

Lecture 8: Document Classification I

Pilsung Kang

School of Industrial Management Engineering

Korea University

AGENDA

01 Document Classification: Overview

02 Naive Bayesian Classifier

03 k-Nearest Neighbor Classifier

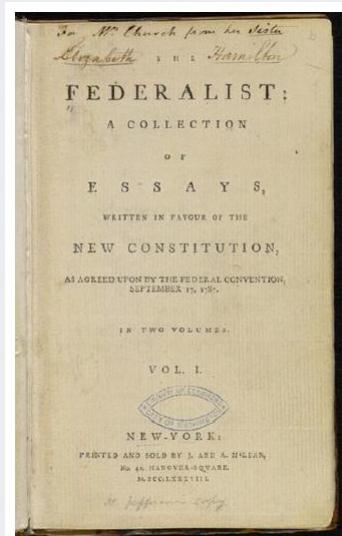
04 Classification Tree

05 Support Vector Machine

06 R Exercise

Document Classification: Examples

- Who wrote which federalist papers?
 - ✓ 1787-8: anonymous essays try to convince New York to ratify U.S Constitution: Jay, Madison, Hamilton
 - ✓ Authorship of 12 of the letters in dispute
 - ✓ 1963: solved by Mosteller and Wallace using Bayesian methods



James Madison



Alexander Hamilton

Document Classification: Examples

- Positive or Negative reviews?



Tied for the best movie I have ever seen

★★★★★

Author: carflo from Texas



A classic piece of unforgettable film-making.

★★★★★

Author: Justin M (kaspen12) from Vancouver, Canada



Simply amazing. The best film of the 90's.

★★★★★

Author: Thomas Peluso (tpeluso@gmail.com) from Long Island, NY



The best story ever told on film

★★★★★

Author: Si Cole



It lacks surprises or excitement.

★★★★★

Author: Comeuppance Reviews from United States Minor Outlying Islands
3 December 2014



The Last Samurai: Hacked to Death

★★★★★

Author: Jon-Jokerjon from County Durham

Document Classification: Examples

- Spam or not?

From: "" <takworlId@hotmail.com>
Subject: real estate is the only way... gem [oalvgkay](#)

Anyone can buy real estate with no money down
Stop paying rent TODAY !
There is no need to spend hundreds or even thousands for similar courses
I am 22 years old and I have already purchased 6 properties using the methods outlined in this truly INCREDIBLE ebook.
Change your life NOW !

=====

Click Below to order:
<http://www.wholesaledaily.com/sales/nmd.htm>

=====



Document Classification: Examples

- What is the subject of this article?

MEDLINE Article



MeSH Subject Category Hierarchy

- Antagonists and Inhibitors
- Blood Supply
- Chemistry
- Drug Therapy
- Embryology
- Epidemiology
- ...

Document Classification: Applications

- Document Classification/Categorization
 - ✓ Assigning subject categories, topics, or genres
 - ✓ Spam detection
 - ✓ Authorship identification
 - ✓ Age/gender identification
 - ✓ Language identification
 - ✓ Sentiment analysis
 - ✓ ...

Document Classification: Definition

- Document Classification

- ✓ Input (I)

- A document d
 - A fixed set of classes $C = \{c_1, c_2, \dots, c_j\}$

- ✓ Output (O)

- A predicted class c in C

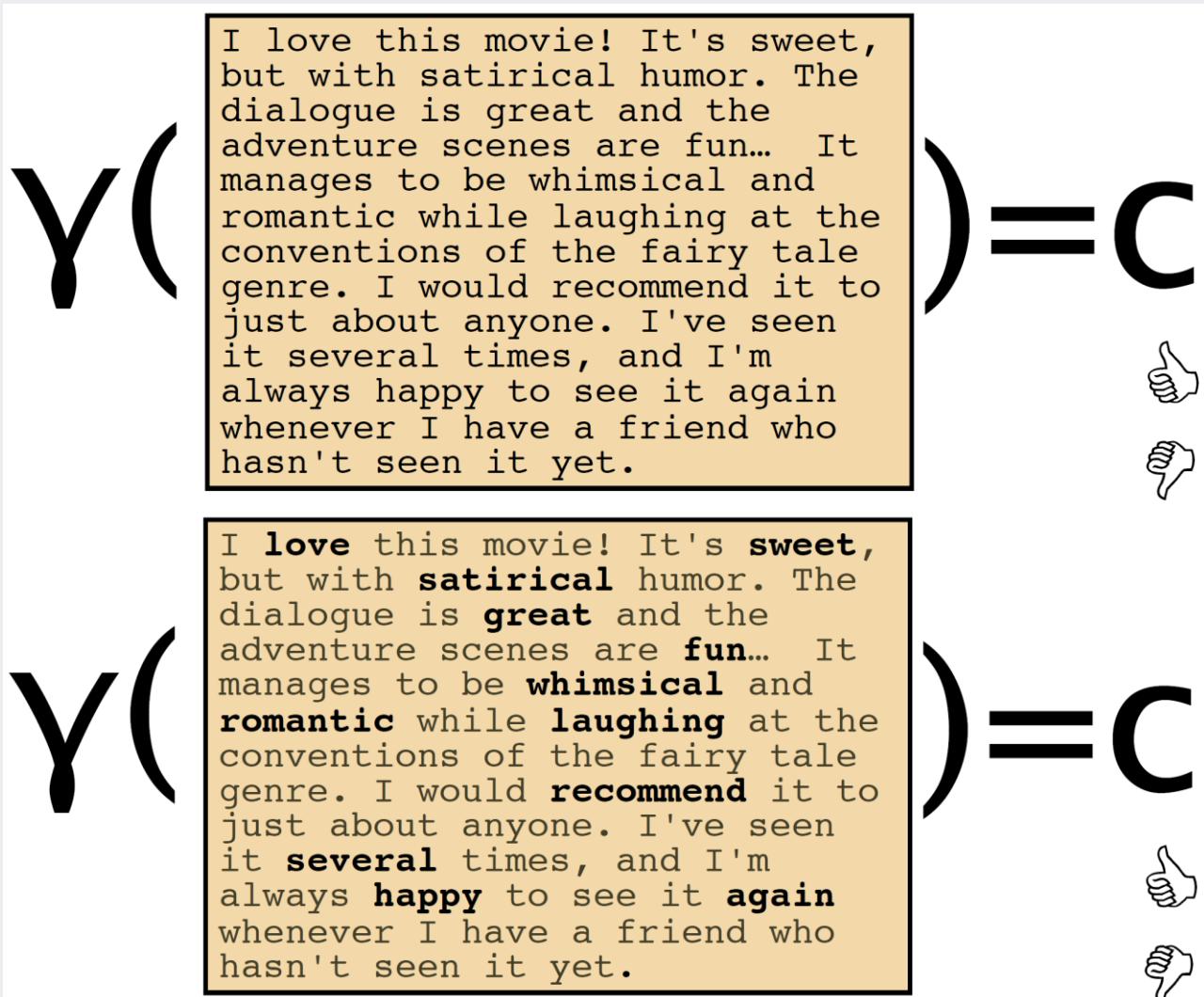
- Supervised Machine Learning

- ✓ Input: I + a training set of m hand-labeled documents: $(d_1, c_1), \dots, (d_m, c_m)$

- ✓ Output: a learned classifier $y: f(d) = c$

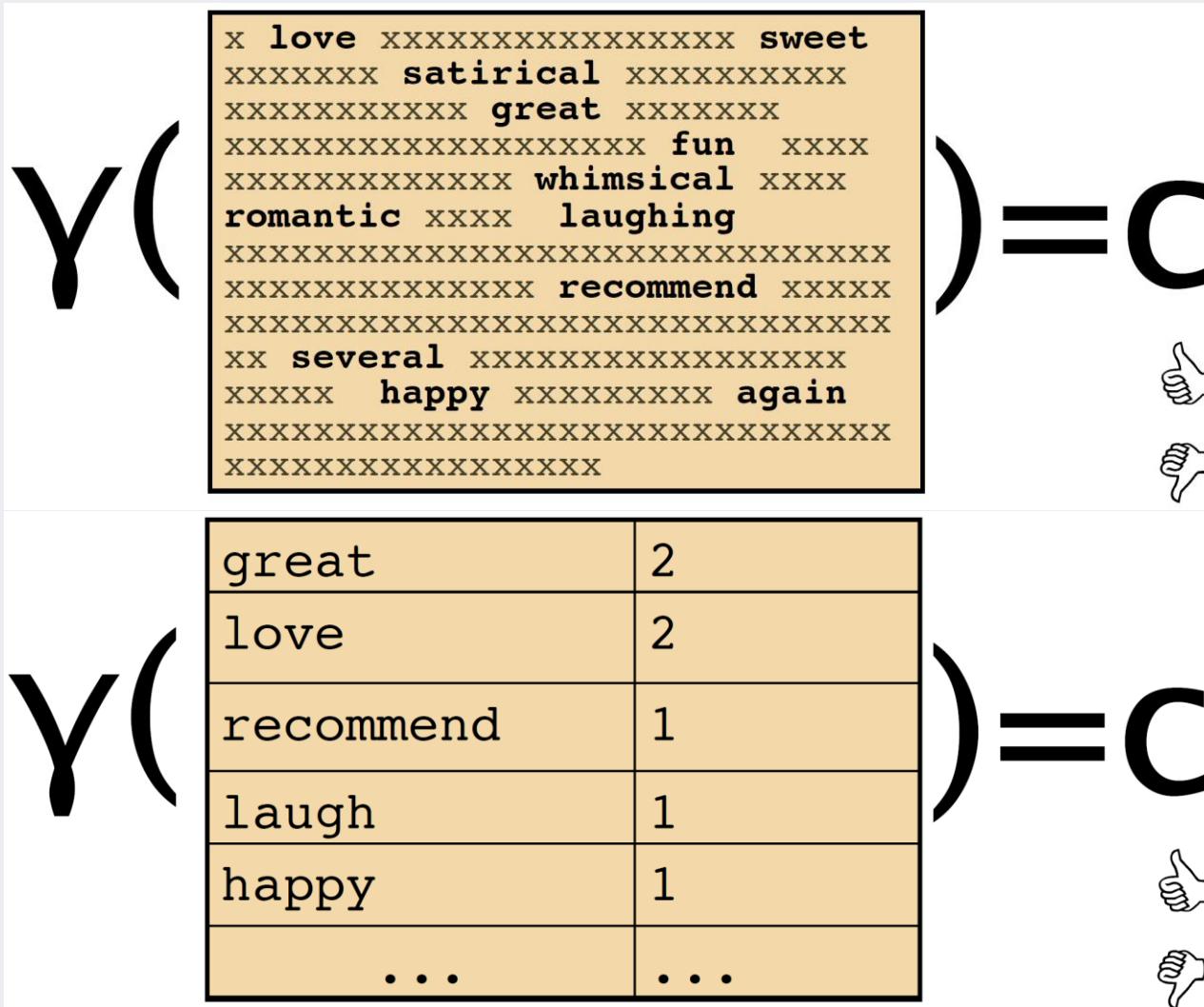
Document Classification: Definition

- Supervised Machine Learning with Bag-of-Words representation



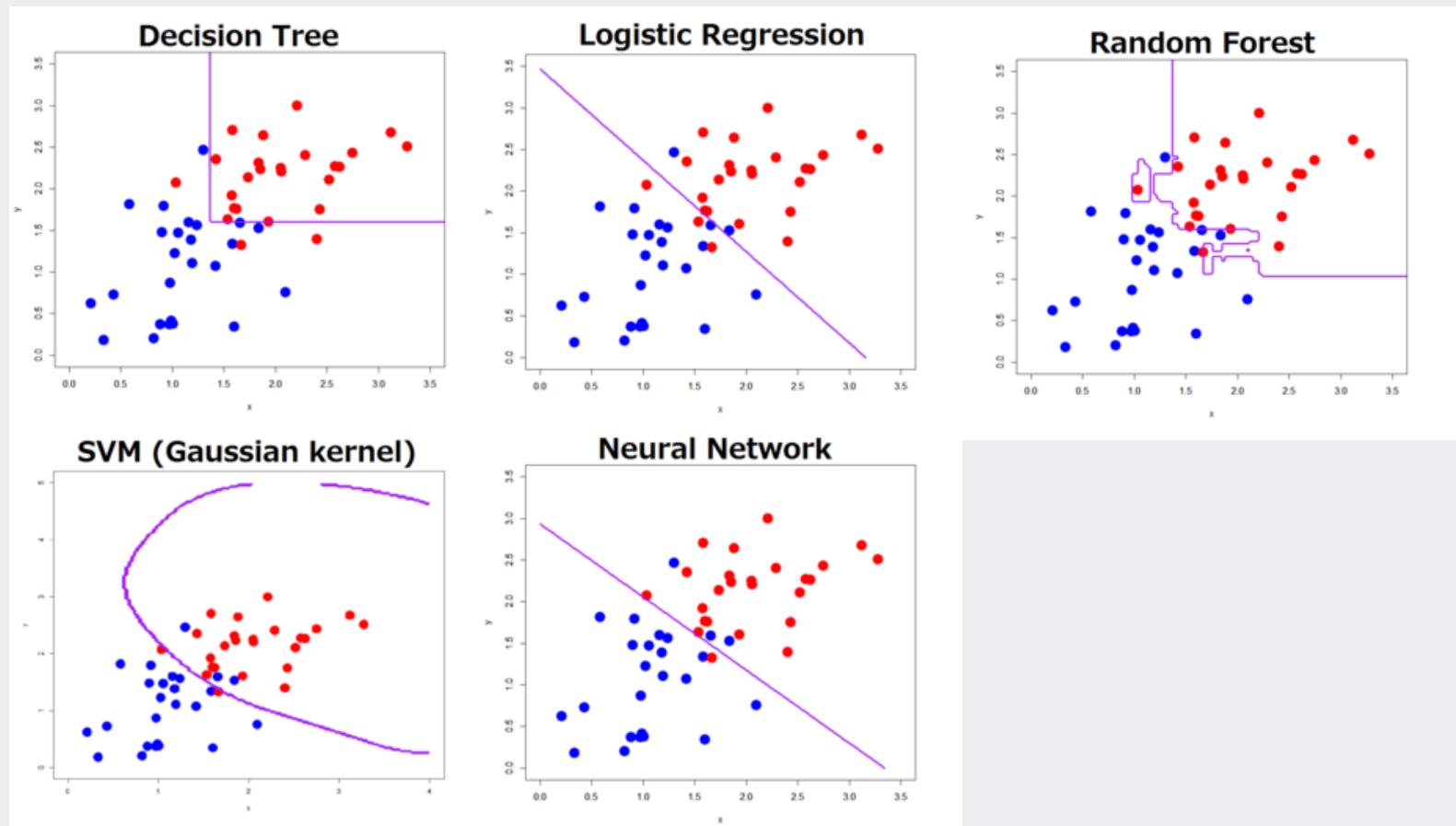
Document Classification: Definition

- Supervised Machine Learning with Bag-of-Words representation



Why Are There So Many Classifiers?

- We cannot guarantee that a single classifier is always better than the others



AGENDA

01 Document Classification: Overview

02 Naive Bayesian Classifier

03 k-Nearest Neighbor Classifier

04 Classification Tree

05 Support Vector Machine

06 R Exercise

Naive Bayesian Classifier

- Baye's Rule (one of the most important rules in statistics)

$$P(C_i|x_1, x_2) = \frac{P(x_1, X_2|C_i) \cdot P(C_i)}{P(x_1, x_2)}$$

- Naive: Let's assume that all variables are statistically independent to each other

$$= \frac{P(x_1|C_i) \cdot P(x_2|C_i) \cdot P(C_i)}{P(x_1, x_2)}$$

Naive Bayesian Classifier

- Maximum likelihood estimates
 - ✓ Simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{N.\text{Doc}(C = c_j)}{\text{Total number of documents}}$$

$$\hat{P}(w_i | c_j) = \frac{\text{count}(w_i, c_j)}{\sum_{w \in V} \text{count}(w, c_j)}$$

Fraction of times word w_i appears among all words in documents of class c_j

- Limitation
 - ✓ What if we have seen no training document with the word “fantastic” and classified in in the class “positive”?

$$\hat{P}(\text{fantastic} | \text{positive}) = \frac{\text{count}(\text{fantastic}, \text{positive})}{\sum_{w \in V} \text{count}(w, \text{positive})} = 0$$

- ✓ Zero probabilities cannot be conditioned away, no matter the other evidence!

Naive Bayesian Classifier

- Example: Classifying movie reviews into Positive/Negative class
 - ✓ Review 1: This movie was awesome! I really enjoyed it.
 - ✓ Review 2: This movie was boring and waste of time.
- Step 1: Estimate the conditional probabilities for each class

| Words | P(Word/Positive) | P(Word/Negative) |
|---------|------------------|------------------|
| This | 0.1 | 0.1 |
| Movie | 0.1 | 0.1 |
| Was | 0.1 | 0.1 |
| Awesome | 0.4 | 0.01 |
| I | 0.2 | 0.2 |
| Really | 0.3 | 0.05 |
| enjoyed | 0.5 | 0.05 |
| It | 0.1 | 0.1 |
| Boring | 0.02 | 0.3 |
| And | 0.1 | 0.1 |
| Waste | 0.02 | 0.35 |
| Of | 0.02 | 0.02 |
| Time | 0.15 | 0.15 |

Naive Bayesian Classifier

- For the Review I

✓ Review I: This movie was awesome! I really enjoyed it.

$$\prod_i P(Word_i | Pos) = 120 \times 10^{-8} > \prod_i P(Word_i | Neg) = 0.5 \times 10^{-8}$$

| Words | P(Word/Positive) | P(Word/Negative) |
|---------|------------------|------------------|
| This | 0.1 | 0.1 |
| Movie | 0.1 | 0.1 |
| Was | 0.1 | 0.1 |
| Awesome | 0.4 | 0.01 |
| I | 0.2 | 0.2 |
| Really | 0.3 | 0.05 |
| enjoyed | 0.5 | 0.05 |
| It | 0.1 | 0.1 |
| Boring | 0.02 | 0.3 |
| And | 0.1 | 0.1 |
| Waste | 0.02 | 0.35 |
| Of | 0.02 | 0.02 |
| Time | 0.15 | 0.15 |

Naive Bayesian Classifier

- For the Review |

✓ Review 2: This movie was boring and waste of time.

$$\prod_i P(Word_i | Pos) = 0.012 \times 10^{-8} < \prod_i P(Word_i | Neg) = 3.15 \times 10^{-8}$$

| Words | P(Word/Positive) | P(Word/Negative) |
|---------|------------------|------------------|
| This | 0.1 | 0.1 |
| Movie | 0.1 | 0.1 |
| Was | 0.1 | 0.1 |
| Awesome | 0.4 | 0.01 |
| I | 0.2 | 0.2 |
| Really | 0.3 | 0.05 |
| enjoyed | 0.5 | 0.05 |
| It | 0.1 | 0.1 |
| Boring | 0.02 | 0.3 |
| And | 0.1 | 0.1 |
| Waste | 0.02 | 0.35 |
| Of | 0.02 | 0.02 |
| Time | 0.15 | 0.15 |

Naive Bayesian Classifier

- Smoothing Techniques

- ✓ Laplace (add-1) smoothing

$$\hat{P}(w_i|c_j) = \frac{\text{count}(w_i, c_j) + 1}{\sum_{w \in V} (\text{count}(w, c_j) + 1)} = \frac{\text{count}(w_i, c_j) + 1}{\sum_{w \in V} \text{count}(w, c_j) + |V|}$$

- Example

| Model pos | | Model neg | | | | | | |
|-----------|------|-----------|------|------|--|--|--|--|
| 0.1 | I | 0.2 | I | I | | | | |
| 0.1 | love | 0.001 | love | love | | | | |
| 0.01 | this | 0.01 | this | this | | | | |
| 0.05 | fun | 0.005 | fun | fun | | | | |
| 0.1 | film | 0.1 | film | film | | | | |

