Data Mining

Course No: CSE 4221

Topic 3: Classification, Clustering and Prediction

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by **labels** indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Prediction Problems: Classification vs. Numeric Prediction

- Classification
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Numeric Prediction
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - 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: Definition

- Given a collection of records (training set)
  - Each record is by characterized by a tuple ($\mathbf{x}$,y), where $\mathbf{x}$ is the attribute set and y is the class label
    - $\mathbf{x}$: attribute, predictor, independent variable, input
    - y: class, response, dependent variable, output
- Task:
  - Learn a model that maps each attribute set $\mathbf{x}$ into one of the predefined class labels y

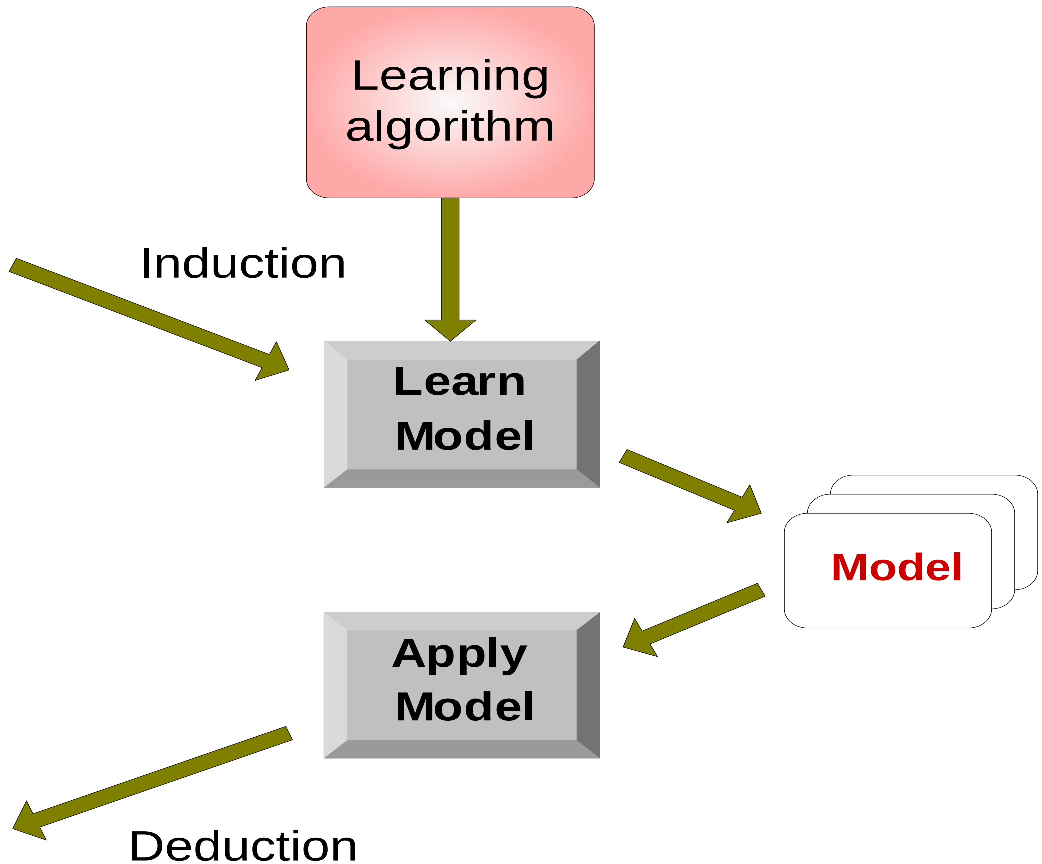
Examples of Classification Task

Task	Attribute set, *x*	Class label, *y*
Categorizing email messages	Features extracted from email message header and content	spam or non-spam
Identifying tumor cells	Features extracted from MRI scans	malignant or benign cells
Cataloging galaxies	Features extracted from telescope images	Elliptical, spiral, or irregular-shaped galaxies

General Approach for Building Classification Model

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Classification—A Two-Step Process

- 1st step: Model construction – describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae

Classification—A Two-Step Process

- 2nd step: Model usage – for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set (otherwise overfitting)
  - If the accuracy is acceptable, use the model to classify new data
- Note: If the test set is used to select models, it is called validation (test) set

Process (1): Model Construction

Training Data

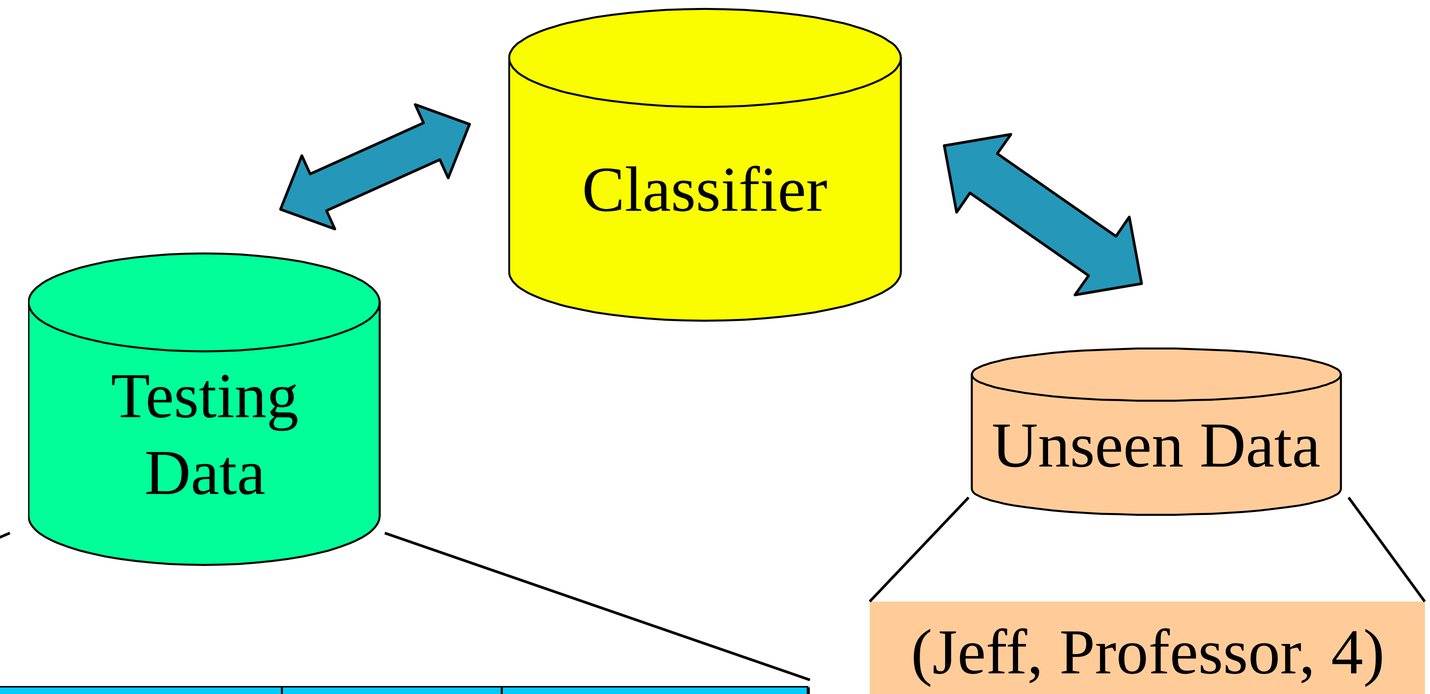
Classification Algorithms

Classifier (Model)

NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Process (2): Using the Model in Prediction



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Classification Techniques

- Base Classifiers
  - Decision Tree based Methods
  - Rule-based Methods
  - Nearest-neighbor
  - Neural Networks
  - Deep Learning
  - Naïve Bayes and Bayesian Belief Networks
  - Support Vector Machines

- Ensemble Classifiers
  - Boosting, Bagging, Random Forests

Decision Tree

- **Why Decision trees?**
  - Decision tress often mimic the human level thinking so its so simple to understand the data and make some good interpretations.
  - Decision trees actually make you see the logic for the data to interpret(not like black box algorithms like SVM, NN, etc..)

Decision Tree

- A decision tree is a tree with the following properties
  - An inner node represents an attribute.
  - An edge represents a test on the attribute of the father node.
  - A leaf represents one of the classes.
- Use of decision tree: Classifying an unknown sample
  - Test the attribute values of the sample against the decision tree
- Construction of a decision tree
  - Based on the training data
  - Top-Down strategy

Decision Tree

Training Data Set

Outlook	Temp	Humidity	Windy	Class
Sunny	79	90	true	No play
Sunny	56	70	False	Play
Sunny	79	75	True	Play
Sunny	60	90	True	No Play
Overcast	88	88	False	Play
Overcast	63	75	True	Play
Overcast	88	95	False	Play
Rain	78	60	False	Play
Rain	66	70	False	No Play
Rain	68	60	True	No Play

- Example:
  - The data set has five attributes.
  - There is a special attribute: the attribute **class** is the class label.
  - The attributes, temp (temperature) and humidity are numerical attributes
  - Other attributes are categorical, that is, they cannot be ordered.

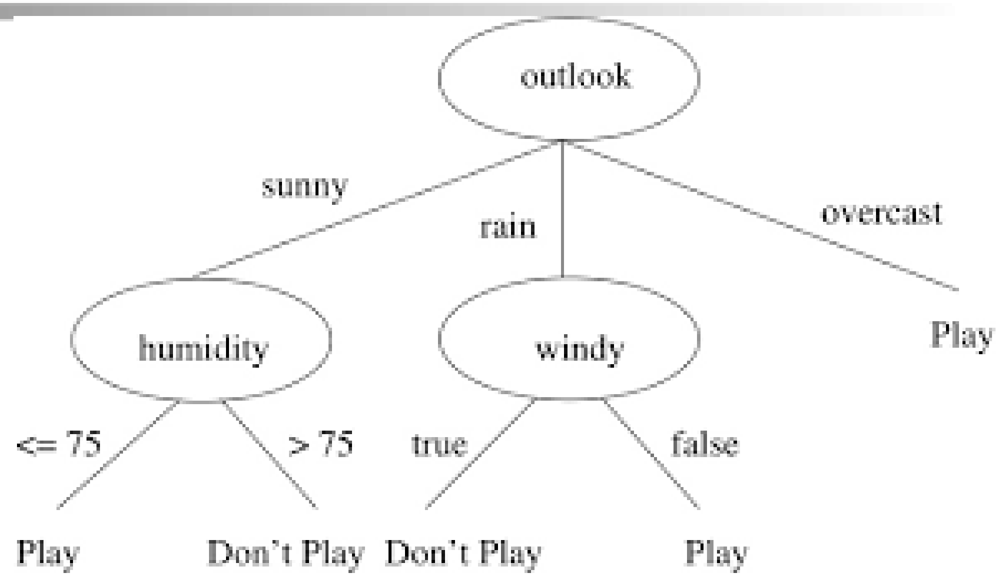
Decision Tree

Training Data Set

Outlook	Temp	Humidity	Windy	Class
Sunny	79	90	true	No play
Sunny	56	70	False	Play
Sunny	79	75	True	Play
Sunny	60	90	True	No Play
Overcast	88	88	False	Play
Overcast	63	75	True	Play
Overcast	88	95	False	Play
Rain	78	60	False	Play
Rain	66	70	False	No Play
Rain	68	60	True	No Play

- Example (cont.):
  - Based on the training data set, we want to find a set of rules to know what values of outlook, temperature, humidity and wind, determine whether or not to play golf.

Decision Tree

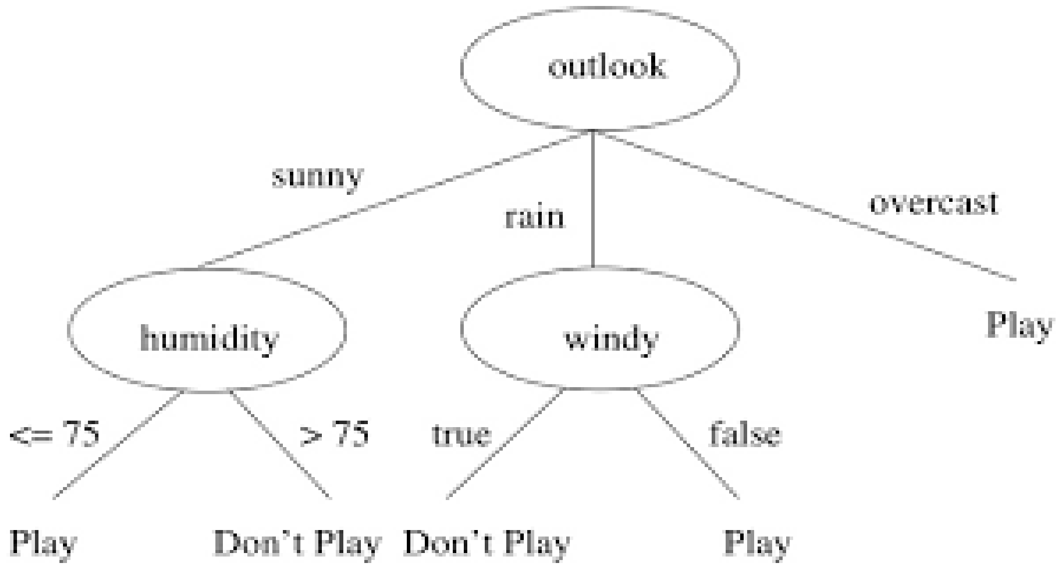


- Example (cont.):
  - We have five leaf nodes.
  - In a decision tree, each leaf node represents a rule.
  - We have the following rules corresponding to the tree given in Figure.
    - *RULE 1* *If it is sunny and the humidity is not above 75%, then play.*
    - *RULE 2* *If it is sunny and the humidity is above 75%, then do not play.*
    - *RULE 3* *If it is overcast, then play.*
    - *RULE 4* *If it is rainy and not windy, then play.*
    - *RULE 5* *If it is rainy and windy, then don't play.*

Decision Tree

- Example (cont.): Classification
  - The classification of an unknown input vector is done by traversing the tree from the root node to a leaf node.
  - A record enters the tree at the root node.
  - At the root, a test is applied to determine which child node the record will encounter next.
  - This process is repeated until the record arrives at a leaf node.
  - All the records that end up at a given leaf of the tree are classified in the same way.
  - There is a unique path from the root to each leaf.
  - The path is a rule which is used to classify the records.

Decision Tree



- Example (cont.):
  - In our tree, we can carry out the classification for o an unknown record as follows.
  - Let us assume, for the record, that we know the values of the first four attributes (but we do not know the value of class attribute) as
  - outlook= rain; temp = 70; humidity = 65; and windy= true.

Decision Tree

- Example (cont.):
  - We start from the root node to check the value of the attribute associated at the root node.
  - This attribute is the splitting attribute at this node.
  - For a decision tree, at every node there is an attribute associated with the node called the splitting attribute.
  - In our example, outlook is the splitting attribute at root.
  - Since for the given record, outlook = rain, we move to the rightmost child node of the root.
  - At this node, the splitting attribute is windy and we find that for the record we want classify, windy = true.
  - Hence, we move to the left child node to conclude that the class label Is "no play".

Decision Tree

- Example (cont.):
  - The accuracy of the classifier is determined by the percentage of the test data set that is correctly classified.
  - We can see that for Rule 1 there are two records of the test data set satisfying outlook= sunny and humidity < 75, and only one of these is correctly classified as play.
  - Thus, the accuracy of this rule is 0.5 (or 50%). Similarly, the accuracy of Rule 2 is also 0.5 (or 50%). The accuracy of Rule 3 is 0.66.

Decision Tree

- Concept of Categorical Attributes:
  - Consider the following training data set.
  - There are three attributes, namely, age, pincode and class.
  - The attribute class is used for class label.

ID	AGE	PINCODE	CLASS
1	30	5600046	C1
2	25	5600046	C1
3	21	5600023	C2
4	43	5600046	C1
5	18	5600023	C2
6	33	5600023	C1
7	29	5600023	C1
8	55	5600046	C2
9	48	5600046	C1

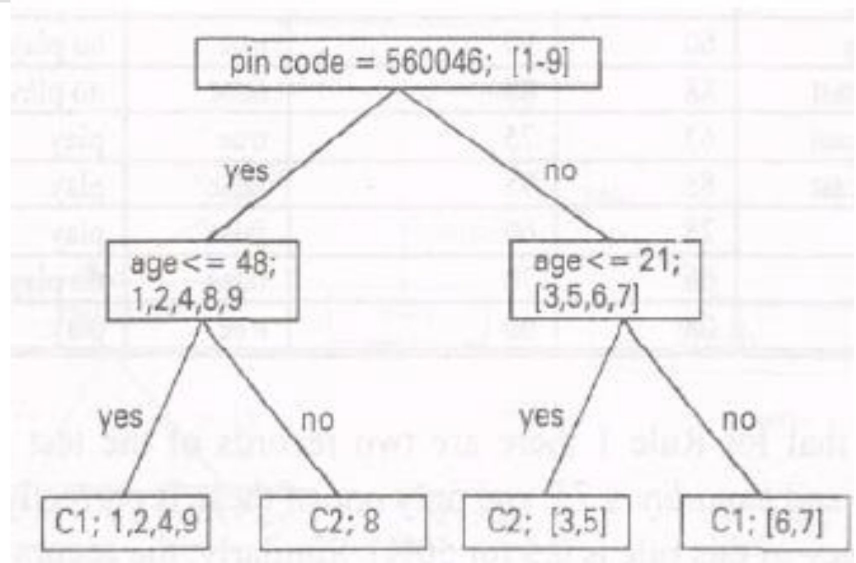
The attribute age is a numeric attribute, whereas pincode is a categorical one.

