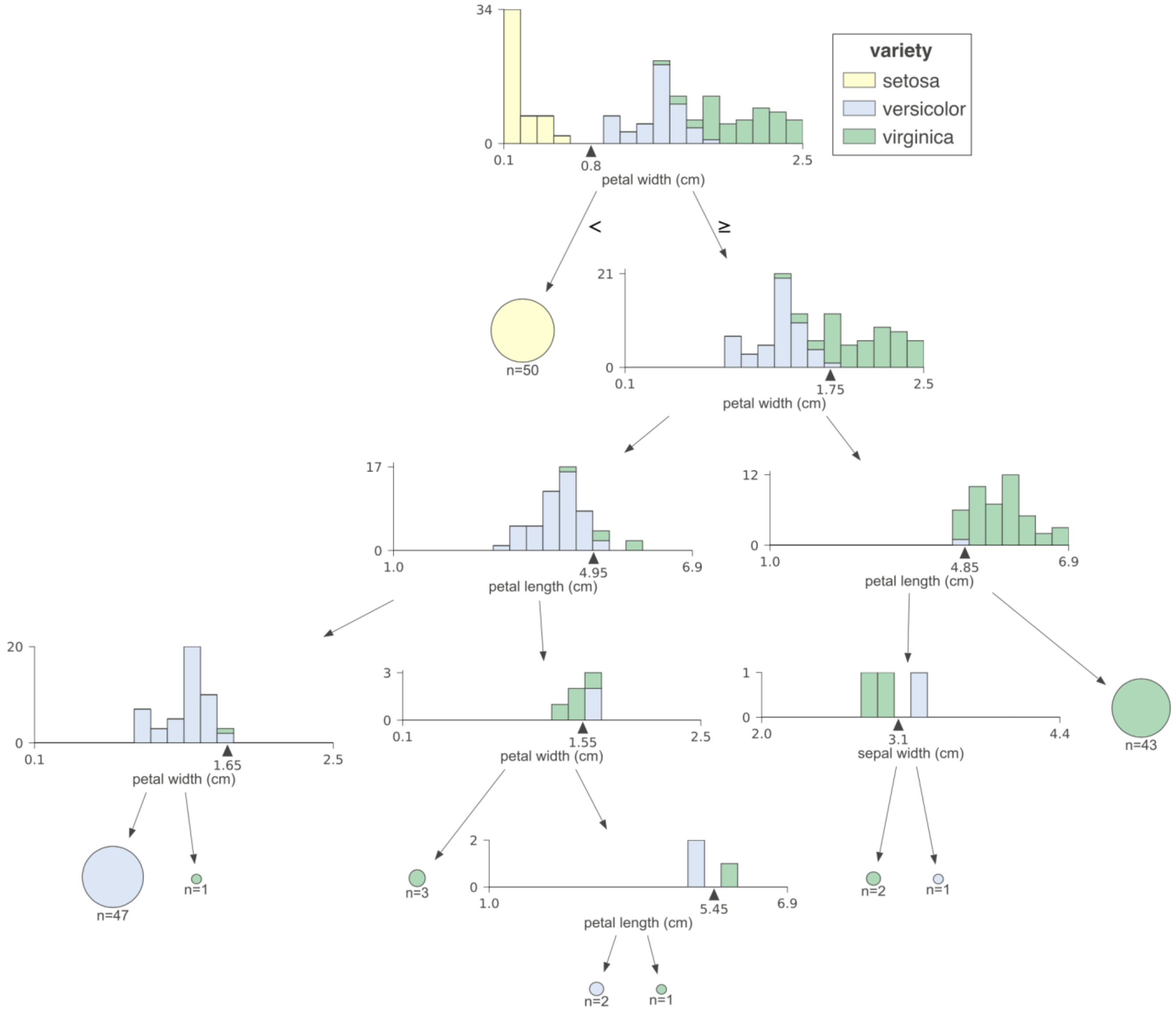
| windy | Play Golf | |
|---|---|---|
| | yes | no |
| False | 6 | 2 |
| True | 3 | 3 |

Its AVC Sets

# Visualization of a Decision Tree (in scikit-learn)



# Visualization of a Decision Tree



# Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Linear Classifier
- Model Evaluation and Selection
- Imbalanced Data Sets
- Additional Concepts on Classification
- Summary

# Bayes' Theorem: Basics

- Total probability Theorem:

$$p(B) = \sum_{i} p\,(B|A_i)p(A_i)$$

- Bayes' Theorem:

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

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 1: Cancer Tests

| | Cancer (1%) | No Cancer (99%) |
|---|---|---|
| Test Pos | 80% | 9.6% |
| Test Neg | 20% | 90.4% |

- Only 1% people have cancer
- How accurate is the test?
  - 80%? 99%? 1%?

| | Cancer (1%) | No Cancer (99%) |
|---|---|---|
| Test Pos | True Pos<br>1% x 80% = .008 | False Pos<br>99% x 9.6% = .09504 |
| Test Neg | False Neg<br>1% x 20% = .002 | True Neg<br>99% x 90.4% = .89496 |

- $P(XH) = P(X|H)P(H)$
- Chance of true positive is thus

$1\% * 80\% = 0.008$

- According to Bayes' Theorem, $P(H|X) = P(X|H)P(H)/P(X)$, the chance of having a cancer given positive test results is

True pos / (True pos + False pos) = 0.008 / (0.008+0.09504) = 7.76%

- The Theorem lets us correct for the skewness introduced by false positives

# Bayes' Theorem Example 2: 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%

- Again, 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 and reality, which is the most important trick in Bayesian Inference

# Naïve Bayes Classifier: Making a Naïve 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 assumption to simplify the complex dependencies: *features are conditionally independent!*
  - $P(C|X) \propto P(X|C)P(C) \approx P(x1|C)P(x2|C)...P(xn|C)P(C)$

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

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

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

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

# 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: <u>Likelihood Tables</u>

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 | ? |

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

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

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

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

# Naïve Bayes Classifier: Example 2

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 | 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 |

# Naïve Bayes Classifier: Example 2

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

| 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 |

# Naïve Bayes Classifier: Example 2

**X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

**$P(X|C_i)$ :**

**P(X|buys_computer = "yes")** =
P(age = "<=30"|buys_computer = "yes")
P(income = "medium" | buys_computer = "yes")
P(student = "yes" | buys_computer = "yes)
P(credit_rating = "fair" | buys_computer = "yes")
= 0.222 x 0.444 x 0.667 x 0.667 = 0.044

**P(X|buys_computer = "no")** =
P(age = "<= 30"|buys_computer = "no")
P(income = "medium" | buys_computer = "no")
P(student = "yes" | buys_computer = "no")
P(credit_rating = "fair" | buys_computer = "no")
= 0.6 x 0.4 x 0.2 x 0.4 = 0.019

| $P(X\|C_i)$ | C1 = yes | C2 = no |
|---|---|---|
| age <= 30 | 0.222 | 0.6 |
| Inc. = med. | 0.444 | 0.4 |
| Stu. = yes | 0.667 | 0.2 |
| Credit = fair | 0.667 | 0.4 |

Conditional probability

**$P(X|C_i)*P(C_i)$ :**

P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028
P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

**Therefore, X is classified to class ("buys_computer = yes")**

# Avoiding the Zero-Probability Problem

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

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

- Example. 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/(1000 + 3)

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

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

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

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

# 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?

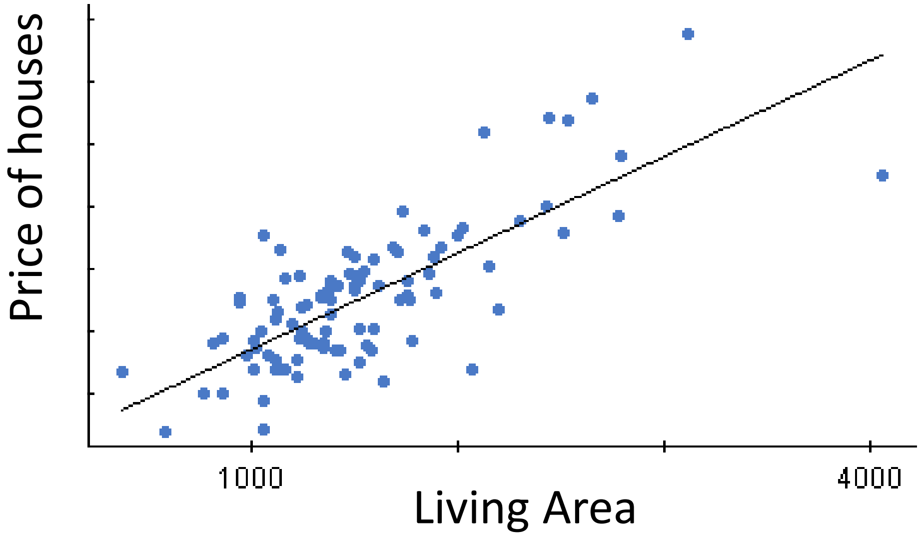
    Use Bayesian Belief Networks (to be covered in the next chapter)

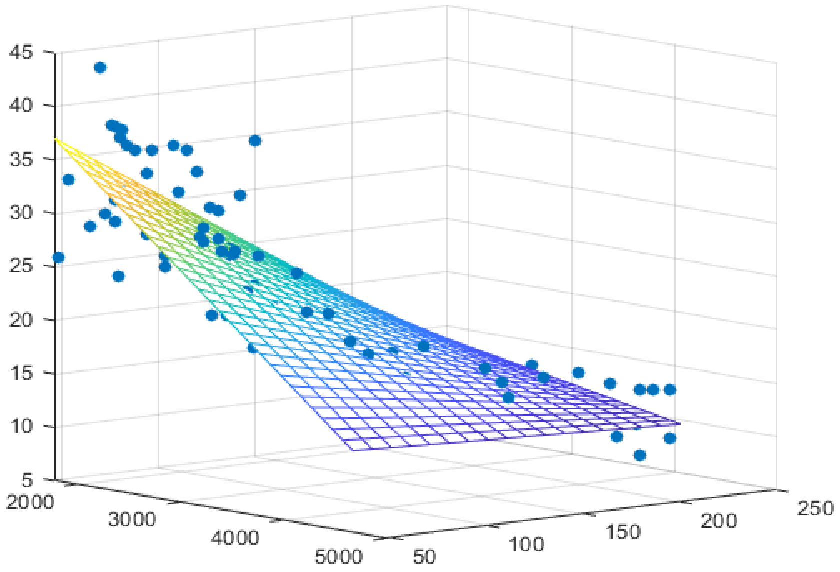
# Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Linear Classifier
- Model Evaluation and Selection
- Imbalanced Data Sets
- Additional Concepts on Classification
- 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

