

# Classification fundamentals



Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis

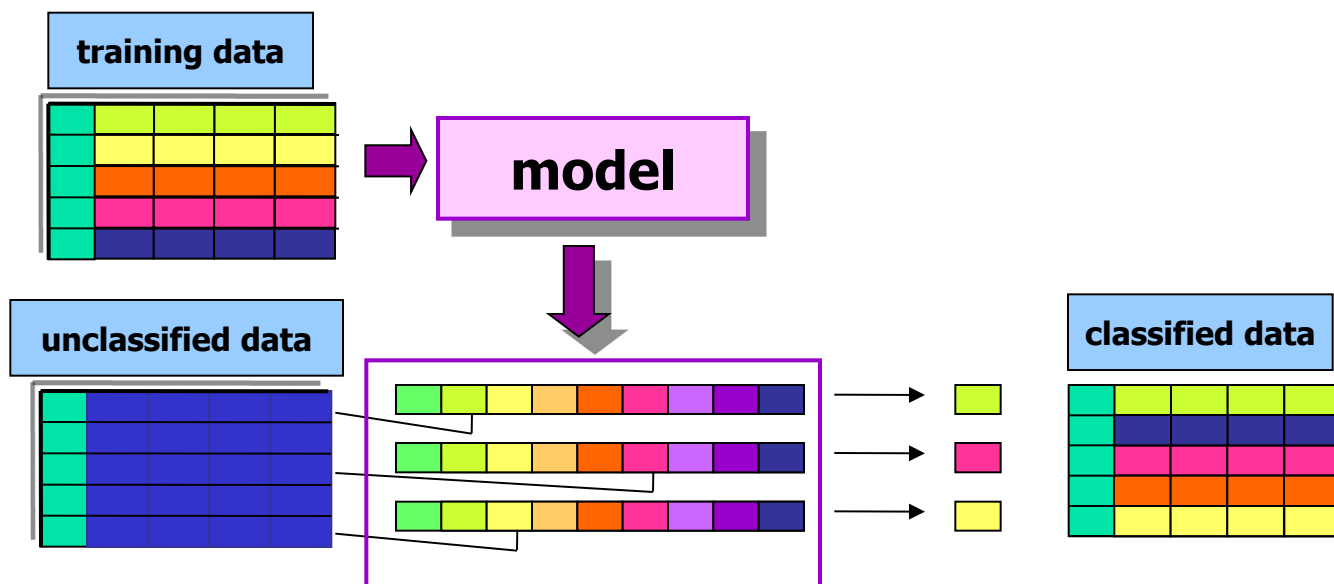
*Politecnico di Torino*



# Classification

## ■ Objectives

- prediction of a class label
- definition of an interpretable model of a given phenomenon

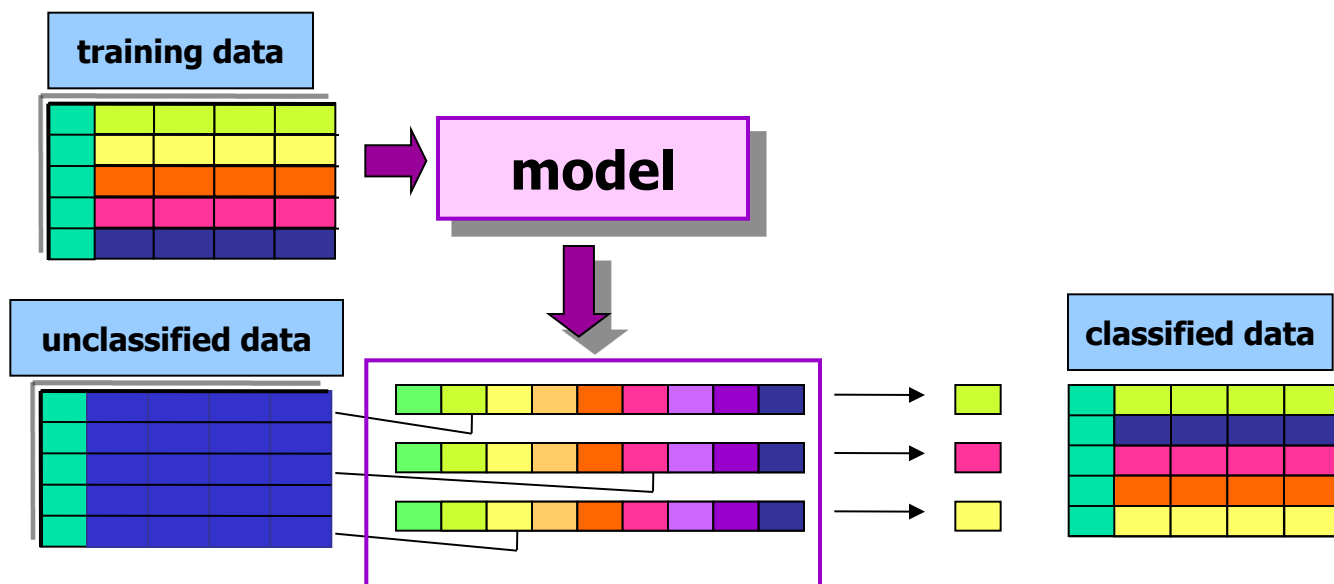




# Classification

## ■ Applications

- detection of customer propensity to leave a company (churn or attrition)
- fraud detection
- classification of different pathology types
- ...





# Classification: definition

- Given
  - a collection of class labels
  - a collection of data objects labelled with a class label
- Find a descriptive profile of each class, which will allow the assignment of unlabeled objects to the appropriate class



# Definitions

- Training set
  - Collection of labeled data objects used to learn the classification model
- Test set
  - Collection of labeled data objects used to validate the classification model



# Classification techniques

- Decision trees
- Classification rules
- Association rules
- Neural Networks
- Naïve Bayes and Bayesian Networks
- k-Nearest Neighbours (k-NN)
- Support Vector Machines (SVM)
- ...



# Evaluation of classification techniques

- Accuracy
  - quality of the prediction
- Interpretability
  - model interpretability
  - model compactness
- Incrementality
  - model update in presence of newly labelled record
- Efficiency
  - model building time
  - classification time
- Scalability
  - training set size
  - attribute number
- Robustness
  - noise, missing data

# Decision trees



Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis  
*Politecnico di Torino*

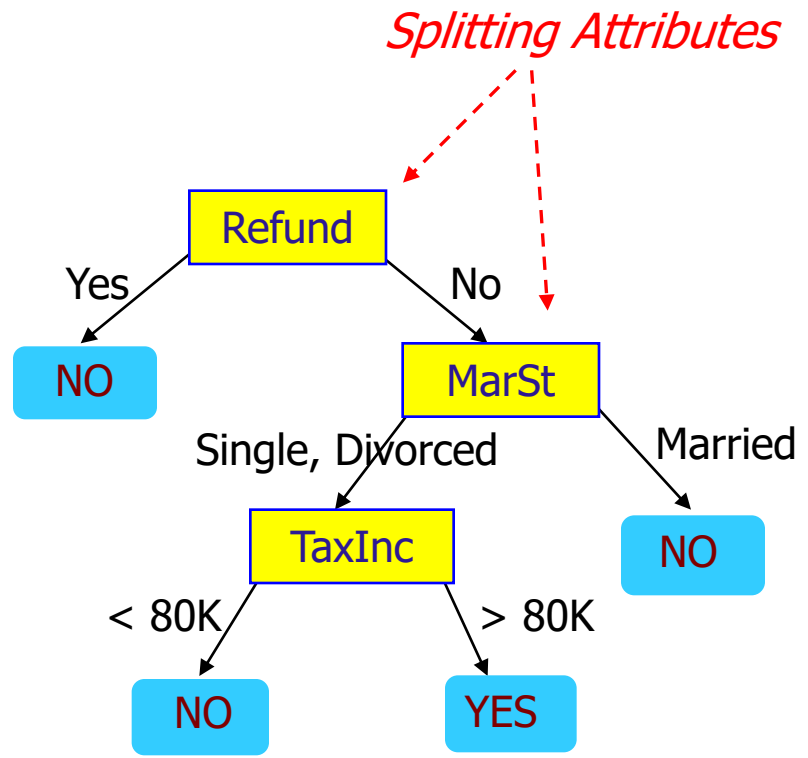


# Example of decision tree

categorical  
categorical  
continuous  
class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data



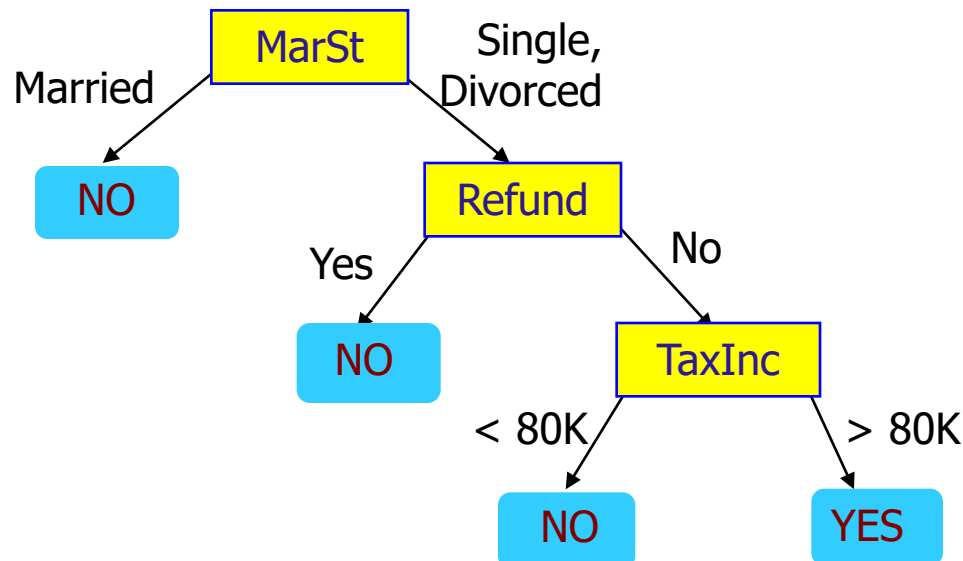
Model: Decision Tree



# Another example of decision tree

categorical  
categorical  
continuous  
class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

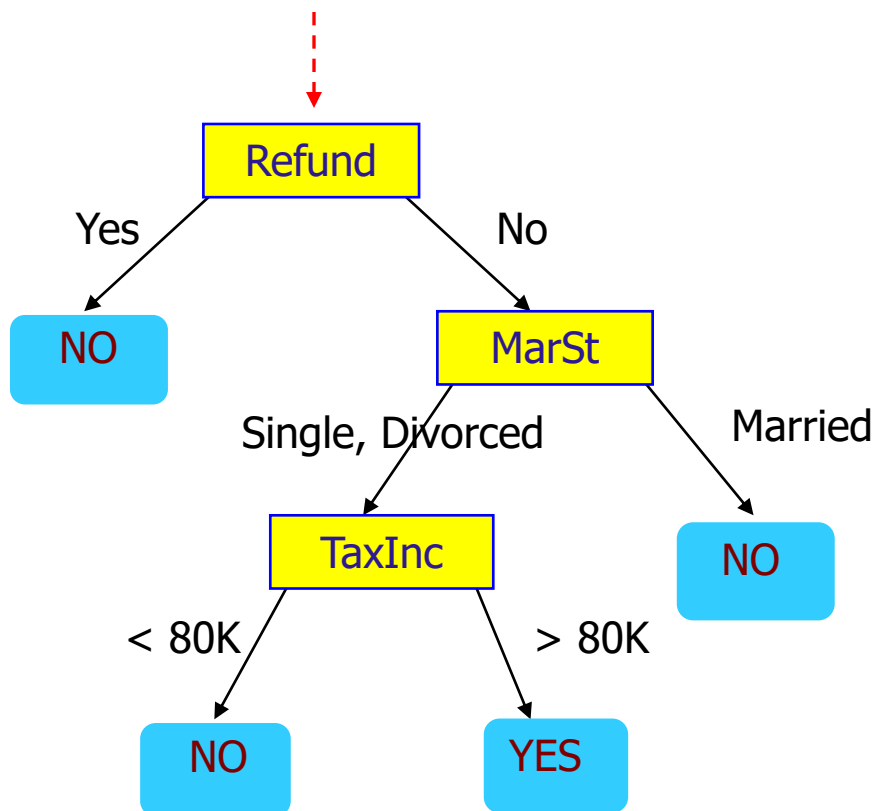


There could be more than one tree that fits the same data!



# Apply Model to Test Data

Start from the root of tree.