Though the domain of pincode is numeric, no ordering can be defined among pincode values.

You cannot derive any useful information if one pin-code is greater than another pincode.

Decision Tree

- Concept of Categorical Attributes (cont.):
  - Figure gives a decision tree for the training data.
  - The splitting attribute at the root is pincode and the splitting criterion here is pincode = 500 046.
  - Similarly, for the left child node, the splitting criterion is age < 48 (the splitting attribute is age).
  - Although the right child node has the same attribute as the splitting attribute, the splitting criterion is different.



At root level, we have 9 records.

The associated splitting criterion is *pincode* = 500 046.

As a result, we split the records into two subsets. Records 1, 2, 4, 8, and 9 are to the left child note and remaining to the right node.

The process is repeated at every node.

Decision Tree

- **Tree construction Principle**
  - Splitting Attribute
  - Splitting Criterion
- **3 main phases**
  - construction Phase
  - Pruning Phase
  - Processing the pruned tree to improve the understandability

Decision Tree

- **Decision Tree Construction Algorithms**
  - Hunt's Algorithm (one of the earliest)
  - CART(Classification And Regression Tree) → uses Gini Index(Classification) as metric.
  - ID3(Iterative Dichotomizer 3) → uses Entropy function and Information gain as metrics.
  - C4.5
  - SLIQ
  - SPRINT

Design Issues of Decision Tree Induction

- How should training records be split?
  - Method for specifying test condition
    - depending on attribute types
  - Measure for evaluating the goodness of a test condition

- How should the splitting procedure stop?
  - Stop splitting if all the records belong to the same class or have identical attribute values
  - Early termination

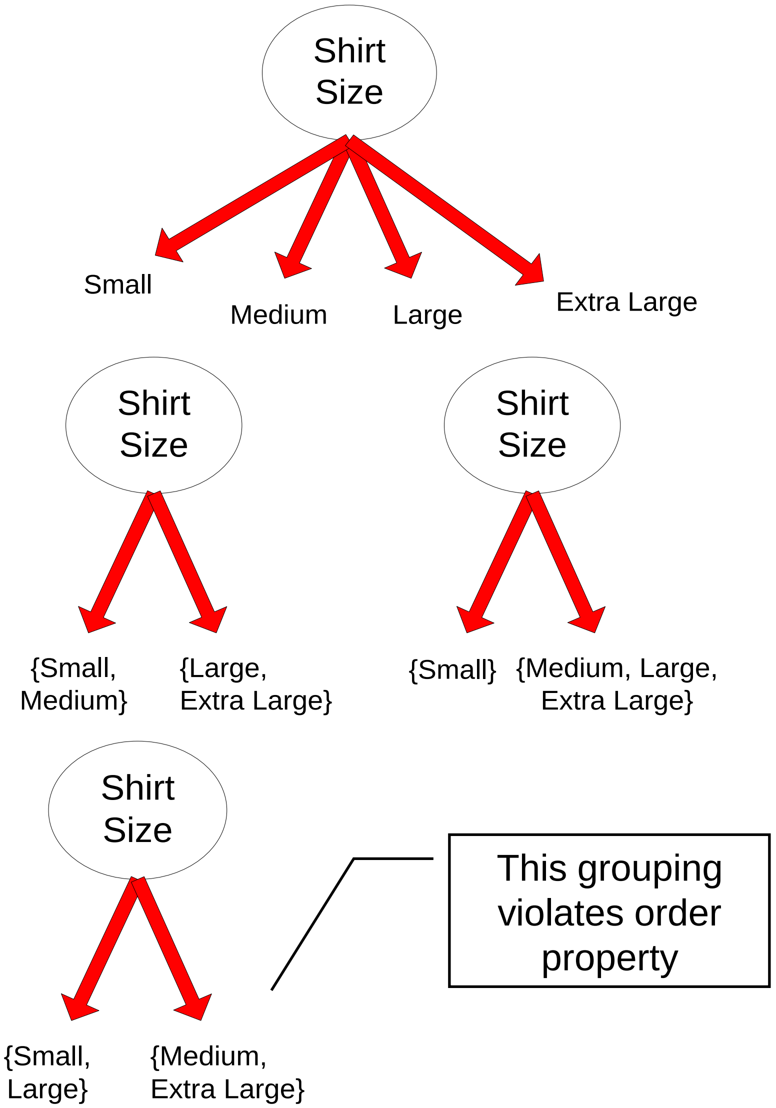
Methods for Expressing Test Conditions

- Depends on attribute types
  - Binary
  - Nominal
  - Ordinal
  - Continuous

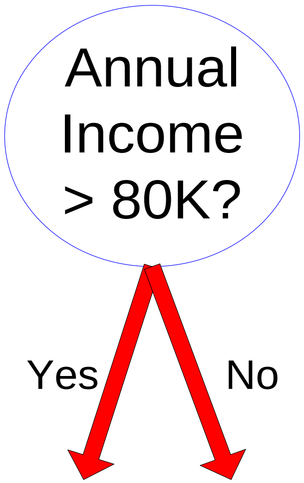
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Test Condition for Ordinal Attributes

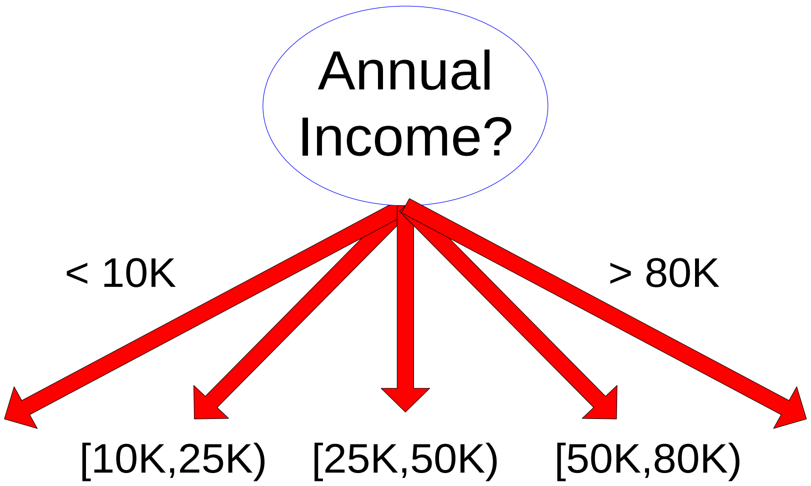
- Multi-way split:
  - Use as many partitions as distinct values
- Binary split:
  - Divides values into two subsets
  - Preserve order property among attribute values



Test Condition for Continuous Attributes



(i) Binary split

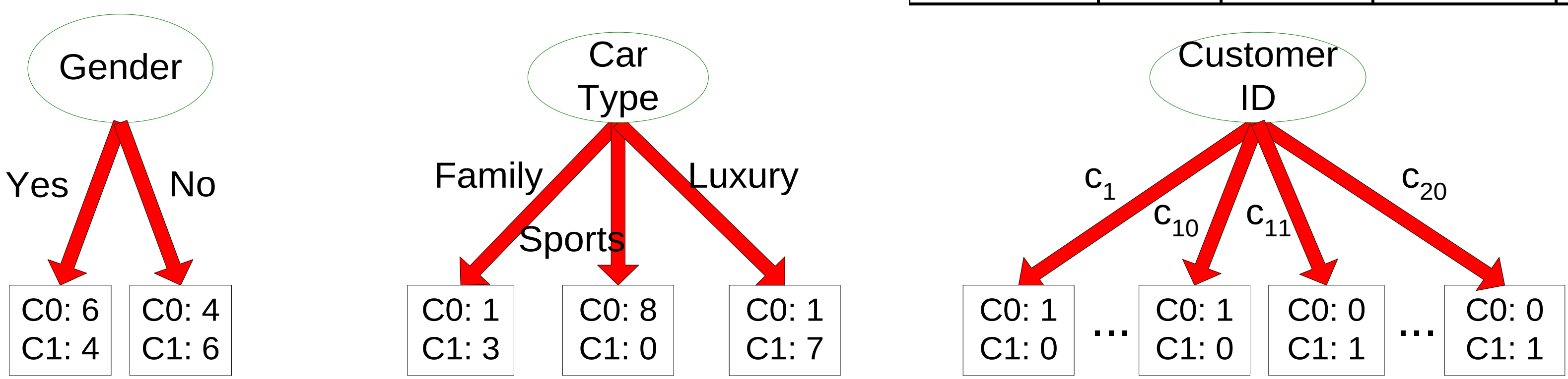


(ii) Multi-way split

How to determine the Best Split

Customer Id	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

Before Splitting: 10 records of class 0,
10 records of class 1



Which test condition is the best?

How to determine the Best Split

- Greedy approach:
  - Nodes with purer class distribution are preferred

- Need a measure of node impurity:

C0: 5
C1: 5

High degree of impurity

C0: 9
C1: 1

Low degree of impurity

How to determine the Best Split

- Gini Index

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

- Entropy

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

- Misclassification error

$$Error(t) = 1 - \max_i P(i \mid t)$$

Finding the Best Split

- Compute impurity measure (P) before splitting
- Compute impurity measure (M) after splitting
  - Compute impurity measure of each child node
  - M is the weighted impurity of children
- Choose the attribute test condition that produces the highest gain

$$\text{Gain} = P - M$$

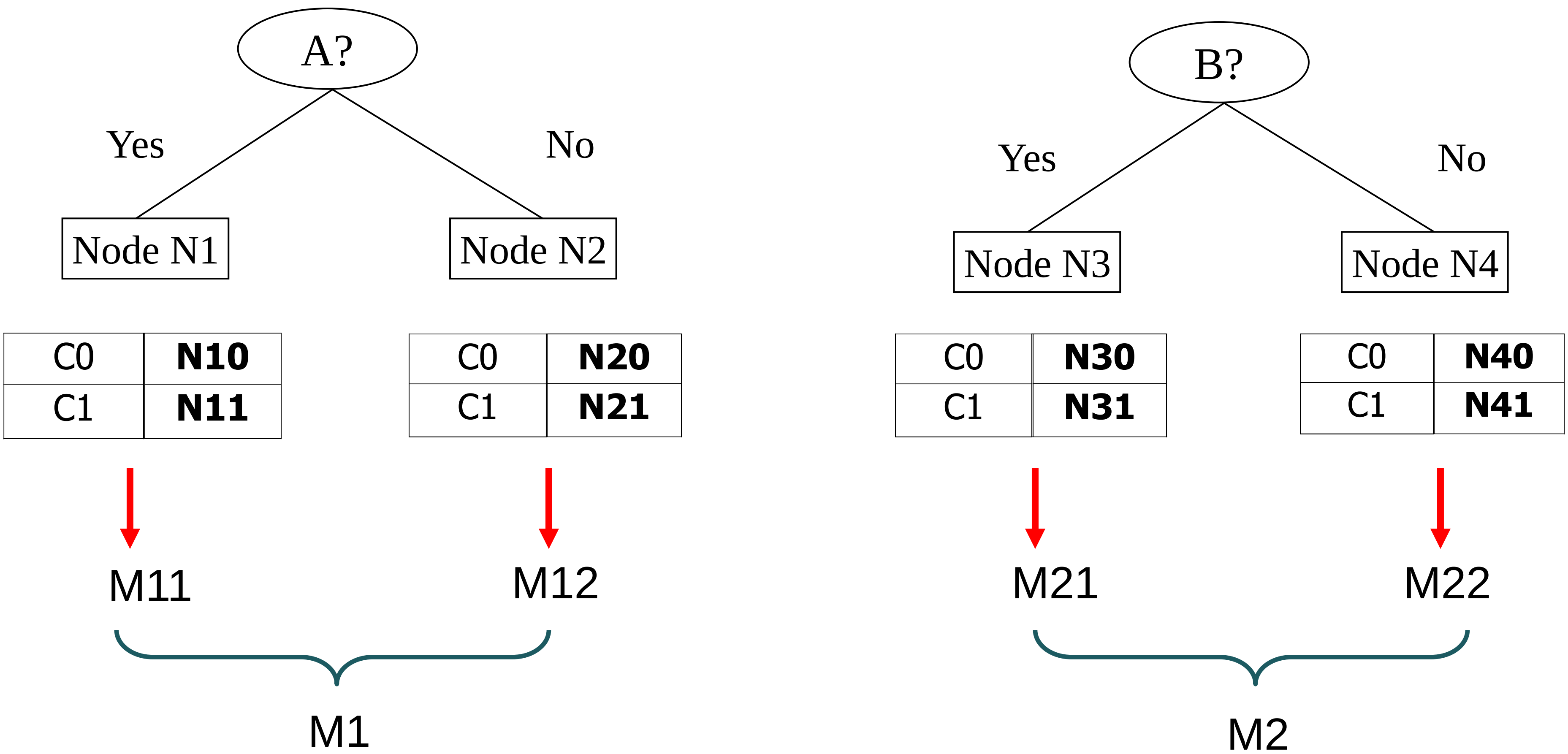
or equivalently, lowest impurity measure after splitting (M)

Finding the Best Split

Before Splitting:

C0	**N00**
C1	**N01**

→ P



Gain = P – M1 vs P – M2

4 steps of Measure of Impurity: GINI

1. If a data set *D* contains examples from *n* classes, gini index, gini(D) is defined as:

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

where p_j = count of specific class level / total count of D

2. 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

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

3. Reduction in Impurity: $\Delta gini(A) = gini(D) - gini_A(D)$

4. Select best attribute whose impurity is less will be selected.

GINI Index

- It uses a binary split of each attribute.
- Finds all possible subsets using all possible values
  - If attribute A have v possible values then there are 2^v *possible subsets*.
  - Attribute: Income
    - Values: {low, medium, high}
  - Subset: 2^3 = 8 = {low, medium, high}, {low, medium}, {low, high}, {medium, high}, {low}, {medium}, {high}, {}

age	income	student	credit_rating	buys_computer
young	high	no	fair	no
young	high	no	excellent	no
middle	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle	low	yes	excellent	yes
young	medium	no	fair	no
young	low	yes	fair	yes
senior	medium	yes	fair	yes
young	medium	yes	excellent	yes
middle	medium	no	excellent	yes
middle	high	yes	fair	yes
senior	medium	no	excellent	no

- Gini (D) = 1 - $\left(\frac{9}{14}\right)^2$ - $\left(\frac{5}{14}\right)^2$

GINI Index

- If a binary split on income, partition D into D_1 and D_2. The gini index of it is calculated by

$$Gini_{income\,\epsilon\,\{low,medium\}}(D)$$

$$= \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

$$= \frac{10}{14} Gini(D_1) + \frac{4}{14} Gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right)$$

10 low and medium income

4 for high income

6 for low and medium income for yes class

4 for low and medium income for no class

2 for high income for yes class

2 for high income for no class

age	income	student	credit_rating	buys_computer
young	high	no	fair	no
young	high	no	excellent	no
middle	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle	low	yes	excellent	yes
young	medium	no	fair	no
young	low	yes	fair	yes
senior	medium	yes	fair	yes
young	medium	yes	excellent	yes
middle	medium	no	excellent	yes
middle	high	yes	fair	yes
senior	medium	no	excellent	no

GINI Index: Example

- Step 1: compute the impurity of D
- Total tuples are 14
- 9 tuples belonging to class buys_computer = yes
- 5 tuples belonging to class buys_computer = no
- Using formula 1:

Gini (D) = $1 - (\frac{9}{14})^2 - (\frac{5}{14})^2 = 0.459$

It is the impurity of total dataset.

- We need to compute gini index of each attribute (age, income, credit_rating, student)

age	income	student	credit_rating	buys_computer
young	high	no	fair	no
young	high	no	excellent	no
middle	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle	low	yes	excellent	yes
young	medium	no	fair	no
young	low	yes	fair	yes
senior	medium	yes	fair	yes
young	medium	yes	excellent	yes
middle	medium	no	excellent	yes
middle	high	yes	fair	yes
senior	medium	no	excellent	no

GINI Index: Example

- Lets take income first
  - Possible splitting subsets: {low, medium}, {low, high}, {medium, high}, {low}, {high}, {medium}.
- Lets take {Low, medium} first
- Total tuples where income ϵ {low, medium} = 10 (D_1)
- Rest left = 4 (D_2)
- Now compute

$$Gini_{income\ \epsilon\,\{low,medium\}}(D)$$

$$= \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right)$$

= 0.450 [it will be same for high]

age	income	student	credit_rating	buys_computer
young	high	no	fair	no
young	high	no	excellent	no
middle	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle	low	yes	excellent	yes
young	medium	no	fair	no
young	low	yes	fair	yes
senior	medium	yes	fair	yes
young	medium	yes	excellent	yes
middle	medium	no	excellent	yes
middle	high	yes	fair	yes
senior	medium	no	excellent	no

GINI Index: Example

	Tuples in D1		Tuples in D2		Gini index
{low, medium} or {High}	Tuples in D1 {low,medium}		Tuples in D2 (high)		.450
	10		4		
	Buys_computer (yes)	Buys_computer (no)	Buys_computer (yes)	Buys_computer (no)	
	6	4	2	2	

GINI Index: Example

	Tuples in D1		Tuples in D2		Gini index
{low,high} or {medium}	Tuples in D1 {low,high}		Tuples in D2 (medium)		.315
	8		6		
	Buys_computer (yes)	Buys_computer (no)	Buys_computer (yes)	Buys_computer (no)	
	6	5	4	2	

GINI Index: Example

<table>
<tr><th></th><th colspan="2">Tuples in D1</th><th colspan="2">Tuples in D2</th><th>Gini index</th></tr>
<tr><td rowspan="5">{medium,high}

or

{low}</td><td colspan="2">Tuples in D1 {medium,high}</td><td colspan="2">Tuples in D2 (low)</td><td rowspan="5">.300</td></tr>
<tr><td colspan="2">10</td><td colspan="2">4</td></tr>
<tr><td>Buys_computer (yes)</td><td>Buys_computer (no)</td><td>Buys_computer (yes)</td><td>Buys_computer (no)</td></tr>
<tr><td>6</td><td>4</td><td>3</td><td>1</td></tr>
</table>

GINI Index: Example

- Best binary split for income is {medium, high} or {low} with minimum gini index.
- Now do the same for attribute age, student and credit_rating

Attribute	Split	Gini index	Reduction in impurity $\Delta G = gini(D) - gini_A(D)$
income	{medium,high} or {low}	.300	.459-.300 = .159
age	{youth_senior} or {middle aged}	.375	.459-.375 =.084
Student	Binary	.367	.459-.367 =.092
Credit_rating	binary	.429	.459-.429 =.03

- Income is selected with minimum gini index and highest reduction in impurity.