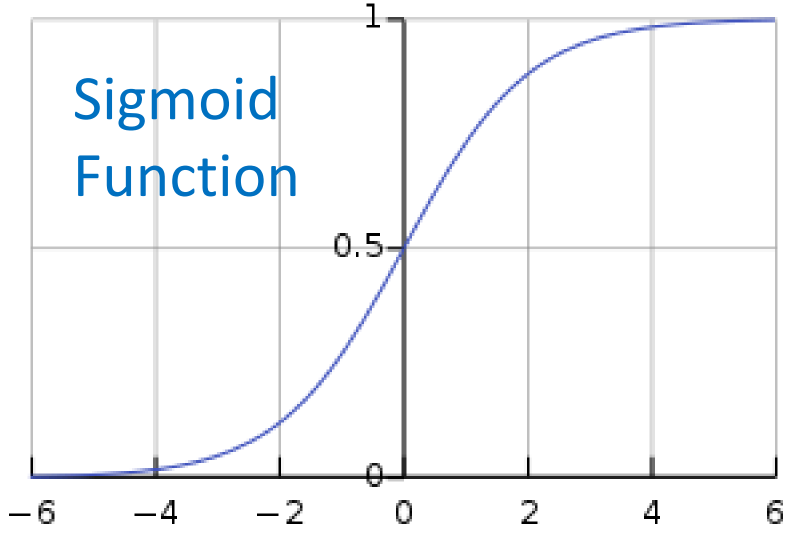
- Linear regression
  - Data: n independent objects
    - Observed Value: $y_i, i = 1,2,3,\cdots,n$
    - p-dimensional attributes: $x_i = \left(x_{i1}, x_{i2}, \cdots, x_{ip}\right)^T,\ i = 1,2,3\cdots,n$
  - Model:
    - Weight vector: $w = \left(w_1, w_2, \cdots, w_p\right)$
    - $y_i = w^T x_i +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

# Linear Regression Model: Solution

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

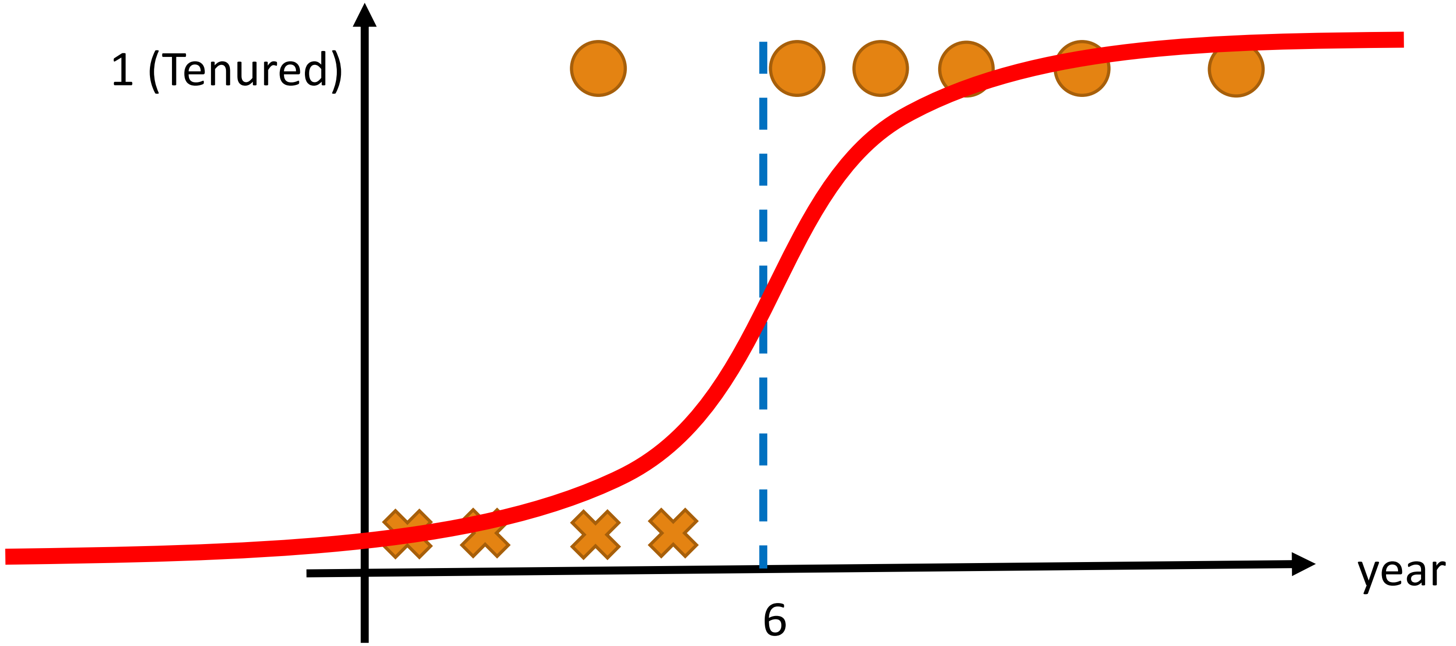
# Logistic Regression: General Ideas

- 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) :
  - $\sigma(y) = \frac{1}{1+e^{-y}} = \frac{e^{y}}{e^{y}+1}$
  - Projects $(-\infty, +\infty)$ to $[0, 1]$
  - Not only LR uses this function, but also neural network, deep learning
- The projected value change sharply around zero point
- Notice that $\ln \frac{y}{1-y} = w^{T}x + b$

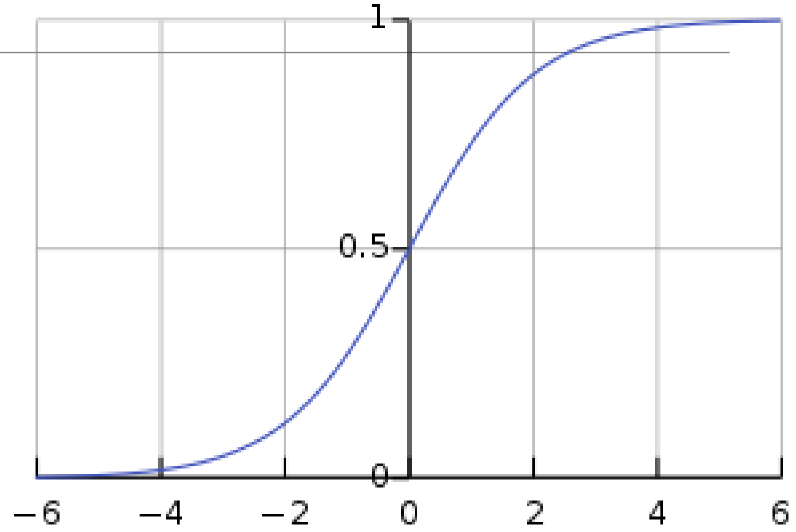


# Logistic Regression: An Example

- 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:
  - $p(Y = 1\ |X = x; \boldsymbol{w}) \ = \ Sigmoid(w_0 + \sum_{i=1}^{n} w_i \cdot x_i)$
  - $\boldsymbol{w} = (w_0, w_1, w_2, \dots, w_n)$ are the parameters

- A single data object with attributes $x_i$ and class label $y_i$
  - Suppose the probability of $p(\widehat{y_i} = 1|x_i, w) = p_i$, then $p(\widehat{y_i} = 0|x_i, w) = 1 - p_i$
  - $p(\widehat{y_i} = y_i) = p_i^{y_i}(1 - p_i)^{1-y_i}$

- 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:

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

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

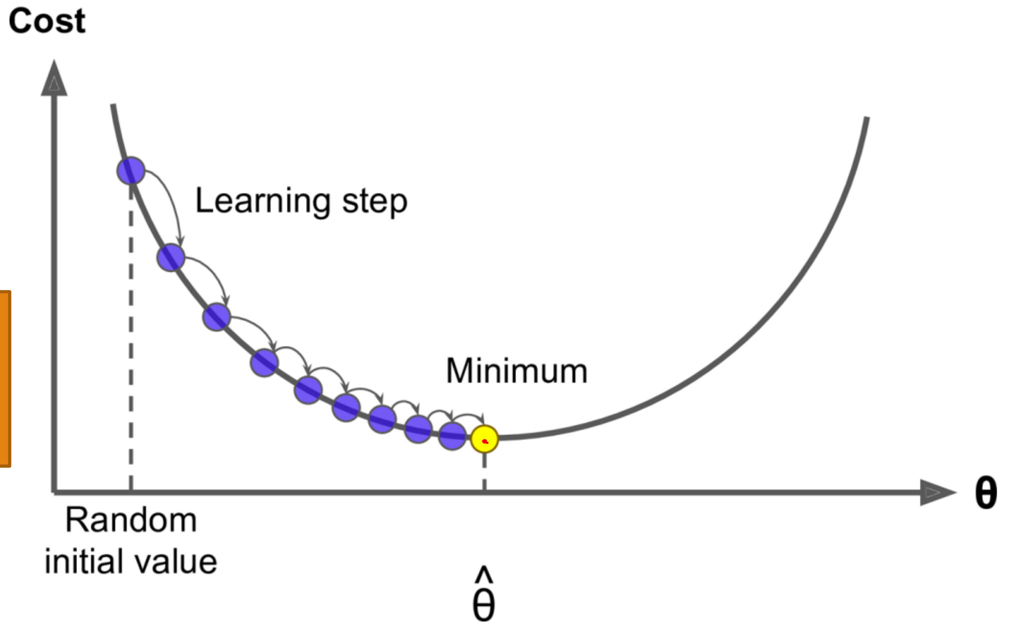
- There's no closed form solution
  - Gradient Descent