Test Data

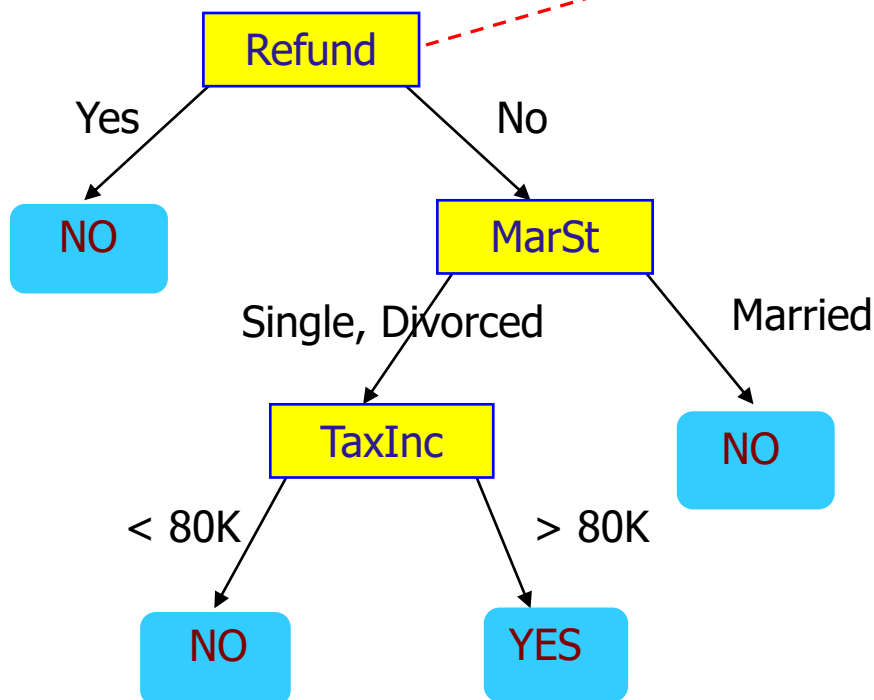
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



# Apply Model to Test Data

## Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

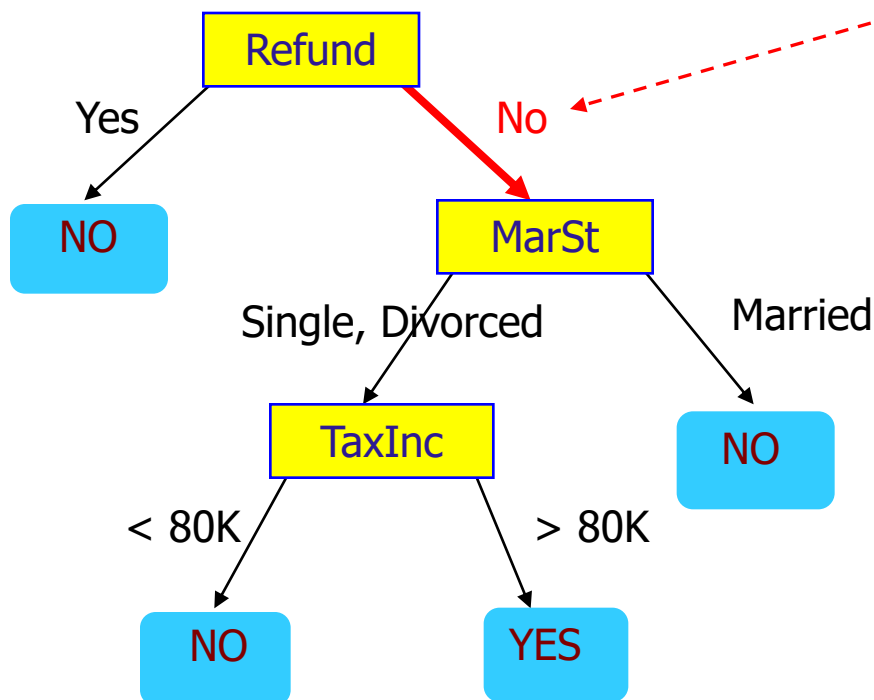




# Apply Model to Test Data

## Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

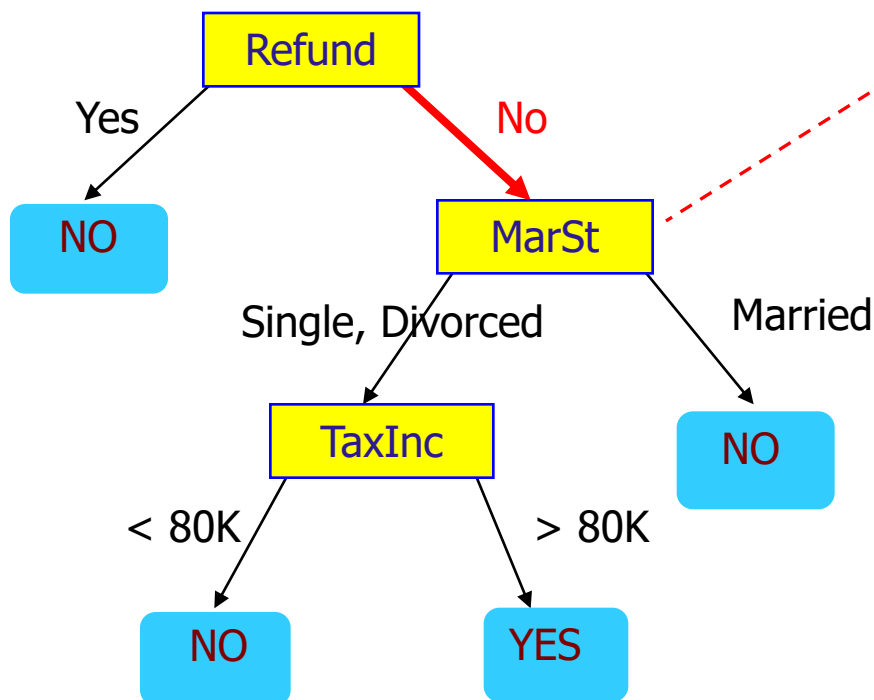




# Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

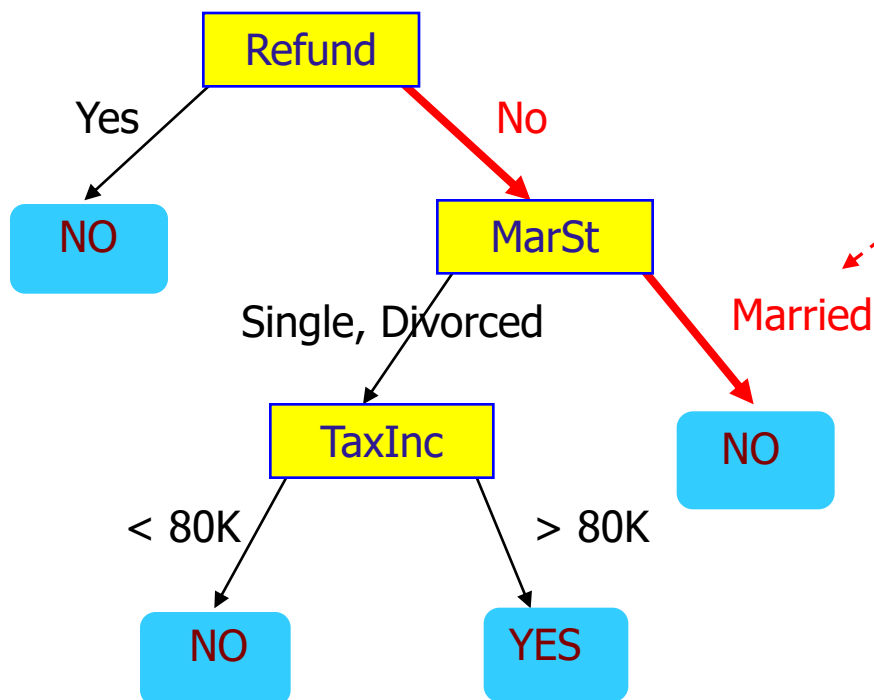




# Apply Model to Test Data

## Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

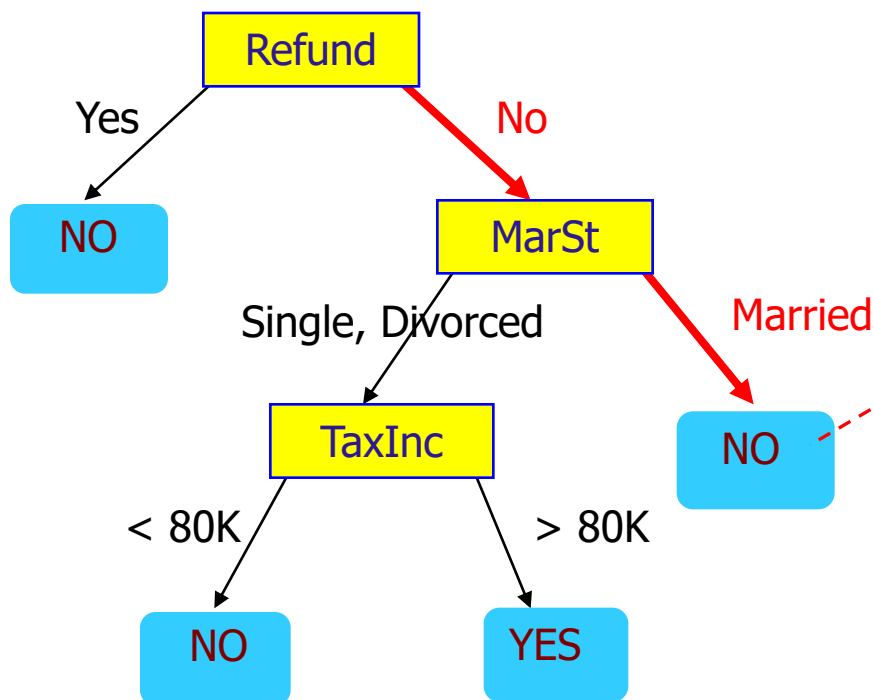




# Apply Model to Test Data

## Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Assign Cheat to "No"



# Decision tree induction

- Many algorithms to build a decision tree
  - Hunt's Algorithm (one of the earliest)
  - CART
  - ID3, C4.5, C5.0
  - SLIQ, SPRINT



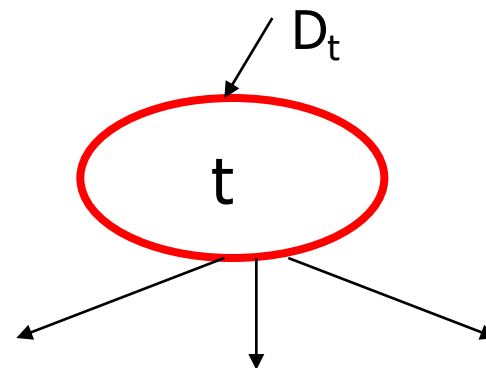
# General structure of Hunt's algorithm

## Basic steps

- If  $D_t$  contains records that belong to more than one class
  - select the “best” attribute  $A$  on which to split  $D_t$  and label node  $t$  as  $A$
  - split  $D_t$  into smaller subsets and recursively apply the procedure to each subset
- If  $D_t$  contains records that belong to the same class  $y_t$ 
  - then  $t$  is a leaf node labeled as  $y_t$
- If  $D_t$  is an empty set
  - then  $t$  is a leaf node labeled as the default (majority) class,  $y_d$

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

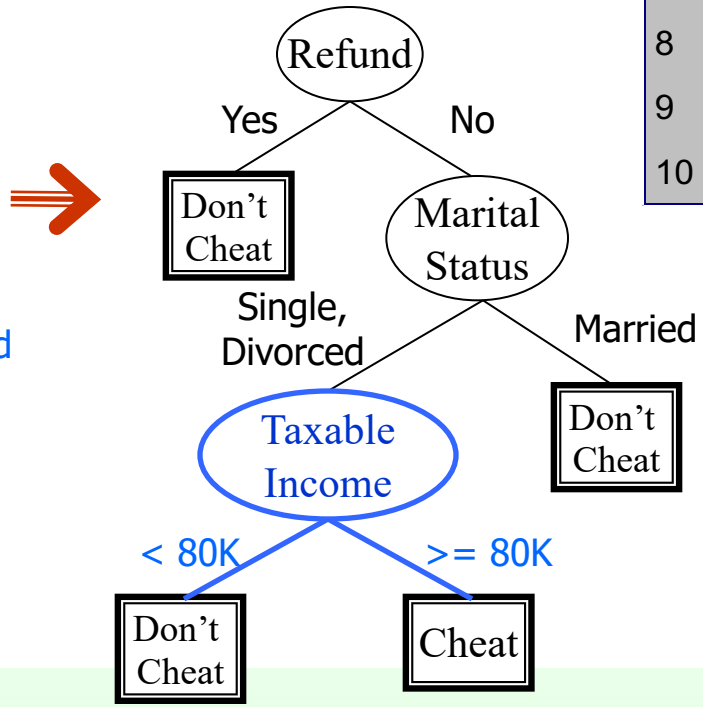
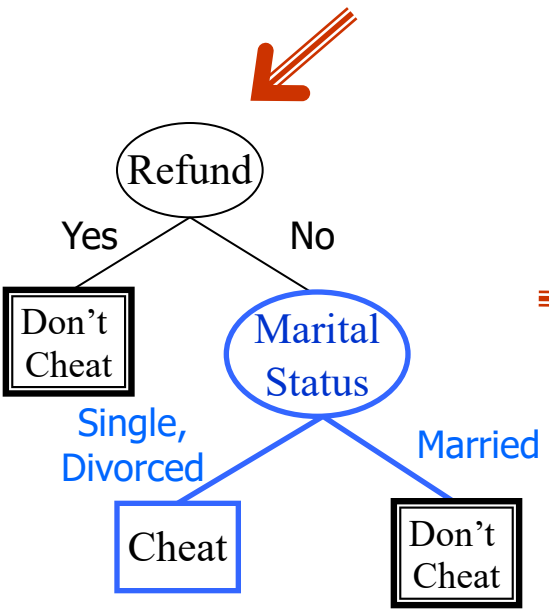
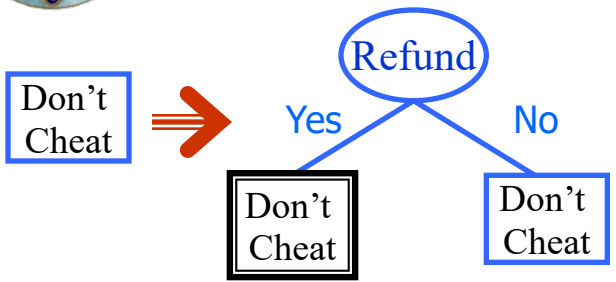
$D_t$ , set of training records that reach a node  $t$





# Hunt's algorithm

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes





# Decision tree induction

- Adopts a greedy strategy
  - “Best” attribute for the split is selected locally at each step
    - not a global optimum
- Issues
  - Structure of test condition
    - Binary split versus multiway split
  - Selection of the best attribute for the split
  - Stopping condition for the algorithm