Measure of Impurity: Entropy

- Entropy at a given node t:

$$Entropy(t) = -\sum_j p(j\,|\,t)\log p(j\,|\,t)$$

(NOTE: $p(j\,|\,t)$ is the relative frequency of class j at node t).

  - Maximum ($\log n_c$) when records are equally distributed among all classes implying least information
  - Minimum (0.0) when all records belong to one class, implying most information

- Entropy based computations are quite similar to the GINI index computations

Computing Entropy of a Single Node

$$Entropy(t) = -\sum_j p(j\,|\,t)\log_2 p(j\,|\,t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0   P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6   P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

C1	**2**
C2	**4**

P(C1) = 2/6   P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Computing Information Gain After Splitting

- Information Gain:

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

Parent Node, p is split into k partitions;

n_i is number of records in partition i

- Choose the split that achieves most reduction (maximizes GAIN)

- Used in the ID3 and C4.5 decision tree algorithms

Computing Information Gain After Splitting

- Example:

Age	Competition	Type	Profit
old	yes	S/w	Down
old	No	S/w	Down
old	No	H/w	Down
mid	yes	S/w	Down
mid	yes	H/w	Down
mid	No	H/w	Up
mid	No	S/w	Up
new	yes	S/w	Up
new	No	H/w	Up
new	No	S/w	Up

Produces a decision tree from this table.

Computing Information Gain After Splitting

- Example:
  - First find out the target attribute. Here Profit is the target attribute.
  - Need to find out the root node.
  - For this calculate information gain (IG) of target attribute.

IG = Entropy = $\frac{-P}{P+N} log_2(\frac{P}{P+N})$ - $\frac{N}{P+N} log_2(\frac{N}{P+N})$

Age	Competition	Type	Profit
old	yes	S/w	Down
old	No	S/w	Down
old	No	H/w	Down
mid	yes	S/w	Down
mid	yes	H/w	Down
mid	No	H/w	Up
mid	No	S/w	Up
new	yes	S/w	Up
new	No	H/w	Up
new	No	S/w	Up

  - Then calculate entropy of rest all attributes: age, competition and type.
  - At last calculate the Gain and Gain with maximum value construct the root node.
  - Other nodes are constructed with second largest, 3rd largest value and so on.

Computing Information Gain After Splitting

- Example:

IG = Entropy(P) = $\frac{-P}{P+N}log_2(\frac{P}{P+N})$ - $\frac{N}{P+N}log_2(\frac{N}{P+N})$

= -[$\frac{5}{10}log_2(\frac{5}{10})$ +$\frac{5}{10}log_2(\frac{5}{10})$] = 1

- Age

	Down	Up
old	3	0
mid	2	2
new	0	3

I(old) = -[$\frac{3}{3}log_2(\frac{3}{3})$ +$\frac{0}{3}log_2(\frac{0}{3})$] = 0*$\frac{3}{10} = 0$

I(mid) = -[$\frac{2}{4}log_2(\frac{2}{4})$ +$\frac{2}{4}log_2(\frac{2}{4})$] = 1*$\frac{4}{10} = 0.4$

I(new) = -[$\frac{0}{3}log_2(\frac{0}{3})$ +$\frac{3}{3}log_2(\frac{3}{3})$] = 0*$\frac{3}{10} = 0$

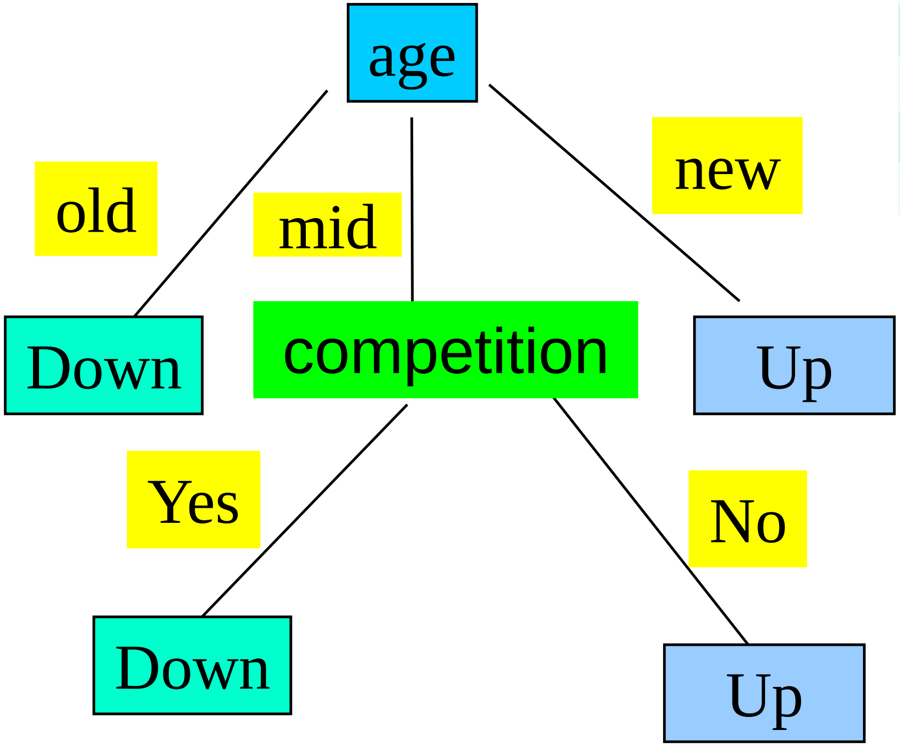
E(Age) = I(old) + I(mid) + I(new) = 0.4

Age	Competition	Type	Profit
old	yes	S/w	Down
old	No	S/w	Down
old	No	H/w	Down
mid	yes	S/w	Down
mid	yes	H/w	Down
mid	No	H/w	Up
mid	No	S/w	Up
new	yes	S/w	Up
new	No	H/w	Up
new	No	S/w	Up

Computing Information Gain After Splitting

- Example:
  - Gain(Age) = IG - E(Age) = 1 – 0.4 = 0.6
  - Gain(competition) = 0.124
  - Gain(type) = 0

Age	Competition	Type	Profit
old	yes	S/w	Down
old	No	S/w	Down
old	No	H/w	Down
mid	yes	S/w	Down
mid	yes	H/w	Down
mid	No	H/w	Up
mid	No	S/w	Up
new	yes	S/w	Up
new	No	H/w	Up
new	No	S/w	Up



Iterative Dichotomizer (ID3)

- Quinlan (1986)
- Each node corresponds to a splitting attribute
- Each arc is a possible value of that attribute.
- At each node the splitting attribute is selected to be the most informative among the attributes not yet considered in the path from the root.
- Entropy is used to measure how informative is a node.
- The algorithm uses the criterion of information gain to determine the goodness of a split.
  - The attribute with the greatest information gain is taken as the splitting attribute, and the data set is split for all distinct values of the attribute.

Iterative Dichotomizer (ID3) – Example

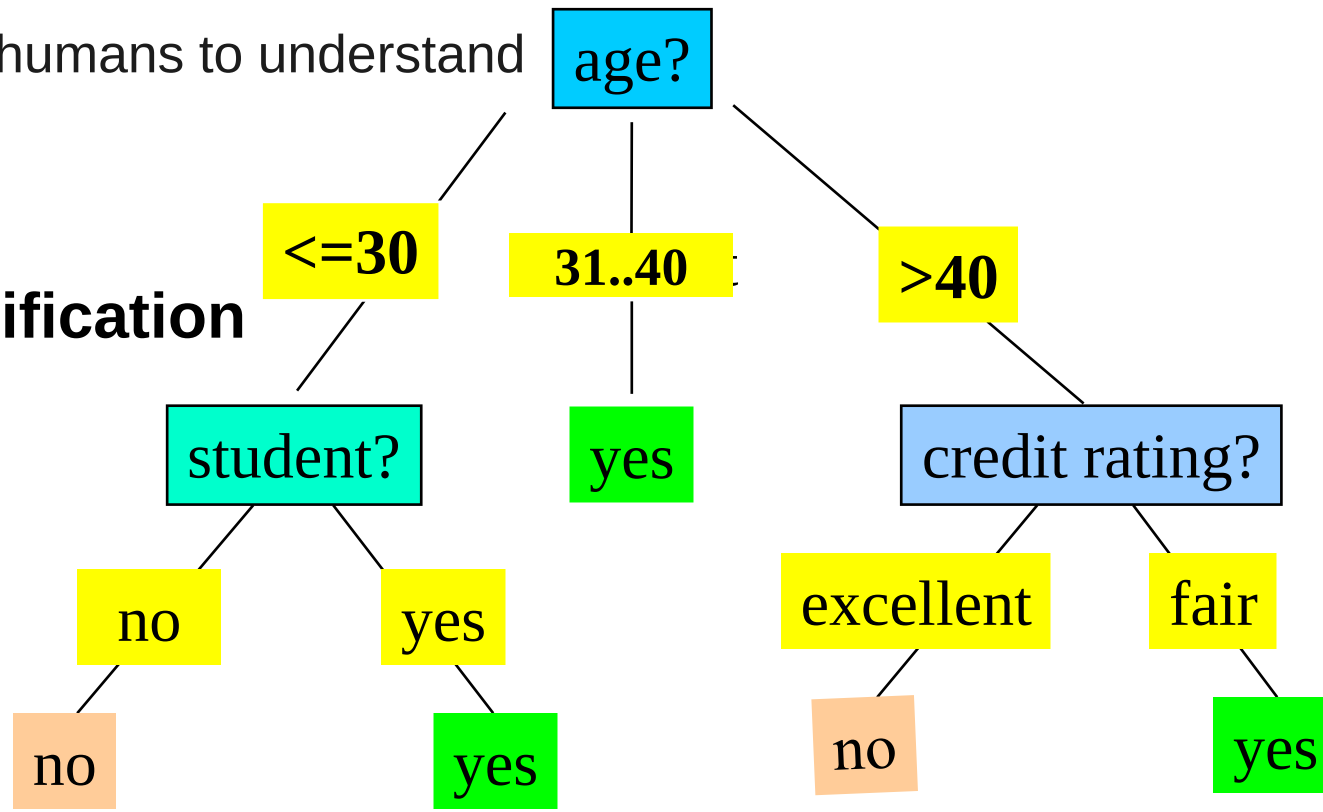
- The class label attribute, *buys_computer*, has two distinct values.
- Thus there are two distinct classes. (m =2)
- Class C1 corresponds to *yes and class C2 corresponds to no.*
- There are 9 samples of class yes and 5 samples of class no.

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

Iterative Dichotomizer (ID3) – Example

- Represent the knowledge in the form of IF-THEN rules
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction
- The leaf node holds the class prediction
- Rules are easier for humans to understand

Extracting Classification Rules from Trees



Iterative Dichotomizer (ID3) – Example

- **Solution (Rules):**

IF *age* = “<=30” AND *student* = “*no*” THEN *buys_computer* = “*no*”

IF *age* = “<=30” AND *student* = “*yes*” THEN *buys_computer* = “*yes*”

IF *age* = “31…40” THEN *buys_computer* = “*yes*”

IF *age* = “>40” AND *credit_rating* = “*excellent*” THEN *buys_computer* = “*yes*”

IF *age* = “<=30” AND *credit_rating* = “*fair*” THEN *buys_computer* = “*no*”

Iterative Dichotomizer (ID3) – Algorithm

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)

Iterative Dichotomizer (ID3) – Algorithm

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

Advantages of Decision Tree

- A decision tree construction process is concerned with identifying the splitting attributes and splitting criterion at every level of the tree.
- Major strengths are:
  - Decision tree able to generate understandable rules.
  - They are able to handle both numerical and categorical attributes.
  - They provide clear indication of which fields are most important for prediction or classification.

Shortcomings of Decision Tree

- Weaknesses are:
  - The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field is examined before its best split can be found.
  - Some decision tree can only deal with binary-valued target classes.

Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: *Halt tree construction early*-do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: *Remove branches* from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"

Bayesian Classification

- **A statistical classifier:** performs probabilistic prediction, i.e., predicts class membership probabilities
- **Foundation:** Based on Bayes' Theorem.
- **Performance:** A simple Bayesian classifier, naïve Bayesian classifier, has comparable performance with decision tree and selected neural network classifiers
- **Incremental:** Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- **Standard:** Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Bayesian Theorem: Basics

- Let **X** be a data sample class label is unknown
- Let H be a *hypothesis* that X belongs to class C
- Classification is to determine P(H|**X**), the probability that the hypothesis holds given the observed data sample **X**
  - Posterior Probability
- P(H) (*prior probability*), the initial probability
- P(**X**): probability that sample data is observed
- P(**X**|H) (*posteriori probability*), the probability of observing the sample **X**, given that the hypothesis holds
- X – Round and Red Fruit H - Apple

Bayesian Theorem

- Given training data **X**, *posteriori probability of a hypothesis* H, P(H|**X**), follows the Bayes theorem

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

- Informally, this can be viewed as

  posteriori = likelihood x prior/evidence

- Predicts **X** belongs to C_i if and only if the probability P(C_i|**X**) is the highest among all the P(C_K|X) for all the *k* classes
- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

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 $\mathbf{X} = (x_1, x_2, \ldots, x_n)$
- 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

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

needs to be maximized

Bayesian Classification

- **Naïve Bayesian Classifier**
  - Class Conditional Independence
  - Effect of an attribute value on a given class is independent of the values of other attributes
  - Simplifies Computations
- **Bayesian Belief Networks**
  - Graphical models
  - Represent dependencies among subsets of attributes

Naïve Bayesian Classifier

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector $\mathbf{X} = (x_1, x_2, \ldots, x_n)$
- 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

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

Naïve Bayesian Classifier

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

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

  needs to be maximized
- Can assume that all classes are equally likely and maximize $P(X|C_i)$
- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

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

Naïve Bayes Classifier: Training Dataset

Class:

C1:buys_computer = 'yes'

C2:buys_computer = 'no'

Data to be classified:

X = (age <=30,

Income = medium,

Student = yes

Credit_rating = Fair)

age	income	student	credit_rating	_comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayes Classifier: An Example

age	income	student	credit_rating	_comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- $P(C_i)$:

  P(buys_computer = “yes”) = 9/14 = 0.643

  P(buys_computer = “no”) = 5/14= 0.357

- Compute $P(X|C_i)$ for each class

  P(age = “<=30” | buys_computer = “yes”) = 2/9 = 0.222

  P(age = “<= 30” | buys_computer = “no”) = 3/5 = 0.6

  P(income = “medium” | buys_computer = “yes”) = 4/9 = 0.444

  P(income = “medium” | buys_computer = “no”) = 2/5 = 0.4

  P(student = “yes” | buys_computer = “yes) = 6/9 = 0.667

  P(student = “yes” | buys_computer = “no”) = 1/5 = 0.2

  P(credit_rating = “fair” | buys_computer = “yes”) = 6/9 = 0.667

  P(credit_rating = “fair” | buys_computer = “no”) = 2/5 = 0.4

Naïve Bayes Classifier: An Example

age	income	student	credit_rating	_comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

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

P(X|C_i) : P(X|buys_computer = “yes”) = 0.222 x 0.444 x 0.667 x 0.667 = 0.044

P(X|buys_computer = “no”) = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

P(X|C_i)*P(C_i) : P(X|buys_computer = “yes”) * P(buys_computer = “yes”) = 0.028

P(X|buys_computer = “no”) * P(buys_computer = “no”) = 0.007

Therefore, X belongs to class (“buys_computer = yes”)

Avoiding the Zero-Probability Problem

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

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10)
- Use **Laplacian correction** (or Laplacian estimator)
  - Adding 1 to each case

    Prob(income = low) = 1/1003

    Prob(income = medium) = 991/1003

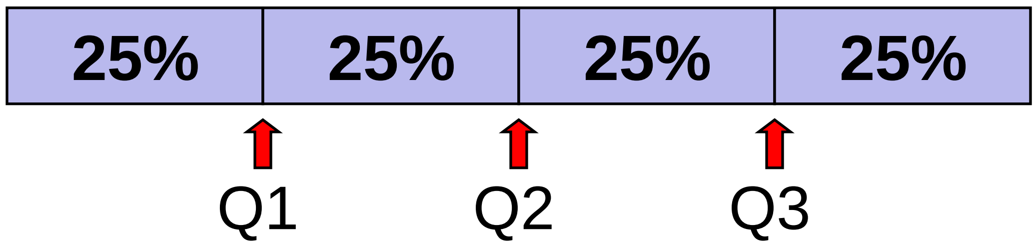
    Prob(income = high) = 11/1003
  - The "corrected" prob. estimates are close to their "uncorrected" counterparts

Naïve Bayesian Classifier

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc. Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayesian Classifier

Classification by Back propagation

- Quartiles split the ranked data into 4 segments with an equal number of values per segment