# 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

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




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

# Linear Regression VS. Logistic Regression

- Linear Regression
  - Y: Continuous Value $\in [-\infty, +\infty]$
  - $Y = W^T X + b$
  - Often used in value prediction problems

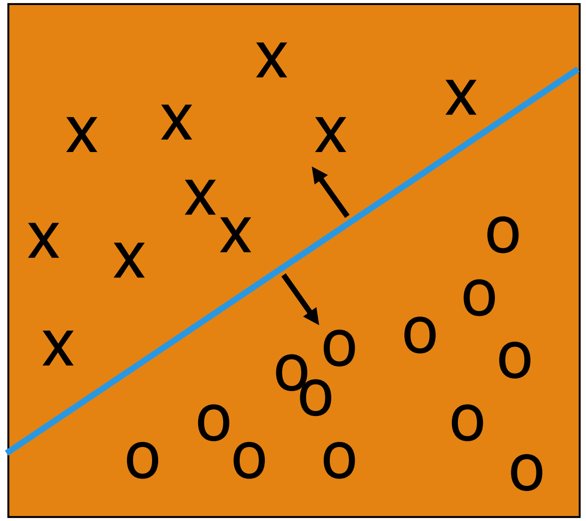
- Logistic Regression
  - Y: A discrete value from m classes
  - $P(Y = C_i) \in [0,1]\ \ and\ \ \Sigma_{i=1}^{m} P(Y = C_i) = 1$
  - Often used in classification problems

# Comments on Logistic Regression

- Pros
  - Can handle multiple types of features
  - Fast and easy
  - Generally speaking, more robust and better performance than tree
  - Interpretable: both weights and predicted value
    - Predicted value: probability
    - Weights: effect of the feature. Unit change of log odds
- Cons
  - Linear model: if the decision boundary is not linear, then LR is not good

# Linear Classifier: General Ideas

- Binary Classification
- $f(x)$ is a linear function based on the example's attribute values
  - The prediction is based on the value of $f(x)$
  - Data above the blue line belongs to class 'x' (i.e., $f(x) > 0$)
  - Data below blue line belongs to class 'o' (i.e., $f(x) < 0$)
- Classical Linear Classifiers
  - Logistic Regression
  - Linear Discriminant Analysis (LDA)
  - Perceptron
  - SVM

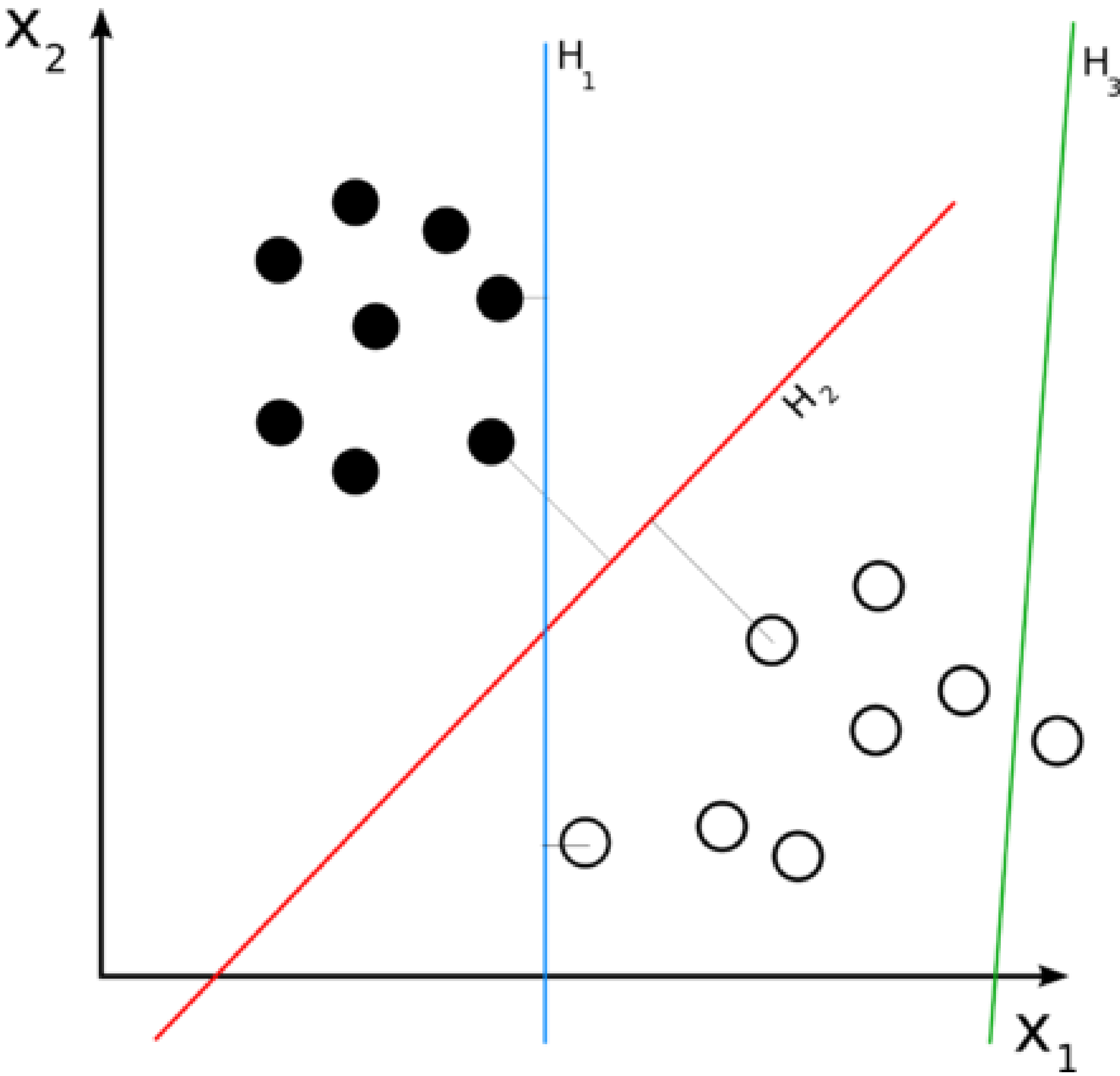


# Linear Classifier: An Example

- A toy rule to determine whether a faculty member has tenure
  - Year >= 6 or Title = "Professor" ⇔ Tenure
- How to express the rule as a linear classifier?
- Features
  - $\mathrm{x}_1 (x_1 \geq 0)$ is an integer denoting the year
  - $\mathrm{x}_2$ is a Boolean denoting whether the title is "Professor"
- A feasible linear classifier: $f(x) = (x_1 - 5) + 6 \cdot x_2$
  - When $x_2$ is True, because $x_1 \geq 0$, $f(x)$ is always greater than 0
  - When $x_2$ is False, because $f(x) > 0 \Leftrightarrow x_1 \geq 6$
- There are many more feasible classifiers
  - $f(x) = (x_1 - 5.5) + 6 \cdot x_2$
  - $f(x) = 2 \cdot (x_1 - 5) + 11 \cdot x_2$
  - ……

# Key Question: Which Line Is Better?

- There might be many feasible linear functions
  - Both H1 and H2 will work
- Which one is better?
  - H2 looks "better" in the sense that it is also furthest from both groups
  - We will introduce more in the SVM section



# Generative vs. Discriminative Classifiers

- X: observed variables (features)
- Y: target variables (class labels)
- A generative classifier models p(Y, X)
    - It models how the data was "generated"? "what is the likelihood this or that class generated this instance?" and pick the one with higher probability
  - Naïve Bayes
  - Bayesian Networks
- A discriminative classifier models p(Y|X)
    - It uses the data to create a decision boundary
  - Logistic Regression
  - Support Vector Machines

# Further Comments on Discriminative Classifiers

- Strength
  - Prediction accuracy is generally high
    - As compared to generative models
    - Robust, works when training examples contain errors
  - Fast evaluation of the learned target function
    - Comparing to (covered in future) Bayesian networks (which are normally slow)
- Criticism
  - Long training time
  - Difficult to understand the learned function (weights)
    - Bayesian networks can be used easily for pattern discovery
  - Not easy to incorporate domain knowledge
    - Easy in the form of priors on the data or distributions

# Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Linear Classifier
- Model Evaluation and Selection
- Imbalanced Data Sets
- Additional Concepts on Classification
- 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
- 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 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**

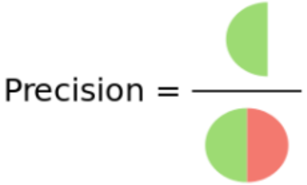
- **Class imbalance problem**
  - One class may be *rare*
    - E.g., fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class
  - Measures handle the class imbalance problem
    - **Sensitivity** (recall): True positive recognition rate
      - **Sensitivity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

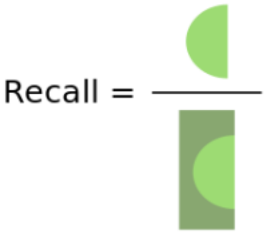
- **Precision**: Exactness: what % of tuples that the classifier labeled as positive are actually positive?