$P(s|pos) > P(s|neg)$

AGENDA

01 Document Classification: Overview

02 Naive Bayesian Classifier

03 k-Nearest Neighbor Classifier

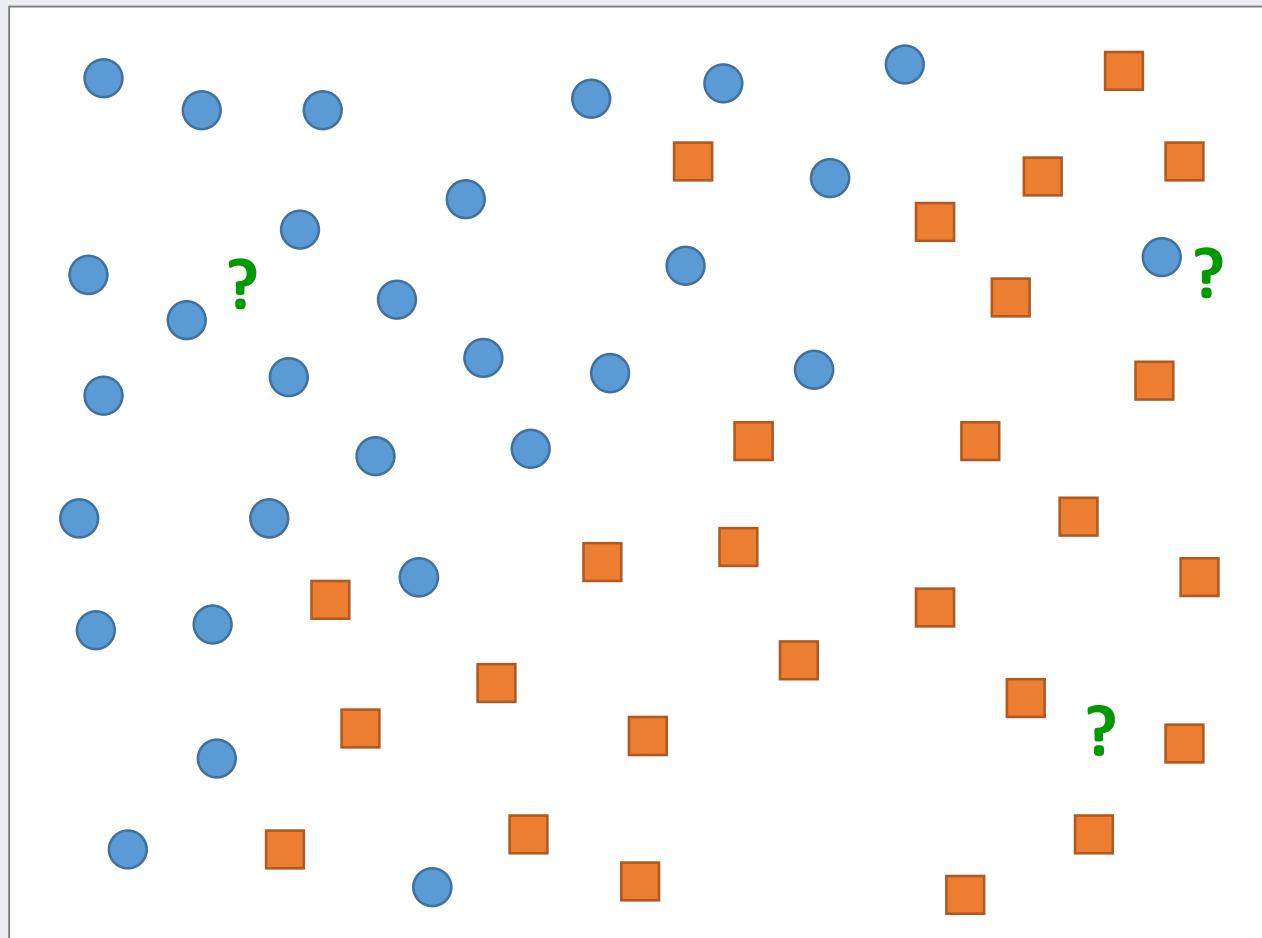
04 Classification Tree

05 Support Vector Machine

06 R Exercise

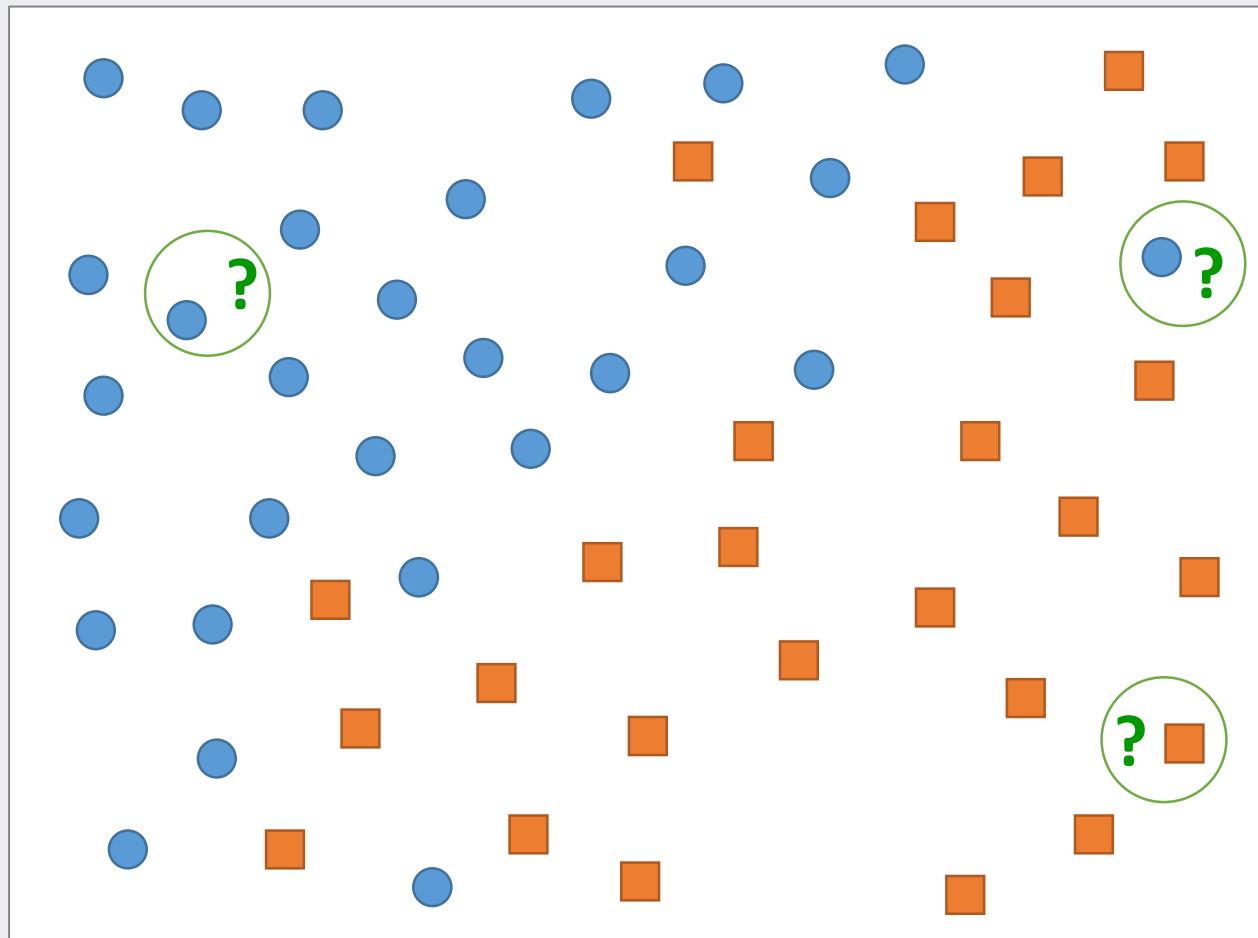
k-Nearest Neighbor Classification

- Which class does the question mark belong to?



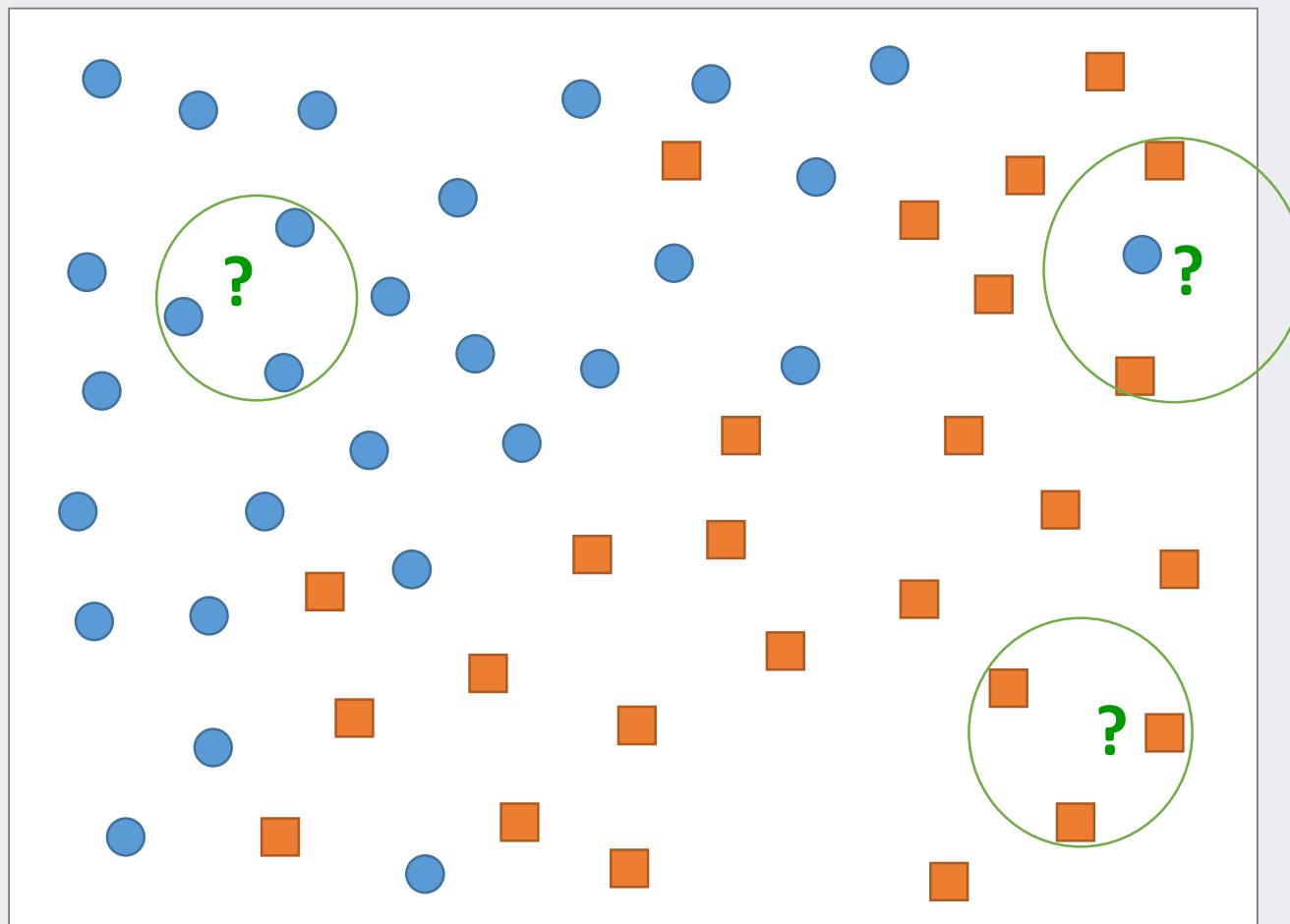
k-Nearest Neighbor Classification

- Which class does the question mark belong to?



k-Nearest Neighbor Classification

- Which class does the question mark belong to?



k-Nearest Neighbor Classification

- Motivation

類類相從 近墨者黑

“Birds of a feather flock together”



k-Nearest Neighbor Classification

k-NN Classification Process

- Step 1: Prepare the reference data

✓ Define attributes

- BoW representation or distributed representation

✓ Collect sufficient number of records from each class

| Doc. | Vision | Finance | ... | Class |
|------|--------|---------|-----|-------|
| 1 | 2.54 | 0.15 | ... | TPAMI |
| 2 | 1.78 | 0 | ... | TPAMI |
| 3 | 0.12 | 2.65 | ... | JoF |
| 4 | 0.25 | 3.52 | ... | JoF |
| ... | ... | ... | ... | ... |
| N | 3.12 | 0.14 | ... | TPAMI |

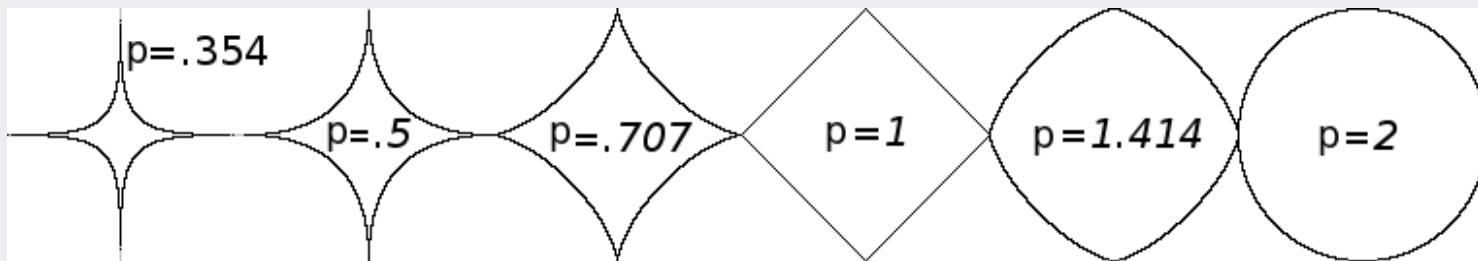
k-Nearest Neighbor Classification

k-NN Classification Process

- Step 2: Define the similarity measure

- ✓ Similarity $\propto 1/\text{distance}$
- ✓ Minkovski distance with order p

$$\text{distance}(P = (x_1, x_2, \dots, x_n), Q(y_1, y_2, \dots, y_n)) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}}$$



- ✓ $p=2$: Euclidean distance
- ✓ $p=1$: Manhattan distance

k-Nearest Neighbor Classification

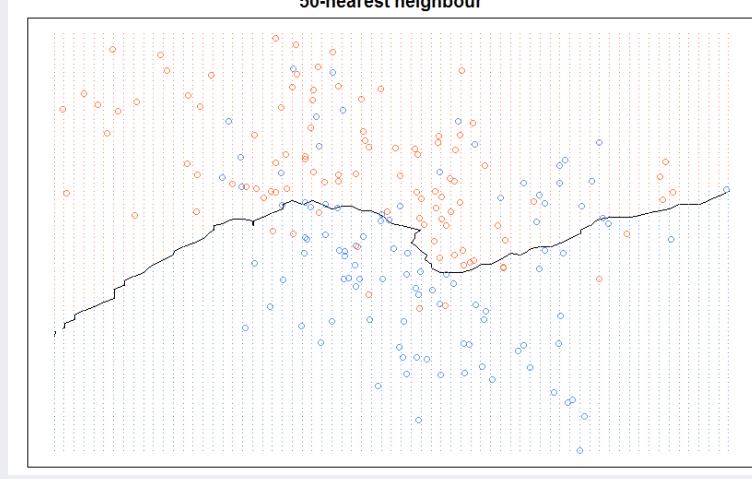
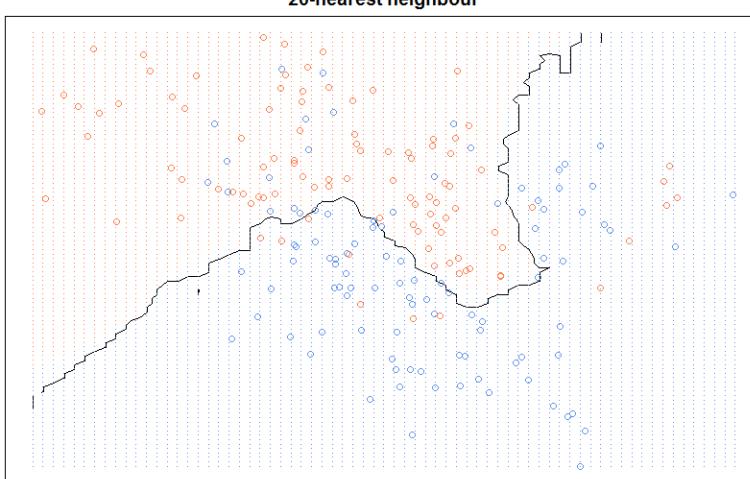
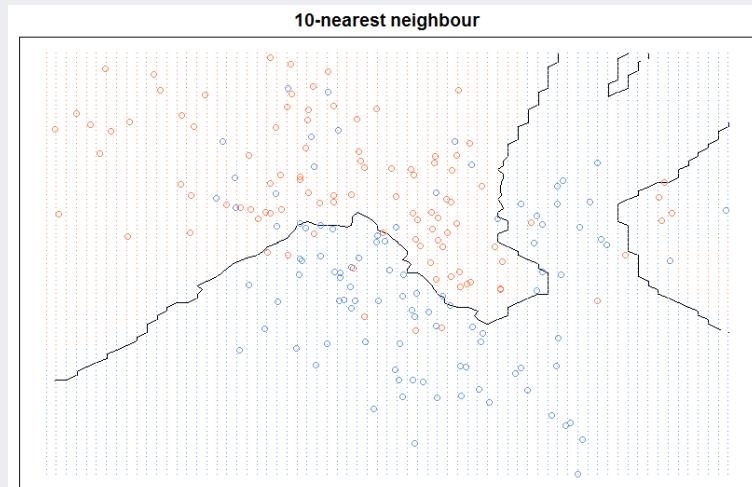
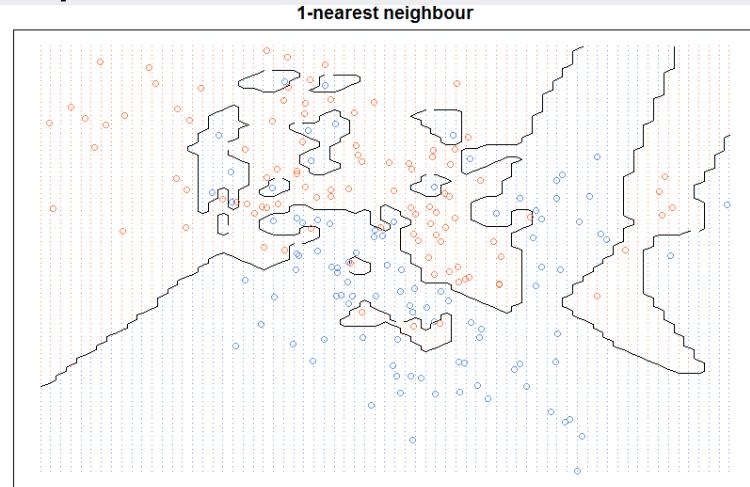
k-NN Classification Process

- Step 3: Initialize the set of candidate values for k
 - ✓ If k is too **small**, then the classification will be highly locally sensitive (**over-fitting**).
 - ✓ If k is too **large**, then it will lose the ability to capture the local structure (**under-fitting**).
 - ✓ A proper k should be chosen among a set of candidates.
 - ✓ Use the validation data.

k-Nearest Neighbor Classification

k-NN Classification Process

- Step 3: Initialize the set of candidate values for k



k-Nearest Neighbor Classification

k-NN Classification Process

- Step 4: Determine the combining rule

- ✓ Majority voting vs. Weighted voting

| | Neighbor | Class | Distance | 1/distance | Weight |
|----------------------|----------|-------|----------|------------|--------|
| For a new data | N1 | TPAMI | 1 | 1.00 | 0.44 |
| | N2 | JoF | 2 | 0.50 | 0.22 |
| | N3 | TPAMI | 3 | 0.33 | 0.15 |
| | N4 | JoF | 4 | 0.25 | 0.11 |
| | N5 | JoF | 5 | 0.20 | 0.08 |

- ✓ Majority voting: $P(X \text{ in TPAMI}) = 2/5 = 0.4$
 - ✓ Weighted voting: $P(X \text{ in TPAMI}) = 0.59$
 - ✓ If the cut-off is set to 0.5 X is classified as **TPAMI** by the majority voting while classified as **JoF** by the weighted voting

k-Nearest Neighbor Classification

k-NN Classification Process

- Step 5: Find the best k using the validation dataset

| Value of k | % Error Training | % Error Validation |
|------------|------------------|--------------------|
| 1 | 0.00 | 33.33 |
| 2 | 16.67 | 33.33 |
| 3 | 11.11 | 33.33 |
| 4 | 22.22 | 33.33 |
| 5 | 11.11 | 33.33 |
| 6 | 27.78 | 33.33 |
| 7 | 22.22 | 33.33 |
| 8 | 22.22 | 16.67 |
| 9 | 22.22 | 16.67 |
| 10 | 22.22 | 16.67 |
| 11 | 16.67 | 33.33 |
| 12 | 16.67 | 16.67 |
| 13 | 11.11 | 33.33 |
| 14 | 11.11 | 16.67 |
| 15 | 5.56 | 33.33 |
| 16 | 16.67 | 33.33 |
| 17 | 11.11 | 33.33 |
| 18 | 50.00 | 50.00 |

k-Nearest Neighbor Classification

- k-NN Issue I: Normalization

- ✓ Normalization or scaling must be done before finding k-nearest neighbors
- ✓ If not, variables with large measuring units are over-emphasized while variables with small measuring units are under-evaluated

| [Before Normalization] | | | | | [After Normalization] | | | | |
|------------------------|--------|--------|-----|--------|-----------------------|--------|--------|-------|--------|
| No. | Height | Weight | BFS | Gender | No. | Height | Weight | BFS | Gender |
| 1 | 187 | 93 | 15 | M | 1 | 1.47 | 2.80 | -1.00 | M |
| 2 | 165 | 51 | 25 | F | 2 | 0.00 | -1.40 | 1.00 | F |
| 3 | 174 | 68 | 14 | M | 3 | 0.60 | 0.30 | -1.20 | M |
| 4 | 156 | 48 | 29 | F | 4 | -0.60 | -1.70 | 1.80 | F |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| N | 168 | 59 | 12 | M | N | 0.20 | -0.60 | -1.60 | M |
| Avg. | 165 | 65 | 20 | - | | | | | |
| Stdev. | 15 | 10 | 5 | - | | | | | |

k-Nearest Neighbor Classification

- k-NN Issue 2: Cut-off
 - ✓ Consider the prior probability of each class
 - ✓ Assume that $N(M) = 100, N(F) = 400$

| | Neighbor | Class | |
|--------------------------------------|----------|-------|-----------------|
| For a new data X | N1 | M | |
| | N2 | F | Majority voting |
| | N3 | M | $P(X=M)=0.4$ |
| | N4 | F | |
| | N5 | F | |

- ✓ If the cut-off is set to 0.5 (assuming equal class distribution), then X is classified as F.
- ✓ If the cut-off is set to 0.2 (proportion of M among the people), then X is classified as M.