- The first quartile, Q1, is the value for which 25% of the observations are smaller and 75% are larger
- Q2 is the same as the median (50% of the observations are smaller and 50% are larger)
- Only 25% of the observations are greater than the third quartile

Support Vector Machines

- Solution provided SVM is
  - Theoretically elegant
  - Computationally Efficient
  - Error rate is 1.1%
  - Very effective in many Large practical problems
- It has a simple geometrical interpretation in a high-dimensional feature space that is nonlinearly related to input space
- By using kernels all computations keep simple

Types of Data in Classification

- Linearly Separable Data
- Linearly Non-separable Data

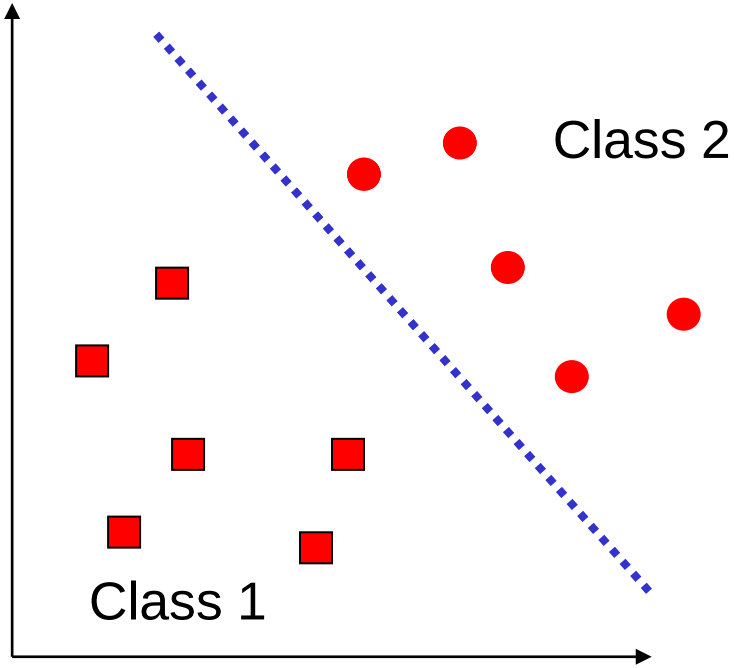


Figure 1: Linearly Separable Data

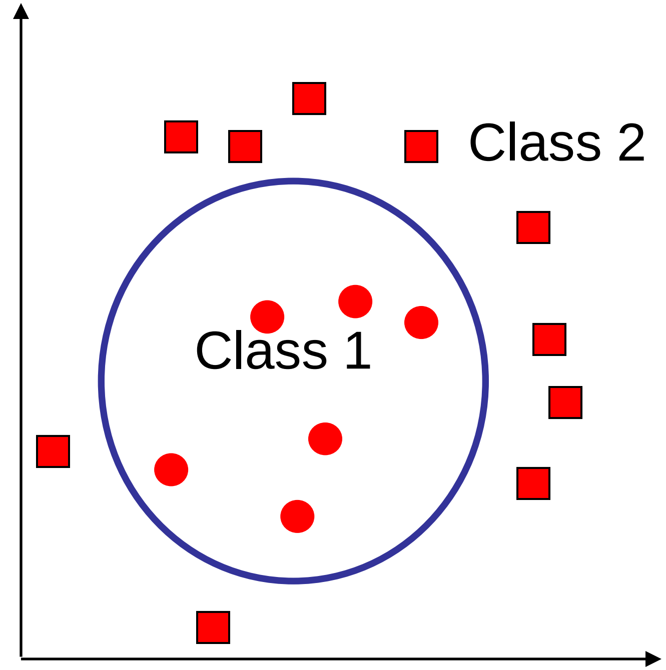


Figure 2: Linearly Non-separable Data

Types of Data in Classification

- Linear Classifier
- Non-linear Classifier

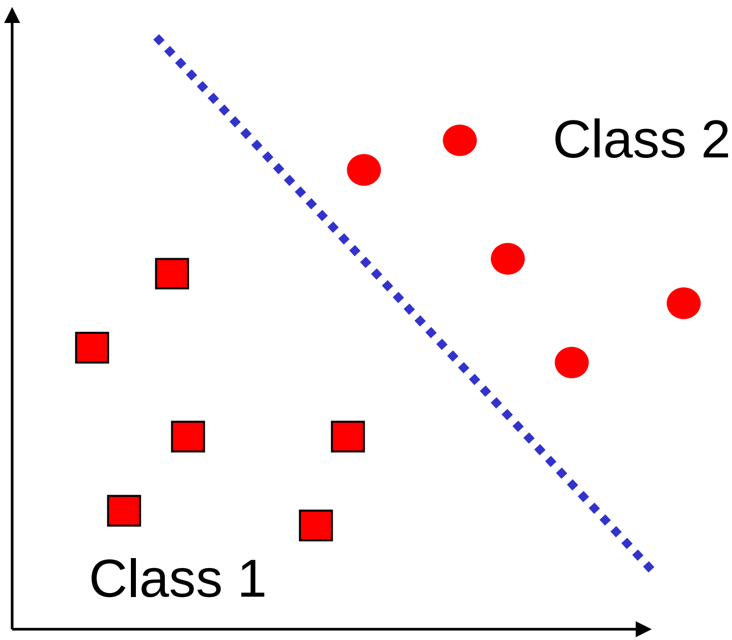


Figure 3: Linear Classifier

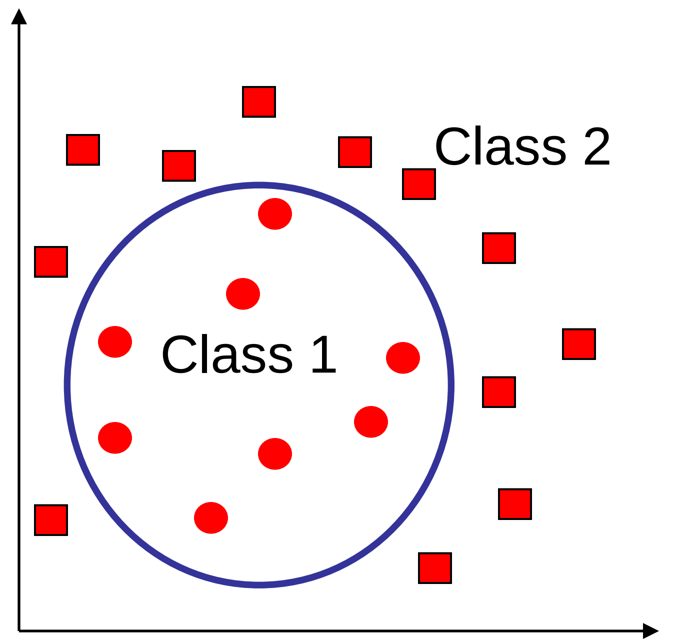
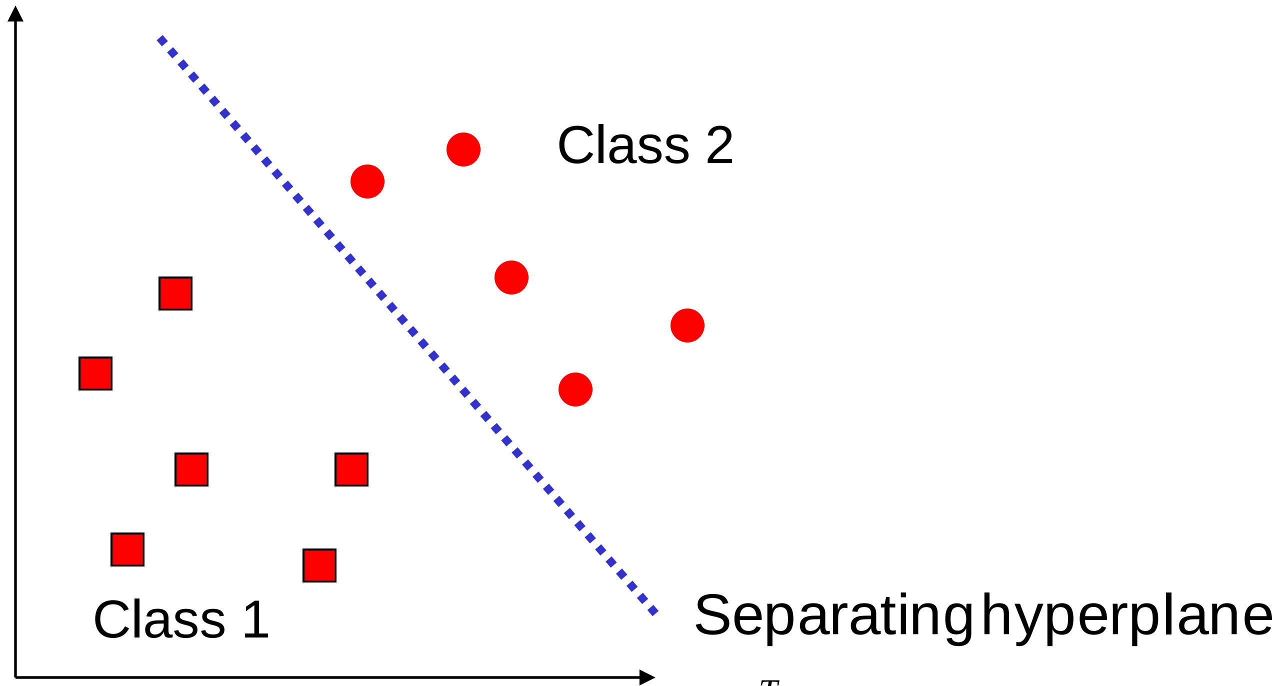


Figure 4: Non-Linear Classifier

Basic Concept of Classification Function

- Consider linear separable case
- Training data two classes



$$\mathrm{w}^T x + b = 0$$

Basic Concept of Classification Function

Equation of a hyperplane:

$$\mathrm{w}^T x + b = 0$$

$$2-D: w_1 x_1 + w_2 x_2 + b = 0$$

$$example:$$

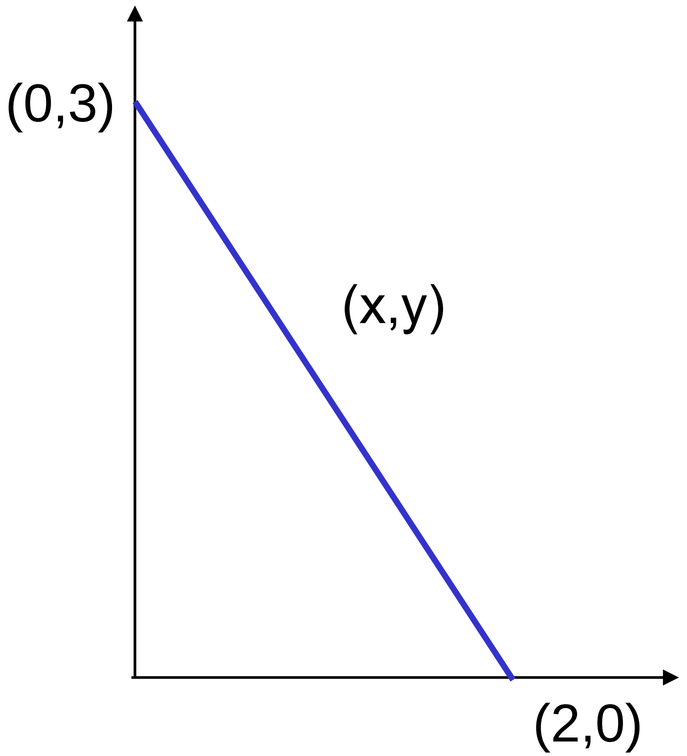
$$\frac{y-0}{x-2} = \frac{3-0}{0-2} = \frac{-3}{2}$$

$$2y = -3x + 6$$

$$\Rightarrow 3x + 2y - 6 = 0$$

$$\mathrm{w}: \text{coefficient}$$

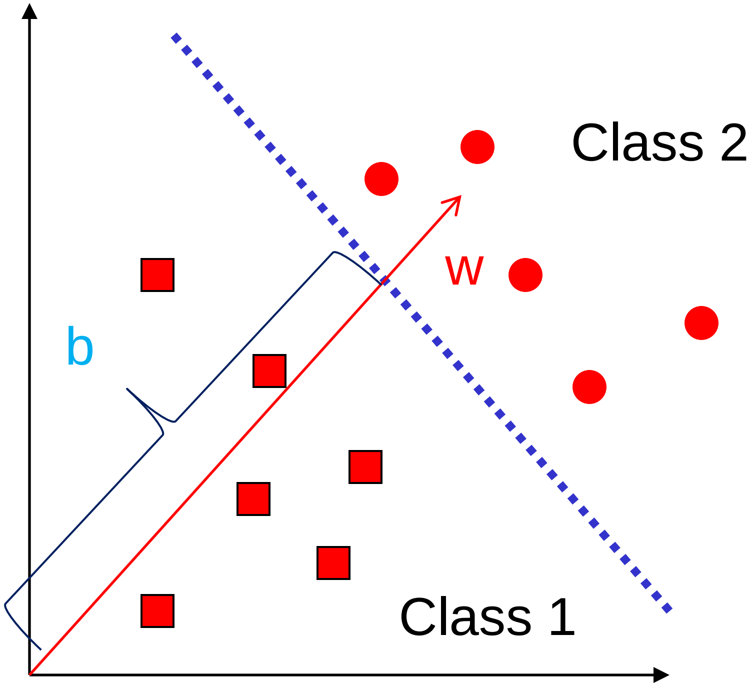
$$\mathrm{b}: \text{constant}$$



Decision Function

$$f(x) \equiv w^T x + b$$

- f(x) > 0 → class 1
- f(x) < 0 → class 2
- How to find good w and b?
- There are many possible (w, b)
- We are looking for (w, b) that will:
  - Classify correctly the classes
  - Give maximum Margins



Separating hyperplane

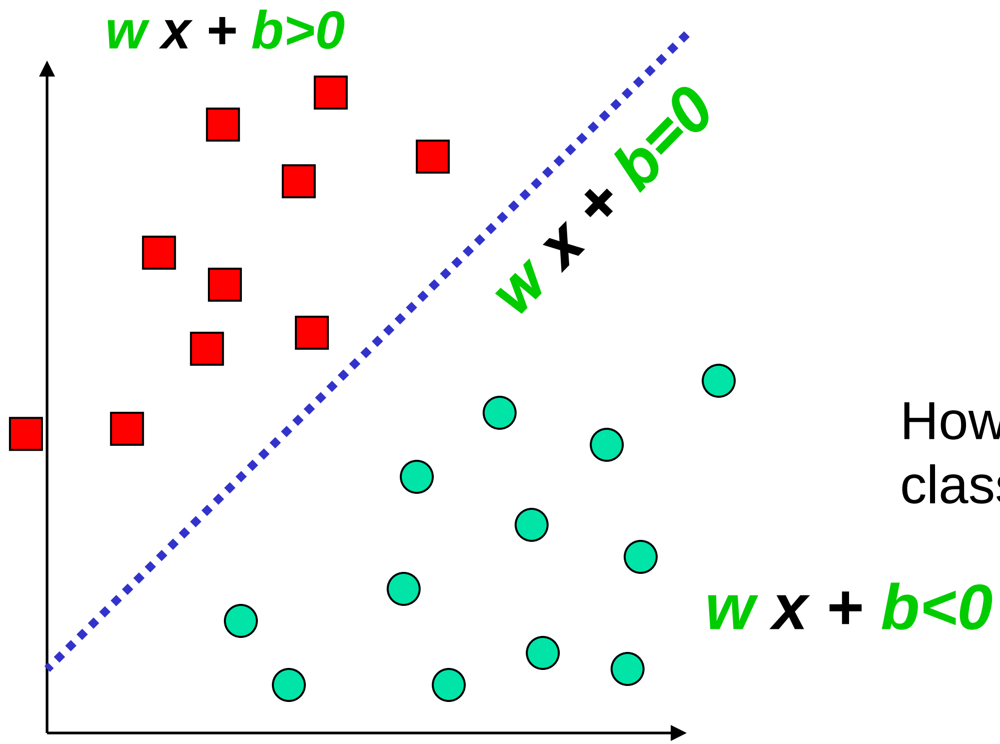
$$w^T x + b = 0$$

Linear Classifiers

$f(\mathbf{x}, w, b) = sign(w\,\mathbf{x} + b)$

■ denotes +1

● denotes -1

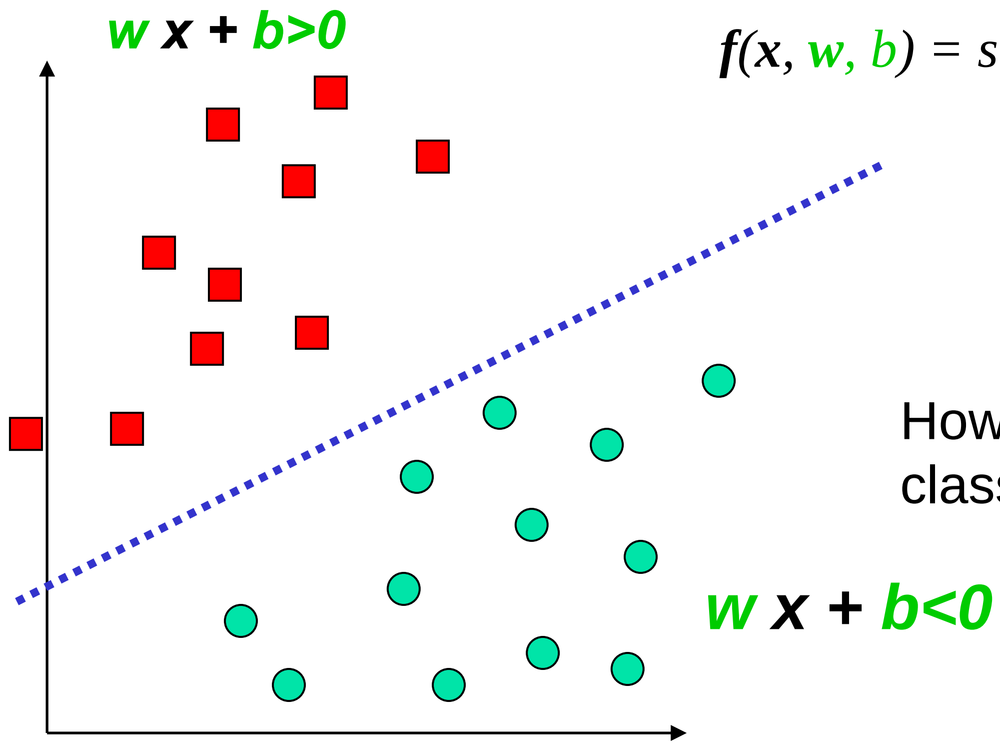


How would you classify this data?