# Structure of test condition

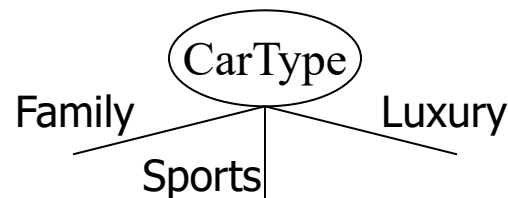
- Depends on attribute type
  - nominal
  - ordinal
  - continuous
- Depends on number of outgoing edges
  - 2-way split
  - multi-way split



# Splitting on nominal attributes

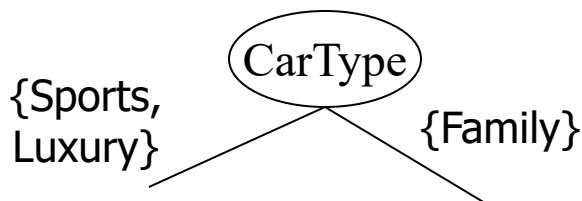
## ■ Multi-way split

- use as many partitions as distinct values

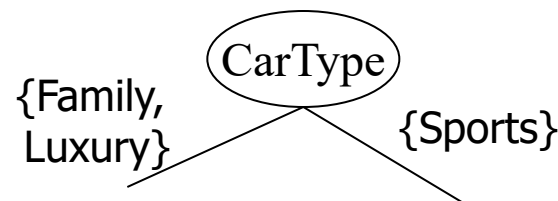


## ■ Binary split

- Divides values into two subsets
- Need to find optimal partitioning



OR





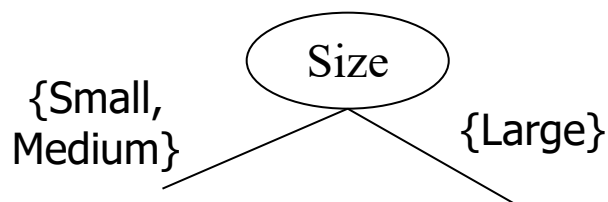
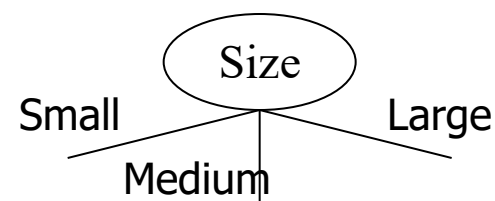
# Splitting on ordinal attributes

## ■ Multi-way split

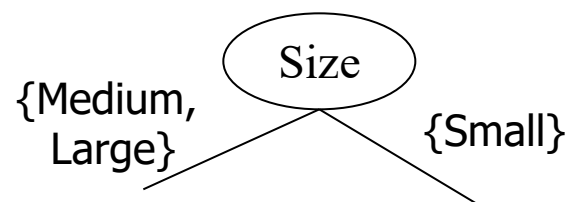
- use as many partitions as distinct values

## ■ Binary split

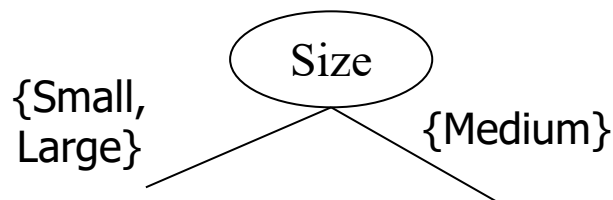
- Divides values into two subsets
- Need to find optimal partitioning



OR



What about this split?





# Splitting on continuous attributes

- Different techniques

- **Discretization** to form an ordinal categorical attribute

- Static – discretize once at the beginning
- Dynamic – discretize during tree induction

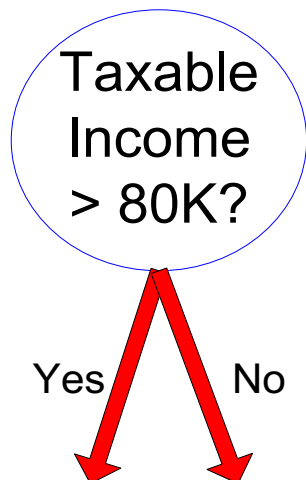
Ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering

- **Binary decision** ( $A < v$ ) or ( $A \geq v$ )

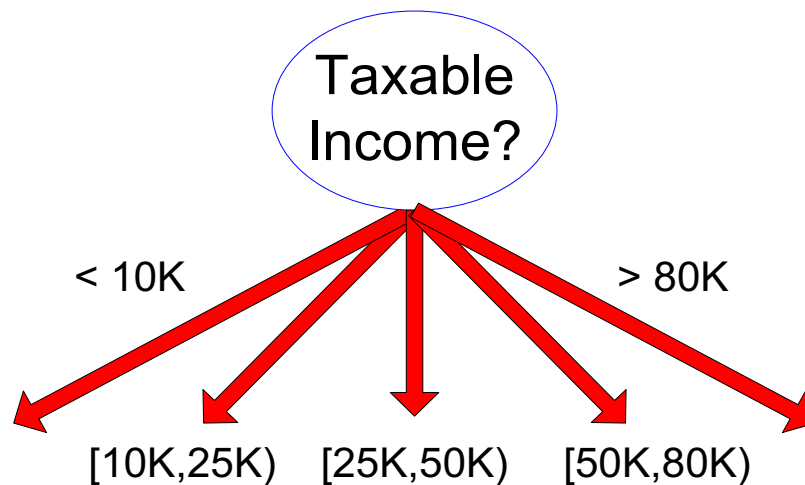
- consider all possible splits and find the best cut
- more computationally intensive



# Splitting on continuous attributes



(i) Binary split

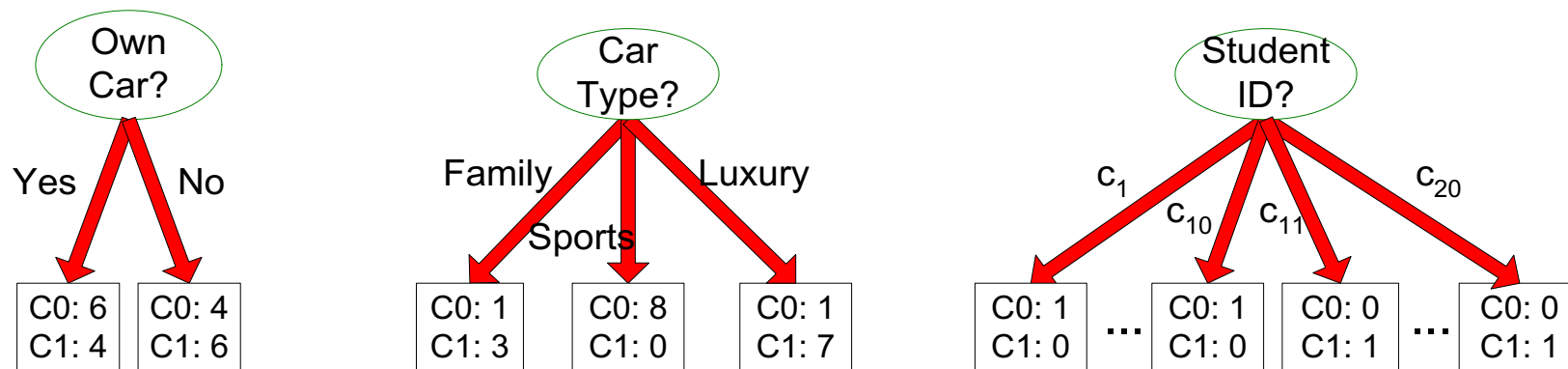


(ii) Multi-way split



# Selection of the best attribute

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



Which attribute (test condition) is the best?



# Selection of the best attribute

- Attributes with *homogeneous* class distribution are preferred
- Need a measure of node impurity

C0: 5  
C1: 5

Non-homogeneous,  
high degree of impurity

C0: 9  
C1: 1

Homogeneous, low  
degree of impurity



# Measures of node impurity

- Many different measures available
  - Gini index
  - Entropy
  - Misclassification error
- Different algorithms rely on different measures

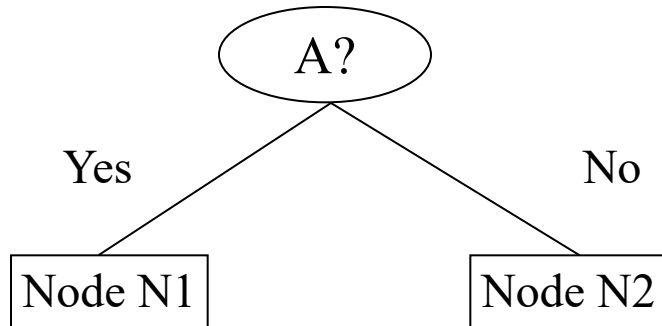


# How to find the best attribute

Before Splitting:

C0	<b>N00</b>
C1	<b>N01</b>

→ M0



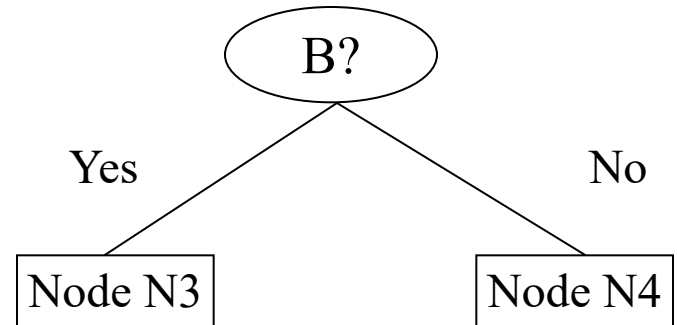
C0	<b>N10</b>
C1	<b>N11</b>

C0	<b>N20</b>
C1	<b>N21</b>

M1

M2

M12



C0	<b>N30</b>
C1	<b>N31</b>

C0	<b>N40</b>
C1	<b>N41</b>

M3

M4

M34

**Gain** = M0 – M12 vs M0 – M34



# GINI impurity measure

- Gini Index for a given node  $t$

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

$p(j | t)$  is the relative frequency of class  $j$  at node  $t$

- Maximum  $(1 - 1/n_c)$  when records are equally distributed among all classes, implying higher impurity degree
- Minimum (0.0) when all records belong to one class, implying lower impurity degree

C1	<b>0</b>
C2	<b>6</b>
<b>Gini=0.000</b>	

C1	<b>1</b>
C2	<b>5</b>
<b>Gini=0.278</b>	

C1	<b>2</b>
C2	<b>4</b>
<b>Gini=0.444</b>	

C1	<b>3</b>
C2	<b>3</b>
<b>Gini=0.500</b>	



# Examples for computing GINI

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

C1	<b>0</b>
C2	<b>6</b>

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$Gini = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$$

C1	<b>1</b>
C2	<b>5</b>

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$Gini = 1 - (1/6)^2 - (5/6)^2 = 0.278$$

C1	<b>2</b>
C2	<b>4</b>

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$Gini = 1 - (2/6)^2 - (4/6)^2 = 0.444$$



# Splitting based on GINI

- Used in CART, SLIQ, SPRINT
- When a node  $p$  is split into  $k$  partitions (children), the quality of the split is computed as

$$GINI_{split} = \sum_{i=1}^k \frac{n_i}{n} GINI(i)$$

where

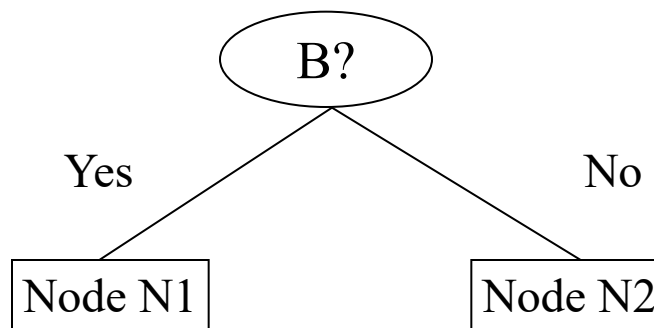
$n_i$  = number of records at child  $i$

$n$  = number of records at node  $p$



# Computing GINI index: Boolean attribute

- Splits into two partitions
  - larger and purer partitions are sought for



	Parent
C1	6
C2	6
<b>Gini = 0.500</b>	

$$\begin{aligned} \text{Gini}(N1) &= 1 - (5/7)^2 - (2/7)^2 \\ &= 0.408 \end{aligned}$$

$$\begin{aligned} \text{Gini}(N2) &= 1 - (1/5)^2 - (4/5)^2 \\ &= 0.32 \end{aligned}$$

	N1	N2
C1	5	1
C2	2	4
<b>Gini=?</b>		

$$\begin{aligned} \text{Gini}(\text{split on B}) &= 7/12 * 0.408 + \\ &\quad 5/12 * 0.32 \\ &= 0.371 \end{aligned}$$



# Computing GINI index: Categorical attribute

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

Two-way split  
(find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

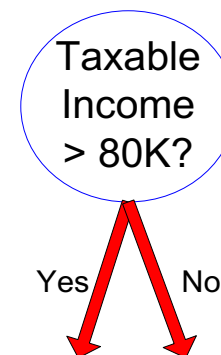
	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	



# Computing GINI index: Continuous attribute

- Binary decision on one splitting value
  - Number of possible splitting values  
= Number of distinct values
- Each splitting value  $v$  has a count matrix
  - class counts in the two partitions
    - $A < v$
    - $A \geq v$

<i>Tid</i>	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes





# Computing GINI index: Continuous attribute

- For each attribute
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index

Sorted Values  
Split Positions

Cheat	Taxable Income																	
	No		No		No		Yes		Yes		Yes		No		No		No	
	60		70		75		85		90		95		100		120		125	
	55		65		72		80		87		92		97		110		122	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2
Gini	0.420		0.400		0.375		0.343		0.417		0.400		<u>0.300</u>		0.343		0.375	