AGENDA

01 Document Classification: Overview

02 Naive Bayesian Classifier

03 k-Nearest Neighbor Classifier

04 Classification Tree

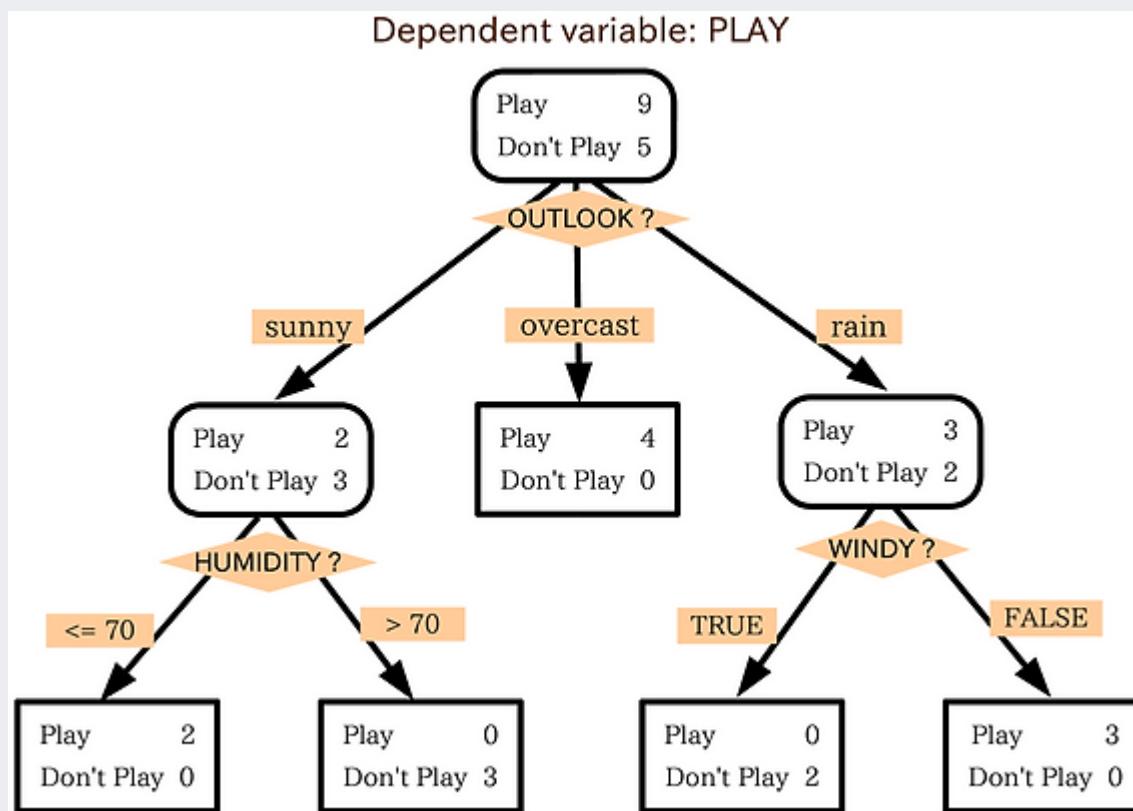
05 Support Vector Machine

06 R Exercise

Classification Tree

- Goal

- ✓ Classify or predict an outcome based on a set of predictors.
- ✓ The output is a set of rules.



Rule example

If outlook is sunny
and if humidity > 70
then he does not play

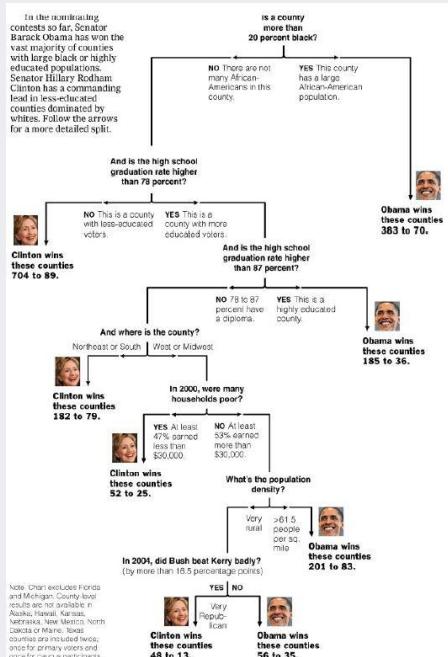
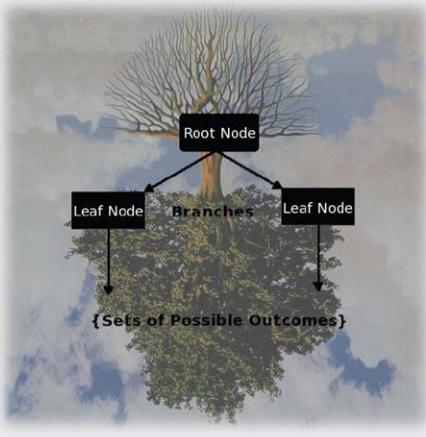
or

If outlook is rainy
and it is not windy
then he does play

Classification Tree

- Why CART?
 - ✓ Simple to understand and interpret.
 - ✓ Requires little data preparation (normalization, missing value treatments, etc.)
 - ✓ Able to handle both numerical and categorical data.
- Key Ideas
 - ✓ Recursive Partitioning
 - Repeatedly split the records into two parts so as to achieve maximum homogeneity within the new parts.
 - ✓ Pruning the Tree
 - Simplify the tree by pruning peripheral branches to avoid over-fitting.

Classification Tree



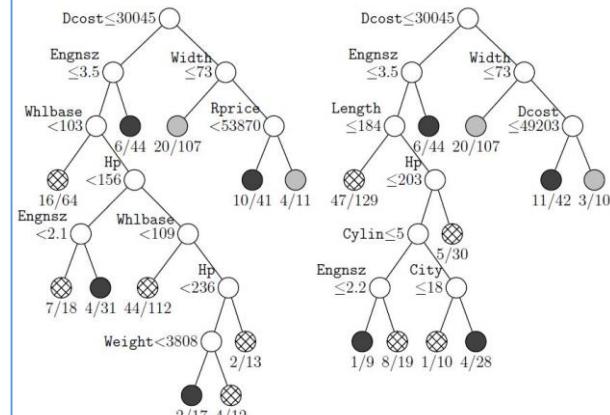
Classification and Regression Tree (CART)

- Generate a set of rules by recursively partitioning the entire datasets to increase the purity of the partitioned area (Breiman, 1984)
- Being able to explain the reason of the prediction result by following the rules to the target leaf node
- Can handle categorical and numerical variables simultaneously

Recursive Partitioning

Pruning

- Partition the data in a parent node into two child nodes using a certain value of a certain variable
- Select the split point to maximize the purity of the child nodes
- Gini-index (for categorical variable) and the variance (for numerical variable) are used to measure the impurity of a node



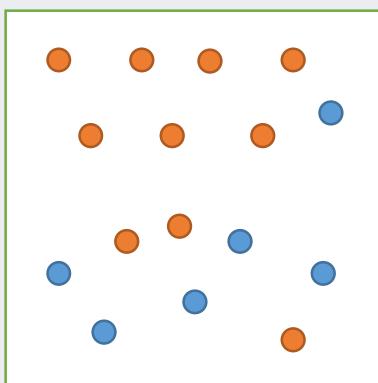
Classification Tree

- Measuring Impurity: Gini Index

- ✓ Gini Index for rectangle A containing m records

$$I(A) = 1 - \sum_{k=1}^m p_k^2$$

- p = proportion of cases in rectangle A that belong to class k.



$$\begin{aligned} I(A) &= 1 - \sum_{k=1}^m p_k^2 \\ &= 1 - \left(\frac{6}{16} \right)^2 - \left(\frac{10}{16} \right)^2 \\ &\approx 0.47 \end{aligned}$$

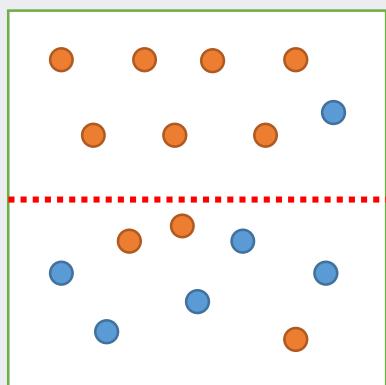
- $I(A) = 0$ when all cases belong to the same class.
- Max value when all classes are equal represented (=0.5 in binary case)

Classification Tree

- Measuring Impurity: Gini Index
 - ✓ When there are more than two rectangles

$$I(A) = \sum_{i=1}^d \left(R_i \left(1 - \sum_{k=1}^m p_{ik}^2 \right) \right)$$

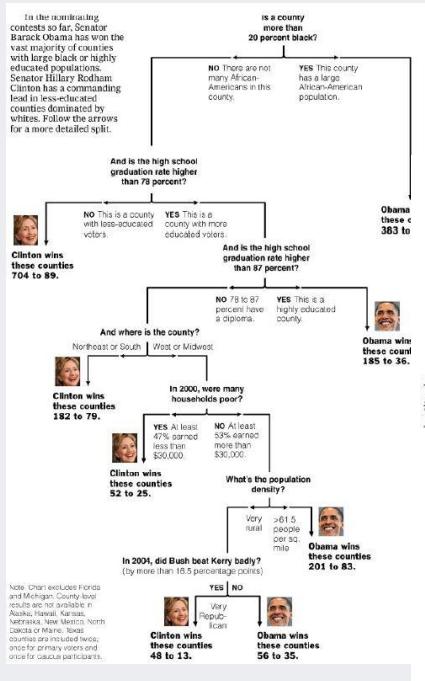
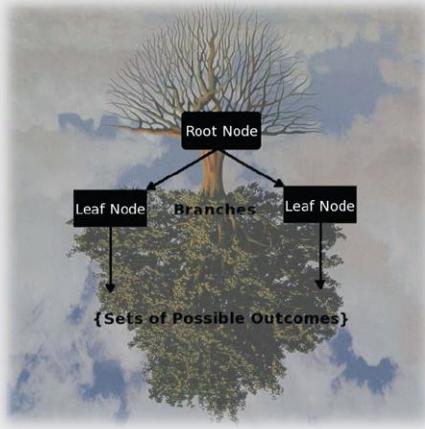
- R_i = proportion of cases in rectangle R_i among the training data.



$$\begin{aligned} I(A) &= 0.5 \times \left(1 - \left(\frac{7}{8} \right)^2 - \left(\frac{1}{8} \right)^2 \right) + 0.5 \times \left(1 - \left(\frac{3}{8} \right)^2 - \left(\frac{5}{8} \right)^2 \right) \\ &= 0.34 \end{aligned}$$

- “Information gain” after splitting: $0.47 - 0.34 = 0.13$

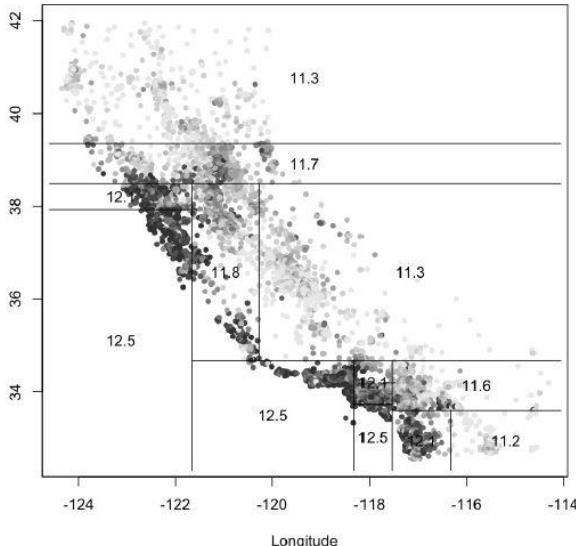
Classification Tree



Classification and Regression Tree (CART)

- Generate a set of rules by recursively partitioning the entire datasets to increase the purity of the partitioned area (Breiman, 1984)
- Being able to explain the reason of the prediction result by following the rules to the target leaf node
- Can handle categorical and numerical variables simultaneously

Recursive Partitioning



Pruning

- Aggregate some child node into a parent node to avoid over-fitting
- Pre-pruning: pruning is done during the tree construction
- Post-pruning: Once a full-tree is constructed, nodes are pruned by taking the validation error and tree complexity

Classification Tree

- Example: Riding Mowers

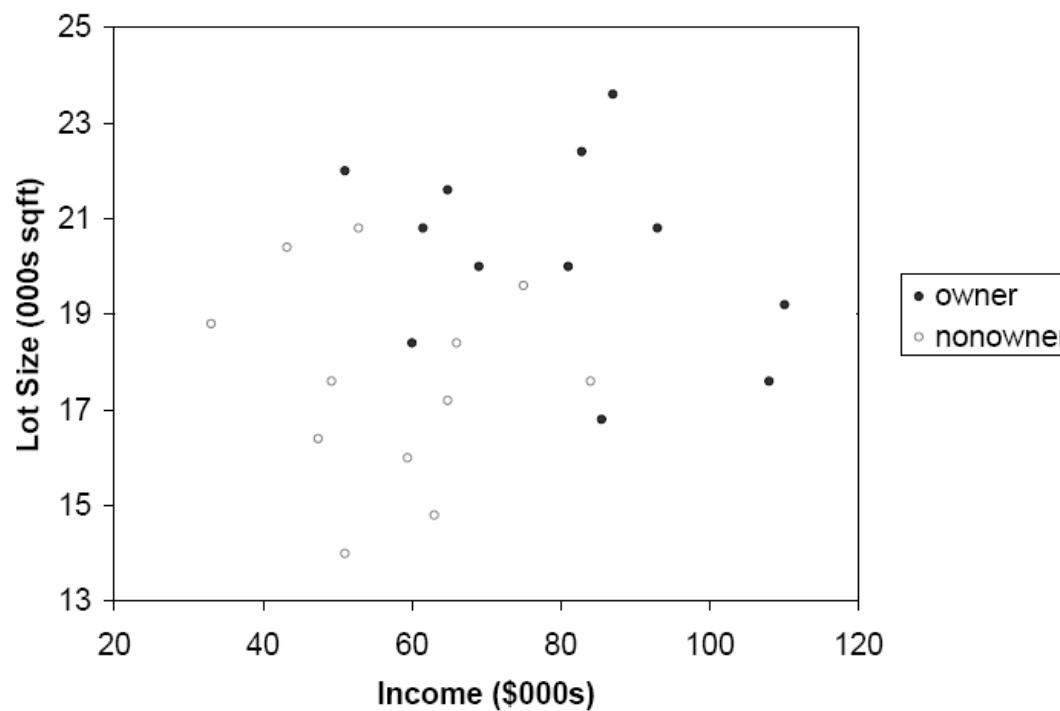
- ✓ Goal: Classify 24 households as owning or not owning riding mowers
- ✓ Predictors: Income, Lot size

| Income | Lot size | Ownership | Income | Lot size | Ownership |
|--------|----------|-----------|--------|----------|-----------|
| 60.0 | 18.4 | Owner | 75.0 | 19.6 | Non-owner |
| 85.5 | 16.8 | Owner | 52.8 | 20.8 | Non-owner |
| 64.8 | 21.6 | Owner | 64.8 | 17.2 | Non-owner |
| 61.5 | 20.8 | Owner | 43.2 | 20.4 | Non-owner |
| 87.0 | 23.6 | Owner | 84.0 | 17.6 | Non-owner |
| 110.1 | 19.2 | Owner | 49.2 | 17.6 | Non-owner |
| 108.0 | 17.6 | Owner | 59.4 | 16.0 | Non-owner |
| 82.8 | 22.4 | Owner | 66.0 | 18.4 | Non-owner |
| 69.0 | 20.0 | Owner | 47.4 | 16.4 | Non-owner |
| 93.0 | 20.8 | Owner | 33.0 | 18.8 | Non-owner |
| 51.0 | 22.0 | Owner | 51.0 | 14.0 | Non-owner |
| 81.0 | 20.0 | Owner | 63.0 | 14.8 | Non-owner |

Classification Tree

Order records according to one variable

- Order the data with regard to lot size



| Income | Lot size | Ownership |
|--------|----------|-----------|
| 51.0 | 14.0 | Non-owner |
| 63.0 | 14.8 | Non-owner |
| 59.4 | 16.0 | Non-owner |
| 47.4 | 16.4 | Non-owner |
| 85.5 | 16.8 | Owner |
| 64.8 | 17.2 | Non-owner |
| 108.0 | 17.6 | Owner |
| 84.0 | 17.6 | Non-owner |
| 49.2 | 17.6 | Non-owner |
| 60.0 | 18.4 | Owner |
| 66.0 | 18.4 | Non-owner |
| 33.0 | 18.8 | Non-owner |
| 110.1 | 19.2 | Owner |
| 75.0 | 19.6 | Non-owner |
| 69.0 | 20.0 | Owner |
| 81.0 | 20.0 | Owner |
| 43.2 | 20.4 | Non-owner |
| 61.5 | 20.8 | Owner |
| 93.0 | 20.8 | Owner |
| 52.8 | 20.8 | Non-owner |
| 64.8 | 21.6 | Owner |
| 51.0 | 22.0 | Owner |
| 82.8 | 22.4 | Owner |
| 87.0 | 23.6 | Owner |

Classification Tree

2

Find midpoints between successive values

- First midpoint = 14.4 ($0.5 * (14.0 + 14.8)$)
- Divide records into those with Lot size > 14.4 and those < 14.4
- Compute the impurity: Gini index

✓ Before splitting:

$$1 - \left(\frac{12}{24} \right)^2 - \left(\frac{12}{24} \right)^2 = 0.5$$

✓ After splitting:

$$\frac{1}{24} \left(1 - \left(\frac{1}{1} \right)^2 \right) + \frac{23}{24} \left(1 - \left(\frac{12}{23} \right)^2 - \left(\frac{11}{23} \right)^2 \right) \approx 0.48$$

✓ Information gain: $0.50 - 0.48 = 0.02$

| Income | Lot size | Ownership |
|--------|----------|-----------|
| 51.0 | 14.0 | Non-owner |
| 63.0 | 14.8 | Non-owner |
| 59.4 | 16.0 | Non-owner |
| 47.4 | 16.4 | Non-owner |
| 85.5 | 16.8 | Owner |
| 64.8 | 17.2 | Non-owner |
| 108.0 | 17.6 | Owner |
| 84.0 | 17.6 | Non-owner |
| 49.2 | 17.6 | Non-owner |
| 60.0 | 18.4 | Owner |
| 66.0 | 18.4 | Non-owner |
| 33.0 | 18.8 | Non-owner |
| 110.1 | 19.2 | Owner |
| 75.0 | 19.6 | Non-owner |
| 69.0 | 20.0 | Owner |
| 81.0 | 20.0 | Owner |
| 43.2 | 20.4 | Non-owner |
| 61.5 | 20.8 | Owner |
| 93.0 | 20.8 | Owner |
| 52.8 | 20.8 | Non-owner |
| 64.8 | 21.6 | Owner |
| 51.0 | 22.0 | Owner |
| 82.8 | 22.4 | Owner |
| 87.0 | 23.6 | Owner |

Classification Tree

2

Find midpoints between successive values

- First midpoint = 14.4 ($0.5 * (14.0 + 14.8)$)
- Divide records into those with Lot size > 14.4 and those < 14.4
- Compute the impurity: Entropy

✓ Before splitting:

$$-\frac{1}{2} \log_2 \left(\frac{1}{2} \right) - \frac{1}{2} \log_2 \left(\frac{1}{2} \right) = 1$$

✓ After splitting:

$$\frac{1}{24} (-\log(1)) + \frac{23}{24} \left(-\frac{12}{23} \log_2 \left(\frac{12}{23} \right) - \frac{11}{23} \log_2 \left(\frac{11}{23} \right) \right) \approx 0.96$$

✓ Information gain: $1 - 0.96 = 0.04$

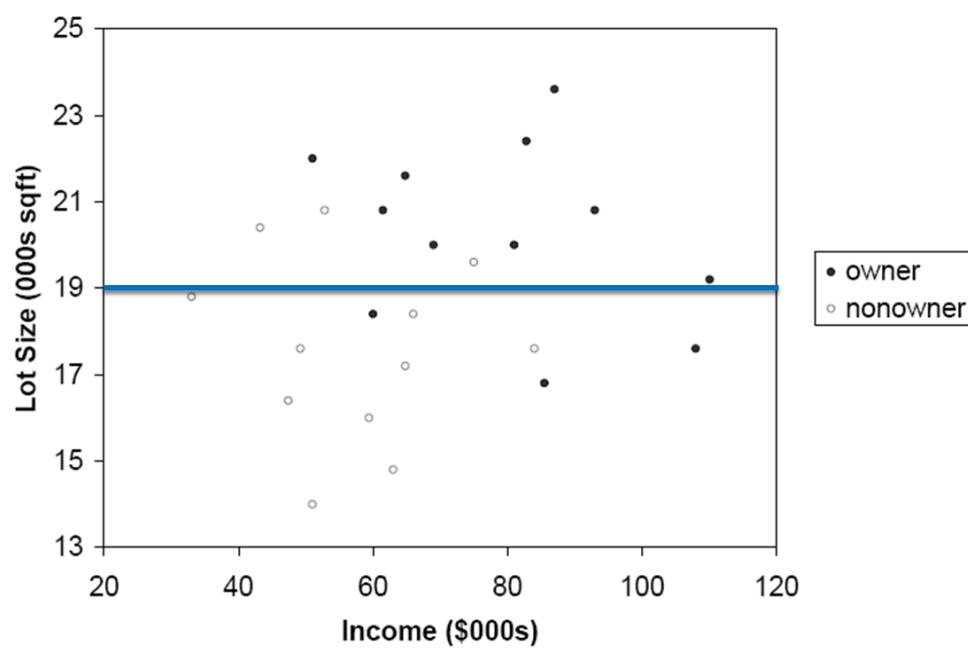
| Income | Lot size | Ownership |
|--------|----------|-----------|
| 51.0 | 14.0 | Non-owner |
| 63.0 | 14.8 | Non-owner |
| 59.4 | 16.0 | Non-owner |
| 47.4 | 16.4 | Non-owner |
| 85.5 | 16.8 | Owner |
| 64.8 | 17.2 | Non-owner |
| 108.0 | 17.6 | Owner |
| 84.0 | 17.6 | Non-owner |
| 49.2 | 17.6 | Non-owner |
| 60.0 | 18.4 | Owner |
| 66.0 | 18.4 | Non-owner |
| 33.0 | 18.8 | Non-owner |
| 110.1 | 19.2 | Owner |
| 75.0 | 19.6 | Non-owner |
| 69.0 | 20.0 | Owner |
| 81.0 | 20.0 | Owner |
| 43.2 | 20.4 | Non-owner |
| 61.5 | 20.8 | Owner |
| 93.0 | 20.8 | Owner |
| 52.8 | 20.8 | Non-owner |
| 64.8 | 21.6 | Owner |
| 51.0 | 22.0 | Owner |
| 82.8 | 22.4 | Owner |
| 87.0 | 23.6 | Owner |

Classification Tree

3

Find the best split

- Find the best split which maximize the (Gini or information gain)

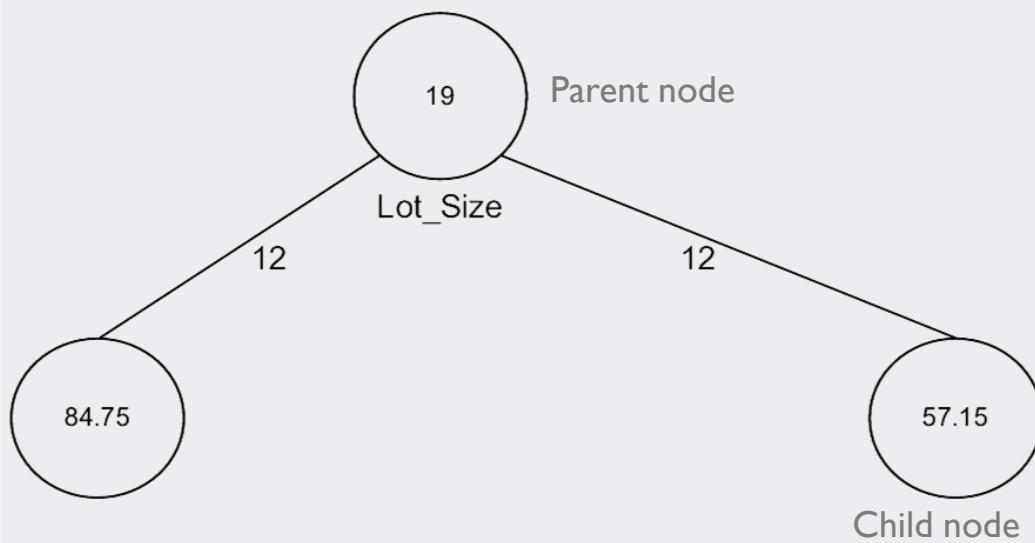


| Income | Lot size | Ownership |
|--------|----------|-----------|
| 51.0 | 14.0 | Non-owner |
| 63.0 | 14.8 | Non-owner |
| 59.4 | 16.0 | Non-owner |
| 47.4 | 16.4 | Non-owner |
| 85.5 | 16.8 | Owner |
| 64.8 | 17.2 | Non-owner |
| 108.0 | 17.6 | Owner |
| 84.0 | 17.6 | Non-owner |
| 49.2 | 17.6 | Non-owner |
| 60.0 | 18.4 | Owner |
| 66.0 | 18.4 | Non-owner |
| 33.0 | 18.8 | Non-owner |
| 110.1 | 19.2 | Owner |
| 75.0 | 19.6 | Non-owner |
| 69.0 | 20.0 | Owner |
| 81.0 | 20.0 | Owner |
| 43.2 | 20.4 | Non-owner |
| 61.5 | 20.8 | Owner |
| 93.0 | 20.8 | Owner |
| 52.8 | 20.8 | Non-owner |
| 64.8 | 21.6 | Owner |
| 51.0 | 22.0 | Owner |
| 82.8 | 22.4 | Owner |
| 87.0 | 23.6 | Owner |

Classification Tree

3

Tree structure



- Split point become nodes on tree (circles with split value in center)
- Rectangles represent “leaves” (terminal points, no future splits, classification value noted)
- Numbers on lines between nodes indicate # cases.