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

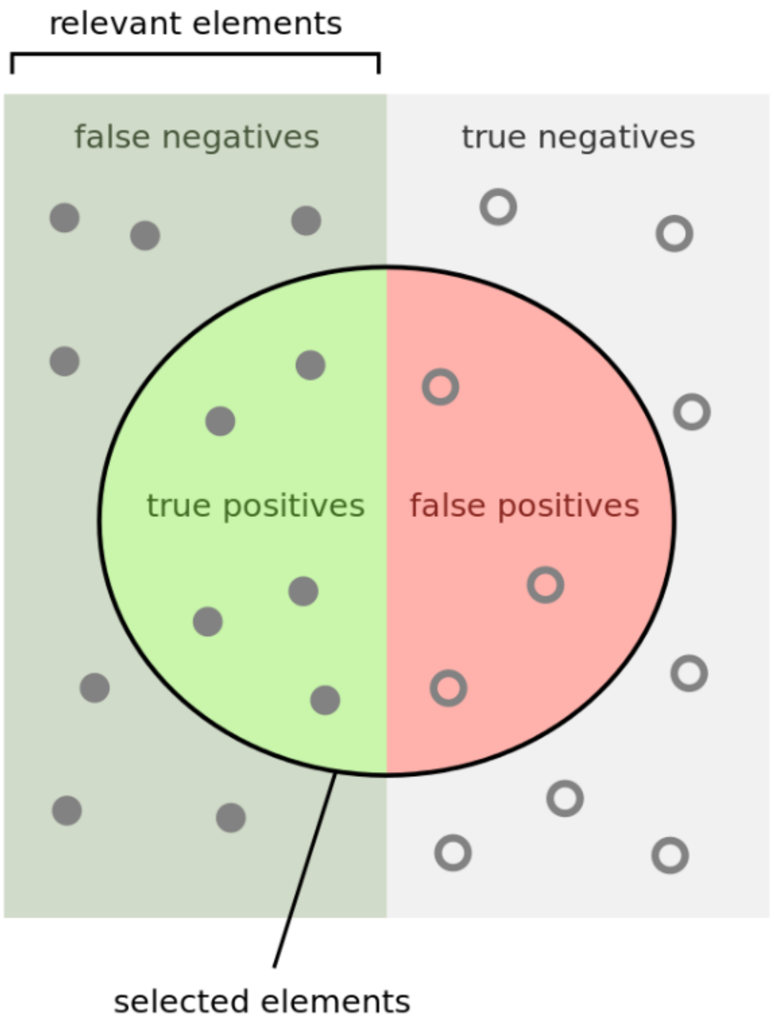



- **Recall:** Completeness: what % of positive tuples did the classifier label as positive?

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




- 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

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

Assigning β times as much weight to recall as to precision)

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

$$\mathrm{F}_1 = \frac{2\mathrm{P} * \mathrm{R}}{\mathrm{P} + \mathrm{R}}$$

# 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 =

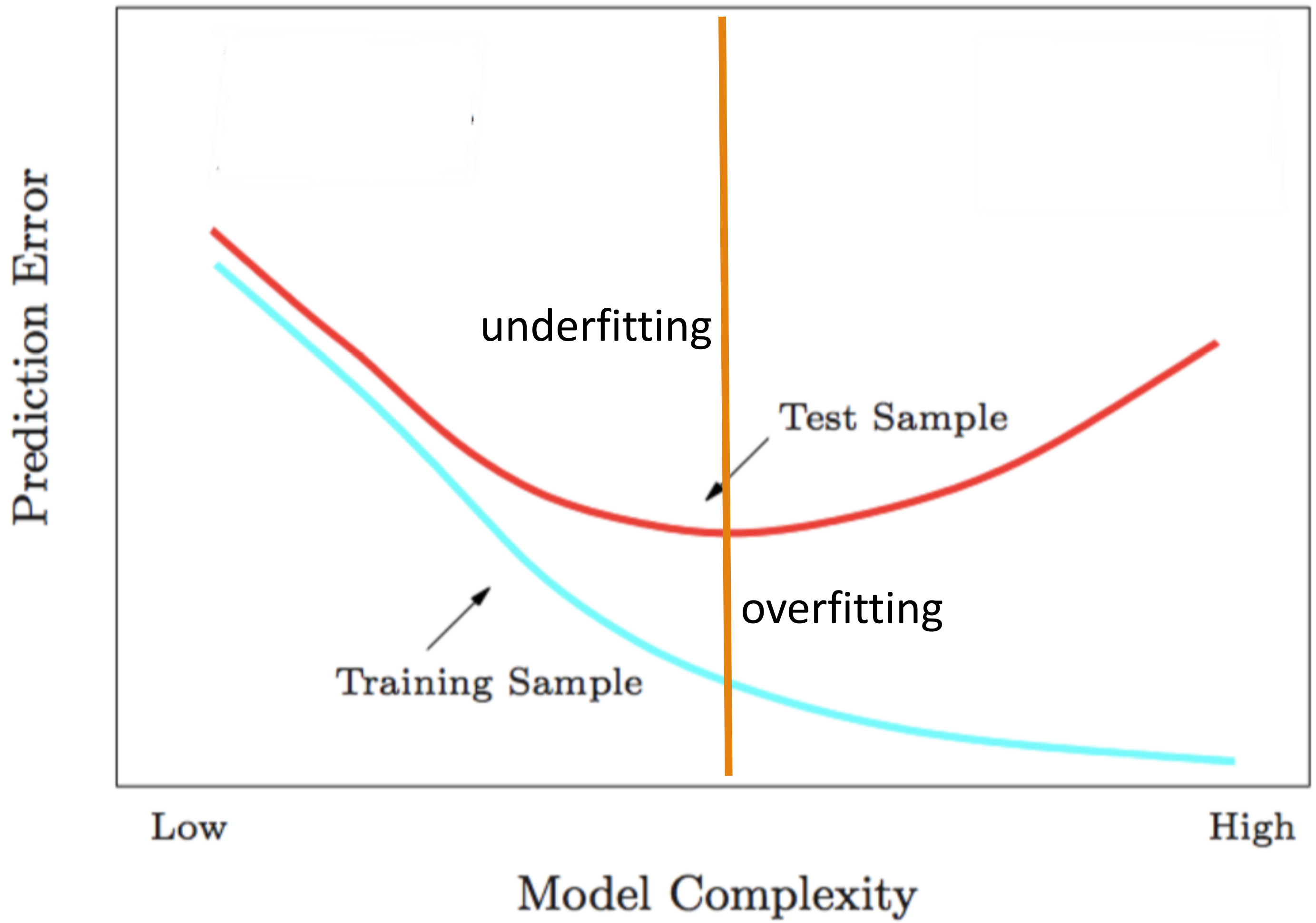
# Classifier Evaluation Metrics: Example

- 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





74

# 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

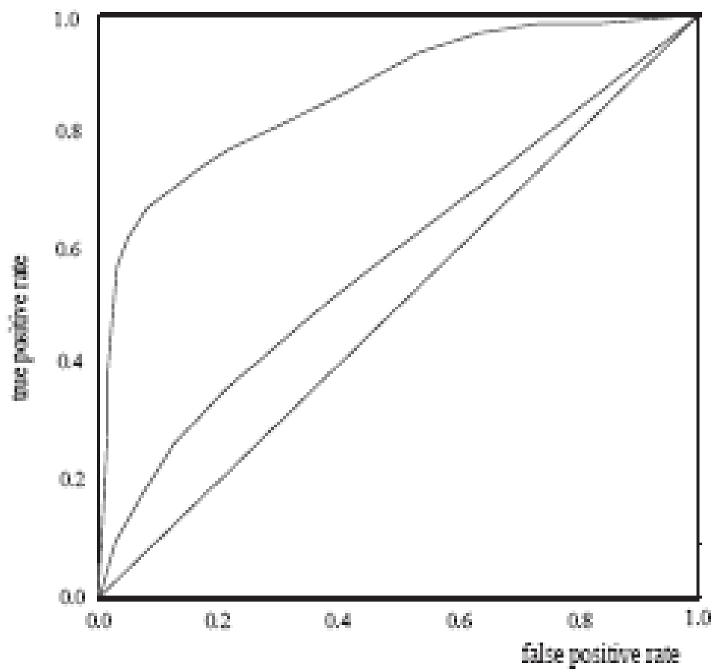
# Classifier Evaluation: Bootstrap

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

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

# Model Selection: ROC Curves

- **ROC** (Receiver Operating Characteristics) curves: for visual comparison of classification models
- 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

# Issues Affecting Model Selection

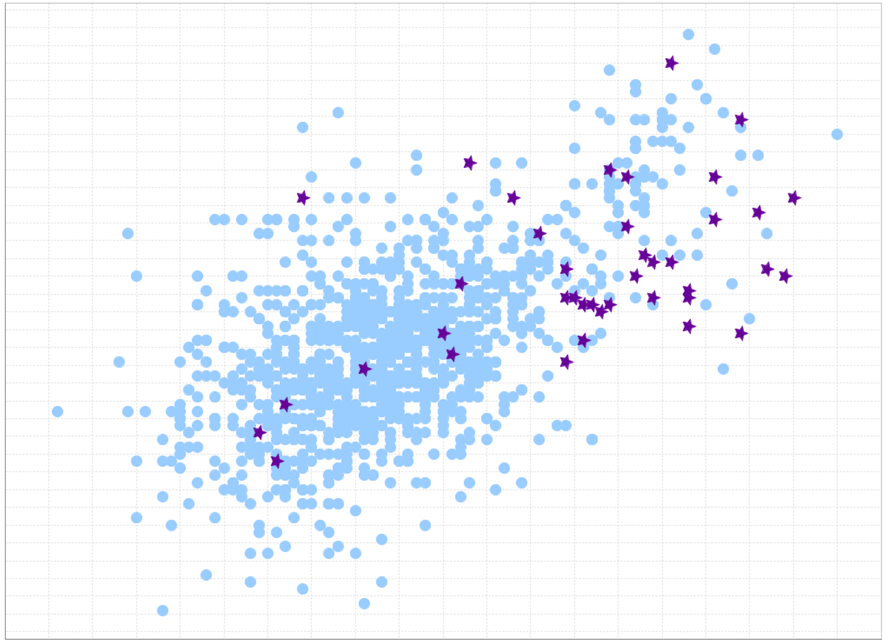
- **Accuracy**
  - classifier accuracy: predicting class label
- **Speed**
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- **Robustness**: handling noise and missing values
- **Scalability**: efficiency in disk-resident databases
- **Interpretability**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

# Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Linear Classifier
- Model Evaluation and Selection
- Imbalanced Data Sets
- Additional Concepts on Classification
- Summary

# Classification of Class-Imbalanced Data Sets

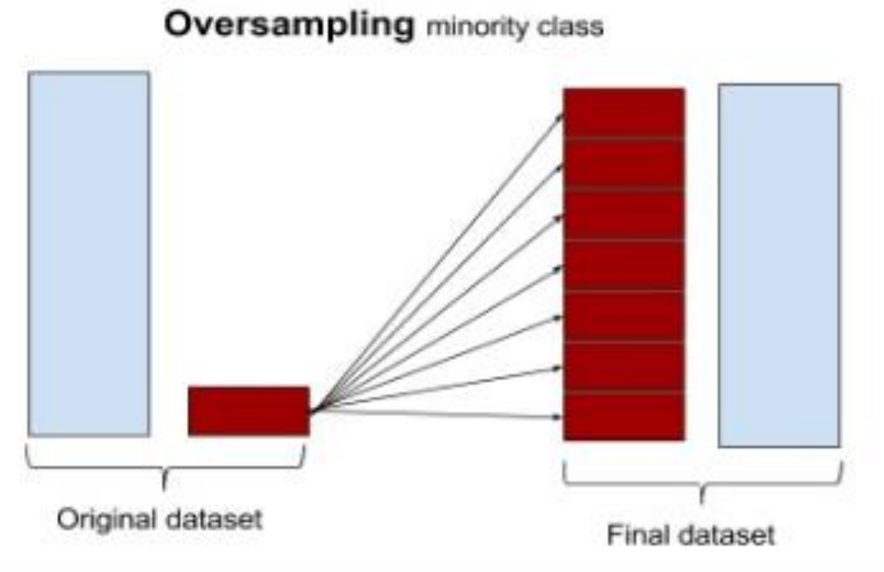
- Traditional methods assume a balanced distribution of classes and equal error costs. But in real world situations, we may face imbalanced data sets, which has rare positive examples but numerous negative ones.