# Entropy impurity measure (INFO)

- Entropy at a given node  $t$

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

$p(j | t)$  is the relative frequency of class  $j$  at node  $t$

- Maximum ( $\log n_c$ ) when records are equally distributed among all classes, implying higher impurity degree
- Minimum (0.0) when all records belong to one class, implying lower impurity degree
- Entropy based computations are similar to GINI index computations



# Examples for computing entropy

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

C1	<b>0</b>
C2	<b>6</b>

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$Entropy = -0 \log 0 - 1 \log 1 = -0 - 0 = 0$$

C1	<b>1</b>
C2	<b>5</b>

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

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

C1	<b>2</b>
C2	<b>4</b>

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

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



# Splitting Based on INFO

- 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  $i$

- Measures reduction in entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
- Used in ID3 and C4.5
- Disadvantage: Tends to prefer splits yielding a large number of partitions, each small but pure



# Splitting Based on INFO

- Gain Ratio

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO}$$

$$SplitINFO = -\sum_{i=1}^k \frac{n_i}{n} \log \frac{n_i}{n}$$

Parent Node, p is split into k partitions

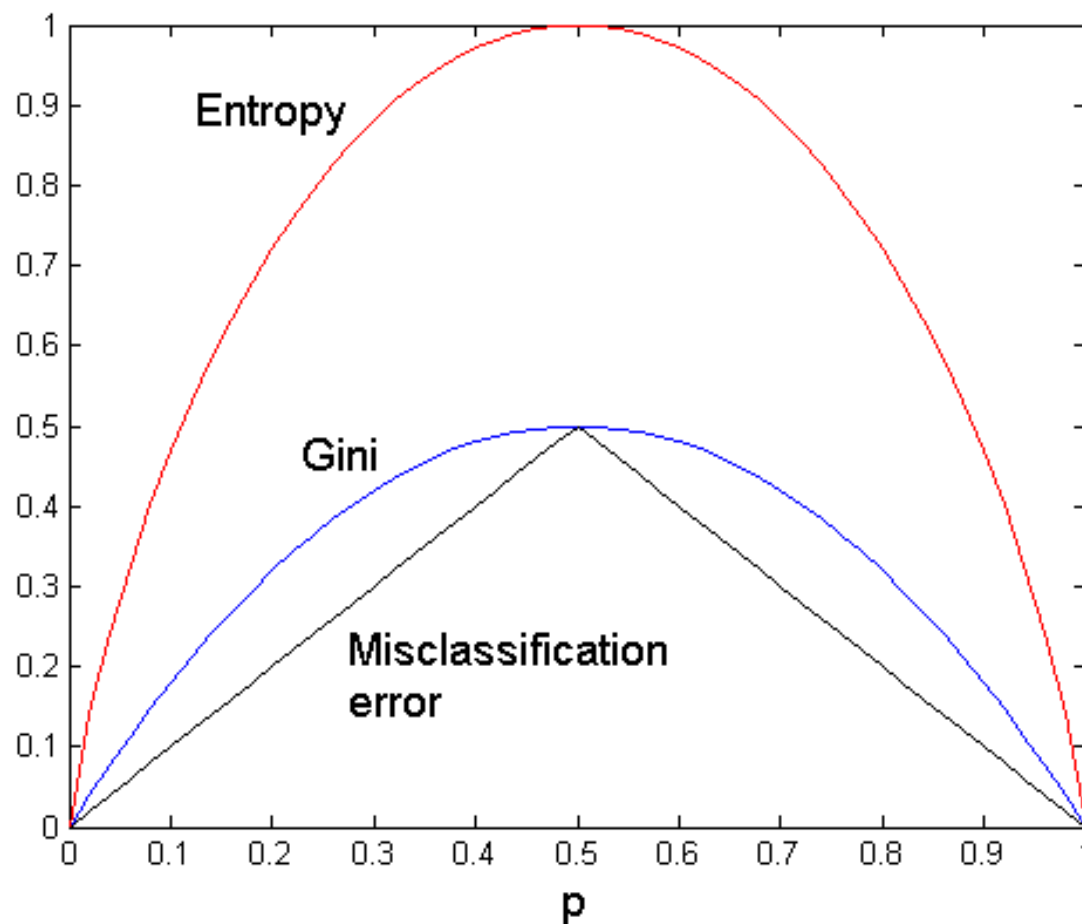
$n_i$  is the number of records in partition i

- Adjusts Information Gain by the entropy of the partitioning (SplitINFO). Higher entropy partitioning (large number of small partitions) is penalized
- Used in C4.5
- Designed to overcome the disadvantage of Information Gain



# Comparison among splitting criteria

For a 2-class problem



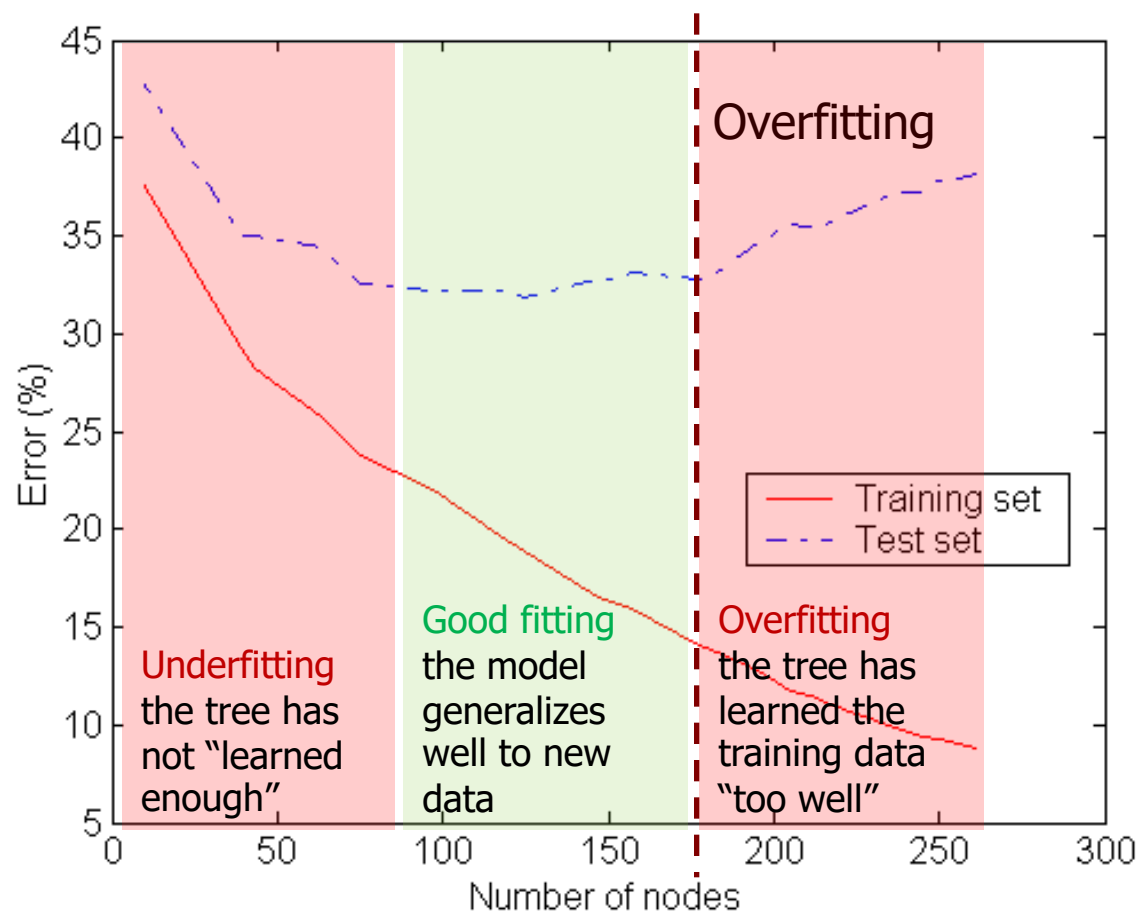


# Stopping Criteria for Tree Induction

- Stop expanding a node when all the records belong to the same class
- Stop expanding a node when all the records have similar attribute values
- Early termination
  - Pre-pruning
  - Post-pruning



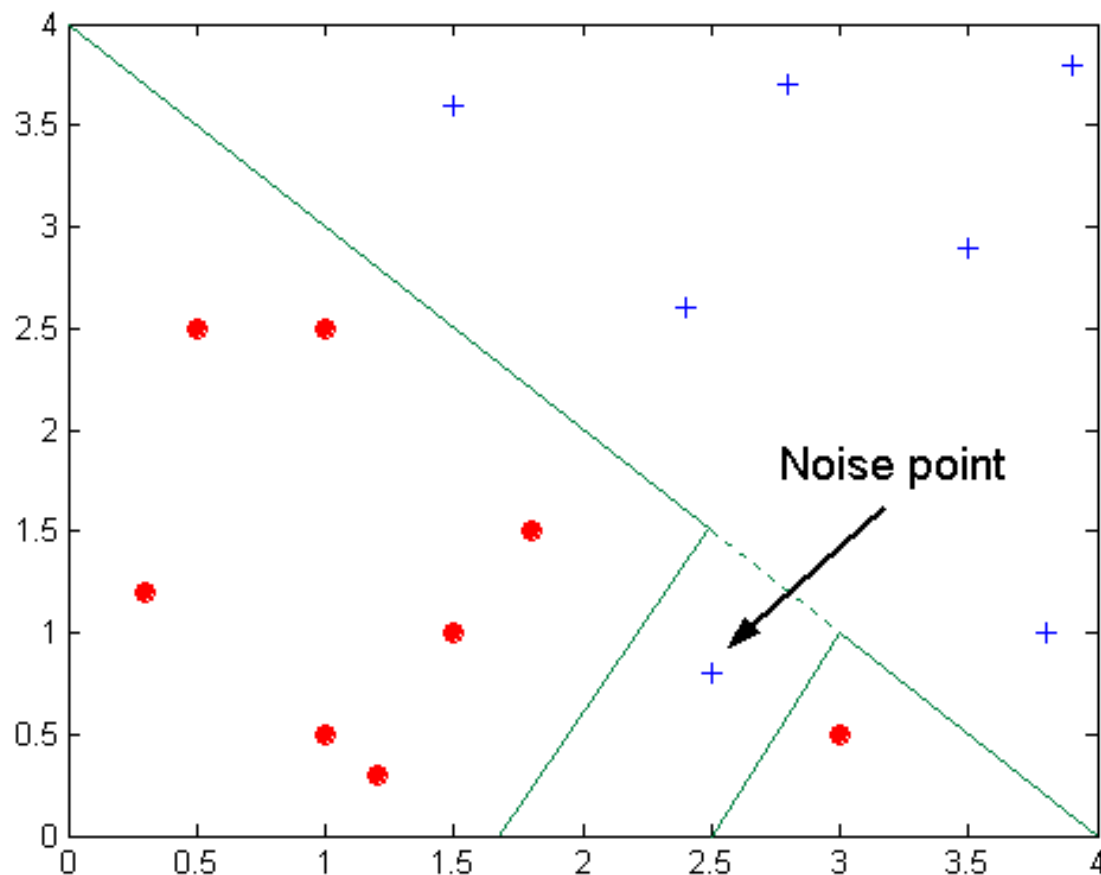
# Underfitting and Overfitting



Underfitting: when model is too simple, both training and test errors are large



# Overfitting due to Noise



Decision boundary is distorted by noise point



# How to address overfitting

- **Pre-Pruning (Early Stopping Rule)**
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same
  - More restrictive conditions
    - Stop if number of instances is less than some user-specified threshold
    - Stop if class distribution of instances are independent of the available features (e.g., using  $\chi^2$  test)
    - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain)



# How to address overfitting

## ■ Post-pruning

- Grow decision tree to its entirety
- Trim the nodes of the decision tree in a bottom-up fashion
- If generalization error improves after trimming, replace sub-tree by a leaf node.
- Class label of leaf node is determined from majority class of instances in the sub-tree



# Data fragmentation

- Number of instances gets smaller as you traverse down the tree
- Number of instances at the leaf nodes could be too small to make any statistically significant decision

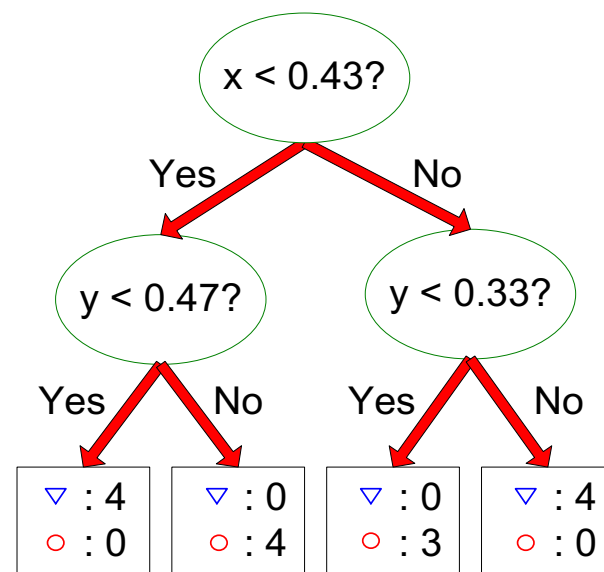
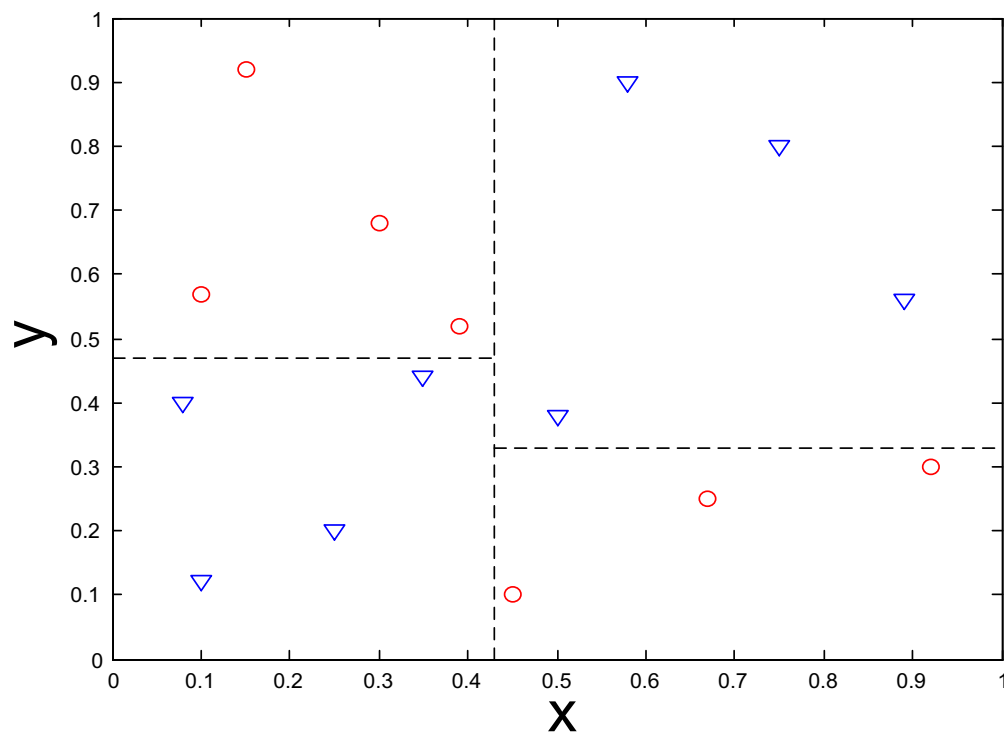