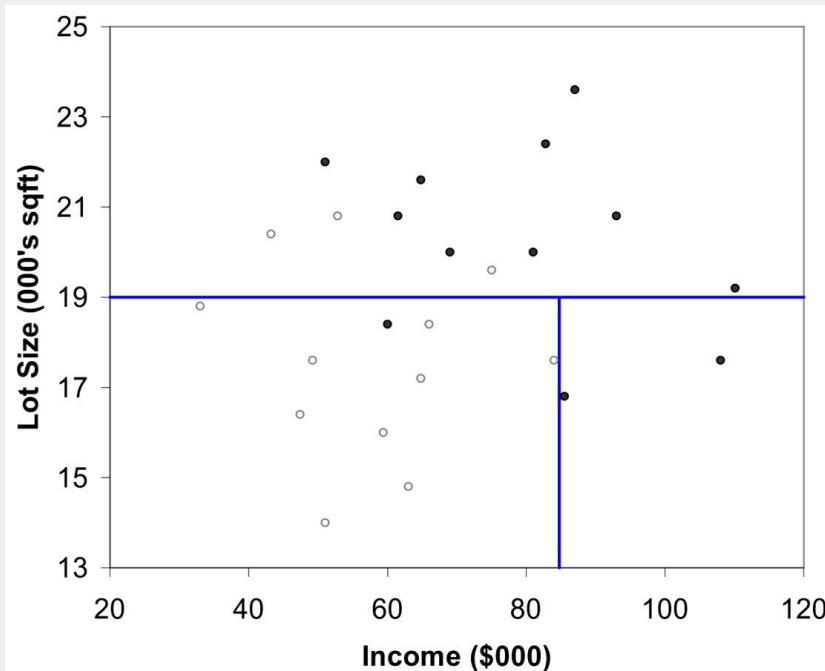
| Income | Lot size | Ownership |
|--------|----------|-----------|
| 51.0 | 14.0 | Non-owner |
| 63.0 | 14.8 | Non-owner |
| 59.4 | 16.0 | Non-owner |
| 47.4 | 16.4 | Non-owner |
| 85.5 | 16.8 | Owner |
| 64.8 | 17.2 | Non-owner |
| 108.0 | 17.6 | Owner |
| 84.0 | 17.6 | Non-owner |
| 49.2 | 17.6 | Non-owner |
| 60.0 | 18.4 | Owner |
| 66.0 | 18.4 | Non-owner |
| 33.0 | 18.8 | Non-owner |
| 110.1 | 19.2 | Owner |
| 75.0 | 19.6 | Non-owner |
| 69.0 | 20.0 | Owner |
| 81.0 | 20.0 | Owner |
| 43.2 | 20.4 | Non-owner |
| 61.5 | 20.8 | Owner |
| 93.0 | 20.8 | Owner |
| 52.8 | 20.8 | Non-owner |
| 64.8 | 21.6 | Owner |
| 51.0 | 22.0 | Owner |
| 82.8 | 22.4 | Owner |
| 87.0 | 23.6 | Owner |

Classification Tree

4

Repeat the splitting for each node

- Repeat the splitting until there is no gain.
- E.g., second split = income = 84.75



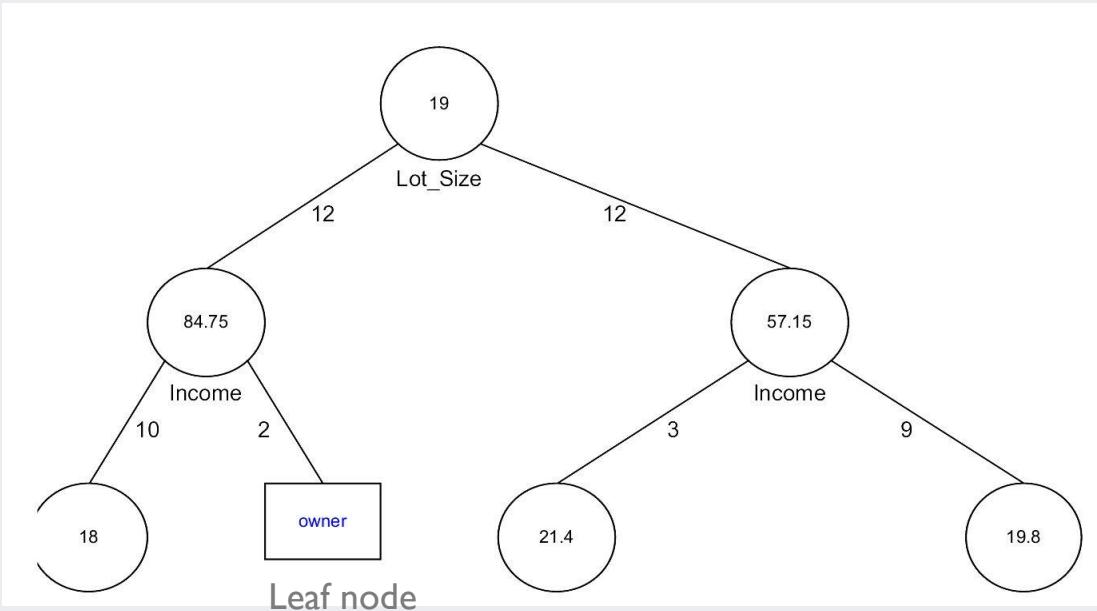
| Income | Lot size | Ownership |
|--------|----------|-----------|
| 33.0 | 18.8 | Non-owner |
| 47.4 | 16.4 | Non-owner |
| 49.2 | 17.6 | Non-owner |
| 51.0 | 14.0 | Non-owner |
| 59.4 | 16.0 | Non-owner |
| 60.0 | 18.4 | Owner |
| 63.0 | 14.8 | Non-owner |
| 64.8 | 17.2 | Non-owner |
| 66.0 | 18.4 | Non-owner |
| 84.0 | 17.6 | Non-owner |
| 85.5 | 16.8 | Owner |
| 108.0 | 17.6 | Owner |

Classification Tree

4

Repeat the splitting for each node

- Repeat the splitting until there is no gain.
- E.g., second split = income = 84.75



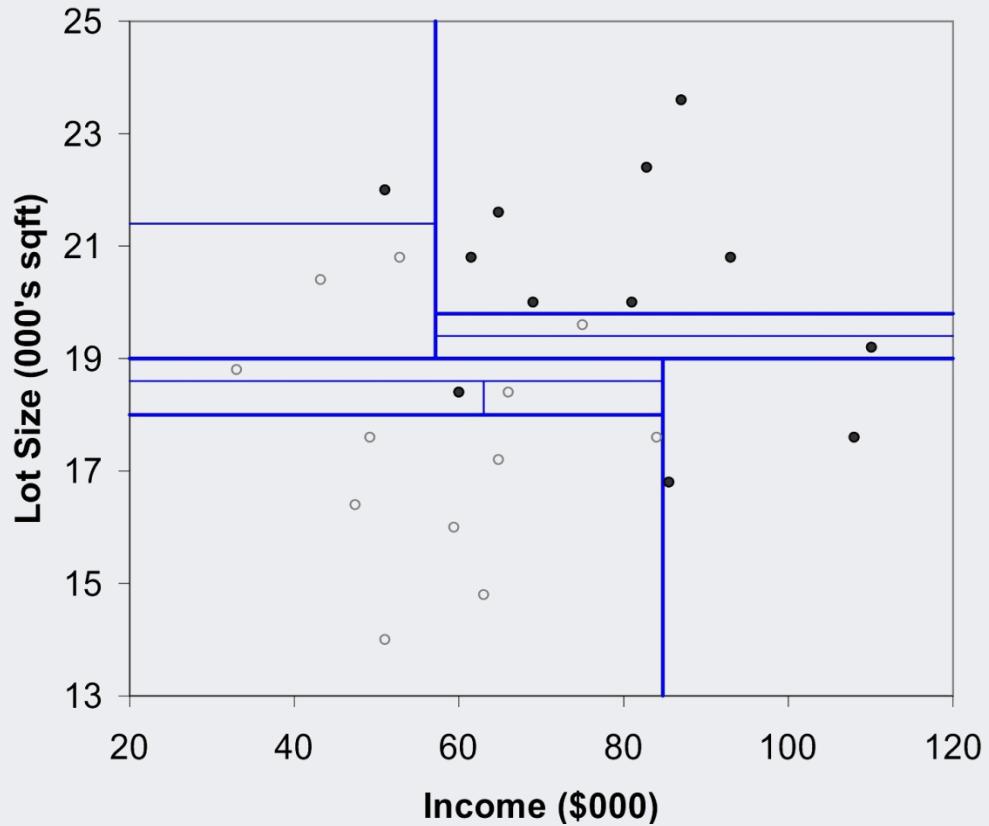
| Income | Lot size | Ownership |
|--------|----------|-----------|
| 33.0 | 18.8 | Non-owner |
| 47.4 | 16.4 | Non-owner |
| 49.2 | 17.6 | Non-owner |
| 51.0 | 14.0 | Non-owner |
| 59.4 | 16.0 | Non-owner |
| 60.0 | 18.4 | Owner |
| 63.0 | 14.8 | Non-owner |
| 64.8 | 17.2 | Non-owner |
| 66.0 | 18.4 | Non-owner |
| 84.0 | 17.6 | Non-owner |
| 85.5 | 16.8 | Owner |
| 108.0 | 17.6 | Owner |

Classification Tree

4

Repeat the splitting for each node

- Repeat the splitting until there is no gain.
- Final splitting



Classification Tree

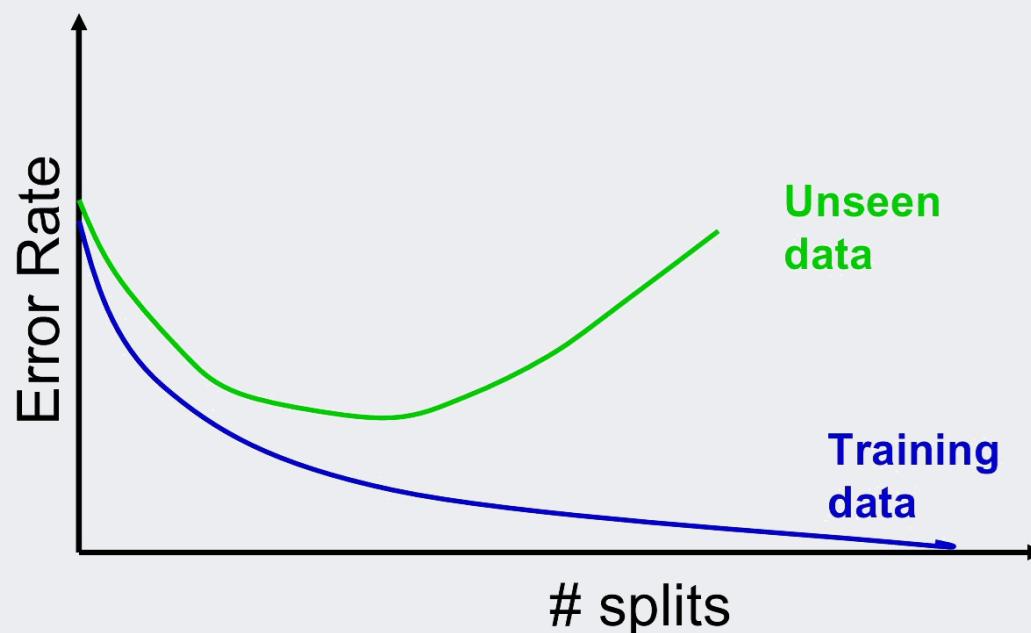
4

Repeat the splitting for each node

- Each leaf node label is determined by “voting” of the records within it, and by the cutoff value.
- Records within each leaf node are from the training data.
- Default cutoff=0.5 means that the leaf node’s label is the majority class.
- Cutoff = 0.75 requires majority of 75% or more “1” records in the leaf to label it a “1” node.

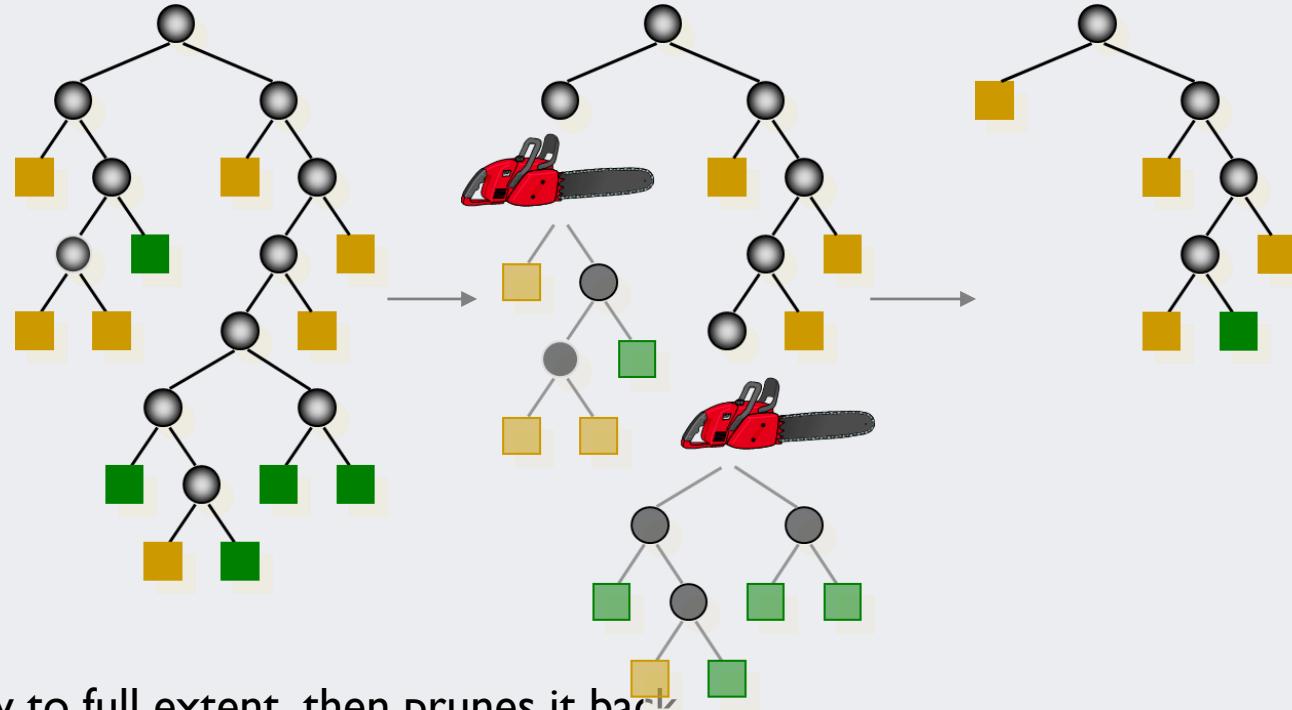
Classification Tree

- Overfitting problem
 - ✓ The end of recursive partitioning process is 100% purity in each leaf
 - ✓ It over-fits the data, ending up fitting noise in the data and leading to low predictive accuracy of new data
 - ✓ Past a certain point, the error rate for the validation data starts to increase



Classification Tree

- Pruning



- ✓ CART lets tree grow to full extent, then prunes it back.
- ✓ Idea is to find that point at which the validation error begins to rise.
- ✓ Generate successively smaller trees by pruning leaves.
- ✓ At each pruning stage, multiple trees are possible.
- ✓ Use “**cost complexity**” to choose the best tree at that stage.

Classification Tree

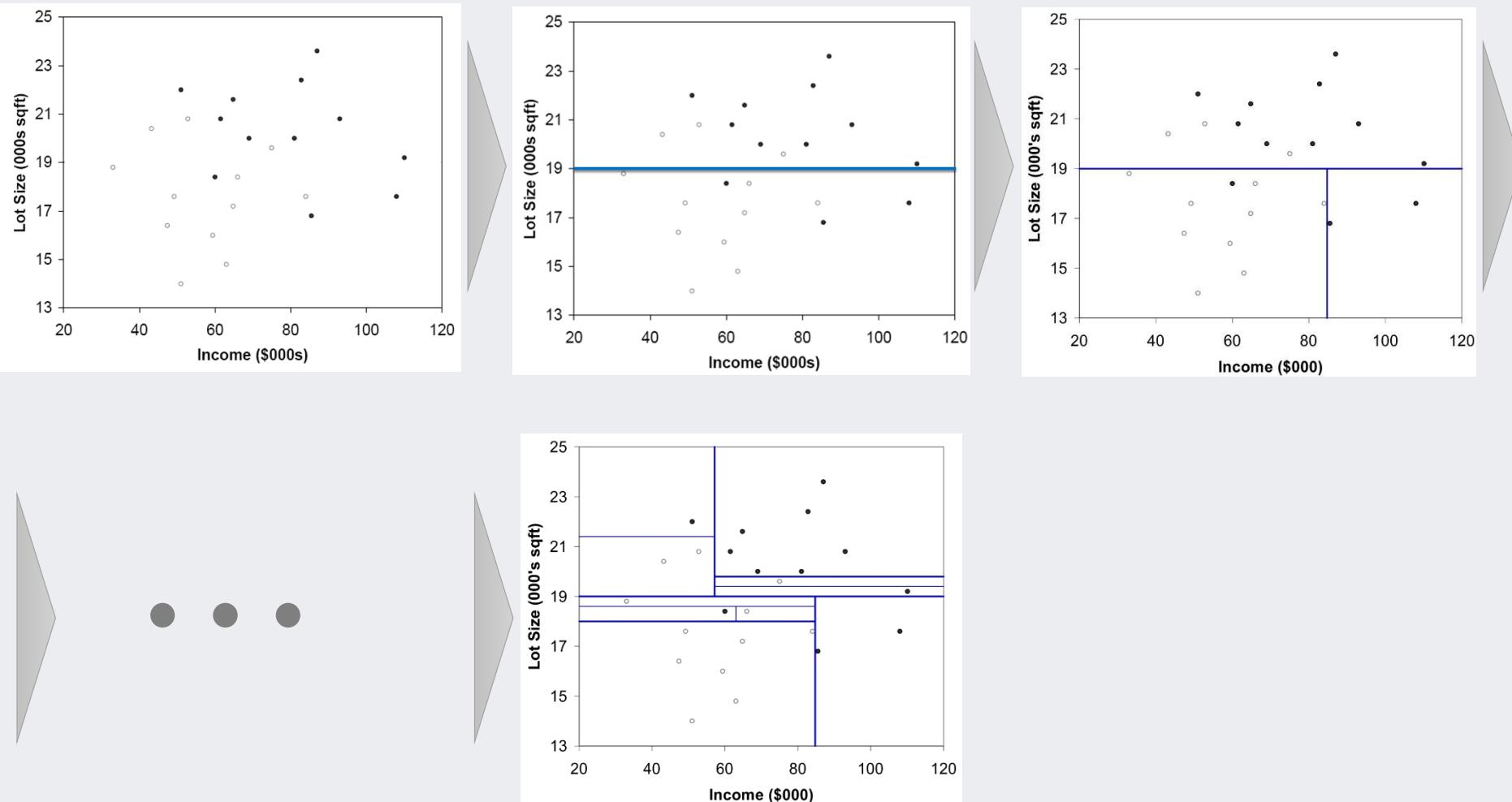
- Cost complexity

$$CC(T) = Err(T) + \alpha \times L(T)$$

- ✓ CC(T) = cost complexity of a tree
- ✓ ERR(T) = proportion of misclassified records in the validation data
- ✓ Alpha = penalty factor attached to the tree size (set by the user)