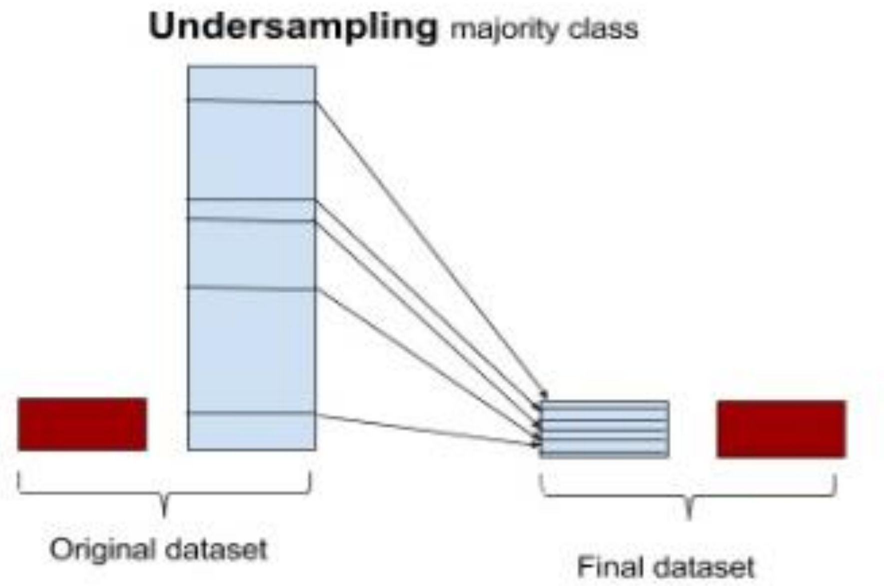


  - Medical diagnosis: Medical screening for a condition is usually performed on a large population of people without the condition, to detect a small minority with it (e.g., HIV prevalence in the USA is ~0.4%)
  - Fraud detection: About 2% of credit card accounts are defrauded per year. (Most fraud detection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

# Classification of Class-Imbalanced Data Sets

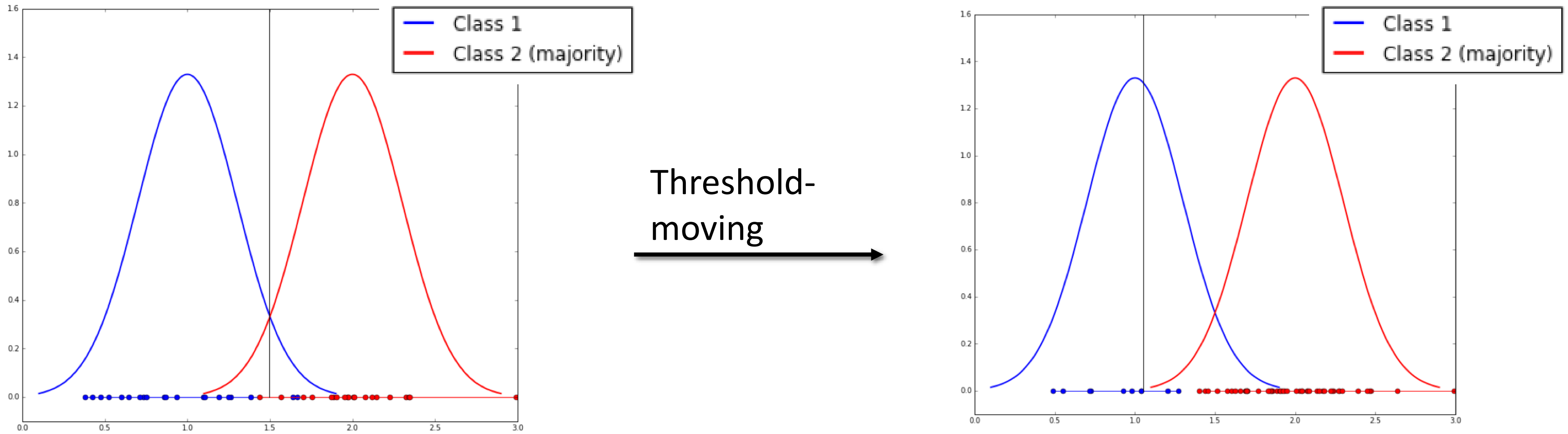
- Typical methods on imbalanced data (Balance the training set)
  - **Oversampling**: Oversample the minority class.
  - **Under-sampling**: Randomly eliminate tuples from majority class
  - **Synthesizing:** Synthesize new minority classes



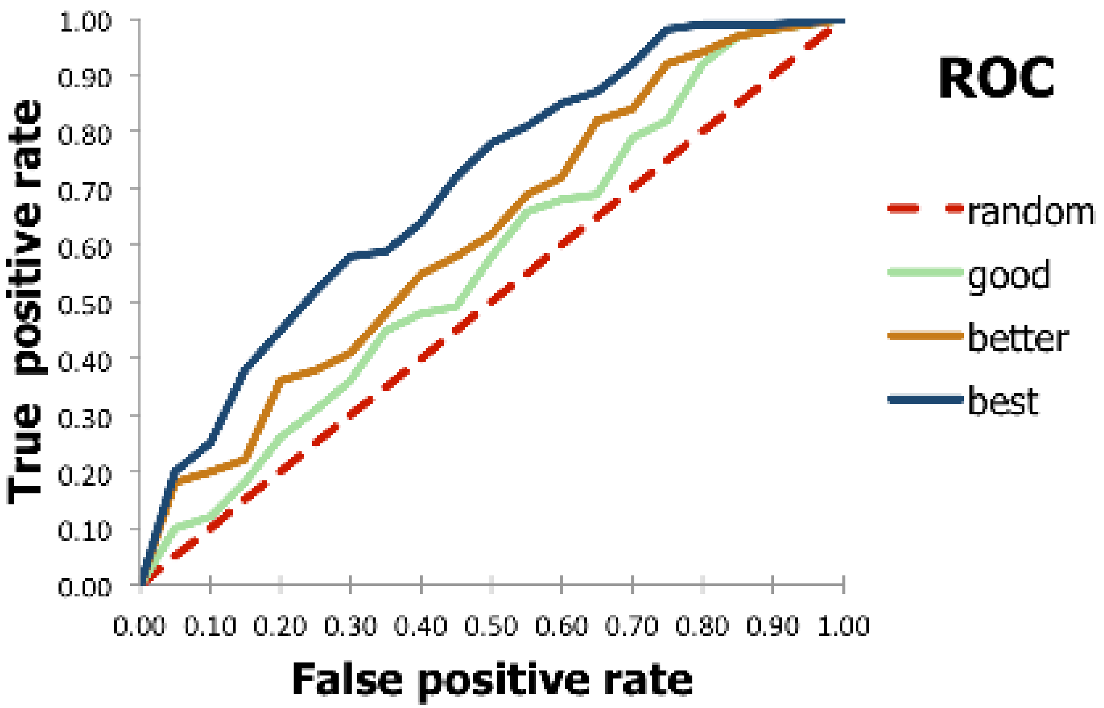
# Classification of Class-Imbalanced Data Sets

- 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

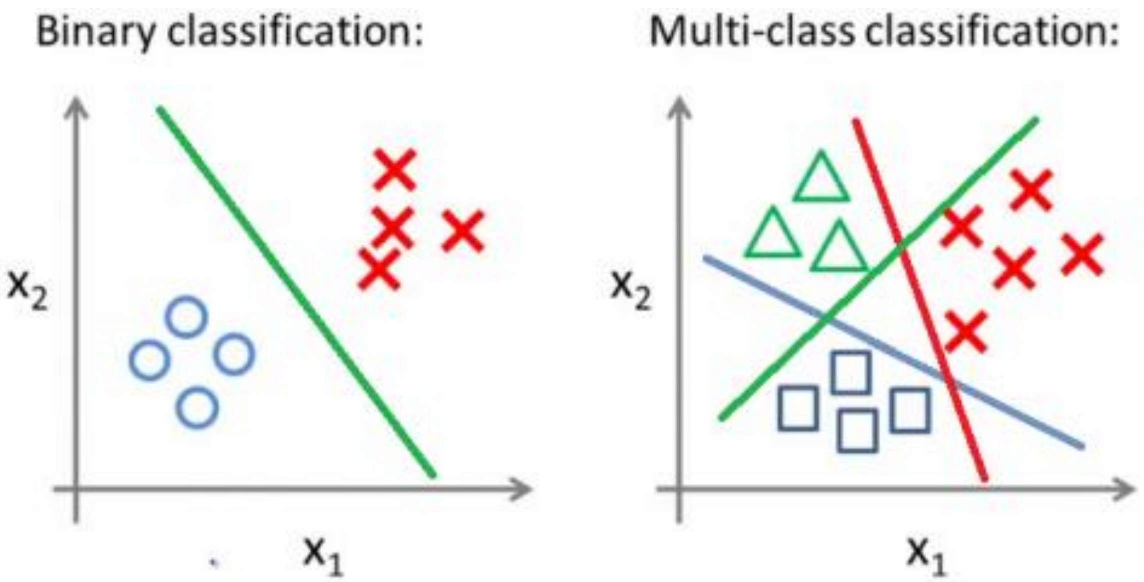


# Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Linear Classifier
- Model Evaluation and Selection
- Imbalanced Data Sets
- Additional Concepts on Classification
- Summary