# Handling missing attribute values

- Missing values affect decision tree construction in three different ways
  - Affect how impurity measures are computed
  - Affect how to distribute instances with missing value to child nodes
  - Affect how a test instance with missing value is classified



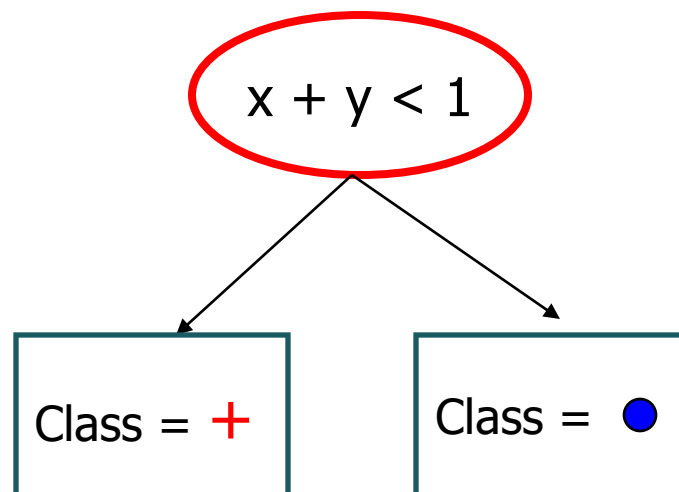
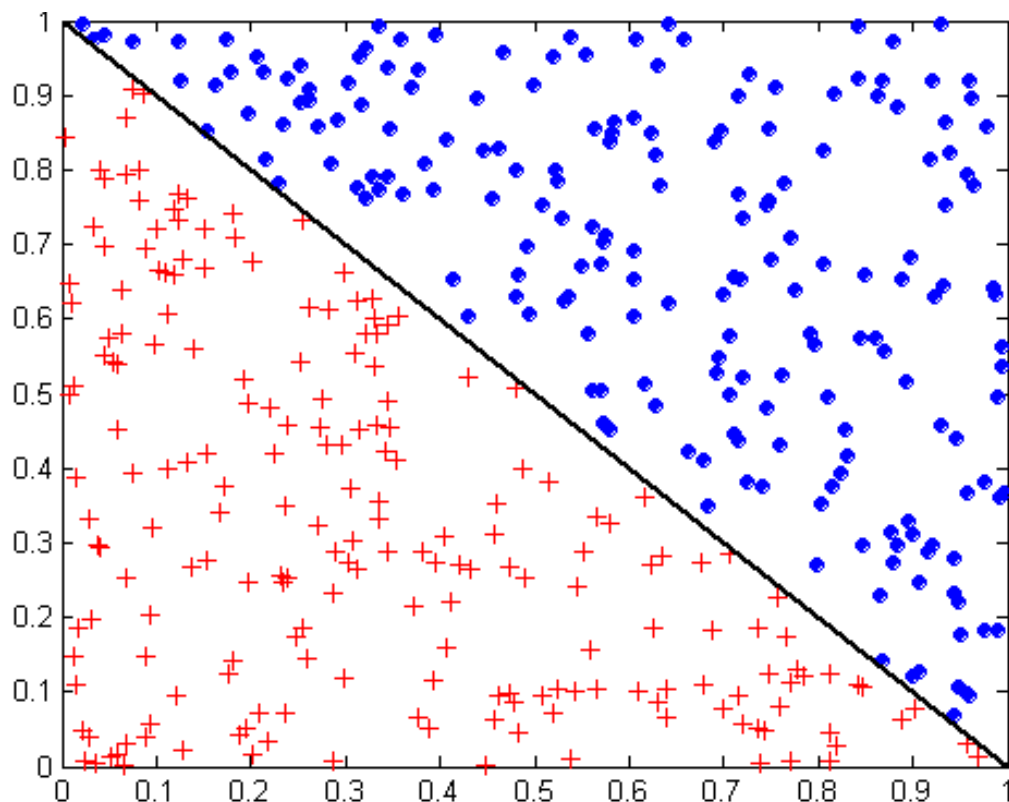
# Decision boundary



- Border line between two neighboring regions of different classes is known as decision boundary
- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time



# Oblique decision trees



- Test condition may involve multiple attributes
- More expressive representation
- Finding optimal test condition is computationally expensive



# Evaluation of decision trees

- Accuracy
  - For simple datasets, comparable to other classification techniques
- Interpretability
  - Model is interpretable for small trees
  - Single predictions are interpretable
- Incrementality
  - Not incremental
- Efficiency
  - Fast model building
  - Very fast classification
- Scalability
  - Scalable both in training set size and attribute number
- Robustness
  - Difficult management of missing data

# Random Forest



Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis  
*Politecnico di Torino*



# Random Forest

- Ensemble learning technique
  - multiple base models are combined
    - to improve accuracy and stability
    - to avoid overfitting
  
- Random forest = set of decision trees
  - a number of decision trees are built at training time
  - the class is assigned by majority voting



# Random Forest

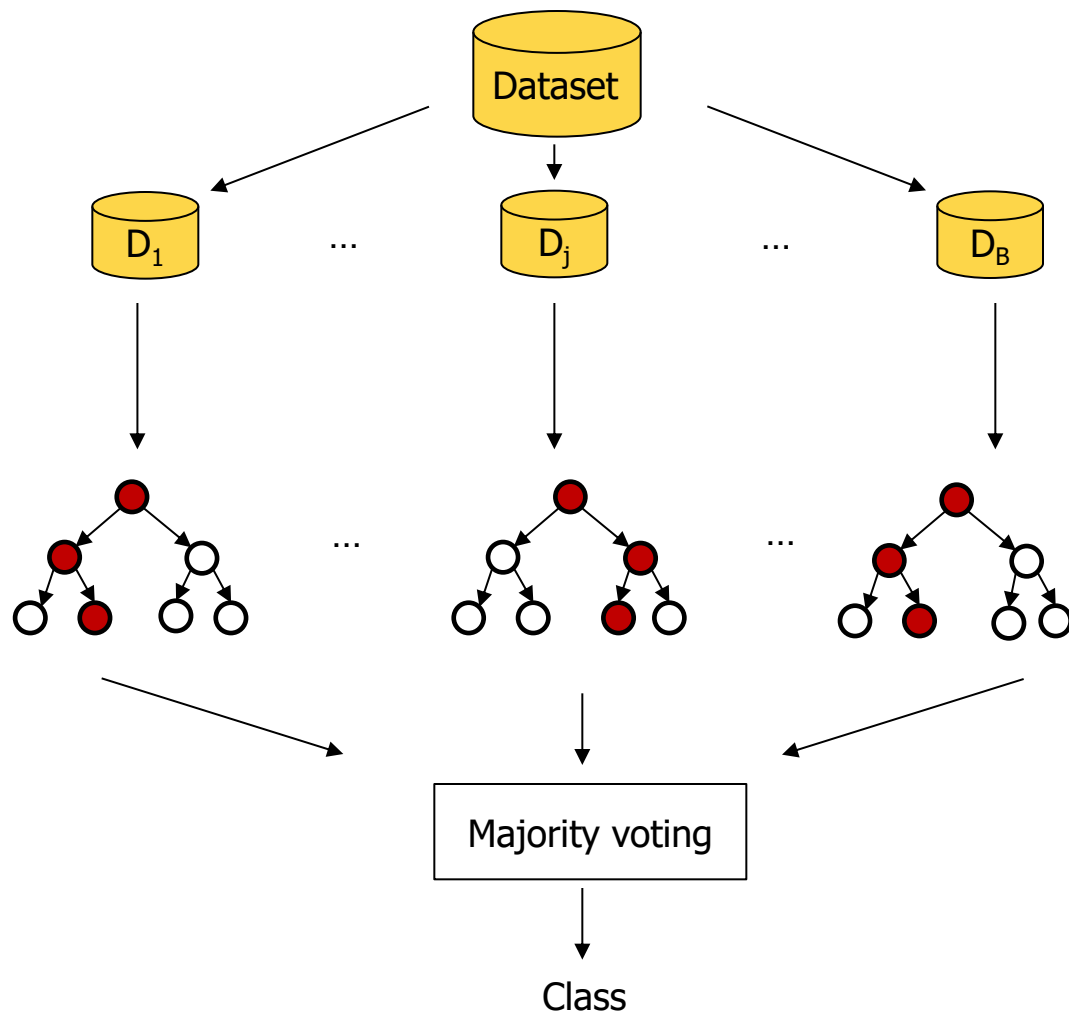
Original Training data

*Random* subsets

Multiple decision trees

For each subset, a tree is learned on a *random* set of features

Aggregating classifiers





# Bootstrap aggregation

- Given a training set  $D$  of  $n$  instances, it selects  $B$  times a *random* sample with replacement from  $D$  and trains trees on these dataset samples
  - For  $b = 1, \dots, B$ 
    - Sample with replacement  $n'$  training examples,  $n' \leq n$ 
      - A dataset subset  $D_b$  is generated
    - Train a classification tree on  $D_b$



# Feature bagging

- Selects, *for each candidate split* in the learning process, a *random* subset of the features
  - Being  $p$  the number of features,  $\sqrt{p}$  features are typically selected
- Trees are decorrelated
  - Feature subsets are sampled randomly, hence different features can be selected as best attributes for the split



# Random Forest – Algorithm Recap

- Given a training set  $D$  of  $n$  instances with  $p$  features
- For  $b = 1, \dots, B$ 
  - Sample randomly with replacement  $n'$  training examples. A subset  $D_b$  is generated
  - Train a classification tree on  $D_b$ 
    - During the tree construction, for each candidate split
      - $m \ll p$  random features are selected (typically  $m \approx \sqrt{p}$ )
      - the best split is computed among these  $m$  features
- Class is assigned by majority voting among the  $B$  predictions



# Evaluation of random forests

- Accuracy
  - Higher than decision trees
- Interpretability
  - Model and prediction are not interpretable
    - A prediction may be given by hundreds of trees
  - Provide global feature importance
    - an estimate of which features are important in the classification
- Incrementality
  - Not incremental
- Efficiency
  - Fast model building
  - Very fast classification
- Scalability
  - Scalable both in training set size and attribute number
- Robustness
  - Robust to noise and outliers

# Rule-based classification



Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis  
*Politecnico di Torino*



# Rule-based classifier

- Classify records by using a collection of “if...then...” rules
- Rule:  $(Condition) \rightarrow y$ 
  - where
    - *Condition* is a conjunction of simple predicates
    - $y$  is the class label
  - *LHS*: rule antecedent or condition
  - *RHS*: rule consequent
- Examples of classification rules
  - $(\text{Blood Type}=\text{Warm}) \wedge (\text{Lay Eggs}=\text{Yes}) \rightarrow \text{Birds}$
  - $(\text{Taxable Income} < 50\text{K}) \wedge (\text{Refund}=\text{Yes}) \rightarrow \text{Cheat}=\text{No}$



# Rule-based Classifier (Example)

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

R1: (Give Birth = no)  $\wedge$  (Can Fly = yes)  $\rightarrow$  Birds

R2: (Give Birth = no)  $\wedge$  (Live in Water = yes)  $\rightarrow$  Fishes

R3: (Give Birth = yes)  $\wedge$  (Blood Type = warm)  $\rightarrow$  Mammals

R4: (Give Birth = no)  $\wedge$  (Can Fly = no)  $\rightarrow$  Reptiles

R5: (Live in Water = sometimes)  $\rightarrow$  Amphibians



# Rule-based classification

- A rule  $r$  **covers** an instance  $x$  if the attributes of the instance satisfy the condition of the rule

R1: (Give Birth = no)  $\wedge$  (Can Fly = yes)  $\rightarrow$  Birds

R2: (Give Birth = no)  $\wedge$  (Live in Water = yes)  $\rightarrow$  Fishes

R3: (Give Birth = yes)  $\wedge$  (Blood Type = warm)  $\rightarrow$  Mammals

R4: (Give Birth = no)  $\wedge$  (Can Fly = no)  $\rightarrow$  Reptiles

R5: (Live in Water = sometimes)  $\rightarrow$  Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