Classification Tree

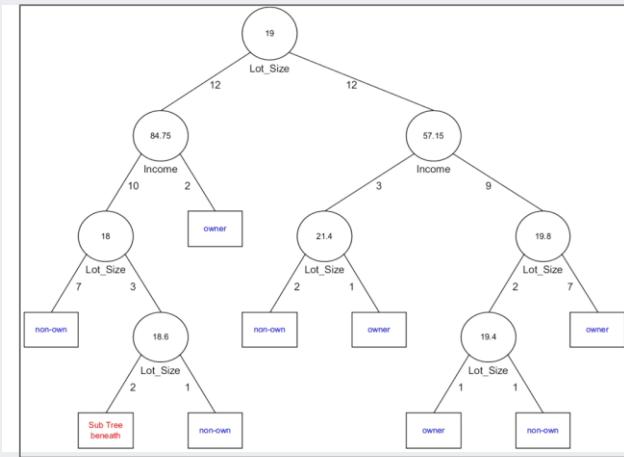
- Full tree constructed by recursive partitioning



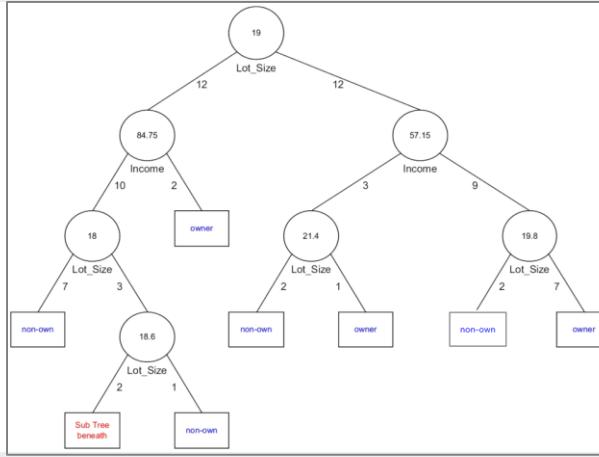
Classification Tree

- Pruning

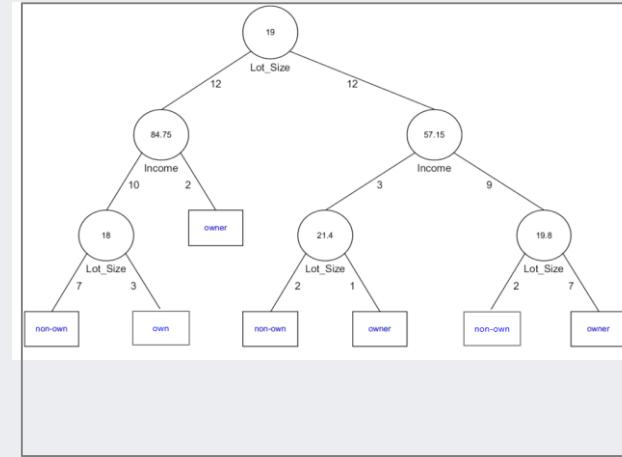
Full Tree



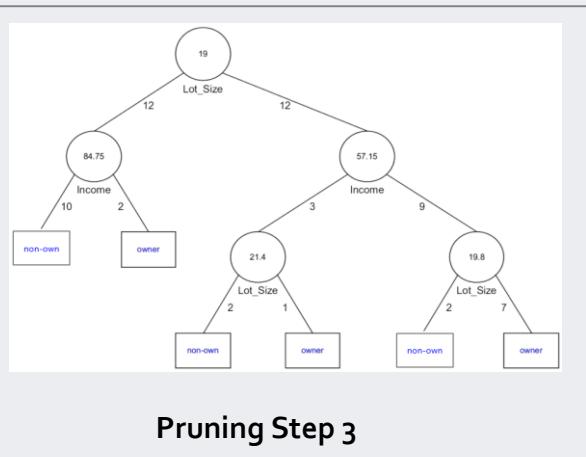
Pruning Step 1



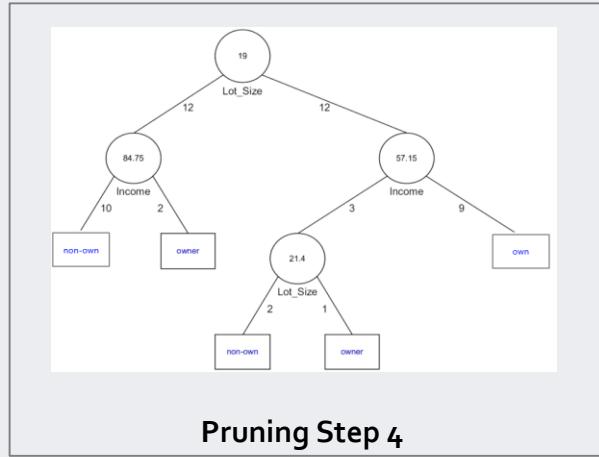
Pruning Step 2



Pruning Step 3



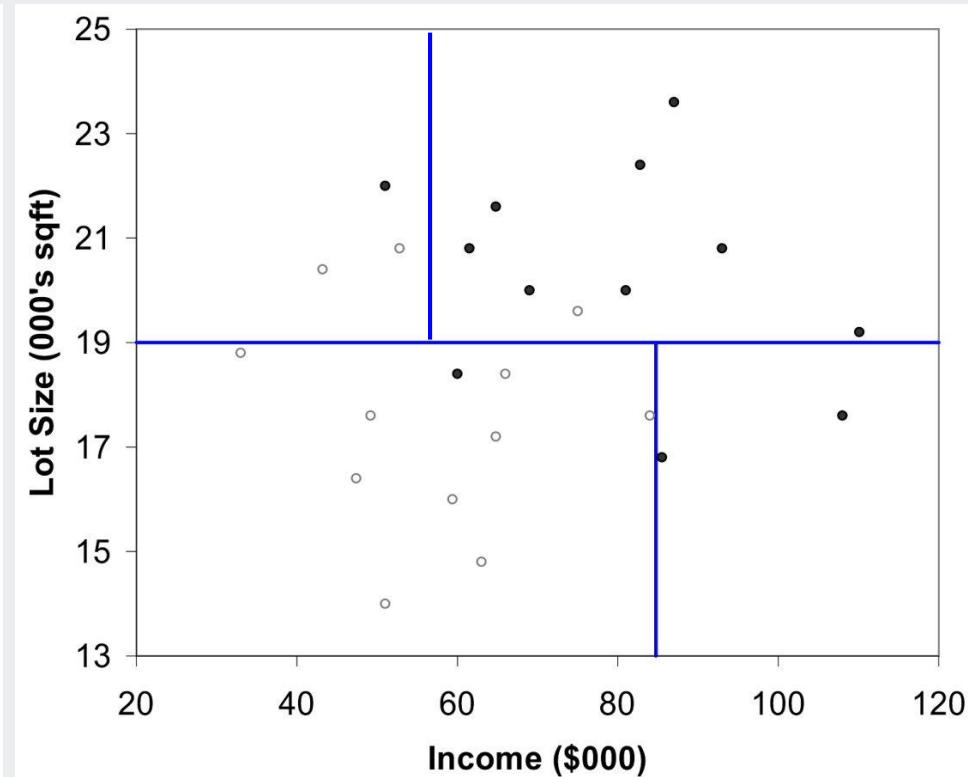
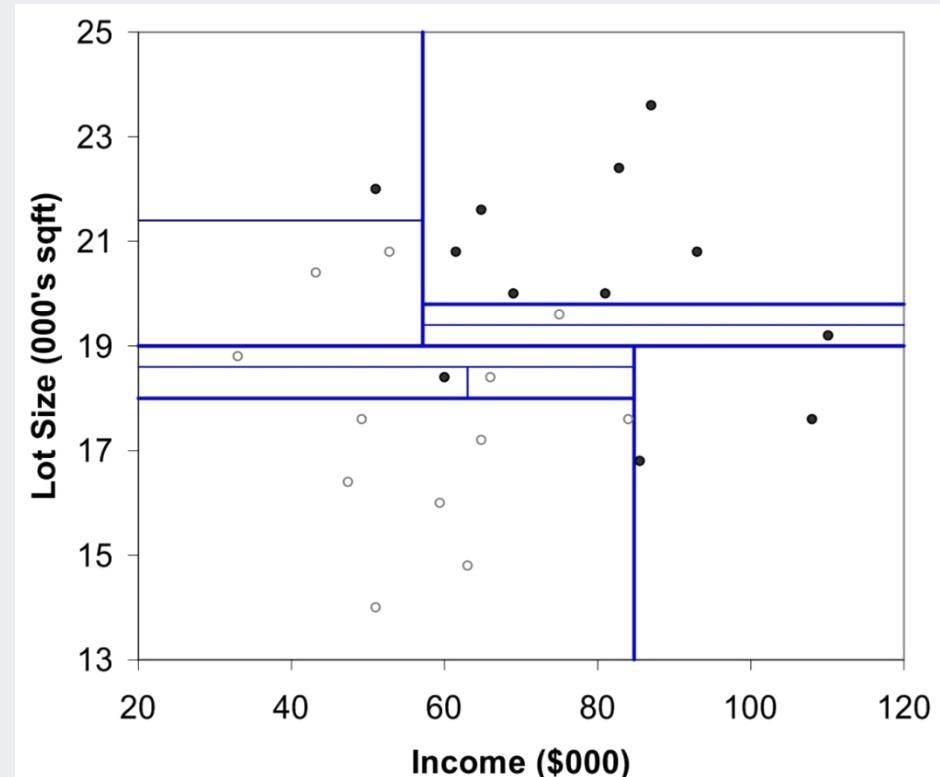
Pruning Step 4



Pruned Tree

Classification Tree

- Full tree vs. Pruned tree



AGENDA

01 Document Classification: Overview

02 Naive Bayesian Classifier

03 k-Nearest Neighbor Classifier

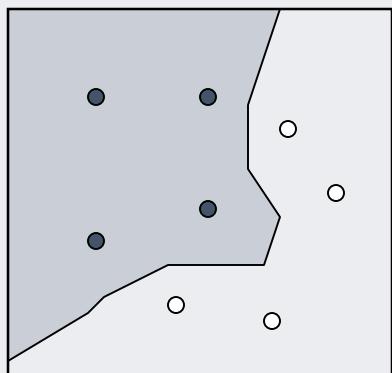
04 Classification Tree

05 Support Vector Machine

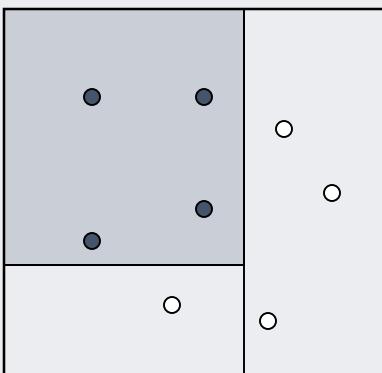
06 R Exercise

Support Vector Machine

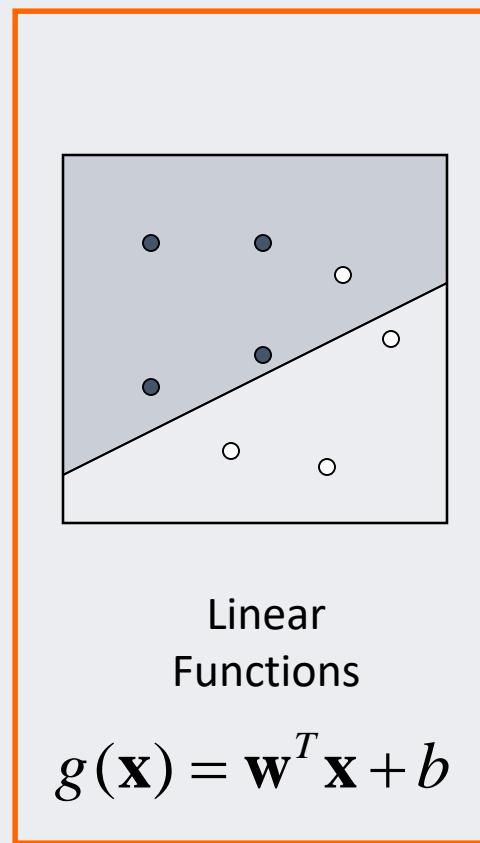
- Discriminant functions in classification



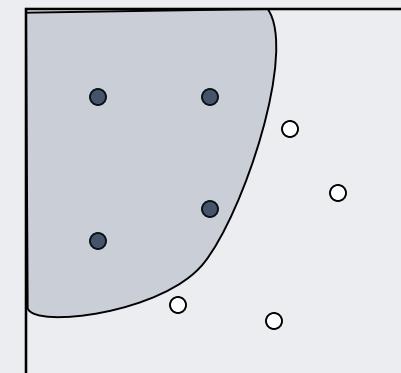
Nearest
Neighbor



Decision
Tree



Linear
Functions

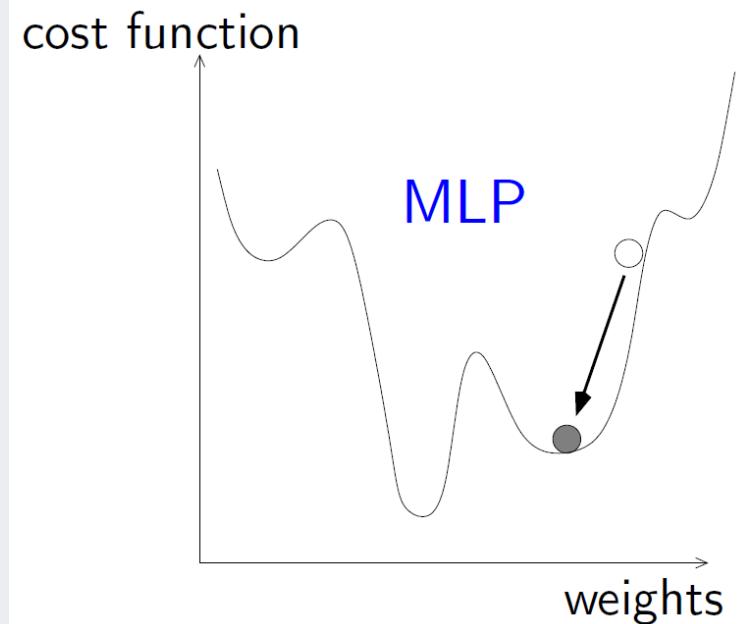
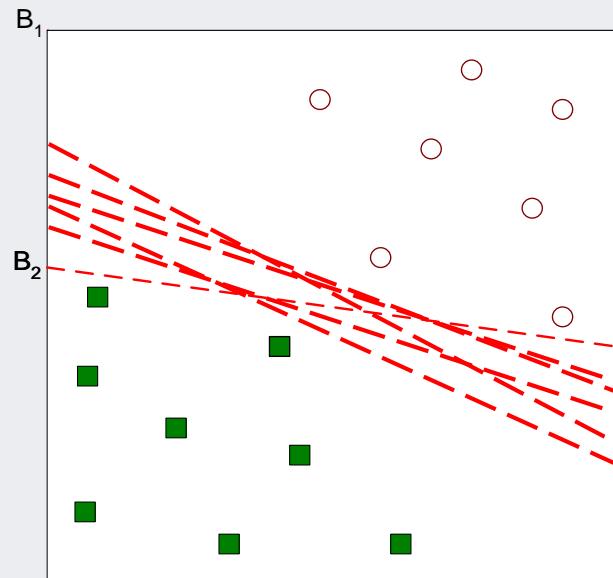


Nonlinear
Functions

Support Vector Machine

Local optimum? Global optimum!

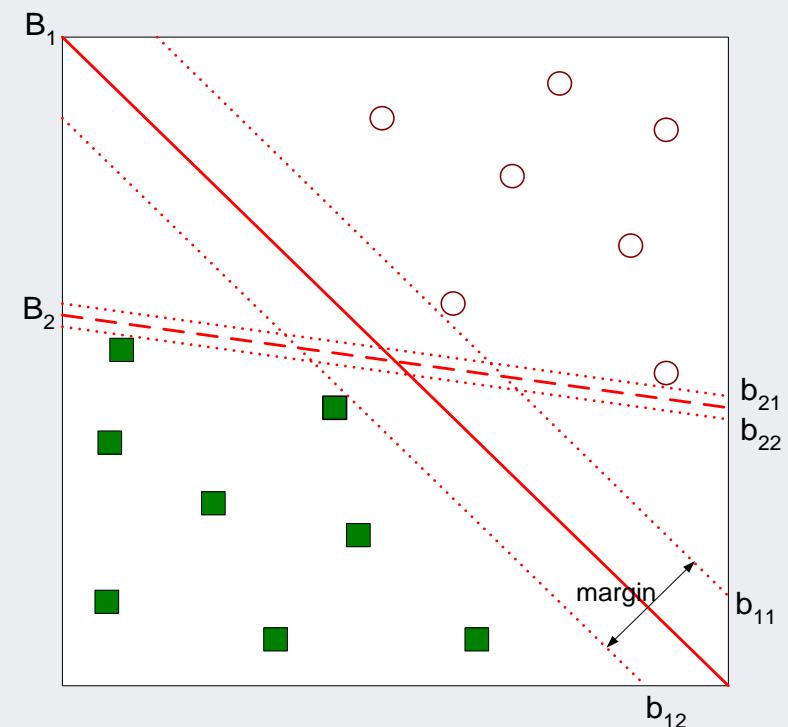
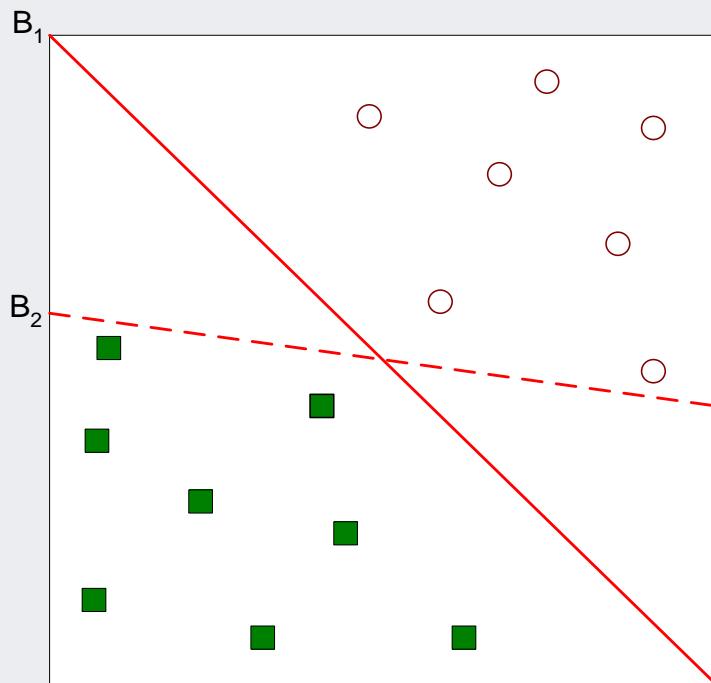
- Many non-linear classification algorithms:
 - ✓ Universal approximation of continuous nonlinear functions
 - ✓ Learning from input-output patterns
 - ✓ Parallel network architecture, multiple inputs and outputs
 - ✓ But, existence of many local optimum!



Support Vector Machine

- Which like is better? B_1 or B_2

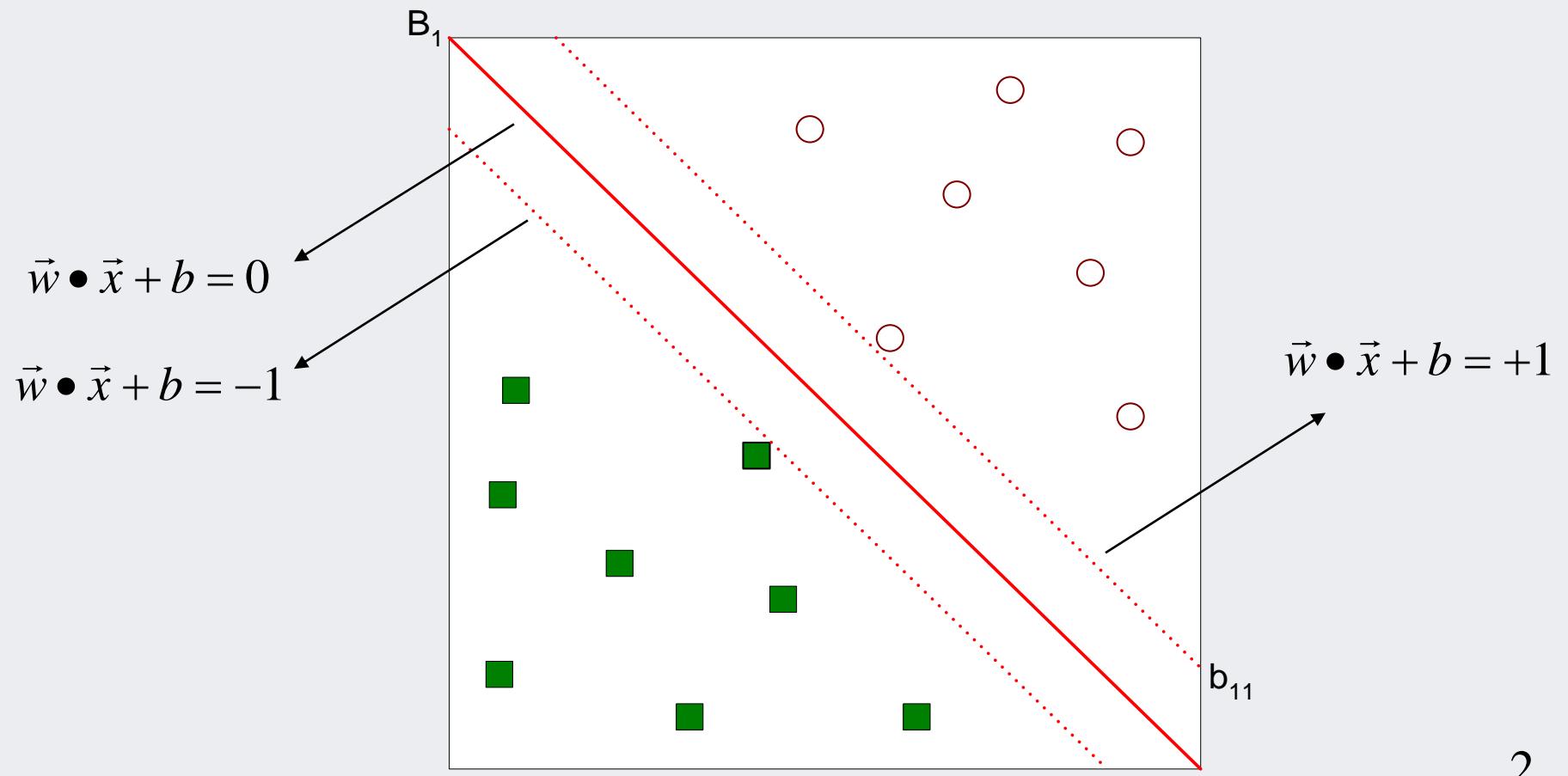
✓ How do you define “better”