Linear Classifiers Cont.

■ denotes +1

● denotes -1



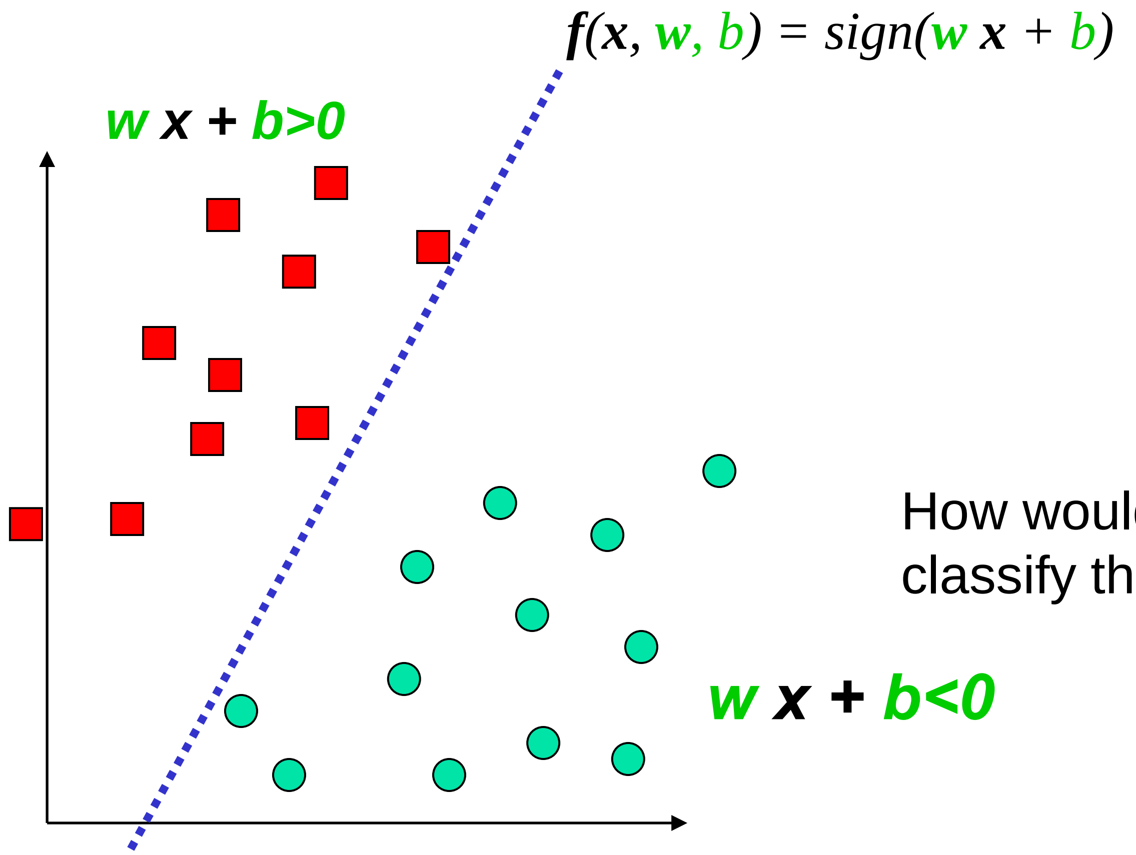
$f(\mathbf{x}, w, b) = sign(w\,\mathbf{x} + b)$

How would you classify this data?

Linear Classifiers Cont.

■ denotes +1

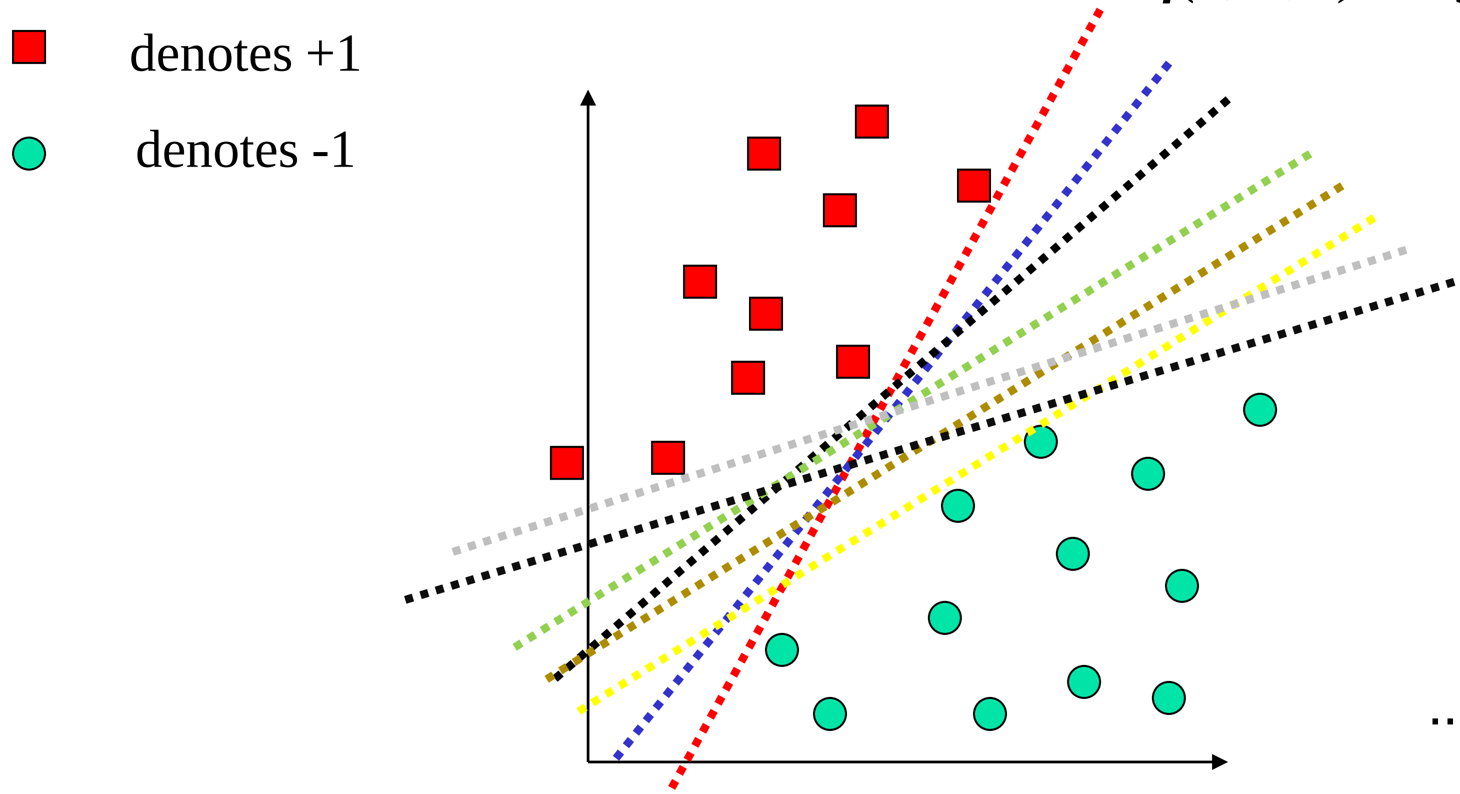
● denotes -1



How would you classify this data?

Linear Classifiers Cont.

$f(\boldsymbol{x}, \mathbf{w}, b) = sign(\mathbf{w}\ \boldsymbol{x} + b)$



Any of these would be fine..

..but which is best?

Support Vector Machines

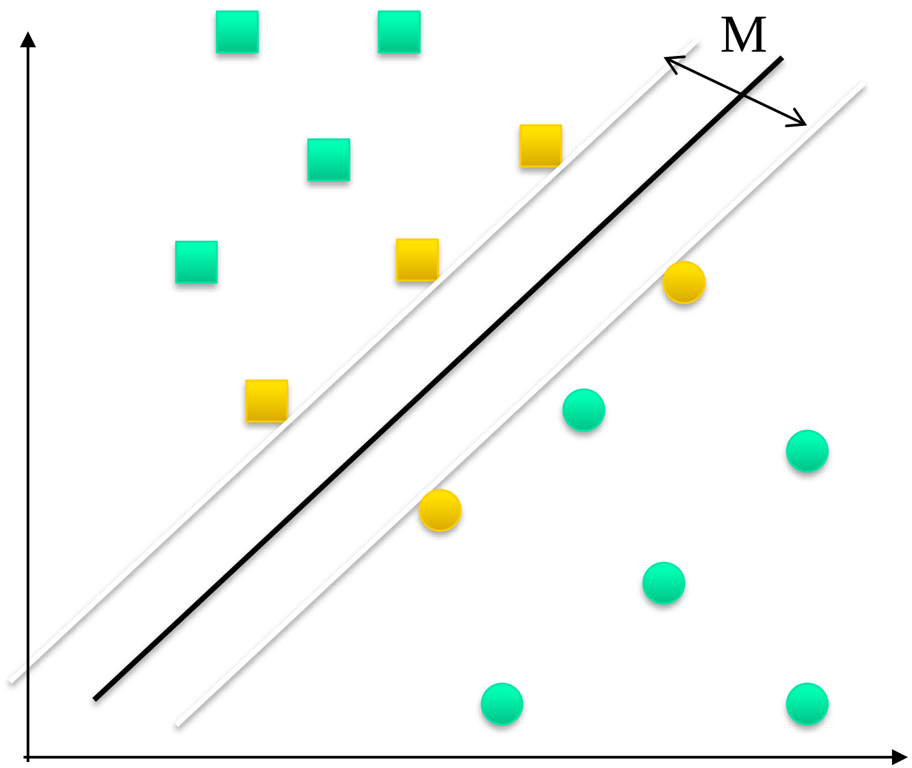
- A promising technique for data classification
- Statistic learning theorem: maximize the distance between two classes
- A new classification method for both linear and nonlinear data
- It uses a nonlinear mapping to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating hyperplane (i.e., “decision boundary”)
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane
- SVM finds this hyperplane using **support vectors** (“essential” training tuples) and **margins** (defined by the support vectors)

Support Vectors

- Support Vectors are those input points (vectors) closest to the decision boundary

  1. They are vectors

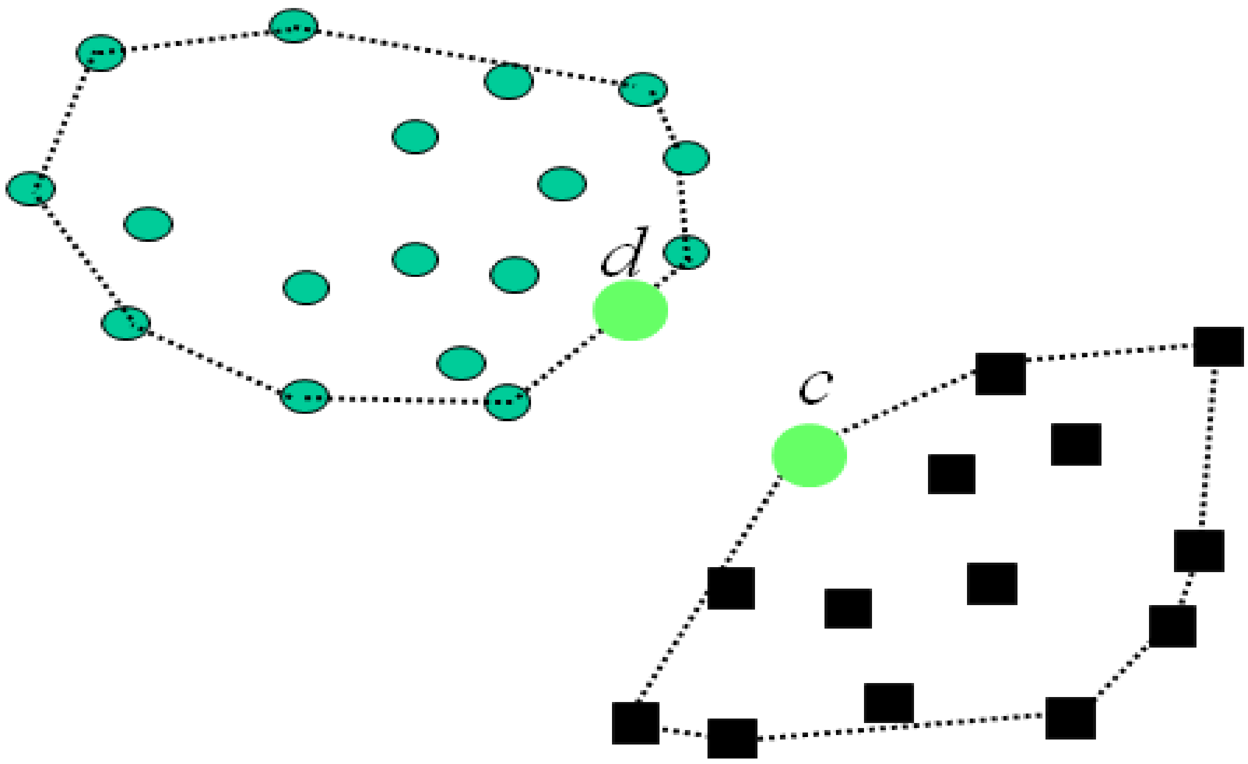
  2. They “support” the decision hyperplane



- Margin M of the separator is the width of separation between classes.