Rule R1 covers a hawk  $\Rightarrow$  Bird

Rule R3 covers the grizzly bear  $\Rightarrow$  Mammal



# Rule-based classification

R1: (Give Birth = no)  $\wedge$  (Can Fly = yes)  $\rightarrow$  Birds

R2: (Give Birth = no)  $\wedge$  (Live in Water = yes)  $\rightarrow$  Fishes

R3: (Give Birth = yes)  $\wedge$  (Blood Type = warm)  $\rightarrow$  Mammals

R4: (Give Birth = no)  $\wedge$  (Can Fly = no)  $\rightarrow$  Reptiles

R5: (Live in Water = sometimes)  $\rightarrow$  Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

A lemur triggers (only) rule R3, so it is classified as a mammal

A turtle triggers both R4 and R5

A dogfish shark triggers none of the rules



# Characteristics of rules

## ■ *Mutually exclusive* rules

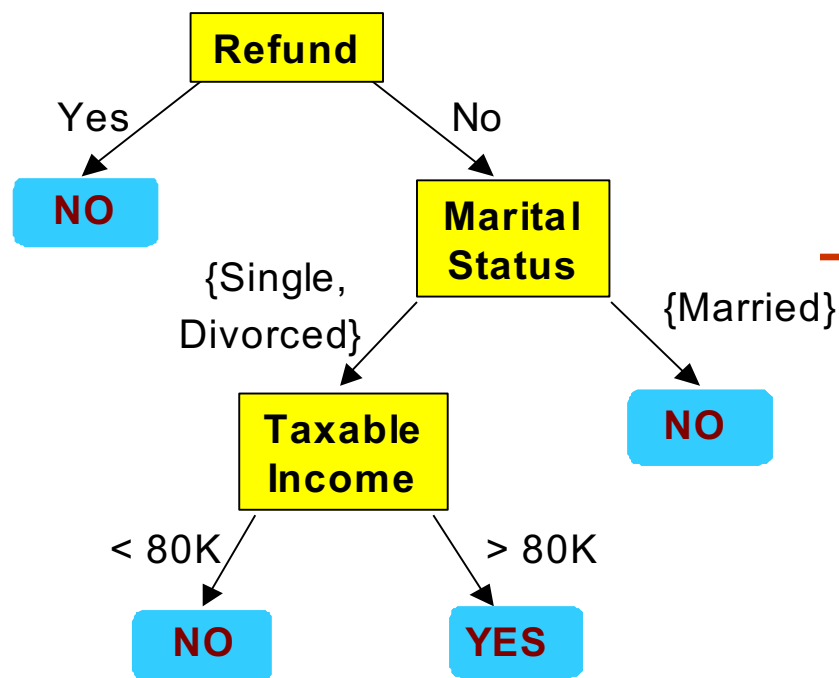
- Two rule conditions can't be true at the same time
- Every record is covered by at most one rule

## ■ *Exhaustive* rules

- Classifier rules account for every possible combination of attribute values
- Each record is covered by at least one rule



# From decision trees to rules



## Classification Rules

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single, Divorced}, Taxable Income<80K) ==> No

(Refund=No, Marital Status={Single, Divorced}, Taxable Income>80K) ==> Yes

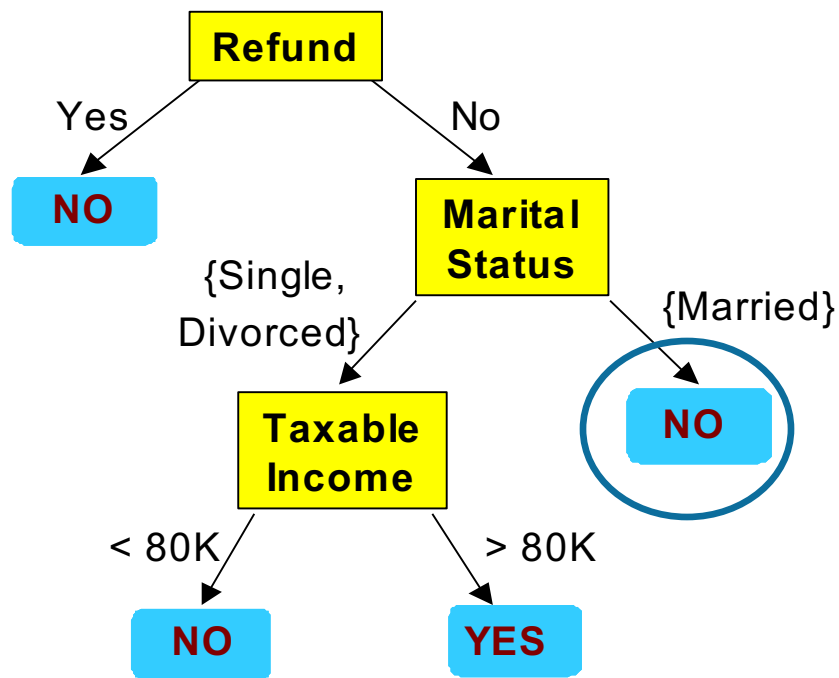
(Refund=No, Marital Status={Married}) ==> No

Rules are mutually exclusive and exhaustive

Rule set contains as much information as the tree



# Rules can be simplified



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Initial Rule:  $(\text{Refund}=\text{No}) \wedge (\text{Status}=\text{Married}) \rightarrow \text{No}$

Simplified Rule:  $(\text{Status}=\text{Married}) \rightarrow \text{No}$



# Effect of rule simplification

- Rules are no longer mutually exclusive
  - A record may trigger more than one rule
  - Solution?
    - Ordered rule set
    - Unordered rule set – use voting schemes
- Rules are no longer exhaustive
  - A record may not trigger any rules
  - Solution?
    - Use a default class



# Ordered rule set

- Rules are rank ordered according to their priority
  - An ordered rule set is known as a decision list
- When a test record is presented to the classifier
  - It is assigned to the class label of the highest ranked rule it has triggered
  - If none of the rules fired, it is assigned to the default class

R1: (Give Birth = no)  $\wedge$  (Can Fly = yes)  $\rightarrow$  Birds

R2: (Give Birth = no)  $\wedge$  (Live in Water = yes)  $\rightarrow$  Fishes

R3: (Give Birth = yes)  $\wedge$  (Blood Type = warm)  $\rightarrow$  Mammals

R4: (Give Birth = no)  $\wedge$  (Can Fly = no)  $\rightarrow$  Reptiles

R5: (Live in Water = sometimes)  $\rightarrow$  Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
turtle	cold	no	no	sometimes	?



# Building classification rules

## ■ Direct Method

- Extract rules directly from data
- e.g.: RIPPER, CN2, Holte's 1R

## ■ Indirect Method

- Extract rules from other classification models (e.g. decision trees, neural networks, etc).
- e.g: C4.5rules



# Evaluation of rule based classifiers

- Accuracy
  - Higher than decision trees
- Interpretability
  - Model and prediction are interpretable
- Incrementality
  - Not incremental
- Efficiency
  - Fast model building
  - Very fast classification
- Scalability
  - Scalable both in training set size and attribute number
- Robustness
  - Robust to outliers

# Associative classification



Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis

*Politecnico di Torino*



# Associative classification

- The classification model is defined by means of association rules

$$(\textit{Condition}) \rightarrow y$$

- rule body is an itemset
- Model generation
  - Rule selection & sorting
    - based on support, confidence and correlation thresholds
  - Rule pruning
    - Database coverage: the training set is covered by selecting topmost rules according to previous sort



# Evaluation of associative classifiers

- Accuracy
  - Higher than decision trees and rule-based classifiers
    - *correlation* among attributes is considered
- Interpretability
  - Model and prediction are interpretable
- Incrementality
  - Not incremental
- Efficiency
  - Rule generation may be slow
    - It depends on support threshold
  - Very fast classification
- Scalability
  - Scalable in training set size
  - Reduced scalability in attribute number
    - Rule generation may become unfeasible
- Robustness
  - Unaffected by missing data
  - Robust to outliers

# K-Nearest Neighbor



Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis  
*Politecnico di Torino*



# Instance-Based Classifiers

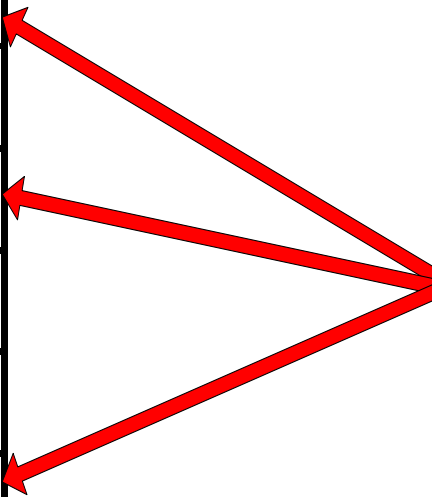
## Set of Stored Cases

Atr1	.....	AtrN	Class
			A
			B
			B
			C
			A
			C
			B

- Store the training records
- Use training records to predict the class label of unseen cases

## Unseen Case

Atr1	.....	AtrN





# Instance Based Classifiers

## ■ Examples

### ■ Rote-learner

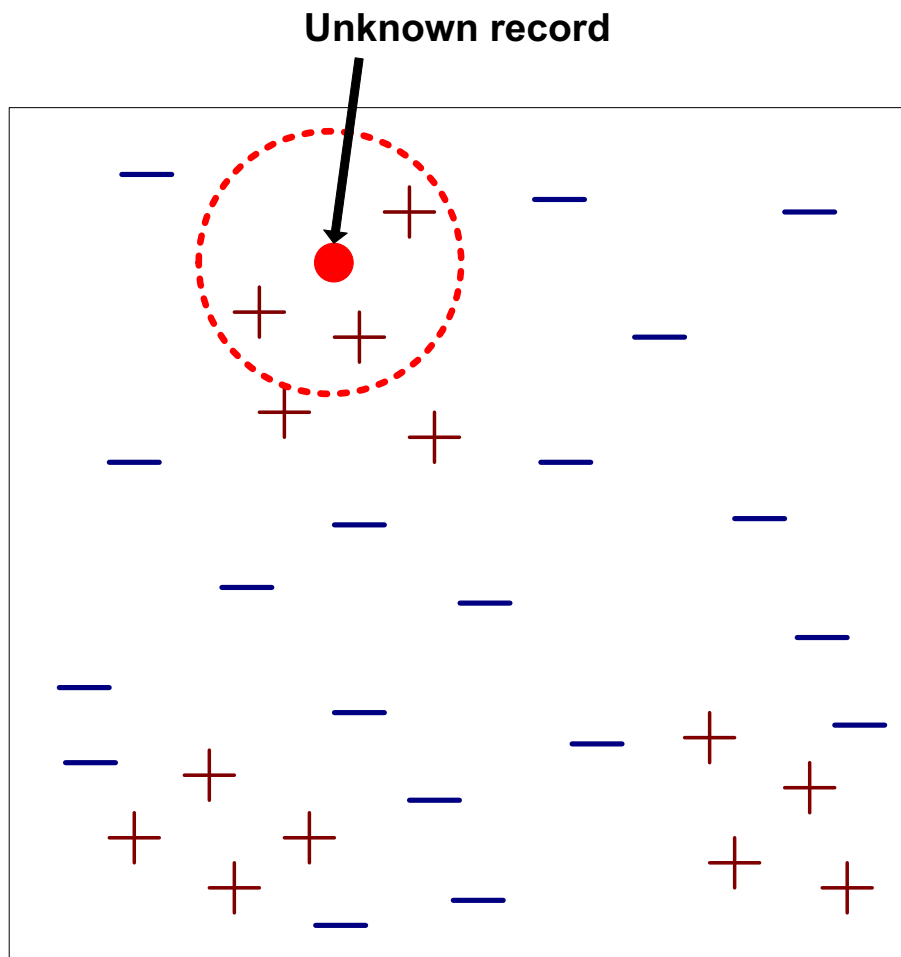
- Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly

### ■ Nearest neighbor

- Uses  $k$  “closest” points (nearest neighbors) for performing classification



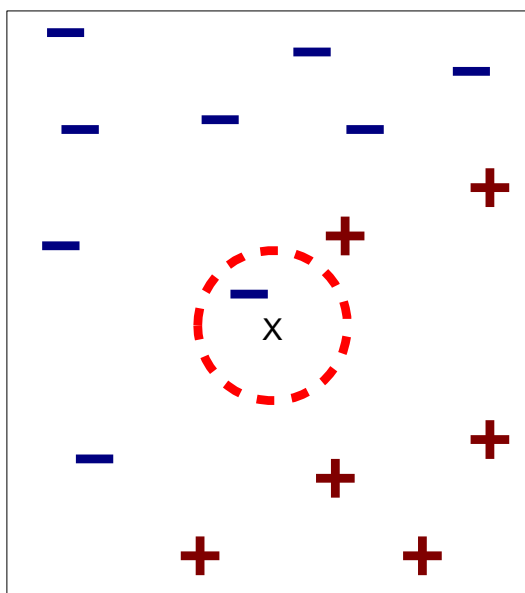
# Nearest-Neighbor Classifiers



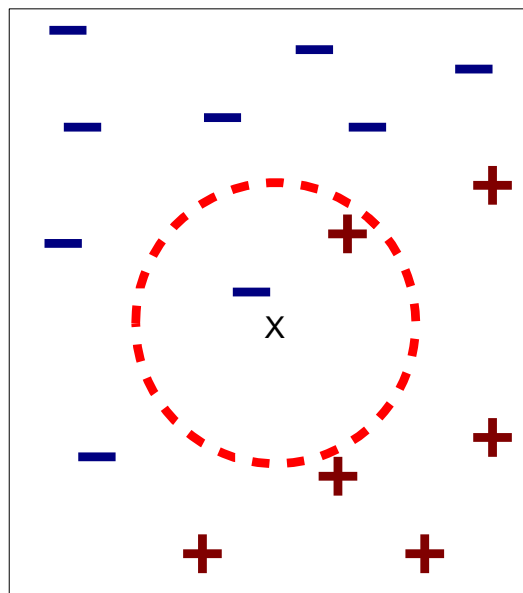
- Requires
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of  $k$ , the number of nearest neighbors to retrieve
- To classify an unknown record
  - Compute distance to other training records
  - Identify  $k$  nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)