- Find the hyperplane that **maximizes the margin!**

Support Vector Machine

- Support Vector Machine Formulation



$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\|\vec{w}\|^2}$$

Support Vector Machine

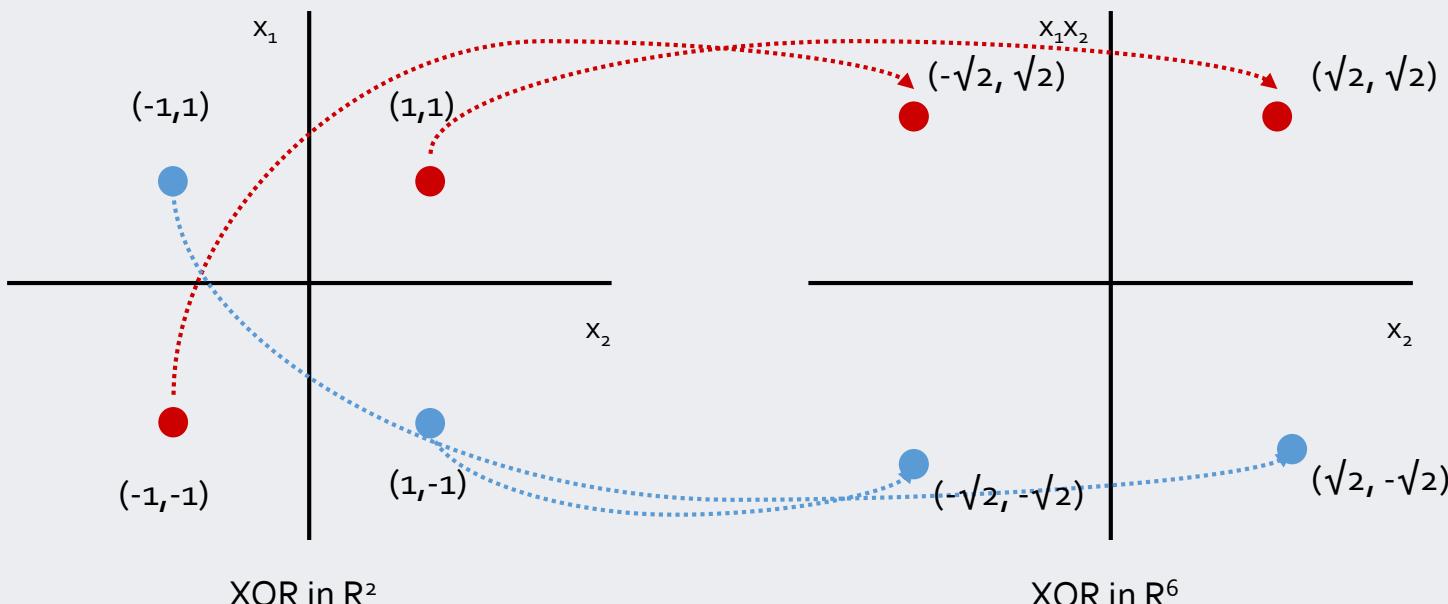
- Support Vector Machine Formulation

| | Hard margin? | Soft margin? |
|-------------------------|----------------------|---|
| Linearly separable? | Basic form | Introduce penalty terms |
| Linearly non-separable? | Utilize Kernel Trick | Introduce penalty terms Utilize Kernel Trick |

Support Vector Machine

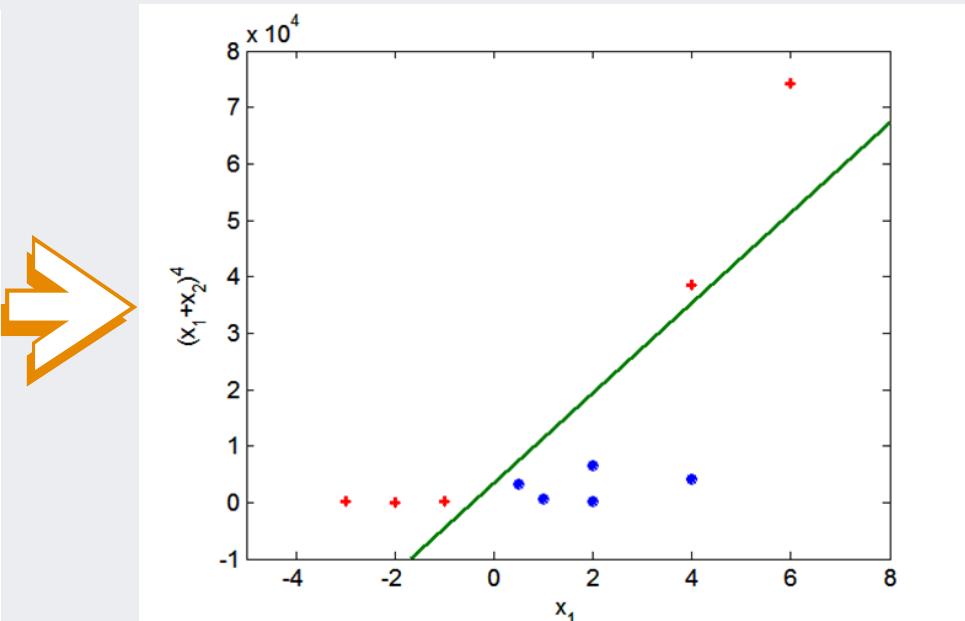
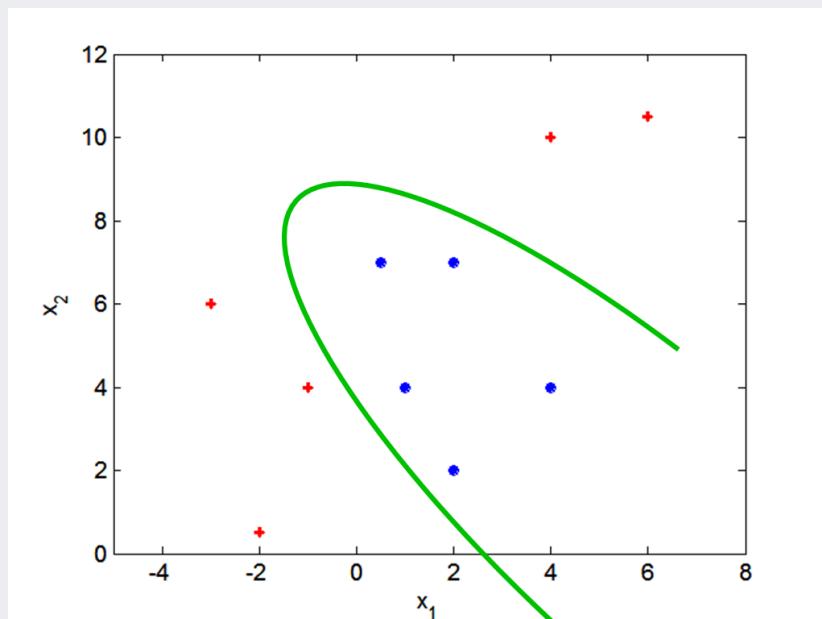
- What if the decision boundary is not linear?
 - ✓ Use a **mapping function** $\Phi(x)$ to transform an input vector from a lower dimensional space into a higher dimensional space

$$\Phi : (\mathbf{x}_1, \mathbf{x}_2) \rightarrow (\mathbf{x}_1^2, \mathbf{x}_2^2, \sqrt{2}\mathbf{x}_1, \sqrt{2}\mathbf{x}_2, \sqrt{2}\mathbf{x}_1\mathbf{x}_2, 1)$$



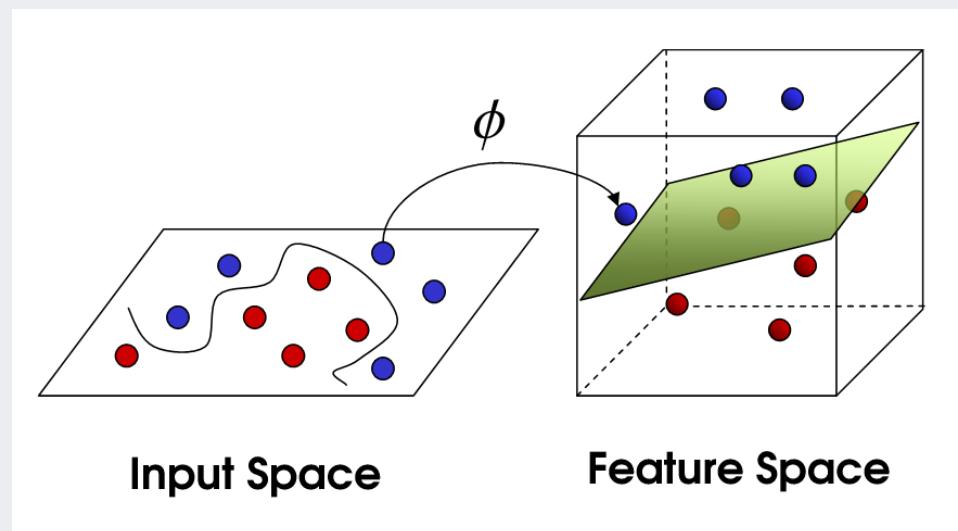
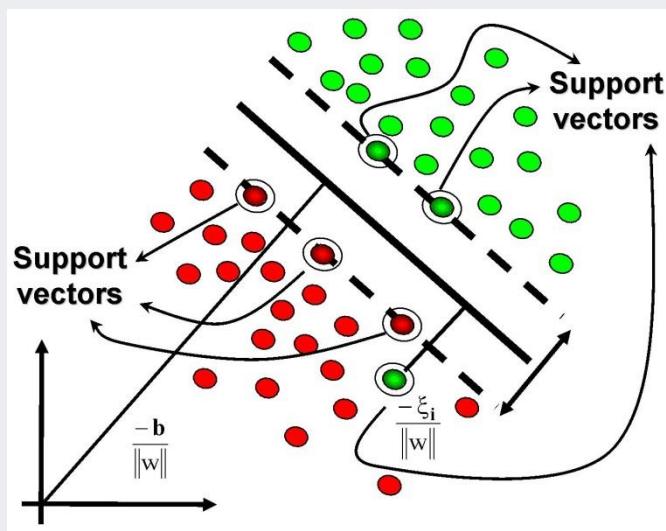
Support Vector Machine

- What if the decision boundary is not linear?
 - ✓ Transform an input vector into a higher feature vector



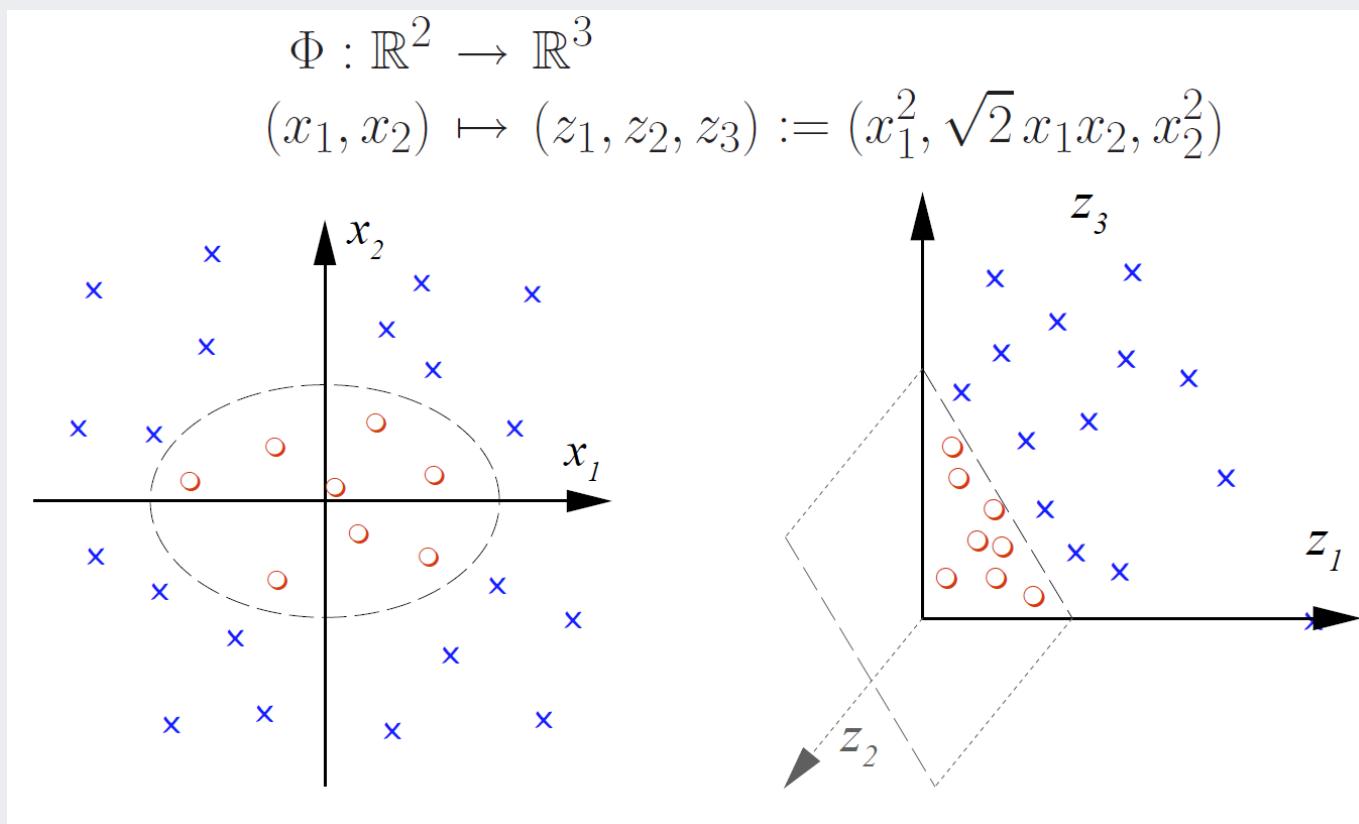
Support Vector Machine

- Goal: To find the best hyperplane that maximizes the margin and minimize the misclassification error
 - ✓ Large margin: preserve the generalization ability
 - ✓ Flexible: can generate an arbitrary shape of boundary in the input space



Support Vector Machine

- Goal: To find the best hyperplane that maximizes the margin and minimize the misclassification error
 - ✓ Large margin: preserve the generalization ability
 - ✓ Flexible: can generate an arbitrary shape of boundary in the input space

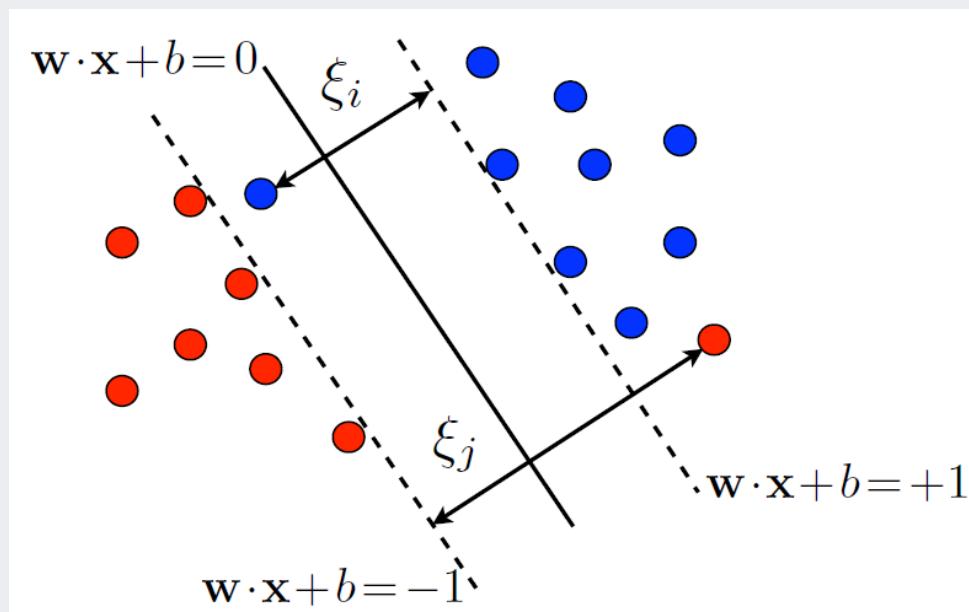


Support Vector Machine

- SVM as an optimization problem

$$\min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i$$

$$s.t. \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i$$



Support Vector Machine

- Lagrangian Problem

$$\begin{aligned} \min \quad L_P(\mathbf{w}, b, \alpha_i) = & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \\ & - \sum_{i=1}^n \alpha_i (y_i (\mathbf{w}^T \Phi(\mathbf{x}_i) + b) - 1 + \xi_i) - \sum_{i=1}^n \mu_i \xi_i \end{aligned}$$

- KKT condition

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^n \alpha_i y_i \Phi(\mathbf{x}_i) \quad \frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^n \alpha_i y_i = 0$$

$$\frac{\partial L_P}{\partial \xi_i} = 0 \Rightarrow C - \alpha_i - \mu_i = 0$$

Support Vector Machine

- Lagrangian Problem (Primal)

$$\begin{aligned} \min \quad L_P(\mathbf{w}, b, \alpha_i) = & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \\ & - \sum_{i=1}^n \alpha_i (y_i (\mathbf{w}^T \Phi(\mathbf{x}_i) + b) - 1 + \xi_i) - \sum_{i=1}^n \mu_i \xi_i \end{aligned}$$

- Lagrangian Problem (Dual)

$$\max \quad L_D = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$$

$$s.t. \quad \sum_{i=1}^n \alpha_i y_i = 0 \quad and \quad 0 \leq \alpha_i \leq C$$

Support Vector Machine

- How to find a mapping function Φ ?

✓ Introduce a Kernel Trick

$$\max L_D = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$$

$$s.t. \quad \sum_{i=1}^n \alpha_i y_i = 0 \quad and \quad 0 \leq \alpha_i \leq C$$

$$\max L_D = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$s.t. \quad \sum_{i=1}^n \alpha_i y_i = 0 \quad and \quad 0 \leq \alpha_i \leq C$$

Support Vector Machine

- **Conditions on Kernel functions**

$K(\mathbf{x}, \mathbf{x}')$ is a valid kernel iff

1. It is symmetric and
2. The matrix:

$$\begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \dots & K(\mathbf{x}_1, \mathbf{x}_N) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \dots & K(\mathbf{x}_2, \mathbf{x}_N) \\ \dots & \dots & \dots & \dots \\ K(\mathbf{x}_N, \mathbf{x}_1) & K(\mathbf{x}_N, \mathbf{x}_2) & \dots & K(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

is **positive semi-definite**

for any $\mathbf{x}_1, \dots, \mathbf{x}_N$ (Mercer's condition)

Support Vector Machine

- Type of Kernel Functions

- ✓ Polynomial

$$K(x, y) = (x \cdot y + c)^d, \quad c > 0$$

- ✓ Gaussian (RBF)

$$K(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right), \quad \sigma \neq 0$$

- ✓ Sigmoid

$$K(x, y) = \tanh(a(x \cdot y) + b), \quad a, b \geq 0$$

Support Vector Machine

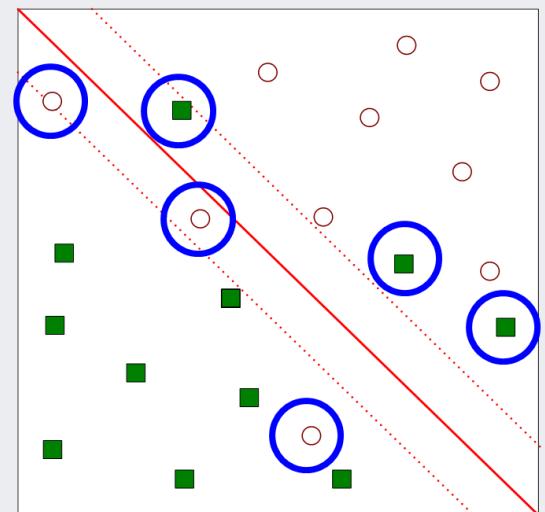
- From the KKT condition, we know that

$$\alpha_i(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i) = 0$$

- ✓ Thus, the only support vectors have $\alpha_i \neq 0$
- ✓ In addition,

$$C - \alpha_i - \mu_i = 0 \quad \& \quad \mu_i \xi_i = 0$$

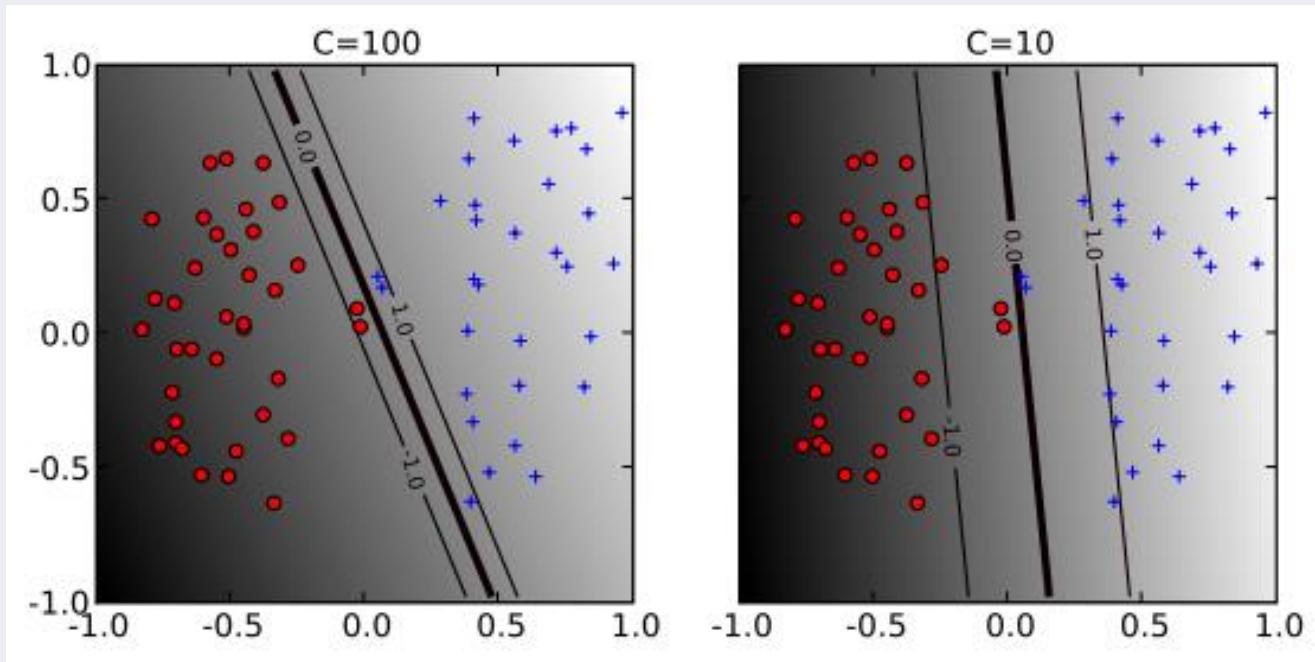
- Case 1: $\alpha_i = 0 \rightarrow$ non-support vectors
- Case 2: $0 < \alpha_i < C \rightarrow$ support vectors on the margin
- Case 3: $\alpha_i = C \rightarrow$ support vectors outside the margin