Best Linear Separator

- Find Closest Points in Convex Hulls

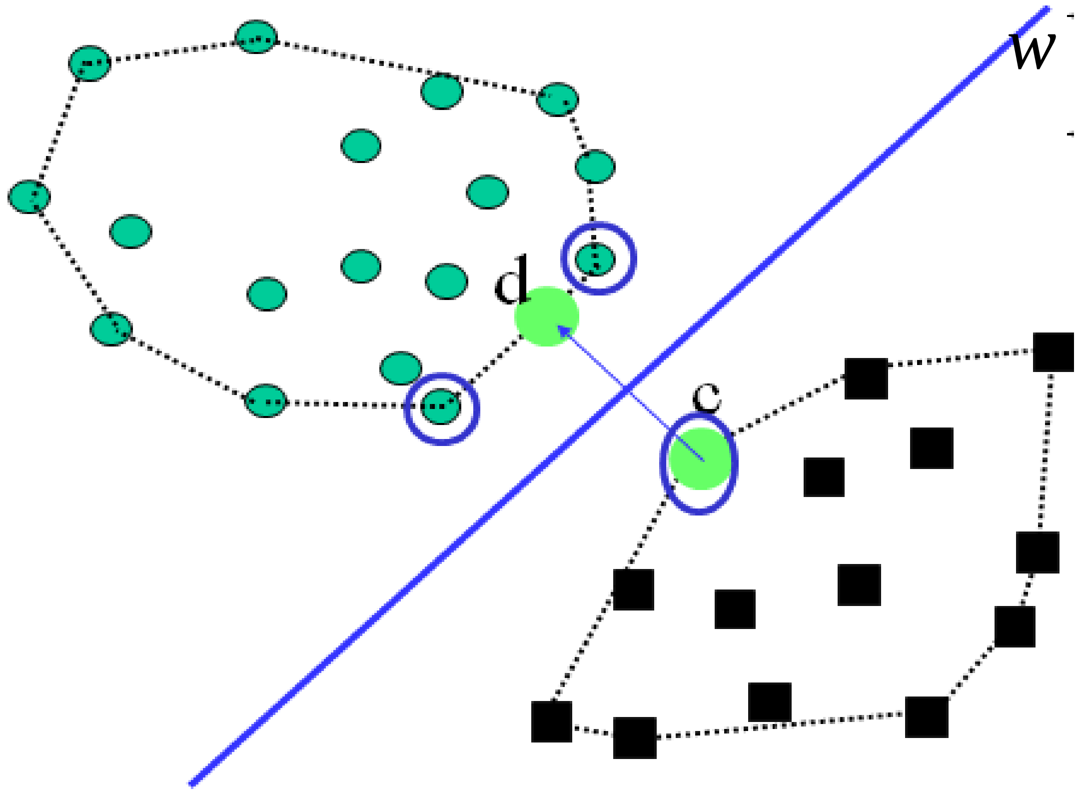


Best Linear Separator

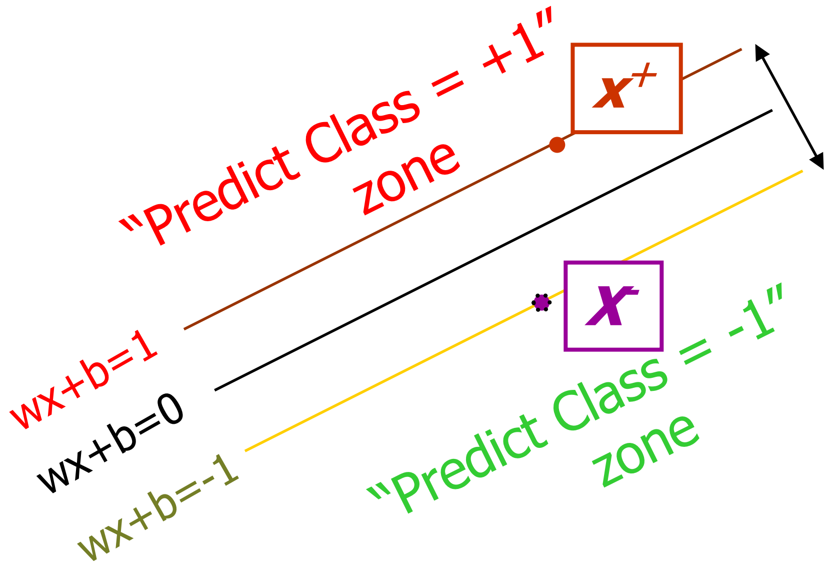
- Plane Bisect Closest Points

$w^T x + b = 0$

$w = d - c$



Linear SVM Mathematically



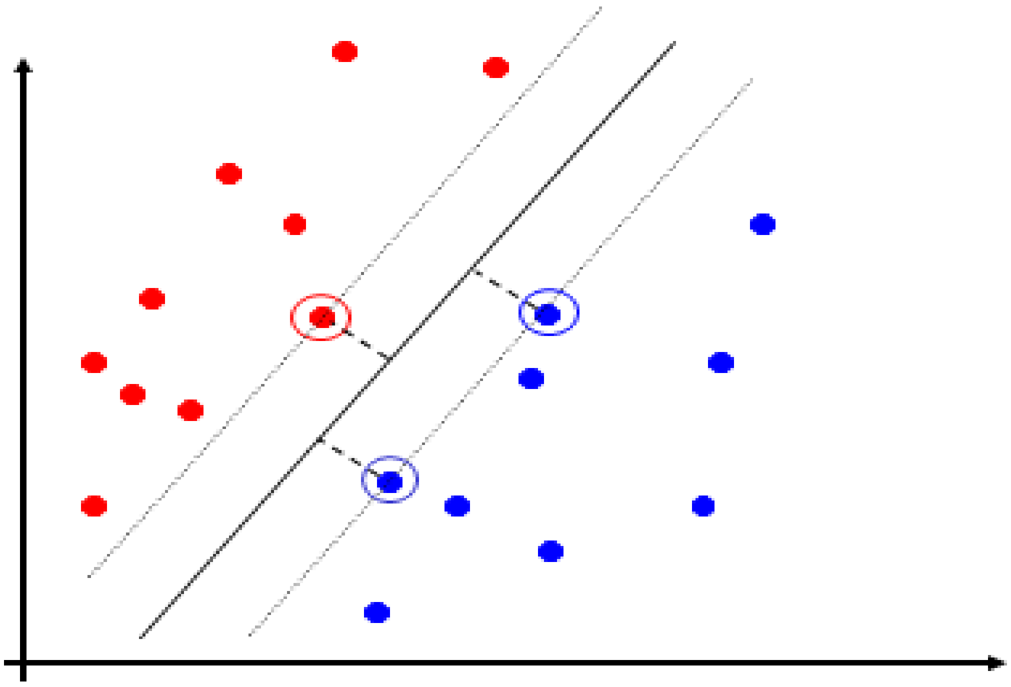
M=Margin Width

What we know:

- $\boldsymbol{w} \,.\, \boldsymbol{x}^+ + b = +1$
- $\boldsymbol{w} \,.\, \boldsymbol{x}^- + b = -1$

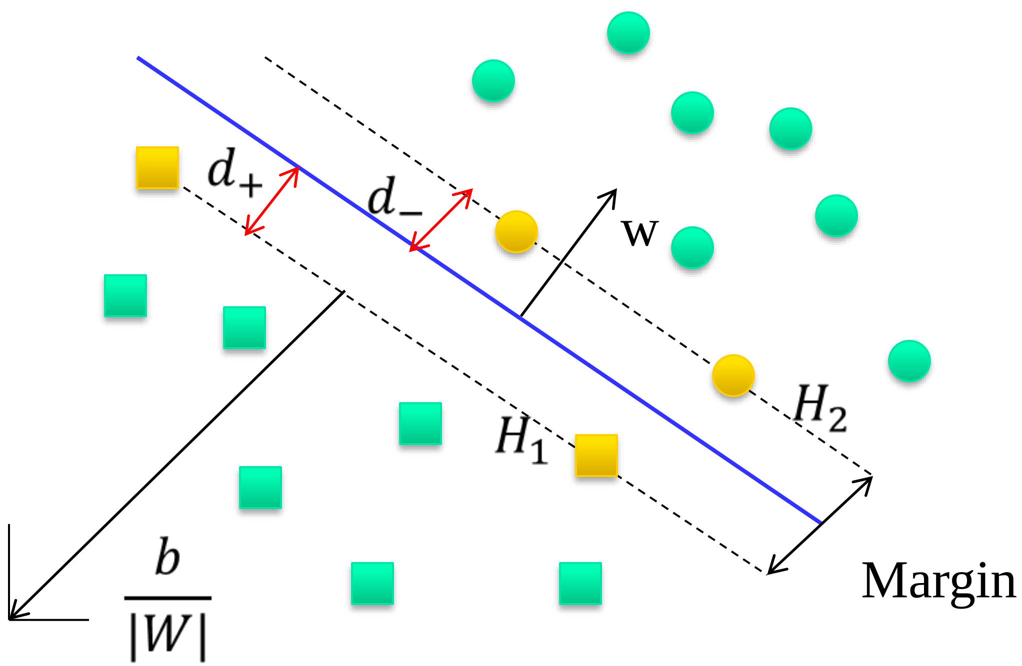
Computing the Margin Width

- Maximizing the margin is good according to intuition and theory.
- Implies that only support vectors are important; other training examples are ignorable.



Computing the Margin Width Cont.

- The points **x** which lie on the hyperplane satisfy $w.x + b = 0$
  - w is normal to the hyperplane,
  - $\frac{|b|}{||w||}$ is the perpendicular distance from the hyperplane to the origin, and
  - $||w||$ is Euclidean norm of **w**.



Computing the Margin Width Cont.

- The points (positive example) which lie on the hyperplane $H_1: w.x_i + b = 1$ with normal **w** and perpendicular distance from the origin $\frac{|1-b|}{\|w\|}$
- Again, the points (negative example) which lie on the hyperplane $H_2: w.x_i + b = -1$ with normal **w** and perpendicular distance from the origin $\frac{|-1-b|}{\|w\|}$
- Let $d_+(d_-)$ be the shortest distance from the separating hyperplane to the closest positive (negative) example.

$$argin\ (M) = d_+ + d$$

$$= \frac{|b|}{\|w\|} - \frac{|1-b|}{\|w\|} = \frac{1}{\|w} \qquad = \frac{|-1-b|}{\|w\|} - \frac{|b|}{\|w\|} = \frac{1}{\|w}$$

$$argin\ (M) = \frac{2}{\|w}$$

Formulate the Decision Boundary

- Let {x_1, ..., x_n} be our data set and let y_i Î {1,-1} be the class label of x_i
- The decision boundary should classify all points correctly

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \qquad \forall i$$

- The decision boundary can be found by solving the following constrained optimization problem

Minimize $\frac{1}{2}||w||^2$

Subject to $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \qquad \forall i$

- This is a constrained optimization problem. Solving it requires some new tools

Recap of Constrained Optimization

- Standard form problem (not necessarily convex)

$$\begin{gathered} minimize\ f(x) \\ subject\ to\ \ h_i(x) \leq 0, i = 1, 2, \dots\dots, m \\ k_i(x) = 0, i = 1, 2, \dots, l \end{gathered}$$

$h_i(x)$ = Series of restrictions

Variables: $x \in R^n$. Optimal Value: p^*. Optimizer: x^*

Lagrangian: augment the objective with a weighted sum of constraints

$$L(x, \lambda, \mu) \cong f(x) + \sum_{i=1}^{m} \lambda_i\, h_i(x) + \sum_{i=1}^{l} \mu_i k_i(x)$$

- $\lambda_i \geq 0$ is Lagrange multiplier associated with $h_i(x) \leq 0$
- μ_i is Lagrange multiplier associated with $k_i(x) = 0$

Lagrange Dual Function

Lagrange dual function: g: $R^m \times R^l \rightarrow R$, (always concave)

$$g(\lambda, \mu) \cong \inf_x L(x, \lambda, \mu)$$
$$= \inf_x \left(f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \sum_{i=1}^{l} \mu_i k_i(x) \right)$$

Lower bound property:

$$g(\lambda, \mu) \leq p^*, \ \ \forall \lambda \geq 0, \mu$$

The optimal solutions of the primal and dual problems are

$$p^* = \min_x \max_{\lambda,\mu} L(x, \lambda, \mu) \text{ (primal)}$$

$$d^* = \max_{\lambda,\mu} \min_x L(x, \lambda, \mu) \ \text{ (dual)}$$

Weak and Strong Duality

- **Weak Duality:**

  The optimal value of the Lagrange dual problem d^* is the best lower bound on the optimal value of the primal problem p^*, i.e., $d^* \le p^*$.

- **Strong Duality**:

  The optimal values of the primal problem and dual problem agrees, i.e. $d^* = p^*$.

Back to Our Problem

$$minimize_{w,b} \frac{1}{2} \|w\|^2$$

Subject to: $y_i(w^T x_i + b) \geq 1, \qquad i = 1,2,3, \dots \dots, n$

Or

$$minimize_{w,b} \frac{1}{2} \|w\|^2$$

Subject to: $1 - y_i(w^T x_i + b) \leq 0, \qquad i = 1, 2, \dots \dots, n$

This is our Primal Problem

Here,

$$f(x) = \frac{1}{2}\|w\|^2 \qquad g_i(x) = 1 - y_i(w^T x + b), \forall 1 \leq i \leq n$$

Primal to Dual Journey

- The Lagrangian is

$$L = f(x) + \sum_{i=1}^{n} \alpha_i g(x)$$

$$= \frac{1}{2} w^T w + \sum_{i=1}^{n} \alpha_i \left(1 - y_i \left(w^T x_i + b\right)\right)$$

  - Note that $\|w\|^2 = w^T \mathrm{w}$ and $\alpha_i \geq 0$

- Setting the gradient of L

$$\frac{\partial}{\partial X}\left(\frac{1}{2}\|w\|^2 + \sum_{i=1}^{n} \alpha_i \left(1 - y_i \left(w^T x_i + b\right)\right)\right) = 0$$

- Differentiate w.r.t. **w**, we have

$$w + \sum_{1=1}^{n} \alpha_i \left(-y_i\right) x_i = 0 \quad \Rightarrow w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

- Differentiate w.r.t. **b**, we have

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

Primal to Dual Journey

- If we substitute $w = \sum_{i=1}^{n} \alpha_i y_i x_i$ to L, we have

$$L = \frac{1}{2}\sum_{i=1}^{n} \alpha_i y_i x_i^T \sum_{j=1}^{n} \alpha_j y_j x_j + \sum_{i=1}^{n} \alpha_i \left(1 - y_i \left(\sum_{j=1}^{n} \alpha_j y_j x_j^T x_i + b\right)\right)$$

$$= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i y_i \sum_{j=1}^{n} \alpha_j y_j x_j^T x_i - b \sum_{i=1}^{n} \alpha_i y_i$$

$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

  - Note that $\sum_{i=1}^{n} \alpha_i y_i = 0$

- This is a function of a_i only

The Dual Problem

- The new objective function is in terms of a_i only
- It is known as the dual problem:

  if we know **w**, we know all a_i
  if we know all a_i, we know **w**

- The original problem is known as the primal problem
- The objective function of the dual problem needs to be maximized!

The Dual Problem (Cont.)

- The dual problem is therefore:

$$\max\ g(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

Subject to: $\alpha_i \geq 0$ $\qquad$ $\sum_{i=1}^{n} \alpha_i y_i = 0$

Properties of α_i when we introduce the Lagrange multipliers

The result when we differentiate the original Lagrangian w.r.t. b

This is a quadratic programming (QP) problem. A global maximum of α_i can always be found.

Finding "w" and "b" for the boundary $w^t x + b$

- Using the KKT(Karus-Kuhn-Tucker) condition:

$$\forall i \qquad \alpha_i(1 - y_i(w^T x_i + b)) = 0$$

- We can calculate "b" by taking " i" such that $\alpha_i > 0$:

Must be $y_i(w^t x_i + b) - 1 = 0 \Rightarrow b = \frac{1}{y_i} - w^t x_i = \underline{y_i - w^t x_i} \quad (y_i \in \{1, -1\})$

- Calculating "w" will be done using what we have found above :

$$w = \sum_i \alpha_i y_i x_i$$

- Usually, Many of the α_i -s are zero so the calculation of "w" has a low complexity.

Solution of this Optimization Problem

- The solution has the form:

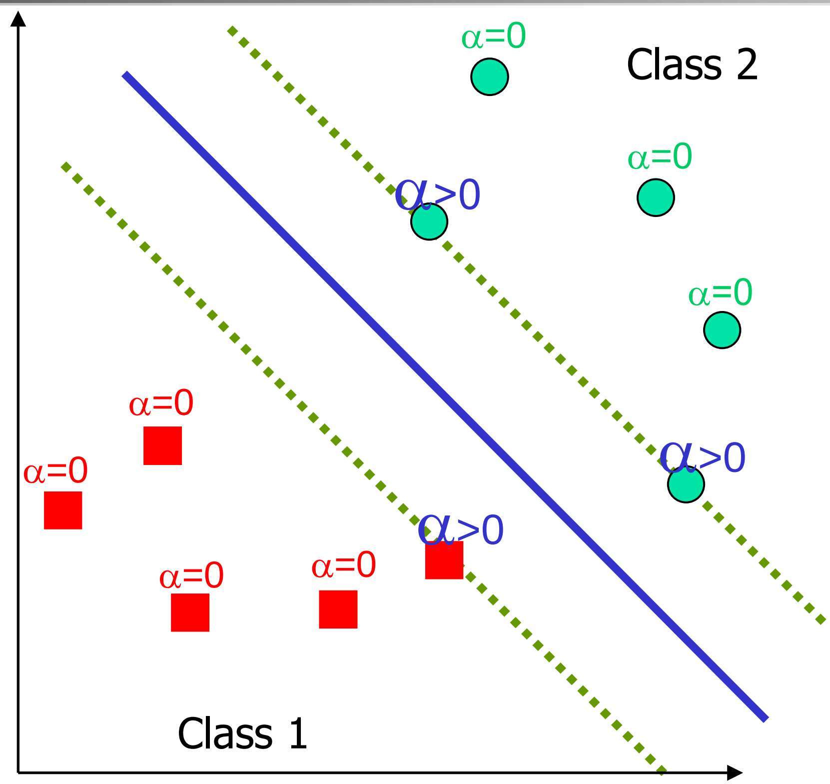
$$\boxed{\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x_i} \qquad b = y_k - \mathbf{w}^\mathbf{T}\mathbf{x_k} \text{ for any } \mathbf{x_k} \text{ such that } \quad \alpha_k \neq 0}$$

- Each non-zero α_i indicates that corresponding $\mathbf{x_i}$ is a support vector.
- Then the classifying function will have the form:

$$\boxed{f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x_i}^\mathbf{T}\mathbf{x} + b}$$

- Notice that it relies on an *inner product* between the test point **x** and the support vectors $\mathbf{x_i}$ – we will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x_i}^\mathbf{T}\mathbf{x_j}$ between all pairs of training points.