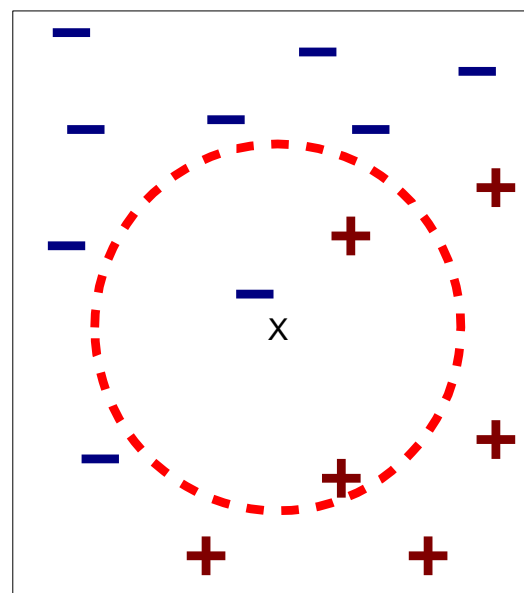
# Definition of Nearest Neighbor



(a) 1-nearest neighbor



(b) 2-nearest neighbor



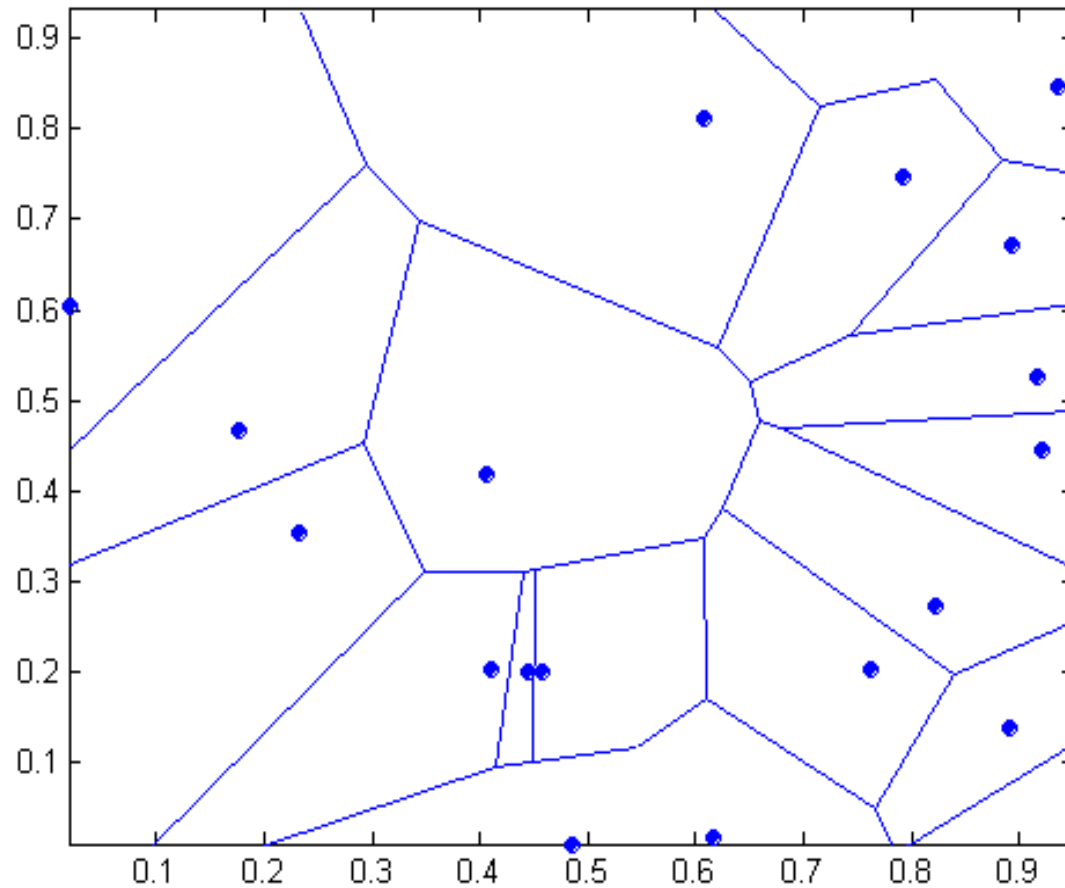
(c) 3-nearest neighbor

K-nearest neighbors of a record  $x$  are data points that have the  $k$  smallest distance to  $x$



# 1 nearest-neighbor

## Voronoi Diagram





# Nearest Neighbor Classification

- Compute distance between two points
  - Euclidean distance

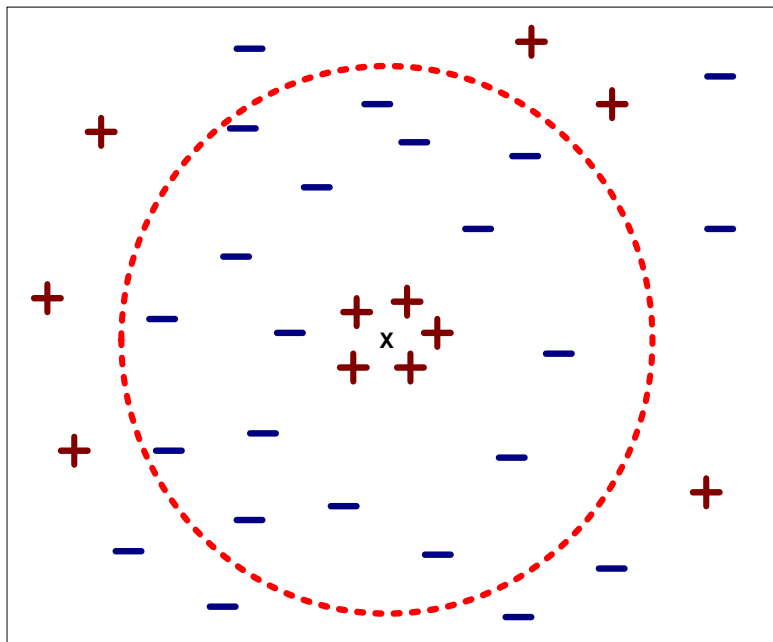
$$d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor,  $w = 1/d^2$



# Nearest Neighbor Classification

- Choosing the value of  $k$ :
  - If  $k$  is too small, sensitive to noise points
  - If  $k$  is too large, neighborhood may include points from other classes





# Nearest Neighbor Classification

## ■ Scaling issues

- Attribute domain should be normalized to prevent distance measures from being dominated by one of the attributes
- Example: height [1.5m to 2.0m] vs. income [\$10K to \$1M]

## ■ Problem with distance measures

- High dimensional data
  - curse of dimensionality



# Evaluation of KNN

## ■ Accuracy

- Comparable to other classification techniques for simple datasets

## ■ Interpretability

- Model is not interpretable
- Single predictions can be "described" by neighbors

## ■ Incrementality

- Incremental
- Training set *must* be available

## ■ Efficiency

- (Almost) no model building
- Slower classification, requires computing distances

## ■ Scalability

- Weakly scalable in training set size
- Curse of dimensionality for increasing attribute number

## ■ Robustness

- Depends on distance computation

# Bayesian Classification



Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis  
*Politecnico di Torino*



# Bayes theorem

- Let  $C$  and  $X$  be random variables

$$P(C, X) = P(C|X) P(X)$$

$$P(C, X) = P(X|C) P(C)$$

- Hence

$$P(C|X) P(X) = P(X|C) P(C)$$

- and also

$$P(C|X) = P(X|C) P(C) / P(X)$$



# Bayesian classification

- Let the class attribute and all data attributes be random variables
  - $C$  = any class label
  - $X = \langle x_1, \dots, x_k \rangle$  record to be classified
- Bayesian classification
  - compute  $P(C|X)$  for all classes
    - probability that record  $X$  belongs to  $C$
  - assign  $X$  to the class with *maximal*  $P(C|X)$
- Applying Bayes theorem

$$P(C|X) = P(X|C) \cdot P(C) / P(X)$$

- $P(X)$  constant for all  $C$ , disregarded for maximum computation
- $P(C)$  a priori probability of  $C$

$$P(C) = N_c / N$$



# Bayesian classification

- How to estimate  $P(X|C)$ , i.e.  $P(x_1, \dots, x_k|C)$ ?

- Naïve hypothesis

$$P(x_1, \dots, x_k|C) = P(x_1|C) P(x_2|C) \dots P(x_k|C)$$

- *statistical independence* of attributes  $x_1, \dots, x_k$
- not always true
  - model quality may be affected

- Computing  $P(x_k|C)$

- for discrete attributes

$$P(x_k|C) = |x_{kC}| / N_c$$

- where  $|x_{kC}|$  is number of instances having value  $x_k$  for attribute  $k$  and belonging to class  $C$
- for continuous attributes, use probability distribution

- Bayesian networks

- allow specifying a subset of dependencies among attributes



# Bayesian classification: Example

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N



# Bayesian classification: Example

outlook	
$P(\text{sunny} p) = 2/9$	$P(\text{sunny} n) = 3/5$
$P(\text{overcast} p) = 4/9$	$P(\text{overcast} n) = 0$
$P(\text{rain} p) = 3/9$	$P(\text{rain} n) = 2/5$
temperature	
$P(\text{hot} p) = 2/9$	$P(\text{hot} n) = 2/5$
$P(\text{mild} p) = 4/9$	$P(\text{mild} n) = 2/5$
$P(\text{cool} p) = 3/9$	$P(\text{cool} n) = 1/5$
humidity	
$P(\text{high} p) = 3/9$	$P(\text{high} n) = 4/5$
$P(\text{normal} p) = 6/9$	$P(\text{normal} n) = 2/5$
windy	
$P(\text{true} p) = 3/9$	$P(\text{true} n) = 3/5$
$P(\text{false} p) = 6/9$	$P(\text{false} n) = 2/5$

$$P(p) = 9/14$$

$$P(n) = 5/14$$



# Bayesian classification: Example

- Data to be labeled

$X = \langle \text{rain}, \text{hot}, \text{high}, \text{false} \rangle$

- For class p

$$\begin{aligned} P(X|p) \cdot P(p) &= \\ &= P(\text{rain}|p) \cdot P(\text{hot}|p) \cdot P(\text{high}|p) \cdot P(\text{false}|p) \cdot P(p) \\ &= 3/9 \cdot 2/9 \cdot 3/9 \cdot 6/9 \cdot 9/14 = 0.010582 \end{aligned}$$

- For class n

$$\begin{aligned} P(X|n) \cdot P(n) &= \\ &= P(\text{rain}|n) \cdot P(\text{hot}|n) \cdot P(\text{high}|n) \cdot P(\text{false}|n) \cdot P(n) \\ &= 2/5 \cdot 2/5 \cdot 4/5 \cdot 2/5 \cdot 5/14 = \mathbf{0.018286} \end{aligned}$$



# Evaluation of Naïve Bayes Classifiers

- Accuracy
  - Similar or lower than decision trees
    - Naïve hypothesis simplifies model
- Interpretability
  - Model and prediction are not interpretable
    - The weights of contributions in a single prediction may be used to explain
- Incrementality
  - Fully incremental
  - Does *not* require availability of training data
- Efficiency
  - Fast model building
  - Very fast classification
- Scalability
  - Scalable both in training set size and attribute number
- Robustness
  - Affected by attribute correlation

# Support Vector Machines

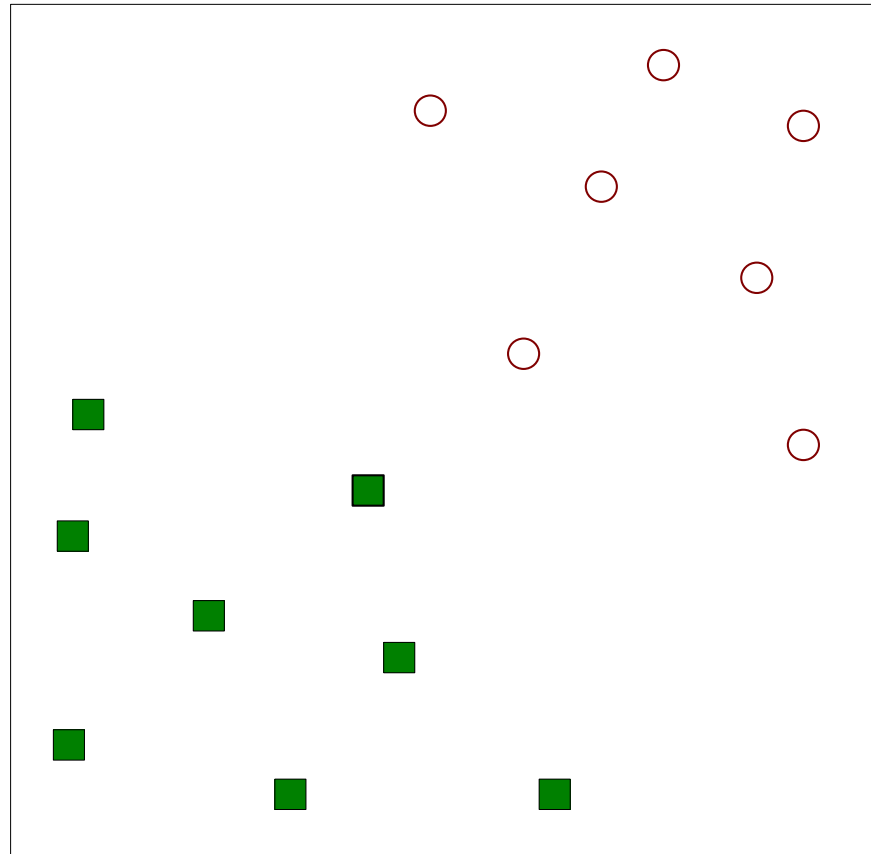


Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis  
*Politecnico di Torino*