Support Vector Machine

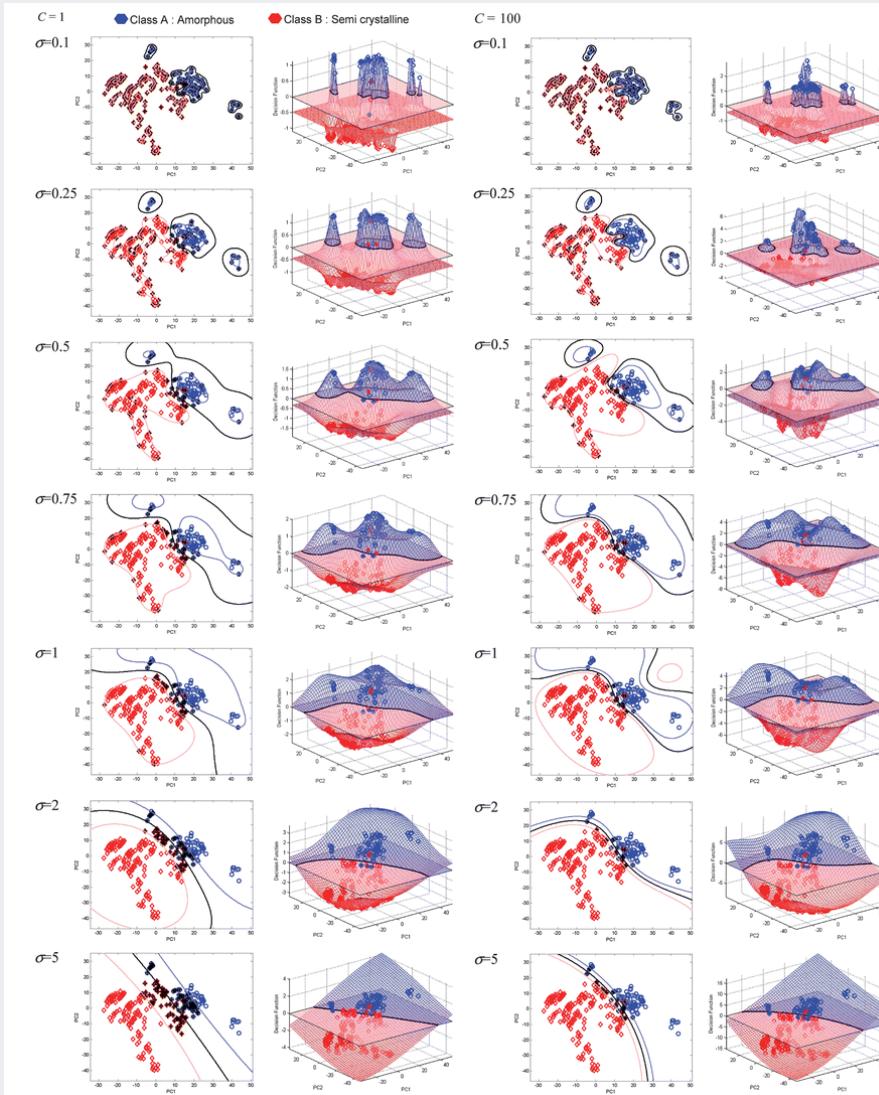
- Regularization cost C
 - ✓ Large C: narrow margin with a few support vectors
 - ✓ Small C: wide margin with many support vectors

$$\min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i$$



Support Vector Machine

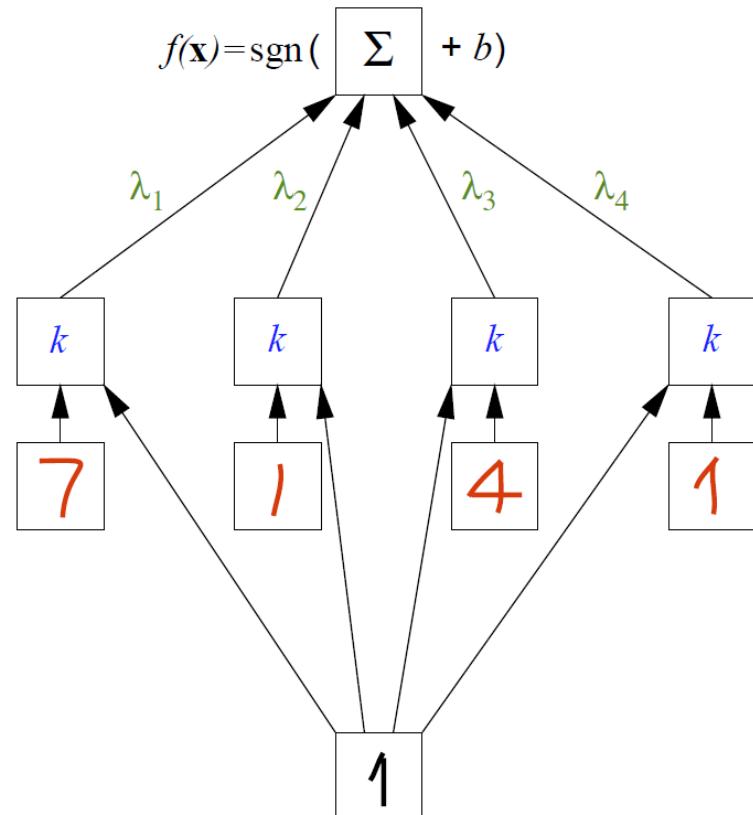
- Decision boundaries according to different σ and C combinations



Support Vector Machine

- Class decision for a new instance \mathbf{x}

$$f(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^N \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \right)$$



classification

weights

comparison: $k(\mathbf{x}, \mathbf{x}_i)$, e.g. $k(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x} \cdot \mathbf{x}_i)^d$

support vectors
 $\mathbf{x}_1 \dots \mathbf{x}_4$

$f(\mathbf{x}) = \text{sgn}(\Sigma \lambda_i k(\mathbf{x}, \mathbf{x}_i) + b)$

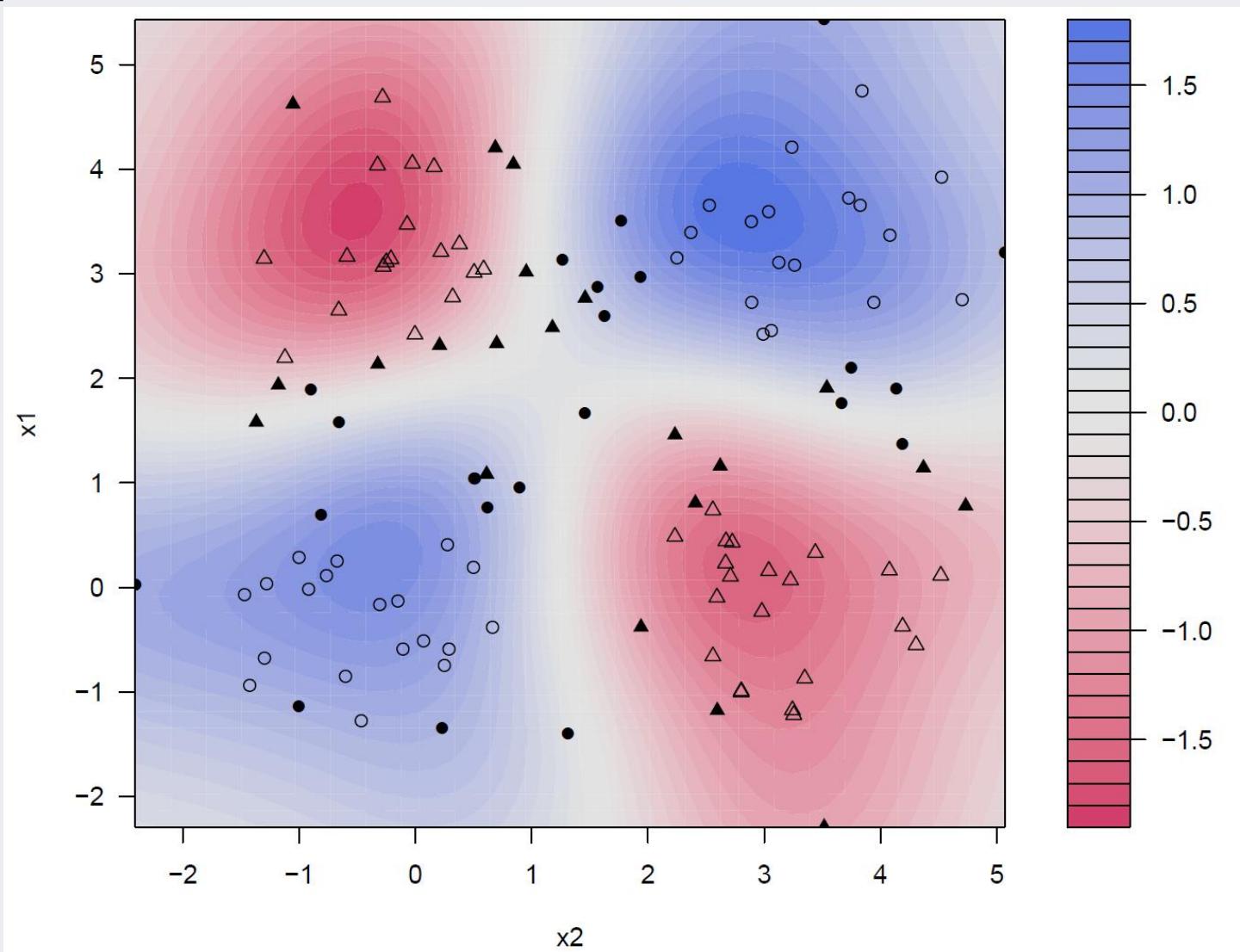
$k(\mathbf{x}, \mathbf{x}_i) = \exp(-\|\mathbf{x} - \mathbf{x}_i\|^2 / c)$

$k(\mathbf{x}, \mathbf{x}_i) = \tanh(\kappa(\mathbf{x} \cdot \mathbf{x}_i) + \theta)$

input vector \mathbf{x}

Support Vector Machine

- XOR problem



AGENDA

01 Document Classification: Overview

02 Naive Bayesian Classifier

03 k-Nearest Neighbor Classifier

04 Classification Tree

05 Support Vector Machine

06 R Exercise

R Exercise

- Data

- ✓ 433 abstracts from TPAMI & 521 abstracts from JoBF
- ✓ 100 features are selected by information gain (IG) and latent semantic indexing (LSI)

The screenshot shows the IEEE Xplore Digital Library homepage. At the top, there are links for 'BROWSE', 'MY SETTINGS', 'GET HELP', and 'WHAT CAN I ACCESS?'. Below is a search bar with 'Enter Search Term' and a 'Search' button. Underneath are buttons for 'Basic Search', 'Author Search', 'Publication Search', 'Advanced Search', and 'Other Search Options'. A banner for 'Pattern Analysis and Machine Intelligence, IEEE Transactions on' is displayed, along with a link to 'Aims & Scope'. On the right, there are three performance metrics: Impact Factor (5.694), Eigenfactor (.04888), and Article Influence Score (3.155). The bottom section contains a brief description of the journal's scope and editorial board.

The screenshot shows the Journal of Banking & Finance website. At the top, there are social media icons and a navigation bar with 'Home', 'Journals', and 'Journal of Banking & Finance'. The main title 'Journal of Banking & Finance' is prominently displayed. Below it, there is a brief description: 'The Journal of Banking and Finance (JBF) publishes theoretical and empirical research papers spanning all the major research fields in finance and banking. The aim of the *Journal of Banking and Finance* is to provide an outlet for the increasing flow of scholarly research concerning financial institutions...'. There are also links for 'Supports Open Access', 'Editors: Carol Alexander, Geert Bekaert', and 'View Editorial Board'. A sidebar on the left lists options like 'Submit Your Paper', 'View Articles', 'Guide for Authors', 'Abstracting/ Indexing', 'Track Your Paper', and 'Order Journal'. At the bottom, there are links for 'Most Downloaded', 'Recent Articles', 'Most Cited', and 'Open Access Articles'. Several article titles are listed at the bottom, such as 'Risk and risk management in the credit card industry' and 'Financial innovation: The bright and the dark sides'.

R Exercise

- Preprocessing

- ✓ Write a performance evaluation function
- ✓ Load the data and transform the IG variables to the TF-IDF format

```
# Performance Evaluation Function -----
perf_eval <- function(cm){

  # Recall
  Recall = cm[2,2]/sum(cm[2,])
  # Precision
  Precision = cm[2,2]/sum(cm[,2])
  # Accuracy
  Acc = (cm[1,1]+cm[2,2])/sum(cm)
  # F1
  F1 = 2*Recall*Precision/(Recall+Precision)

  return(c(Recall, Precision, Acc, F1))
}

# Load the data
load("FSJournal_2017.RData")

# For IG matrix: log(tf)*idf
log_tf <- log(1+t(IG.Mat))

# (Inverse) Document Frequency
N_total <- dim(log_tf)[2]
n_doc <- function(x) length(which(x > 0))
idf_vec <- log(N_total/apply(log_tf, 1, n_doc))
idf_mat <- replicate(N_total, idf_vec)

# TF-IDF
TFIDF <- log_tf*idf_mat
IG.Mat <- as.data.frame(t(TFIDF))

Journal <- c(rep("TPAMI", 433), rep("JoF", 521))
```

R Exercise

- Divide the dataset into training and test dataset
 - ✓ The last 100 abstracts are assigned to the test dataset
 - ✓ Initialize the performance summary matrices

```
# Classification data sets
IG.Data <- data.frame(Journal, IG.Mat)
LSI.Data <- data.frame(Journal, LSI.Mat)

# Training/Validation
trn.idx <- c(1:333, 434:854)
val.idx <- c(334:433, 855:954)

IG.Data.Trn <- IG.Data[trn.idx,]
IG.Data.Val <- IG.Data[val.idx,]

LSI.Data.Trn <- LSI.Data[trn.idx,]
LSI.Data.Val <- LSI.Data[val.idx,]

Class.Perf.IG <- matrix(0, 4, 4)
rownames(Class.Perf.IG) <- c("Naive Bayes", "k-NN", "CTree", "SVM")
colnames(Class.Perf.IG) <- c("Recall", "Precision", "Accuracy", "F1-measure")

Class.Perf.LSI <- matrix(0, 4, 4)
rownames(Class.Perf.LSI) <- c("Naive Bayes", "k-NN", "CTree", "SVM")
colnames(Class.Perf.LSI) <- c("Recall", "Precision", "Accuracy", "F1-measure")
```

R Exercise

- Classifier I: Naive Bayesian Classifier

- ✓ With IG variables

```
# Classification 1: Naive Bayesian Classifier -----
# e1071 package install & call
install.packages("e1071", dependencies = TRUE)
library(e1071)

# Selected Features based on Information Gain
NB.IG <- naiveBayes(Journal ~ ., data = IG.Data.Trn)
NB.IG$apriori
NB.IG$tables

# Predict the new input data based on Naive Bayesian Classifier
NB.IG.Posterior = predict(NB.IG, IG.Data.Val, type = "raw")
NB.IG.PreY = predict(NB.IG, IG.Data.Val, type ="class")

# Confusion Matrix & Performance Evaluation
CF.NB.IG <- table(IG.Data.Val$Journal, NB.IG.PreY)
Class.Perf.IG[1,] <- perf_eval(CF.NB.IG)
Class.Perf.IG

> Class.Perf.IG
      Recall Precision Accuracy F1-measure
Naive Bayes   0.99  0.9428571   0.965  0.9658537
k-NN          0.00  0.0000000   0.000  0.0000000
CTree         0.00  0.0000000   0.000  0.0000000
SVM          0.00  0.0000000   0.000  0.0000000
```

R Exercise

- Classifier I: Naive Bayesian Classifier

- ✓ With LSI variables

```
# Extracted Features based on LSI
NB.LSI <- naiveBayes(Journal ~ ., data = LSI.Data.Trn)
NB.LSI$apriori
NB.LSI$tables

# Predict the new input data based on Naive Bayesian Classifier
NB.LSI.Posterior = predict(NB.LSI, LSI.Data.Val, type = "raw")
NB.LSI.PreY = predict(NB.LSI, LSI.Data.Val, type ="class")

# Confusion Matrix & Performance Evaluation
CF.NB.LSI <- table(LSI.Data.Val$Journal, NB.LSI.PreY)
Class.Perf.LSI[1,] <- perf_eval(CF.NB.LSI)
Class.Perf.LSI
```

```
> Class.Perf.LSI
      Recall Precision Accuracy F1-measure
Naive Bayes   0.68  0.9315068    0.815  0.7861272
k-NN          0.00  0.0000000    0.000  0.0000000
CTree         0.00  0.0000000    0.000  0.0000000
SVM           0.00  0.0000000    0.000  0.0000000
```

R Exercise

- Classifier 2: k-NN

- ✓ Normalize the data

```
# Classification 2: k-Nearest Neighbor Classifier -----
# kknn package install & call
install.packages("kknn", dependencies = TRUE)
library(kknn)

# Normalize the input data
IG.Data.Scaled <- data.frame(Journal, scale(IG.Mat, center = TRUE, scale = TRUE))
LSI.Data.Scaled <- data.frame(Journal, scale(LSI.Mat, center = TRUE, scale = TRUE))

IG.Data.Scaled.Trn <- IG.Data.Scaled[trn.idx,]
IG.Data.Scaled.Val <- IG.Data.Scaled[val.idx,]

LSI.Data.Scaled.Trn <- LSI.Data.Scaled[trn.idx,]
LSI.Data.Scaled.Val <- LSI.Data.Scaled[val.idx,]
```

R Exercise

- Classifier 2: k-NN

- ✓ With the IG variables

- Parameter search: Find the best k & weighting scheme using leave-one-out validation

```
# Leave-one-out validation for finding the best k: Information Gain Features
IG.knn <- train.kknn(Journal ~ ., IG.Data.Scaled.Trn, kmax = 10, distance = 2)
IG.knn$MISCLASS
IG.knn$best.parameters
```

```
> IG.knn$MISCLASS
  optimal
1 0.06631300
2 0.06631300
3 0.06631300
4 0.06631300
5 0.06631300
6 0.06896552
7 0.07161804
8 0.07029178
9 0.07427056
10 0.07427056
```

```
> IG.knn$best.parameters
$kernel
[1] "optimal"

$k
[1] 1
```

R Exercise

- Classifier 2: k-NN
 - ✓ With the IG variables
 - Test with the best parameters

```
# Training the k-nn model with the best parameters
IG.knn <- kknn(Journal ~ ., IG.Data.Scaled.Trn, IG.Data.Scaled.Val, k = IG.knn$best.parameters$k,
                 distance = 2, kernel = IG.knn$best.parameters$kernel)

# Prediction
IG.knn.PreY <- fitted(IG.knn)

# Confusion Matrix & Performance Evaluation
CF.knn.IG <- table(IG.Data.Val$Journal, IG.knn.PreY)
Class.Perf.IG[2,] <- perf_eval(CF.knn.IG)
Class.Perf.IG
```

```
> Class.Perf.IG
      Recall Precision Accuracy F1-measure
Naive Bayes   0.99  0.9428571   0.965  0.9658537
k-NN          0.77  1.0000000   0.885  0.8700565
CTree         0.00  0.0000000   0.000  0.0000000
SVM          0.00  0.0000000   0.000  0.0000000
```

R Exercise

- Classifier 2: k-NN

- ✓ With the LSI variables

- Parameter search: Find the best k & weighting scheme using leave-one-out validation

```
# Leave-one-out validation for finding the best k: LSI Features
LSI.knn <- train.kknn(Journal ~ ., LSI.Data.Scaled.Trn, kmax = 10, distance = 2)
LSI.knn$MISCLASS
LSI.knn$best.parameters
```

```
> LSI.knn$MISCLASS
  optimal
1 0.09681698
2 0.09681698
3 0.09681698
4 0.09681698
5 0.09681698
6 0.08620690
7 0.08222812
8 0.07824934
9 0.08090186
10 0.07559682
```

```
> LSI.knn$best.parameters
  $kernel
[1] "optimal"
  $k
[1] 10
```

R Exercise

- Classifier 2: k-NN
 - ✓ With the LSI variables
 - Test with the best parameters

```
# Tranining the k-nn model with the best parameters
LSI.knn <- kknn(Journal ~ ., LSI.Data.Scaled.Trn, LSI.Data.Scaled.Val, k = LSI.knn$best.parameters$k,
                  distance = 2, kernel = LSI.knn$best.parameters$kernel)

# Prediction
LSI.knn.PreY <- fitted(LSI.knn)

# Confusion Matrix & Performance Evaluation
CF.knn.LSI <- table(LSI.Data.Val$Journal, LSI.knn.PreY)
Class.Perf.LSI[2,] <- perf_eval(CF.knn.LSI)
Class.Perf.LSI
```

```
> Class.Perf.LSI
      Recall Precision Accuracy F1-measure
Naive Bayes   0.68  0.9315068   0.815   0.7861272
k-NN          0.89  0.9569892   0.925   0.9222798
CTree         0.00  0.0000000   0.000   0.0000000
SVM          0.00  0.0000000   0.000   0.0000000
```