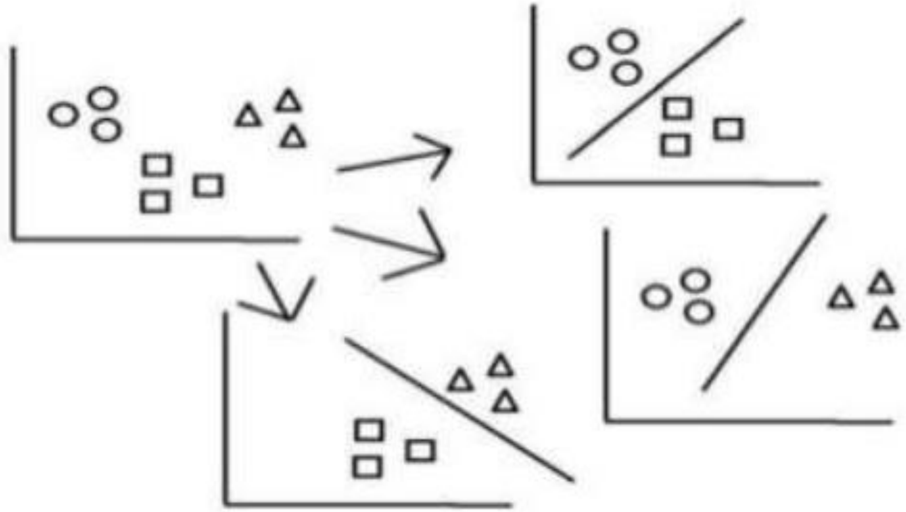
# Multiclass Classification

- Classification involving more than two classes (i.e., > 2 Classes)
- Methodology: Reducing the multi-class problem into multiple binary problems
- Method 1. **One-vs.-rest** (or **one-vs.-all**)
  - Given *m* classes, train *m* classifiers: one for each class
  - Classifier j: treat tuples in class j as *positive* & **all the rest** as *negative*
  - To classify a tuple **X**, the set of classifiers vote as an ensemble



# Multiclass Classification

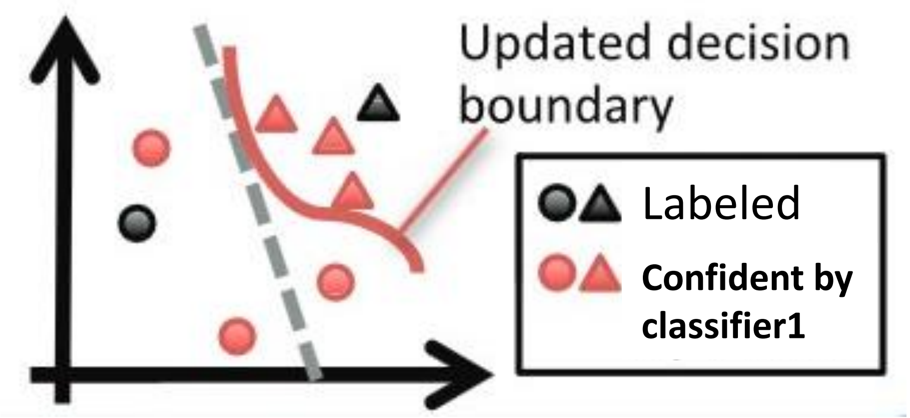
- Method 2. **one-vs.-one** (or **all-vs.-all**): Learn a classifier for each pair of classes
  - Given *m* classes, construct $m(m-1)/2$ binary classifiers
  - A classifier is trained using tuples of the two classes
  - To classify a tuple **X**, each classifier votes
    - X is assigned to the class with maximal vote



- Comparison: One-vs.-one tends to perform better than one-vs.-rest
- Many new algorithms have been developed to go beyond binary classifier method

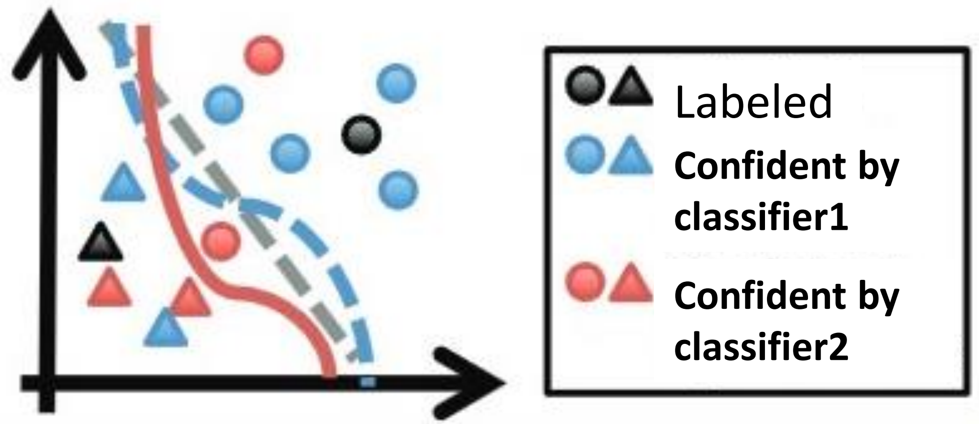
# Semi-Supervised Classification

- Semi-supervised: Uses labeled and unlabeled data to build a classifier
- Self-training
  - 1. Build a classifier using the labeled data
  - 2. Use it to label the unlabeled data, and those with the most confident label prediction are added to the set of labeled data
  - 3. Repeat the step 1 and 2
  - Adv.: easy to understand; Disadv.: may reinforce errors



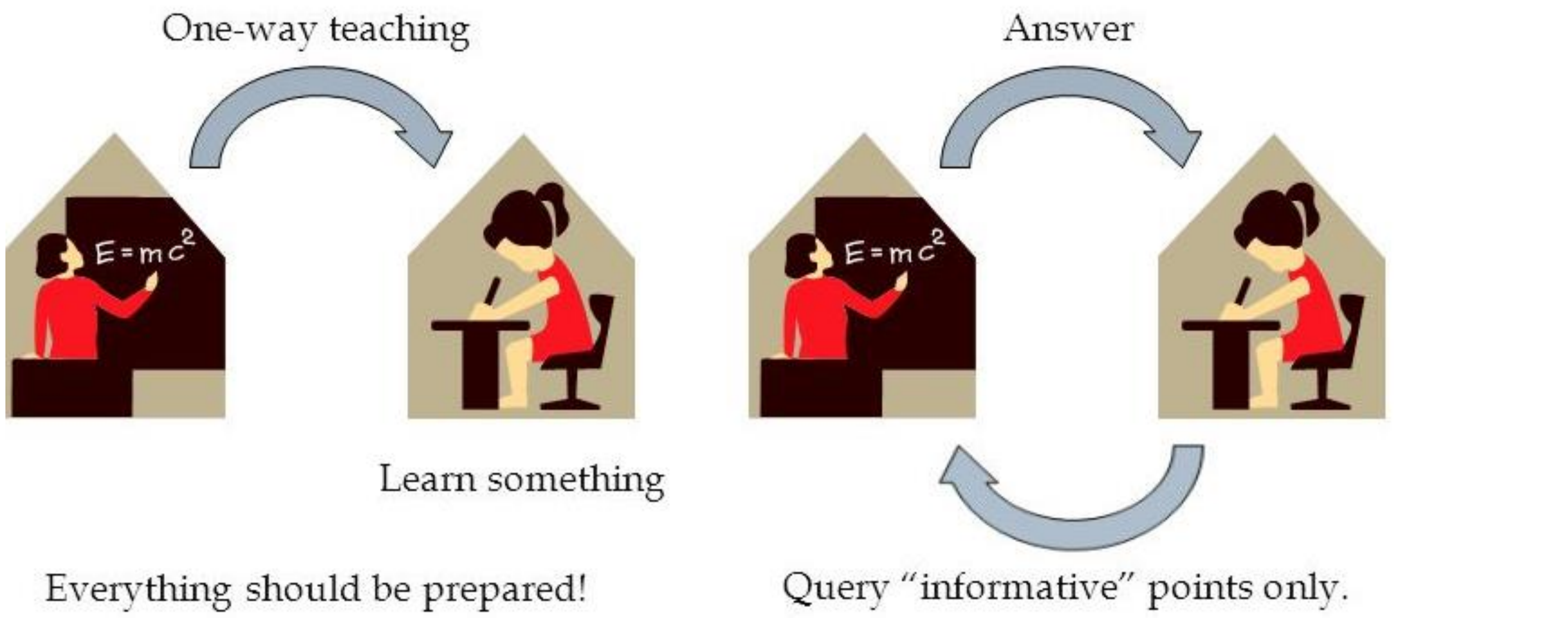
# Semi-Supervised Classification

- Semi-supervised: Uses labeled and unlabeled data to build a classifier
- Co-training: Use two or more classifiers to teach each other
  - Each learner uses a mutually independent set of features of each tuple to train a good classifier, say $f_1$ and $f_2$
  - Then $f_1$ and $f_2$ are used to predict the class label for unlabeled data X
  - Teach each other: The tuple having the most confident prediction from $f_1$ is added to the set of labeled data for $f_2$ & vice versa
- Other methods include joint probability distribution of features and labels



# Active Learning

- A special case of semi-supervised learning
- Active learner: Interactively query teachers (oracle) for labels of “informative” data