Support Vectors



$$w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

$$w = \sum_{i \in SV} \alpha_i y_i x_i$$

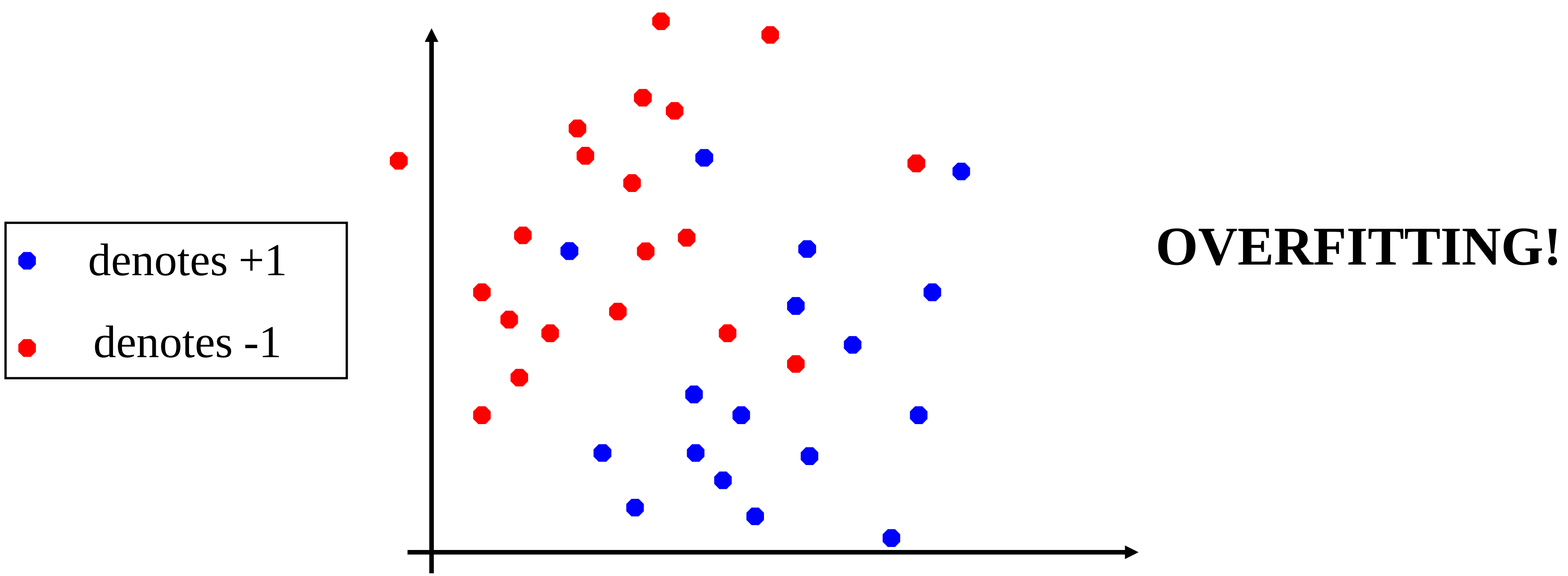
- x_i with α_i >0 are called *support vectors* (SV)
- w is determined only by the SV

Dataset with noise

- **Hard Margin:** So far we require all data points be classified correctly

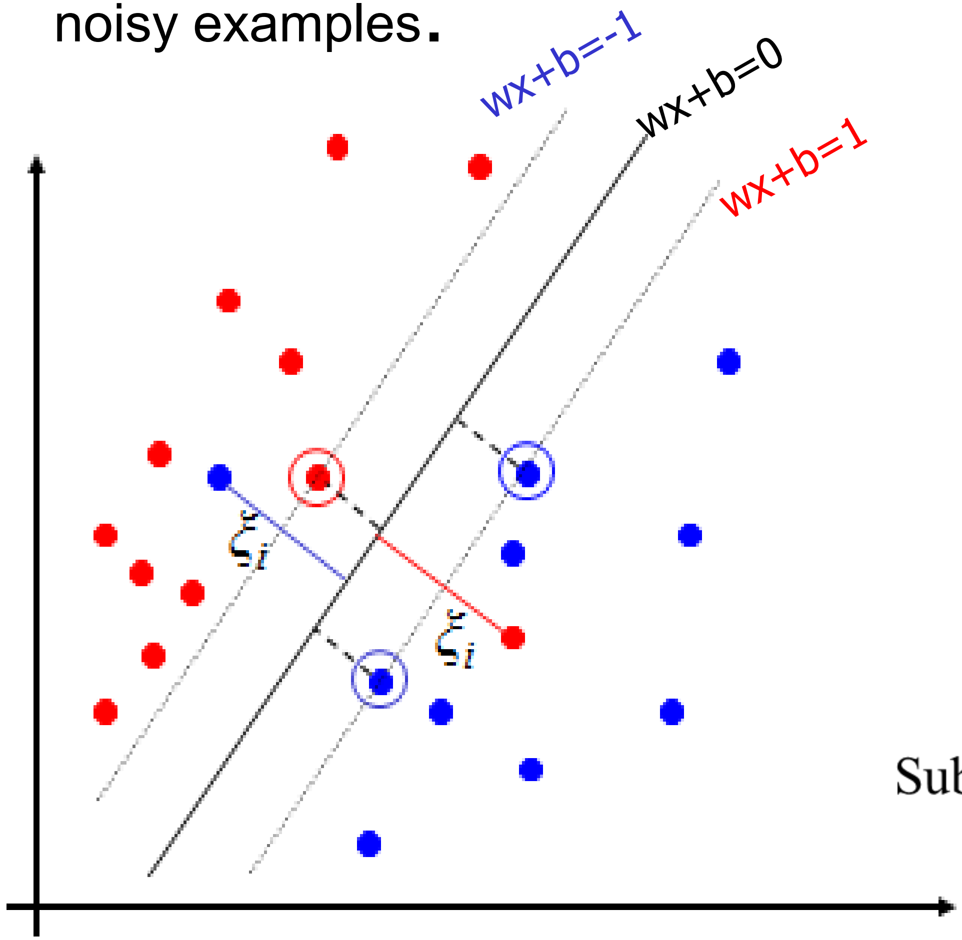
  - No training error

- **What if the training set is noisy?**



Soft Margin Classification

- ✓ ***Slack variables*** ξ_i can be added to allow misclassification of difficult or noisy examples.



What should our quadratic optimization criterion be?

$$min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{Subject to: } y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \forall i$$

$$\xi_i \geq 0 \;\;, \forall i$$

Hard Margin Vs. Soft Margin

- The old formulation:

Find **w** and *b* such that

$\boldsymbol{\Phi}(\mathbf{w}) = \frac{1}{2}\, \mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$

$y_i\,(\mathbf{w^T x_i} + b) \geq 1$

- The new formulation incorporating slack variables:

Find **w** and *b* such that

$\boldsymbol{\Phi}(\mathbf{w}) = \frac{1}{2}\, \mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$

$y_i\,(\mathbf{w^T x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- Parameter *C* can be viewed as a way to control overfitting.

The Optimization Problem
(Soft Margin Classification)

- The dual of this new constrained optimization problem is :

$$\text{Find } \alpha_1 \ldots \alpha_N \text{ such that}$$
$$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \tfrac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \boxed{\mathbf{x_i}^T\mathbf{x_j}} \text{ is maximized and}$$
$$(1)\ \ \Sigma\alpha_i y_i = 0$$
$$(2)\ \ 0 \leq \alpha_i \leq C \text{ for all } \alpha_i$$

- This is very similar to the optimization problem in the linear separable case, except that there is an upper bound C on a_i now. Once again, a QP solver can be used to find a_i
- Neither slack variables ξ_i nor their Lagrange multipliers appear in the dual problem!
- Again, $\mathbf{x_i}$ with non-zero α_i will be support vectors.
- Solution to the dual problem is:

$$\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x_i}$$
$$b = y_k(1 - \xi_k) - \mathbf{w}^T\mathbf{x_k} \text{ where } k = \underset{k}{\operatorname{argmax}}\ \alpha_k$$

But neither **w** nor *b* are needed explicitly for classification!

$$f(\mathbf{x}) = \Sigma\alpha_i y_i \boxed{\mathbf{x_i}^T\mathbf{x}} + b$$

Linear SVMs: Overview

- The classifier is a *separating hyperplane.*
- Most "important" training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $\mathbf{x}_i$ are support vectors with non-zero Lagrangian multipliers α_i.
- Both in the dual formulation of the problem and in the solution training points appear only inside inner products:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \boxed{\mathbf{x_i}^T\mathbf{x_j}}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

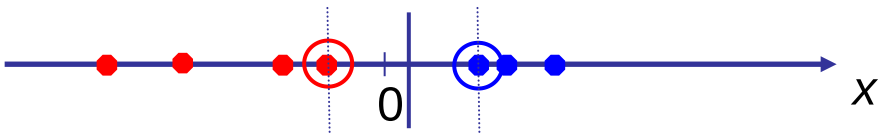
$f(\mathbf{x}) = \Sigma\alpha_i y_i \boxed{\mathbf{x_i}^T\mathbf{x}} + b$

Extension to Non-linear Decision Boundary

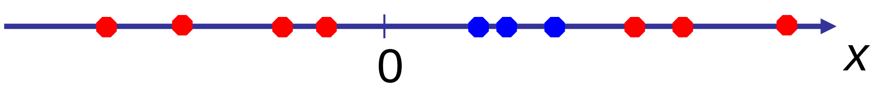
- So far, we have only considered large-margin classifier with a linear decision boundary
- How to generalize it to become nonlinear?

Non-linear SVMs

- Datasets that are linearly separable with some noise work out great:

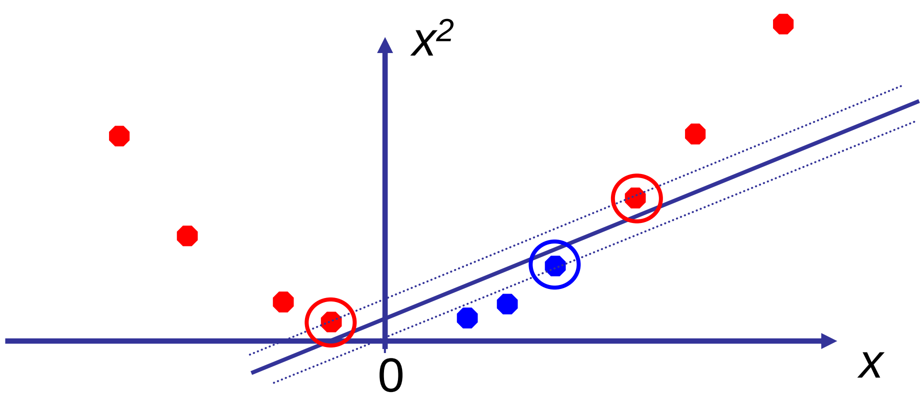


- But what are we going to do if the dataset is just too hard?



- How about… mapping data to a higher-dimensional space:

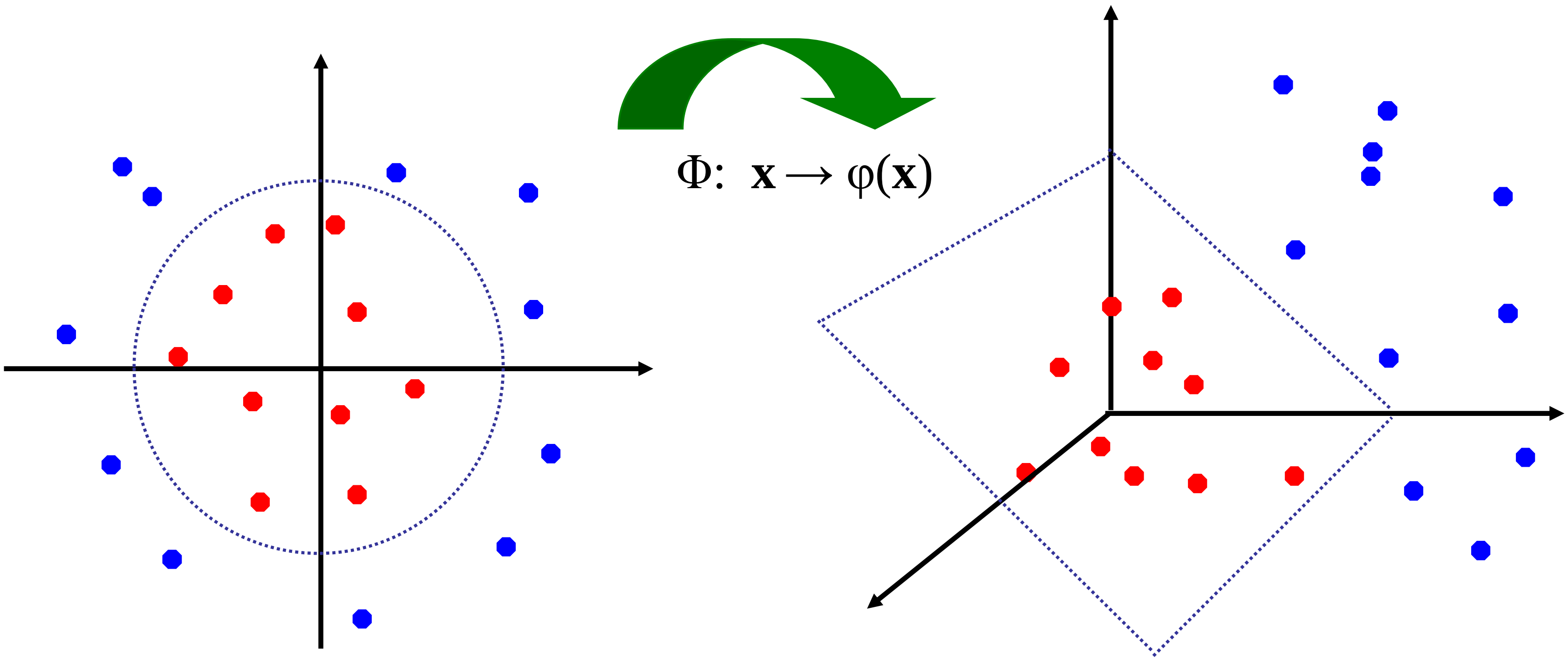
mapping data to two-dimensional space with $\phi(x) = (x, x^2)$



Non-linear SVMs: Feature spaces

- **General idea:**

The original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:

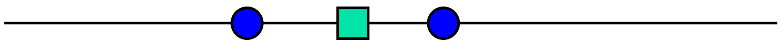


Non-linear SVMs: Feature spaces

- **Key idea:** transform x_i to a higher dimensional space to "make classes linearly separable"
  - **Input space:** the space x_i are inputs
  - **Feature space:** the space of $\varphi(x_i)$ after transformation
- **Why transform?**
  - Linear operation in the feature space is equivalent to non-linear operation in input space
  - The classification task can be "easier" with a proper transformation. Example: XOR

Example: Mapping To Feature Space

$x_1, x_2, x_3 \in R^1$



$x_1 = 0, x_2 = 1, x_3 = 2$ *(nonseparable in* R^1*)*

mapping to higher dim*ension:*

$x \rightarrow \phi(x) = (x^2, \sqrt{2}x, 1)$

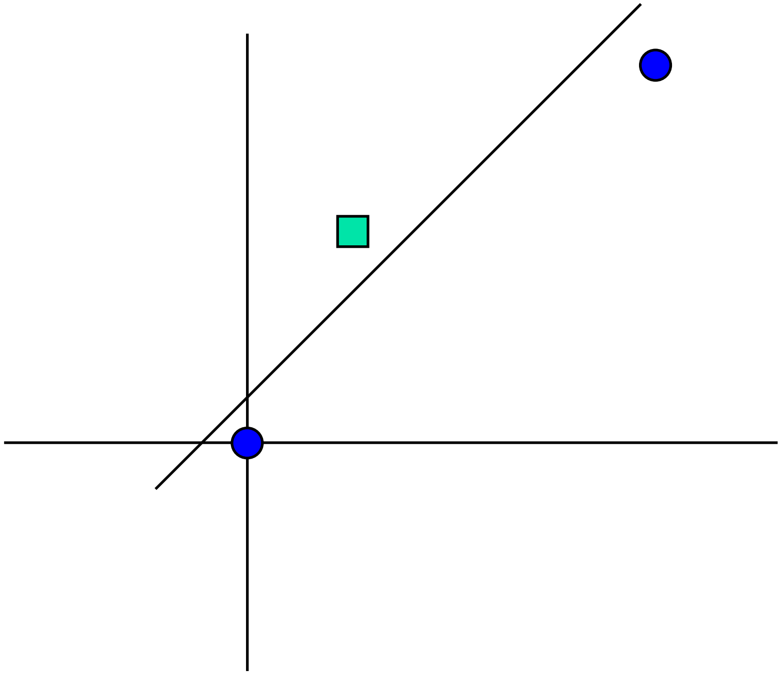
$R^1 \rightarrow R^3$

$0 \rightarrow (0, 0, 1)$

$1 \rightarrow (1, \sqrt{2}, 1)$

$2 \rightarrow (4, 2\sqrt{2}, 1)$

now separable



Classification Problem in Feature Space

$$x_1, x_2, \cdots, x_l \in R^n$$

$$\downarrow$$

$$\phi(x_1), \phi(x_2), \cdots \phi(x_l) \in R^m$$

- Find a linear separating hyperplane

$$\max \frac{2}{\|w\|}$$

$$s.t.\ w^T \phi(x_i) + b \geq 1\ if\ \ y_i = 1$$

$$w^T \phi(x_i) + b \leq 1\ if\ \ y_i = -1$$

Classification Problem in Feature Space

$$\max \frac{2}{\|w\|} \equiv \min \frac{\|w\|}{2} = \min_{w,b} \frac{w^T w}{2}$$

$$subject.to.\ y_i (w^T \phi(x_i) + b) \geq 1\ i = 1, \cdots l$$

- Questions:
  1. How to choose ϕ ?
  2. Is it really better? Yes.

Soft margin Hyperplane

- Some times even in high dimension spaces, Data may still not separable.

→ Allow training error

$$\min_{w,b,\xi} \quad \frac{1}{2} w^T w + C\left(\sum_{i=1}^{l} \xi_i\right)$$

$$y_i((w^T \phi(x_i)) + b) \geq 1 - \xi_i,$$

$$\xi_i \geq 0, i = 1, \cdots, l$$

Optimization Problem to find W and b

- Consider the following primal problem:

$$\text{minimise}_{\xi, w, b} \quad w^T \cdot w + C\sum_{i=1}^{l} \xi_i$$

$$\textit{subject to} \quad y_i (w^T \cdot \phi(x_i) + b) \geq 1 - \xi_i \ , i = 1, \cdots, l$$

$$\xi_i \geq 0 \ , i = 1, \cdots, l$$

- Find $\alpha_1 \ldots \alpha_N$ such that

$$maximize_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i \, y_j \emptyset(x_i)^T \emptyset(x_j)$$

Subject to: $\sum_{i=1}^{n} y_i \alpha_i = 0, 0 < \alpha_i < C$ for all $i = 1,2,3, \ldots \ldots, n$

How to know the mapping ∅?