R Exercise

- Classifier 3: Classification tree

- ✓ Parameter tuning for the IG variables

```
# Classification 3: Classification Tree -----
# party package install & call
install.packages("party")
library(party)

# A Simple validation for finding the best tree parameters: Information Gain Features
# tree parameter settings
Min.Criterion = c(0.7, 0.8, 0.9)
Min.Split = c(3, 5, 10)
Max.Depth = c(0, 5, 10)
IG.LOO.Result = matrix(0,length(Min.Criterion)*length(Min.Split)*length(Max.Depth),8)

iter.cnt = 1

for (i in 1:length(Min.Criterion))
{
  for ( j in 1:length(Min.Split))
  {
    for ( k in 1:length(Max.Depth))
    {

      cat("CART Min criterion:", Min.Criterion[i], ", Min split:", Min.Split[j], ", Max depth:", Max.Depth[k], "\n")
      Tmp.Control = ctree_control(mincriterion = Min.Criterion[i], minsplit = Min.Split[j], maxdepth = Max.Depth[k])

      Tmp.Trn <- IG.Data.Trn[c(1:233, 334:654),]
      Tmp.Val <- IG.Data.Trn[c(234:333, 655:754),]

      Tmp.Tree <- ctree(Journal ~ ., data = Tmp.Trn, controls = Tmp.Control)
      Tmp.PreY <- predict(Tmp.Tree, newdata = Tmp.Val)

      IG.LOO.Result[iter.cnt,1] <- Min.Criterion[i]
      IG.LOO.Result[iter.cnt,2] <- Min.Split[j]
      IG.LOO.Result[iter.cnt,3] <- Max.Depth[k]
      IG.LOO.Result[iter.cnt,4:7] <- perf_eval(table(Tmp.Val$Journal, Tmp.PreY))
      IG.LOO.Result[iter.cnt,8] <- length(nodes(Tmp.Tree, unique(where(Tmp.Tree))))
      iter.cnt = iter.cnt + 1
    }
  }
}
```

R Exercise

- Classifier 3: Classification tree

- ✓ Find the best parameters

```
IG.LOO.Result <- IG.LOO.Result[order(IG.LOO.Result[,7], decreasing = T),]  
IG.LOO.Result
```

```
Best.IG.Criterion <- IG.LOO.Result[1,1]  
Best.IG.Split <- IG.LOO.Result[1,2]  
Best.IG.Depth <- IG.LOO.Result[1,3]
```

```
> IG.LOO.Result  
[.1] [.2] [.3] [.4] [.5] [.6] [.7] [.8]  
[1,] 0.7 3 0 0.88 0.880000 0.88 0.8800000 8  
[2,] 0.7 3 10 0.88 0.880000 0.88 0.8800000 8  
[3,] 0.7 5 0 0.88 0.880000 0.88 0.8800000 8  
[4,] 0.7 5 10 0.88 0.880000 0.88 0.8800000 8  
[5,] 0.7 10 0 0.88 0.880000 0.88 0.8800000 8  
[6,] 0.7 10 10 0.88 0.880000 0.88 0.8800000 8  
[7,] 0.8 3 0 0.88 0.880000 0.88 0.8800000 8  
[8,] 0.8 3 10 0.88 0.880000 0.88 0.8800000 8  
[9,] 0.8 5 0 0.88 0.880000 0.88 0.8800000 8  
[10,] 0.8 5 10 0.88 0.880000 0.88 0.8800000 8  
[11,] 0.8 10 0 0.88 0.880000 0.88 0.8800000 8  
[12,] 0.8 10 10 0.88 0.880000 0.88 0.8800000 8  
[13,] 0.9 3 0 0.88 0.880000 0.88 0.8800000 8  
[14,] 0.9 3 10 0.88 0.880000 0.88 0.8800000 8  
[15,] 0.9 5 0 0.88 0.880000 0.88 0.8800000 8  
[16,] 0.9 5 10 0.88 0.880000 0.88 0.8800000 8  
[17,] 0.9 10 0 0.88 0.880000 0.88 0.8800000 8  
[18,] 0.9 10 10 0.88 0.880000 0.88 0.8800000 8  
[19,] 0.7 3 5 0.86 0.877551 0.87 0.8686869 7  
[20,] 0.7 5 5 0.86 0.877551 0.87 0.8686869 7  
[21,] 0.7 10 5 0.86 0.877551 0.87 0.8686869 7  
[22,] 0.8 3 5 0.86 0.877551 0.87 0.8686869 7  
[23,] 0.8 5 5 0.86 0.877551 0.87 0.8686869 7  
[24,] 0.8 10 5 0.86 0.877551 0.87 0.8686869 7  
[25,] 0.9 3 5 0.86 0.877551 0.87 0.8686869 7  
[26,] 0.9 5 5 0.86 0.877551 0.87 0.8686869 7  
[27,] 0.9 10 5 0.86 0.877551 0.87 0.8686869 7
```

R Exercise

- Classifier 3: Classification tree

- ✓ Train the tree with the best parameters

```
Best.IG.Criterion <- IG.LOO.Result[1,1]
Best.IG.Split <- IG.LOO.Result[1,2]
Best.IG.Depth <- IG.LOO.Result[1,3]

IG.Best.Control = ctree_control(mincriterion = Best.IG.Criterion, minsplit = Best.IG.Split, maxdepth = Best.IG.Depth)

IG.Tree <- ctree(Journal ~ ., data = IG.Data.Trn, controls = IG.Best.Control)
IG.Tree.PreY <- predict(IG.Tree, newdata = IG.Data.Val)

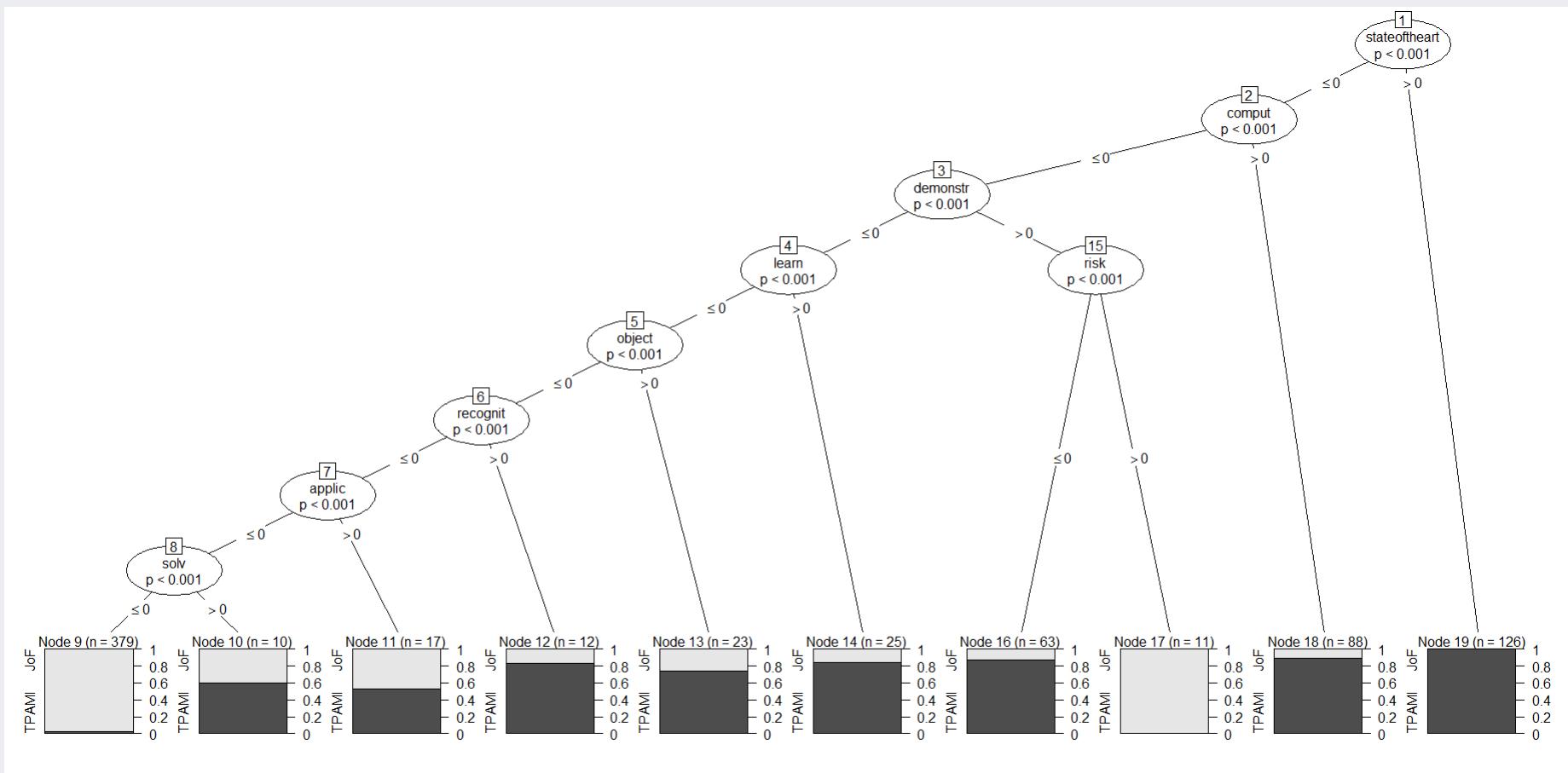
CF.Tree.IG <- table(IG.Data.Val$Journal, IG.Tree.PreY)
Class.Perf.IG[3,] <- perf_eval(CF.Tree.IG)
Class.Perf.IG
|
plot(IG.Tree)
```

```
> Class.Perf.IG
      Recall Precision Accuracy F1-measure
Naive Bayes    0.99  0.9428571   0.965   0.9658537
k-NN           0.77  1.0000000   0.885   0.8700565
CTree          0.92  0.8846154   0.900   0.9019608
SVM            0.00  0.0000000   0.000   0.0000000
```

R Exercise

- Classifier 3: Classification tree

✓ Train the tree with the best parameters



R Exercise

- Classifier 3: Classification tree

- ✓ Parameter tuning for the LSI variables

```
# A Simple validation for finding the best tree parameters: LSI Features
# tree parameter settings
Min.Criterion = c(0.7, 0.8, 0.9)
Min.Split = c(3, 5, 10)
Max.Depth = c(0, 5, 10)
LSI.LOO.Result = matrix(0,length(Min.Criterion)*length(Min.Split)*length(Max.Depth),8)

iter.cnt = 1

for (i in 1:length(Min.Criterion))
{
  for ( j in 1:length(Min.Split))
  {
    for ( k in 1:length(Max.Depth))
    {

      cat("CART Min criterion:", Min.Criterion[i], ", Min split:", Min.Split[j], ", Max depth:", Max.Depth[k], "\n")
      Tmp.Control = ctree_control(mincriterion = Min.Criterion[i], minsplit = Min.Split[j], maxdepth = Max.Depth[k])

      Tmp.Trn <- LSI.Data.Trn[c(1:233, 334:654),]
      Tmp.Val <- LSI.Data.Trn[c(234:333, 655:754),]

      Tmp.Tree <- ctree(Journal ~ ., data = Tmp.Trn, controls = Tmp.Control)
      Tmp.PreY <- predict(Tmp.Tree, newdata = Tmp.Val)

      LSI.LOO.Result[iter.cnt,1] <- Min.Criterion[i]
      LSI.LOO.Result[iter.cnt,2] <- Min.Split[j]
      LSI.LOO.Result[iter.cnt,3] <- Max.Depth[k]
      LSI.LOO.Result[iter.cnt,4:7] <- perf_eval(table(Tmp.Val$Journal, Tmp.PreY))
      LSI.LOO.Result[iter.cnt,8] <- length(nodes(Tmp.Tree, unique(where(Tmp.Tree))))
      iter.cnt = iter.cnt + 1
    }
  }
}
```

R Exercise

- Classifier 3: Classification tree

- ✓ Find the best parameters

```
LSI.LOO.Result <- LSI.LOO.Result[order(LSI.LOO.Result[,7], decreasing = T),]  
LSI.LOO.Result
```

```
Best.LSI.Criterion <- LSI.LOO.Result[1,1]  
Best.LSI.Split <- LSI.LOO.Result[1,2]  
Best.LSI.Depth <- LSI.LOO.Result[1,3]
```

```
> LSI.LOO.Result  
     [,1] [,2] [,3] [,4]      [,5] [,6]      [,7] [,8]  
[1,]  0.7   3    0    1 0.9803922 0.99 0.990099  3  
[2,]  0.7   3    5    1 0.9803922 0.99 0.990099  3  
[3,]  0.7   3   10   1 0.9803922 0.99 0.990099  3  
[4,]  0.7   5    0    1 0.9803922 0.99 0.990099  3  
[5,]  0.7   5    5    1 0.9803922 0.99 0.990099  3  
[6,]  0.7   5   10   1 0.9803922 0.99 0.990099  3  
[7,]  0.7   10   0    1 0.9803922 0.99 0.990099  3  
[8,]  0.7   10   5    1 0.9803922 0.99 0.990099  3  
[9,]  0.7   10   10   1 0.9803922 0.99 0.990099  3  
[10,] 0.8   3    0    1 0.9803922 0.99 0.990099  2  
[11,] 0.8   3    5    1 0.9803922 0.99 0.990099  2  
[12,] 0.8   3   10   1 0.9803922 0.99 0.990099  2  
[13,] 0.8   5    0    1 0.9803922 0.99 0.990099  2  
[14,] 0.8   5    5    1 0.9803922 0.99 0.990099  2  
[15,] 0.8   5   10   1 0.9803922 0.99 0.990099  2  
[16,] 0.8   10   0    1 0.9803922 0.99 0.990099  2  
[17,] 0.8   10   5    1 0.9803922 0.99 0.990099  2  
[18,] 0.8   10   10   1 0.9803922 0.99 0.990099  2  
[19,] 0.9   3    0    1 0.9803922 0.99 0.990099  2  
[20,] 0.9   3    5    1 0.9803922 0.99 0.990099  2  
[21,] 0.9   3   10   1 0.9803922 0.99 0.990099  2  
[22,] 0.9   5    0    1 0.9803922 0.99 0.990099  2  
[23,] 0.9   5    5    1 0.9803922 0.99 0.990099  2  
[24,] 0.9   5   10   1 0.9803922 0.99 0.990099  2  
[25,] 0.9   10   0    1 0.9803922 0.99 0.990099  2  
[26,] 0.9   10   5    1 0.9803922 0.99 0.990099  2  
[27,] 0.9   10   10   1 0.9803922 0.99 0.990099  2
```

R Exercise

- Classifier 3: Classification tree
 - ✓ Train the tree with the best parameters

```
LSI.Best.Control = ctree_control(mincriterion = Best.LSI.Criterion, minsplit = Best.LSI.Split, maxdepth = Best.LSI.Depth)

LSI.Tree <- ctree(Journal ~ ., data = LSI.Data.Trn, controls = LSI.Best.Control)
LSI.Tree.PreY <- predict(LSI.Tree, newdata = LSI.Data.Val)

CF.Tree.LSI <- table(LSI.Data.Val$Journal, LSI.Tree.PreY)
Class.Perf.LSI[3,] <- perf_eval(CF.Tree.LSI)
Class.Perf.LSI

plot(LSI.Tree)
```

```
> Class.Perf.LSI
      Recall Precision Accuracy F1-measure
Naive Bayes   0.68  0.9315068    0.815  0.7861272
k-NN          0.89  0.9569892    0.925  0.9222798
CTree         0.98  0.9800000    0.980  0.9800000
SVM           0.00  0.0000000    0.000  0.0000000
```

R Exercise

- Classifier 3: Classification tree
 - ✓ Train the tree with the best parameters



R Exercise

- Classifier 4: SVM

- ✓ Parameter search with the IG variables

```
# Classification 4: Support Vector Machine -----
# party package install & call
install.packages("e1071")
library(e1071)

# Parameter Search: Information Gain Features
IG.Param.Search <- tune(svm, Journal ~ ., data = IG.Data.Scaled.Trn,
                        ranges = list(gamma = 2-10:0, cost = 2-3:5), tunecontrol = tune.control(sampling = "fix"))

IG.Param.Search
```

```
Best.Gamma <- IG.Param.Search$best.parameters$gamma
Best.Cost <- IG.Param.Search$best.parameters$cost
```

```
> IG.Param.Search
```

```
Parameter tuning of 'svm':
```

- sampling method: fixed training/validation set

- best parameters:

- gamma cost

- 0.0078125 0.25

- best performance: 0.003968254

R Exercise

- Classifier 4: SVM

- ✓ Train the SVM with the best parameters and evaluate its classification performance

```
# Train the SVM and prediction
IG.SVM <- svm(Journal ~ ., data = IG.Data.Scaled.Trn, gamma = Best.Gamma, cost = Best.Cost)
IG.SVM.PreY <- predict(IG.SVM, newdata = IG.Data.Val)

CF.SVM.IG <- table(IG.Data.Val$Journal, IG.SVM.PreY)
Class.Perf.IG[4,] <- perf_eval(CF.SVM.IG)
Class.Perf.IG
```

```
> Class.Perf.IG
      Recall Precision Accuracy F1-measure
Naive Bayes    0.99  0.9428571   0.965  0.9658537
k-NN           0.77  1.0000000   0.885  0.8700565
CTree          0.92  0.8846154   0.900  0.9019608
SVM            0.98  1.0000000   0.990  0.9898990
```

R Exercise

- Classifier 4: SVM

- ✓ Parameter search with the LSI variables

```
# Parameter Search: LSI Features
LSI.Param.Search <- tune(svm, Journal ~ ., data = LSI.Data.Scaled.Trn,
                        ranges = list(gamma = 2^(-10:0), cost = 2^(-3:5)), tunecontrol = tune.control(sampling = "fix"))
```

```
LSI.Param.Search
```

```
Best.Gamma <- LSI.Param.Search$best.parameters$gamma
Best.Cost <- LSI.Param.Search$best.parameters$cost
```

```
> LSI.Param.Search
```

Parameter tuning of 'svm':

- sampling method: fixed training/validation set
- best parameters:

| | |
|------------|------|
| gamma | cost |
| 0.00390625 | 2 |
- best performance: 0.007936508

R Exercise

- Classifier 4: SVM

- ✓ Train the SVM with the best parameters and evaluate its classification performance

```
# Train the SVM and prediction
LSI.SVM <- svm(Journal ~ ., data = LSI.Data.Scaled.Trn, gamma = Best.Gamma, cost = Best.Cost)
LSI.SVM.PreY <- predict(LSI.SVM, newdata = LSI.Data.Val)

CF.SVM.LSI <- table(LSI.Data.Val$Journal, LSI.SVM.PreY)
Class.Perf.LSI[4,] <- perf_eval(CF.SVM.LSI)
Class.Perf.LSI
```

```
> Class.Perf.LSI
      Recall Precision Accuracy F1-measure
Naive Bayes   0.68  0.9315068    0.815  0.7861272
k-NN          0.89  0.9569892    0.925  0.9222798
CTree         0.98  0.9800000    0.980  0.9800000
SVM           0.98  0.9702970    0.975  0.9751244
```

R Exercise

- Compare the classification performances of the four classifiers with two variable sets

```
> Class.Perf.IG
```

| | Recall | Precision | Accuracy | F1-measure |
|-------------|--------|-----------|----------|------------|
| Naive Bayes | 0.99 | 0.9428571 | 0.965 | 0.9658537 |
| k-NN | 0.77 | 1.0000000 | 0.885 | 0.8700565 |
| CTree | 0.92 | 0.8846154 | 0.900 | 0.9019608 |
| SVM | 0.98 | 1.0000000 | 0.990 | 0.9898990 |

```
> Class.Perf.LSI
```

| | Recall | Precision | Accuracy | F1-measure |
|-------------|--------|-----------|----------|------------|
| Naive Bayes | 0.68 | 0.9315068 | 0.815 | 0.7861272 |
| k-NN | 0.89 | 0.9569892 | 0.925 | 0.9222798 |
| CTree | 0.98 | 0.9800000 | 0.980 | 0.9800000 |
| SVM | 0.98 | 0.9702970 | 0.975 | 0.9751244 |

- ✓ In terms of simple accuracy, SVM > Naive Bayes > Classification Tree > k-NN when IG variables are used, while Classification Tree > SVM > k-NN > Naive Bayes when LSI variables are used
- ✓ Supervised variable selection (IG) goes well with Naive Bayesian Classifier



ANY
questions?

References

Research Papers

- Andrews, N. O. and Fox, E.A. (2007). Recent Developments in Document Clustering.
- Choi, S.-S., Cha, S.-H., and Tappert, C. C. (2010). A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics* 8(1): 43-48.
- Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A.Y., Foufou, S., and Bouras, A. (2014). A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE Transactions on Emerging Topics in Computing* 2(3): 267-279.
- Kang, P., & Cho, S. (2009, September). K-means clustering seeds initialization based on centrality, sparsity, and isotropy. In *International Conference on Intelligent Data Engineering and Automated Learning* (pp. 109-117). Springer Berlin Heidelberg.
- Kohonen, T., Kaski, S., Lagus, K., Salojarvi, J., Honkela, J., Paatero, V., Saarela, A. (2000). Self organization of a massive document collection, *IEEE Transactions on Neural Networks* 11(3): 574-585.
- Lee, M., Pincombe, B. and Welsh, M. (2005). An empirical evaluation of models of text document similarity. *Cognitive Science*: 1254-1259.
- Liu, Y., Li, Z., Xiong, H., Gao, X., and Wu, J. (2010). Understanding of internal clustering validation measures, In: *proceedings of the 2010 IEEE International Conference on Data Mining (ICDM'10)*: 911-916.
- Zhao, Y. and Karypis, G. (2002). Comparison of agglomerative and partitional document clustering algorithms. No. TR-02-014. MINNESOTA UNIV MINNEAPOLIS DEPT OF COMPUTER SCIENCE.

References

Other Materials

- Image on the first page: <http://aylien.com/press/aylien-announces-text-analysis-api-suite-nlp-information-retrieval-machine-learning-apis-extract-insights-documents/>
- Jurafsky, D. (2015). [Text Classification and Naïve Bayes](#), CS 124: From Languages to Information, Stanford University.
- Suykens, J. (2003). [Least Squares Support Vector Machines](#). IJCNN 2003 Tutorial.
- Abu-Mostafa, Y. (2012). [Lecture 14: Support Vector Machines](#). Learning From Data. Caltech.
- Burges, C.J.C. (1998). A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2: 121-167.