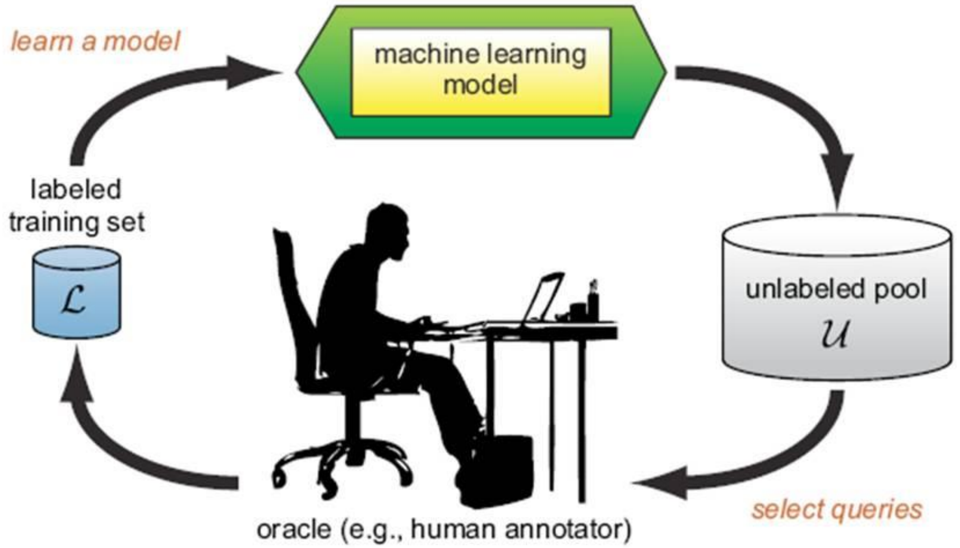



(a) Passive Learning

(b) Active Learning

# Active Learning

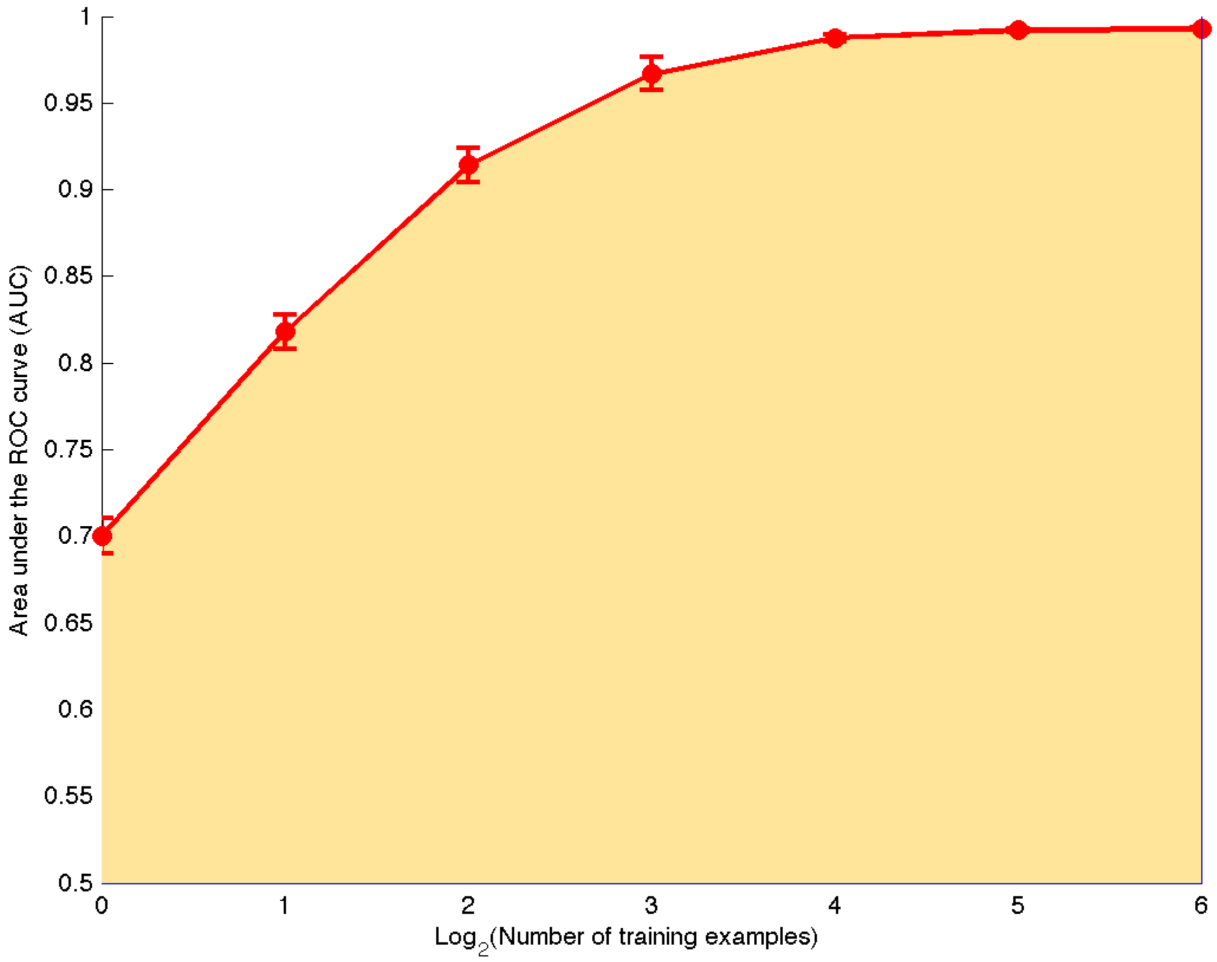
- Pool-based approach: Uses a pool of unlabeled data
  - L: a small subset of D is labeled, U: a pool of unlabeled data in D
  - Use a query function to carefully select one or more tuples from U and request labels from an oracle (a human annotator)
  - The newly labeled samples are added to L, and learn a model
  - Goal: **Achieve high accuracy using as few labeled data as possible**



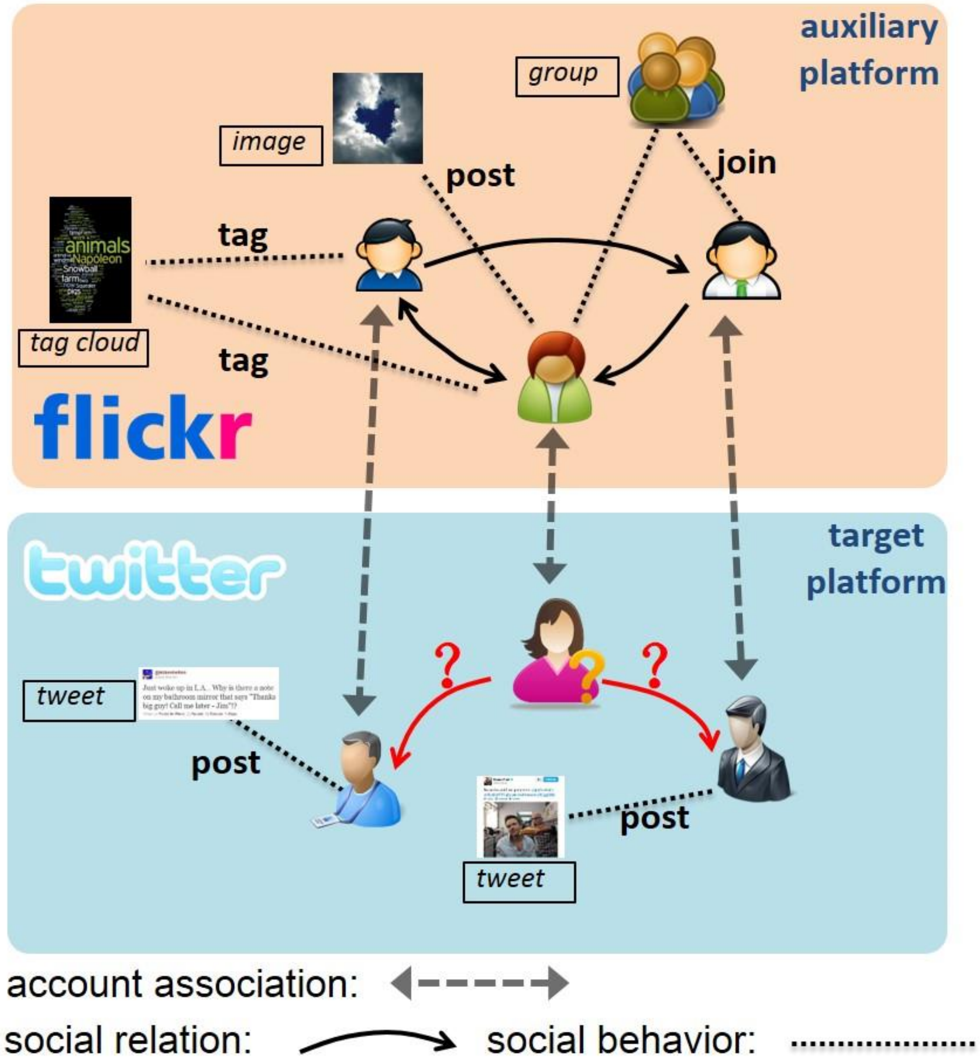

One good choice: unlabeled point closest to the current decision boundary

# Active Learning

- Evaluated using *learning curves*: Accuracy as a function of the number of instances queried (# of tuples to be queried should be small)



# Transfer Learning




- **Traditional learning**: Build a new classifier for each new task
- **Transfer learning:** Extract knowledge from one or more source tasks (e.g., recognizing cars) and apply the knowledge to a target task (e.g., recognizing trucks)
- **Example: Cross-platform friend recommendation** [1]
  - Users' social relation and behavior in one platform(flickr) offers important knowledge about social interest in another platform(Twitter)



[1]http://nlpr-web.ia.ac.cn/mmc/homepage/myan/Project_YanMing/ming_ICME2013/material/ICME2013Final.pdf

# Weak Supervision: A New Programming Paradigm for Machine Learning

- Overcome the training data bottleneck
  - Leverage higher-level and/or noisier input from experts
- Sources of cheaply and efficiently provided weak labels:
  - Higher-level, less precise supervision (e.g., heuristic rules, expected label distributions)
  - Cheaper, lower-quality supervision (e.g. crowdsourcing)
  - Existing resources (e.g. knowledge bases, pre-trained models)

# Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Linear Classifier
- Model Evaluation and Selection
- Imbalanced Data Sets
- Additional Concepts on Classification
- 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)
- Improve Classification Accuracy: Bagging, boosting
- Additional concepts on classification: Multiclass classification, semi-supervised classification, active learning, transfer learning, weak supervision

# Prepare for Exam

- Do calculations Step by step
  - Decision tree, Naive Bayes classification methods
  - Accuracy, sensitivity, specificity, precision, recall, *F* measure
  - Prediction for logistic regression (with provided parameters)
- Concepts
  - What does a term mean
  - Describe the procedure
  - Pros and cons for the classification methods
  - Interpret the results

# 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. 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

# Bayes' Theorem: Basics

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

# Classification Is to Derive the Maximum Posteriori

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

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

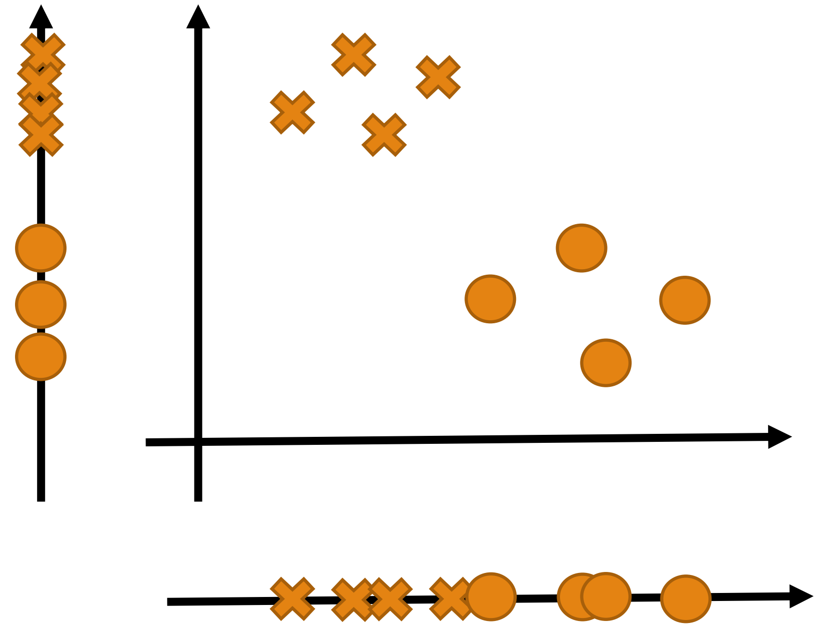
- Since P(X) is constant for all classes, only

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

needs to be maximized

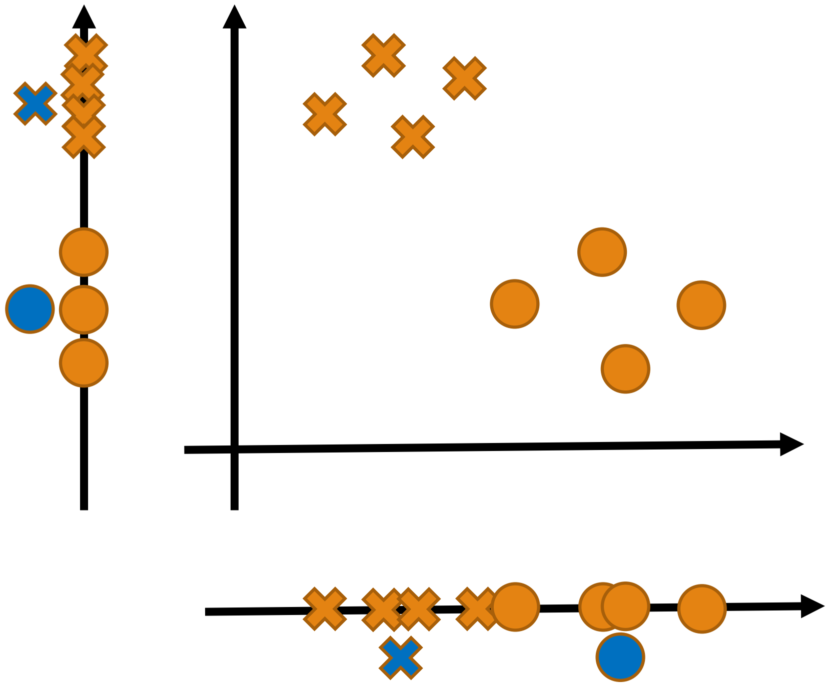
# Linear Discriminant Analysis (LDA)

- Linear Discriminant Analysis (LDA) works when the attributes are all continuous
  - For the categorical attributes, discriminant correspondence analysis is the equivalent technique
- Basic Ideas: Project all samples on a line such that different classes are well separated
- Example: Suppose we have 2 classes and 2-dimensional samples $x_1, \dots, x_n$
  - $n_1$ samples come from class 1
  - $n_2$ samples come from class 2
- Let the line direction be given by unit vector $\boldsymbol{v}$
- There are two candidates of projections
  - Vertical: $\boldsymbol{v} = (0,1)$
  - Horizontal: $\boldsymbol{v} = (1,0)$
- Which one looks better?
- How to mathematically measure it?



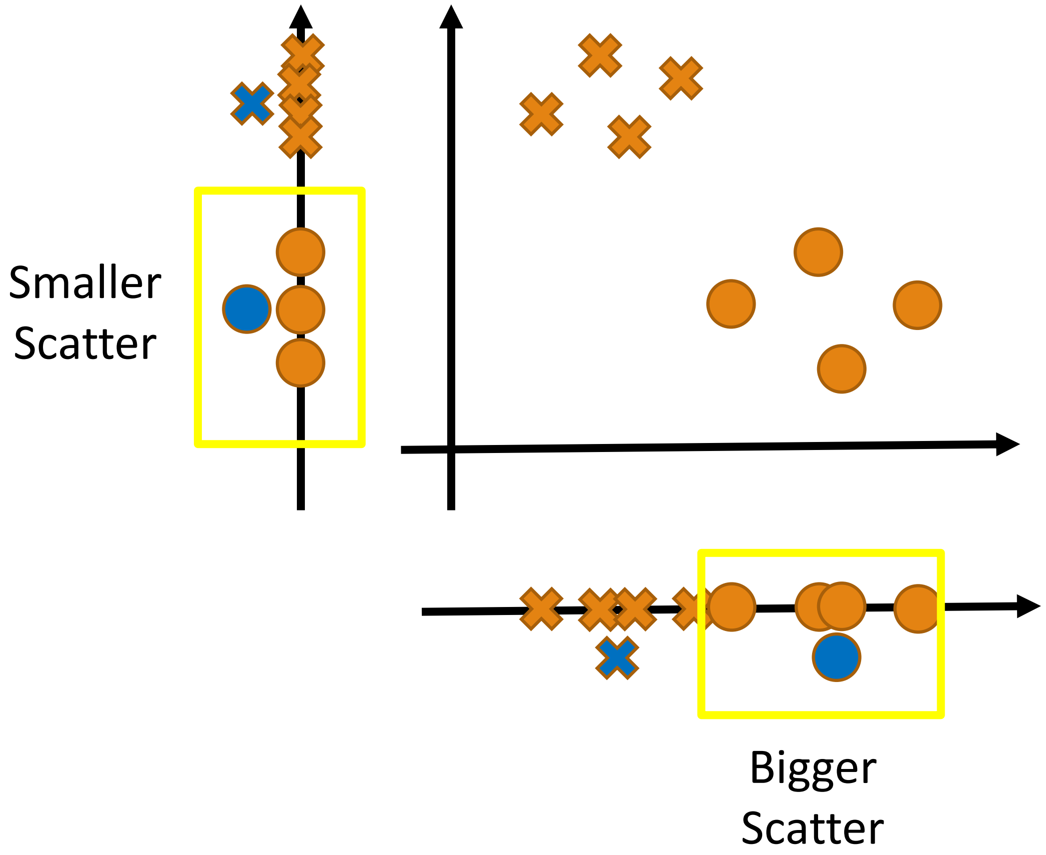
# Fisher's LDA (Linear Discriminant Analysis)

- $\boldsymbol{v}^T\boldsymbol{x_i}$ is the distance of projection of $\boldsymbol{x_i}$ from the origin
- Let $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ be the means of class 1 and class 2 in the original space
  - $\boldsymbol{\mu_1} = \frac{1}{n_1}\sum_{i\in\text{class 1}}\boldsymbol{x_i}$
  - $\boldsymbol{\mu_2} = \frac{1}{n_2}\sum_{i\in\text{class 2}}\boldsymbol{x_i}$
- The distance between the means of the projected points
  - $|\boldsymbol{v}^T\boldsymbol{\mu_1} - \boldsymbol{v}^T\boldsymbol{\mu_2}|$
  - Good? No. Horizontal one may have larger distance



# Fisher's LDA (con't)

- Normalization needed
- Scatter: Sample variance multiplied by $n$
  - $s_1 = \sum_{i \in \text{class } 1} \left(\boldsymbol{v}^T \boldsymbol{x}_i - \boldsymbol{v}^T \boldsymbol{\mu}_1\right)^2$
  - $s_2 = \sum_{i \in \text{class } 2} \left(\boldsymbol{v}^T \boldsymbol{x}_i - \boldsymbol{v}^T \boldsymbol{\mu}_2\right)^2$
- Fisher's LDA
  - Maximize $J(\boldsymbol{v}) = \frac{\left(\boldsymbol{v}^T \boldsymbol{\mu}_1 - \boldsymbol{v}^T \boldsymbol{\mu}_2\right)^2}{s_1 + s_2}$
  - Closed-form optimal solution



# Fisher's LDA: Summary

- Advantages
  - Useful for dimension reduction
  - Easy to extend to multi-classes

- Fisher's LDA will fail
  - When $\boldsymbol{\mu_1} = \boldsymbol{\mu_2}$, $J(\boldsymbol{v})$ is always 0.
  - When classes have large overlap when projected to any line