- The mapped data only occurs as an inner product in the objectives.
- A **kernel function** is defined as a function that corresponds to a dot product of two feature vectors in some expanded feature space:

$$k(x_i, x_j) = \emptyset(x_i)^T \emptyset(x_j)$$

- Now we only need to compute $K(x_i, x_j)$ and we don't need to perform computations in high dimensional space explicitly. This is what is called the Kernel Trick.
- Example:

  2-dimensional vectors $\mathbf{x}=[x_1\ \ x_2]$; let $K(\mathbf{x_i},\mathbf{x_j})=(1 + \mathbf{x_i}^T\mathbf{x_j})^2$,

  Need to show that $K(\mathbf{x_i},\mathbf{x_j})= \varphi(\mathbf{x_i})^T\varphi(\mathbf{x_j})$:

  $K(\mathbf{x_i},\mathbf{x_j})=(1 + \mathbf{x_i}^T\mathbf{x_j})^2 = 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$

  $= [1\ \ x_{i1}^2\ \ \sqrt{2}\, x_{i1}x_{i2}\ \ x_{i2}^2\ \ \sqrt{2}x_{i1}\ \ \sqrt{2}x_{i2}]^T [1\ \ x_{j1}^2\ \ \sqrt{2}\, x_{j1}x_{j2}\ \ x_{j2}^2\ \ \sqrt{2}x_{j1}\ \ \sqrt{2}x_{j2}]$

  $= \varphi(\mathbf{x_i})^T\varphi(\mathbf{x_j})$, where $\varphi(\mathbf{x}) = [1\ \ x_1^2\ \ \sqrt{2}\, x_1x_2\ \ x_2^2\ \ \sqrt{2}x_1\ \ \sqrt{2}x_2]$

Non-linear SVMs Mathematically

- **Dual problem formulation:**

Find $\alpha_1 \ldots \alpha_N$ such that
$Q(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j K(\mathrm{x_i}, \mathrm{x_j})$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

- **The solution is:**

$$\boxed{f(\mathrm{x}) = \Sigma\alpha_i y_i K(\mathrm{x_i}, \mathrm{x_j}) + b}$$

- **Optimization techniques for finding α_i's remain the same!**

Non-linear SVM Cont.

- It is observed that to change from a linear to nonlinear classifier, one must only substitute a kernel evaluation in the objective instead of the original dot product.

- No algorithmic changes are required from the linear case other than substitution of a kernel evaluation for the simple dot product.

Kernel function

Commonly Used Kernel

- Linear kernel: $K(x_i, x_j) = < x_i, x_j >$
- Polynomial kernel: $K(x_i, x_j) = (< x_i, x_j > +1)^d$
- Gaussian kernel: $K(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma})$

Positive Semi-definite Function

Let E be a set and let $K(x,y)$ be a real valued function defined on $E \times E$. Then $K(x,y)$ is called a **positive Semi-definite Function** or a **positive type function** on E if, for any finite set of point $\{x_i\}, i = 1, 2, \ldots, n$ in E and for any $\xi_1, \xi_2, \ldots, \xi_n$, in **C**

$$\sum_{i=1}^{n} \sum_{j=1}^{n} K(x_i, x_j)\, \xi_i \xi_j \geq 0$$

SVM Pros and Cons

- Pros:
  - Easy to integrate different distance functions
  - Fast classification of new objects (depends on SV)
  - Good performance even with small set of examples

- Cons:
  - Slow training (O(n2), n= number of vectors in training set)
  - Separates only 2 classes

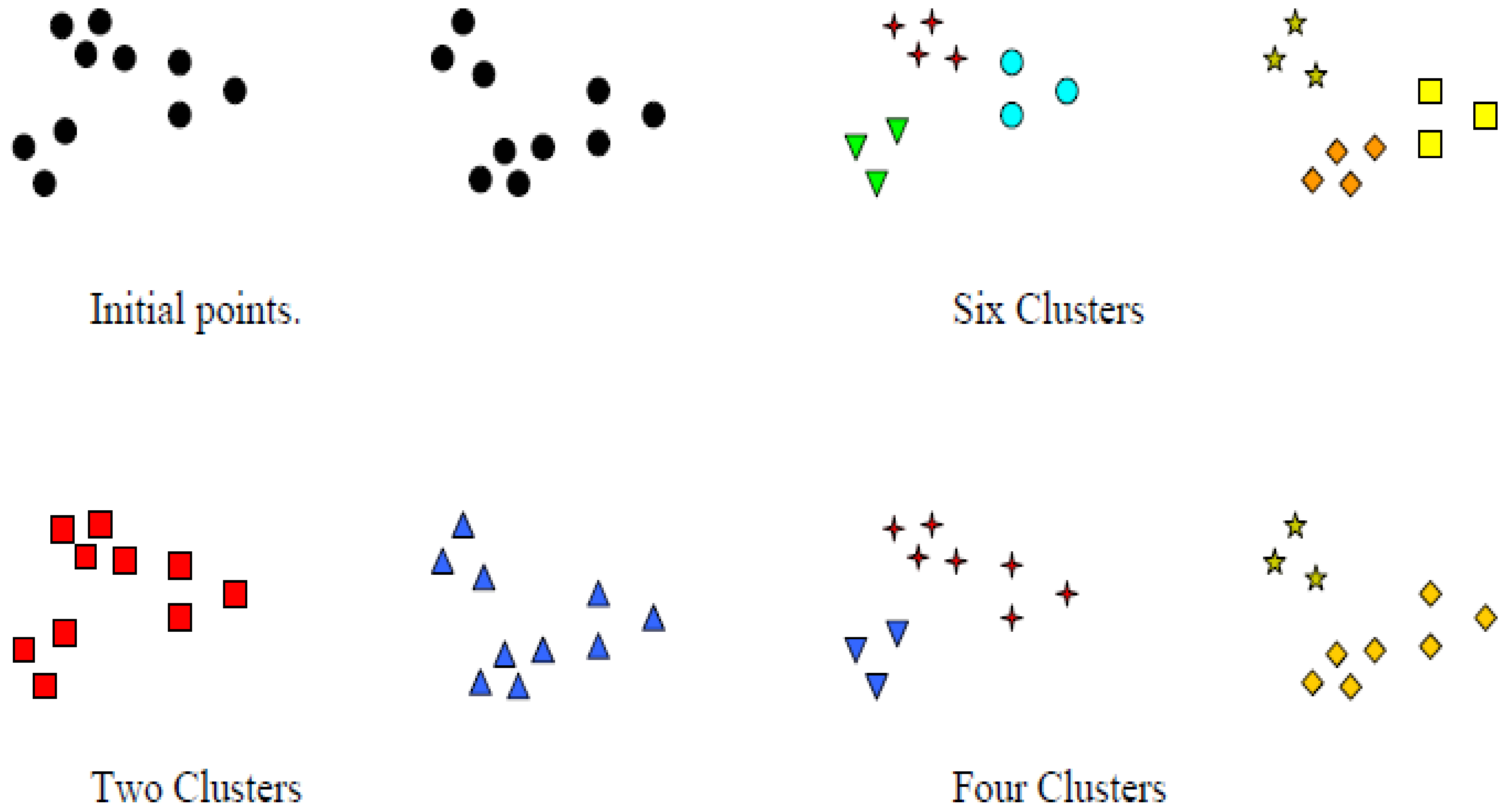
Clustering

- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups.
  - Based on information found in the data that describes the objects and their relationships.
  - Also known as unsupervised classification.
- Many applications
  - **Understanding:** group related documents for browsing or to find genes and proteins that have similar functionality.
  - **Summarization:** Reduce the size of large data sets.
- Web Documents are divided into groups based on a similarity metric.
  - Most common similarity metric is the dot product between two document vectors.

What is not Cluster Analysis?

- Supervised classification.
  - Have class label information.
- Simple segmentation.
  - Dividing students into different registration groups alphabetically, by last name.
- Results of a query.
  - Groupings are a result of an external specification.
- Graph partitioning
  - Some mutual relevance and synergy, but areas are not identical.

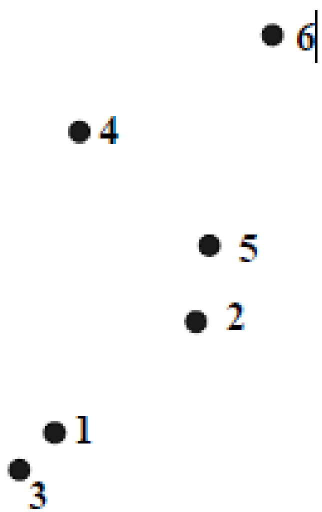
Notion of a Cluster is Ambiguous



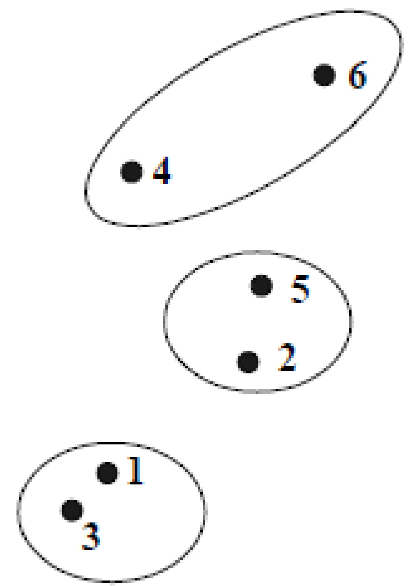
Types of Clusterings

- A clusteringis a set of clusters.
- One important distinction is between hierarchicaland partitionalsets of clusters.
- **Partitional Clustering**
  - A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset.
- **Hierarchical clustering**
  - A set of nested clusters organized as a hierarchical tree.

Partitional Clustering



Original Points



A Partitional Clustering

Hierarchical Clustering

- Variance is a measure of how data points differ from the mean
- Example:

  Data Set 1: 3, 5, 7, 10, 10

  Data Set 2: 7, 7, 7, 7, 7

  What is the mean and median of the above data set?

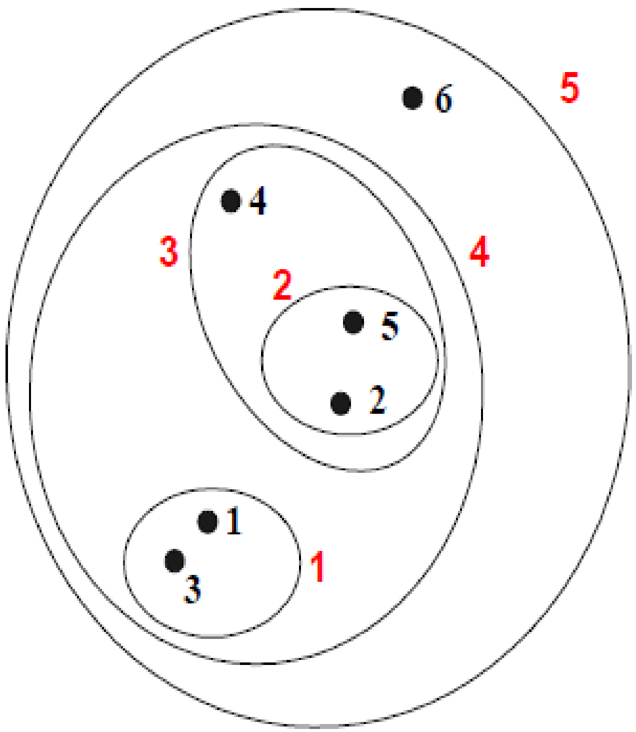
  Data Set 1: mean = 7, median = 7

  Data Set 2: mean = 7, median = 7

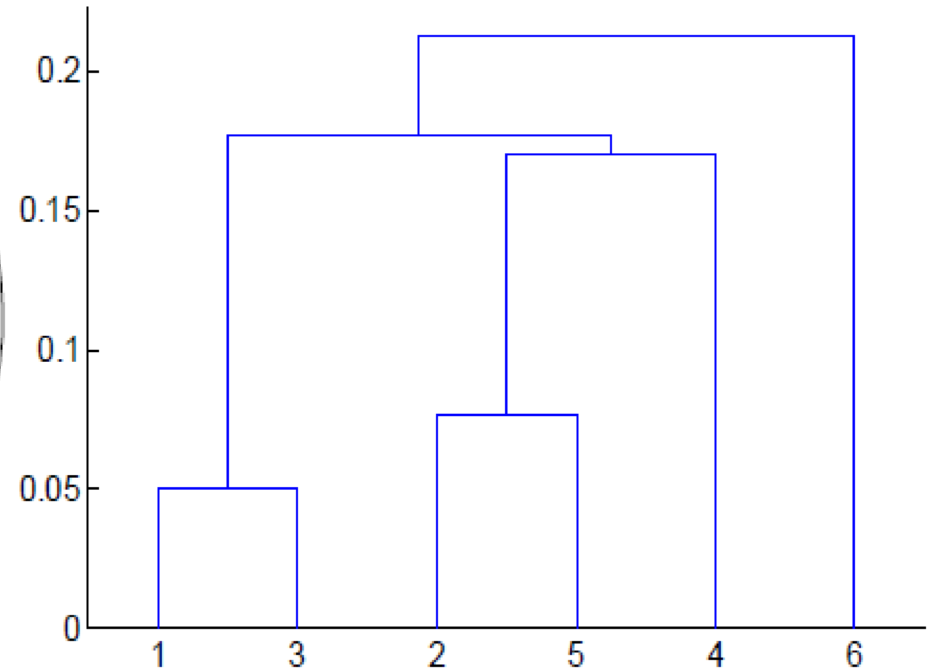
  But we know that the two data sets are not identical! The variance shows how they are different.

  We want to find a way to represent these two data set numerically.

Hierarchical Clustering



Traditional Hierarchical Clustering



Traditional Dendrogram

Clustering by Density Based Methods

- Population variance:

$$\sigma^2 = \frac{\sum_{i=1}^{N}(X_i - \mu)^2}{N}$$

μ = population mean

N = population size

X_i = i^{th} value of the variable X

Clustering by Grid-Based Methods

Calculate the Variance for Ungrouped Data

1. Find the Mean.
2. Calculate the difference between each score and the mean.
3. Square the difference between each score and the mean.
4. Add up all the squares of the difference between each score and the mean.
5. Divide the obtained sum by n – 1.

Clustering by Model-Based Methods

Calculate the Variance for Grouped Data

1. Calculate the mean.
2. Get the deviations by finding the difference of each midpoint from the mean.
3. Square the deviations and find its summation.
4. Substitute in the formula.

Clustering High-Dimensional Data

- It is based on the quartiles so while calculating this may require upper quartile (Q3) and lower quartile (Q1) and then is divided by 2. Hence it is half of the deference between two quartiles it is also a semi inter quartile range.
- Quartile deviation considers only 50% of the item and ignores the other 50% of items in the series.
- The formula of Quartile Deviation is

$$Q\,D = \frac{Q3 - Q1}{2}$$

Outlier analysis

- Mean Deviation is also known as average deviation.
- In this case deviation taken from any average especially Mean, Median or Mode.
- While taking deviation we have to ignore negative items and consider all of them as positive. The formula is given below
- The mean deviation is an average of absolute deviations of individual observations from the central value of a series.
- Mean Deviation = $\frac{\sum_{i=1}^{N} |x_i - \bar{x}|}{N}$

Prediction

- Average deviation about mean

$$MD(\bar{x}) = \frac{\sum_{i=1}^{k} f_i \left| x_i - \bar{x} \right|}{n}$$

k = Number of classes
x_i= Mid point of the i-th class
f_i= frequency of the i-th class

- MD = $\frac{\sum m}{N}$ (deviation taken from median)
- MD = $\frac{\sum z}{N}$ (deviation taken from mode)

Linear Regression

- Example: Find the mean deviation of the set 2,3,6,8,11.

Arithmetic mean $(\bar{x}) = \frac{2+3+6+8+11}{5} = 6$

Mean Deviation = $\frac{\sum_{i=1}^{N}|x_i-\bar{x}|}{N} = \frac{|2-6|+|3-6|+|6-6|+|8-6|+|11-6|}{5}$

$= \frac{|-4|+|-3|+|0|+|2|+|5|}{5}$

$= \frac{4+3+0+2+5}{5} = 2.8$

Nonlinear Regression

- Measures the variation of observations from the mean
- The most common measure of dispersion
- Takes into account every observation
- Measures the ‘average deviation’ of observations from mean
- Works with squares of residuals not absolute values—easier to use in further calculations
- Is the square root of the variance
- Has the same units as the original data

Other Regression-Based Methods of prediction

Standard deviation of a sample s

- In practice, most populations are very large and it is more common to calculate the sample standard deviation.

$$Sample\ standard\ deviation = s = \sqrt{\frac{\sum\left(x-\overline{x}\right)^2}{n-1}}$$

- Where: (n-1) is the number of observations in the sample

Standard deviation of a population δ

- Every observation in the population is used.

$$Standard\ deviation = \delta = \sqrt{\frac{\sum\left(x-\overline{x}\right)^2}{n}}$$

Evaluating the Accuracy and error measures of a Classifier or Predictor

- 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, random subsampling
  - Cross-validation
  - Bootstrap
- Comparing classifiers:
  - Confidence intervals
  - Cost-benefit analysis and 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)**

Example of Confusion Matrix:

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

- Given *m* classes, an entry, $CM_{i,j}$ in a **confusion matrix** indicates # of tuples in class *i* that were labeled by the classifier as class *j*
- May have extra rows/columns to provide totals

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

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

- **Classifier Accuracy,** or recognition rate: percentage of test set tuples that are correctly classified

  Accuracy = (TP + TN)/All

- **Error rate:** *1 – accuracy*, or

  Error rate = (FP + FN)/All

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

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

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

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

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

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

- Perfect score is 1.0
- Inverse relationship between precision & recall
- ***F* measure (*F_1* or *F*-score)**: harmonic mean of precision and recall,

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

- ***$F_ß$*:** weighted measure of precision and recall
  - assigns ß times as much weight to recall as to precision

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

Evaluating Classifier Accuracy: Cross-Validation Methods

- 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 Di as test set and others as training set
  - **Leave-one-out:** k folds where k = # of tuples, for small sized data
  - ***Stratified cross-validation*:** folds are stratified so that class dist. in each fold is approx. the same as that in the initial data