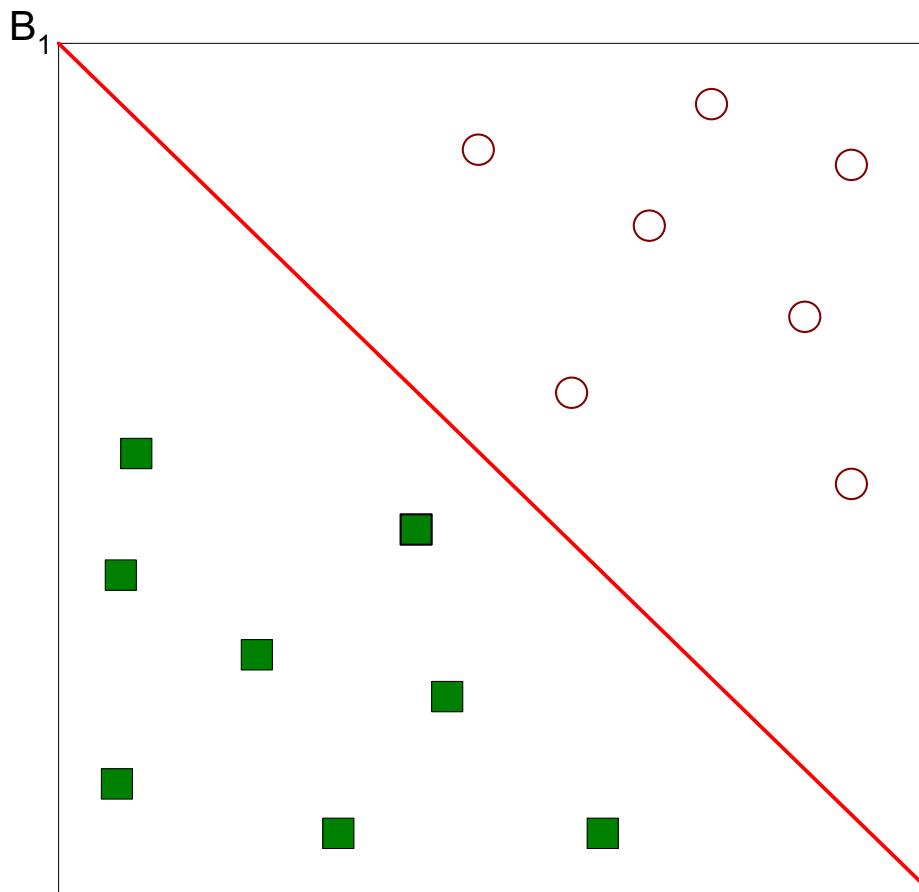
# Support Vector Machines



- Find a linear hyperplane (decision boundary) that will separate the data



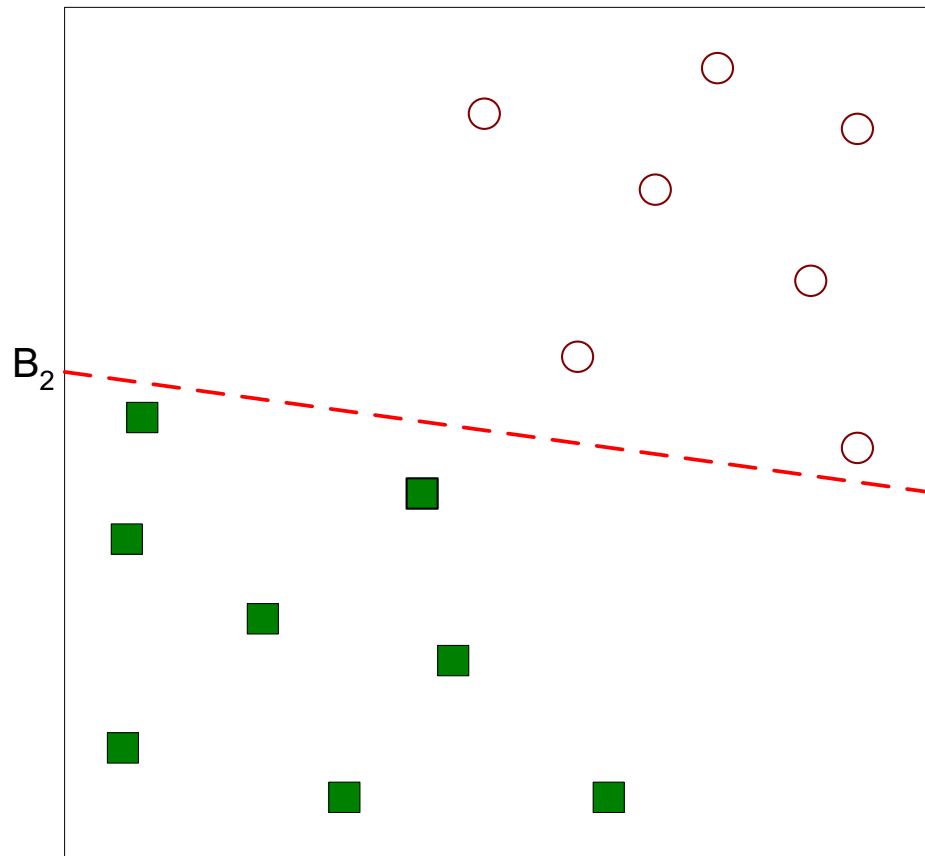
# Support Vector Machines



- One Possible Solution



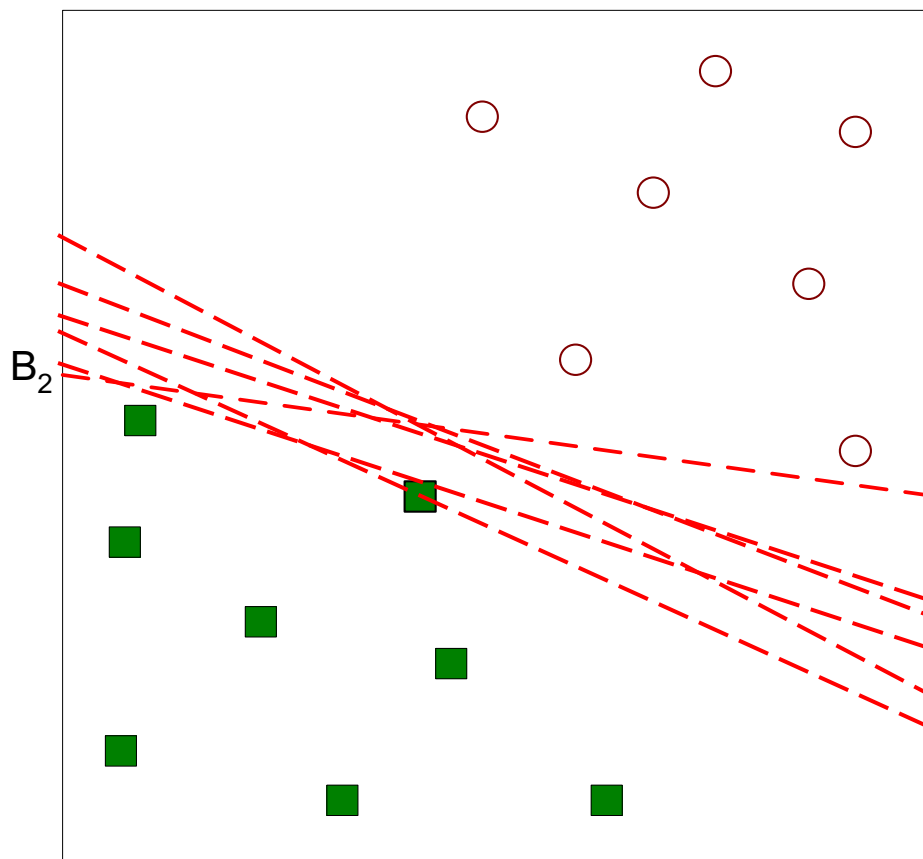
# Support Vector Machines



- Another possible solution



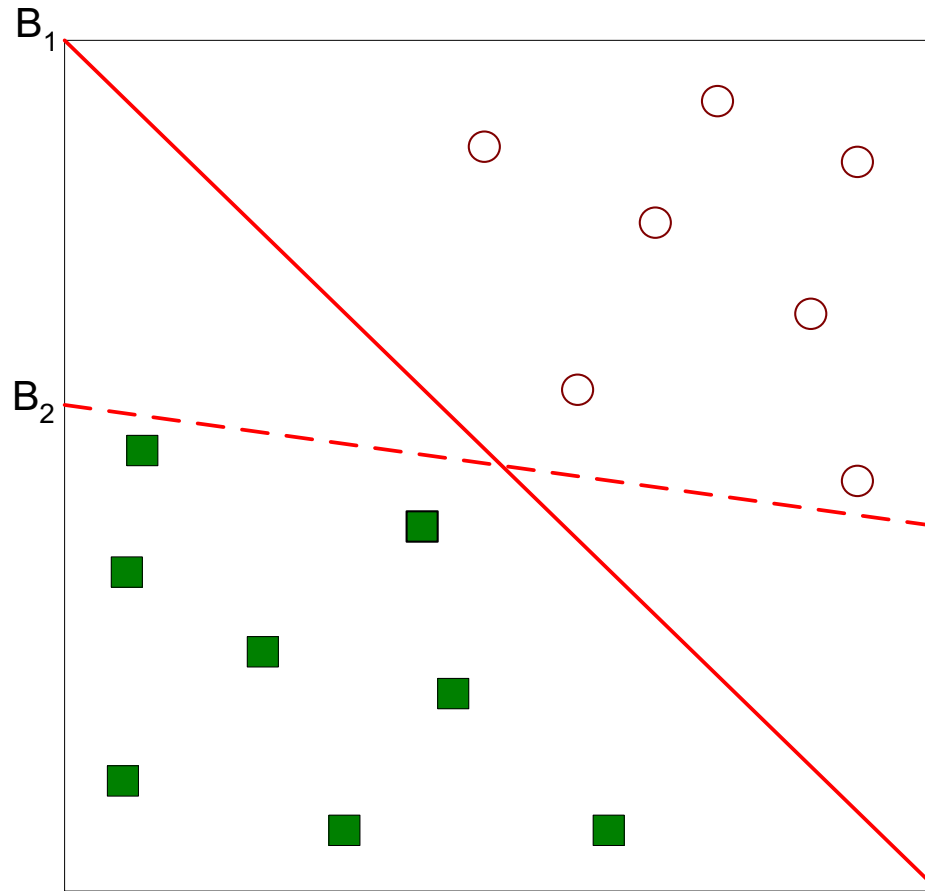
# Support Vector Machines



- Other possible solutions



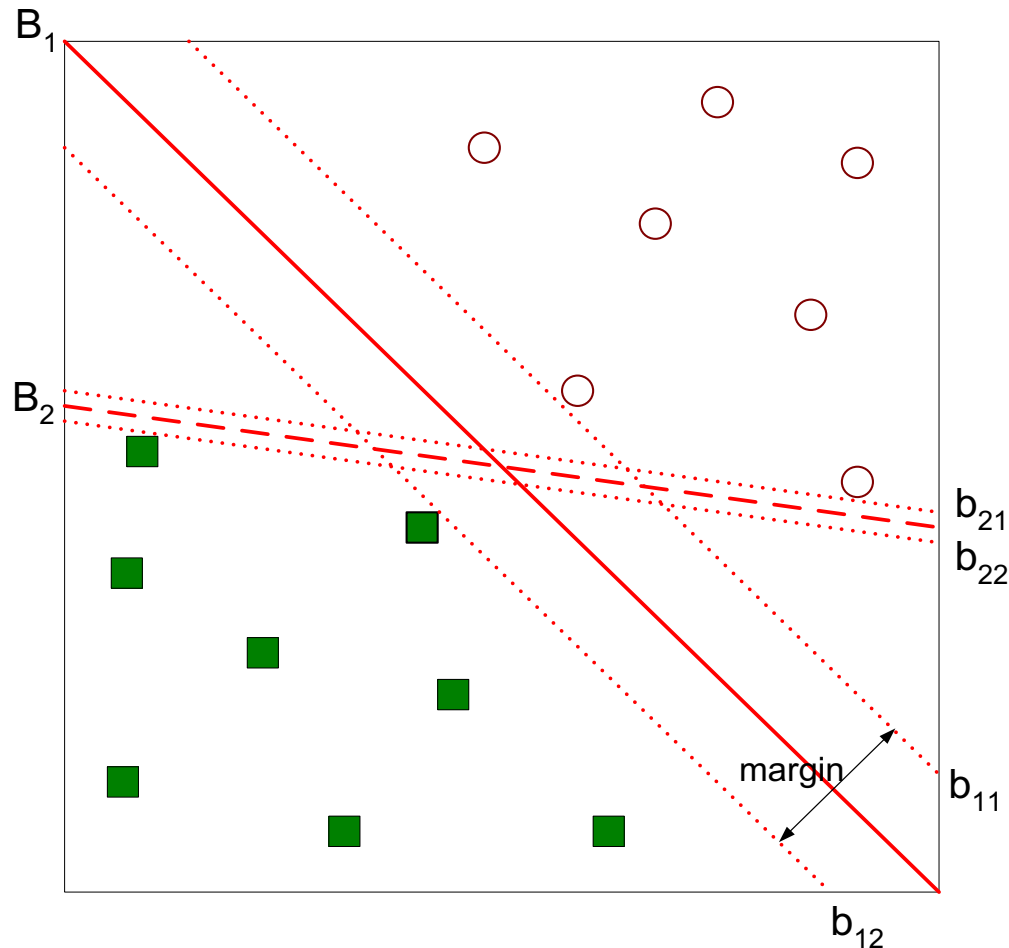
# Support Vector Machines



- Which one is better?  $B_1$  or  $B_2$ ?
- How do you define better?



# Support Vector Machines

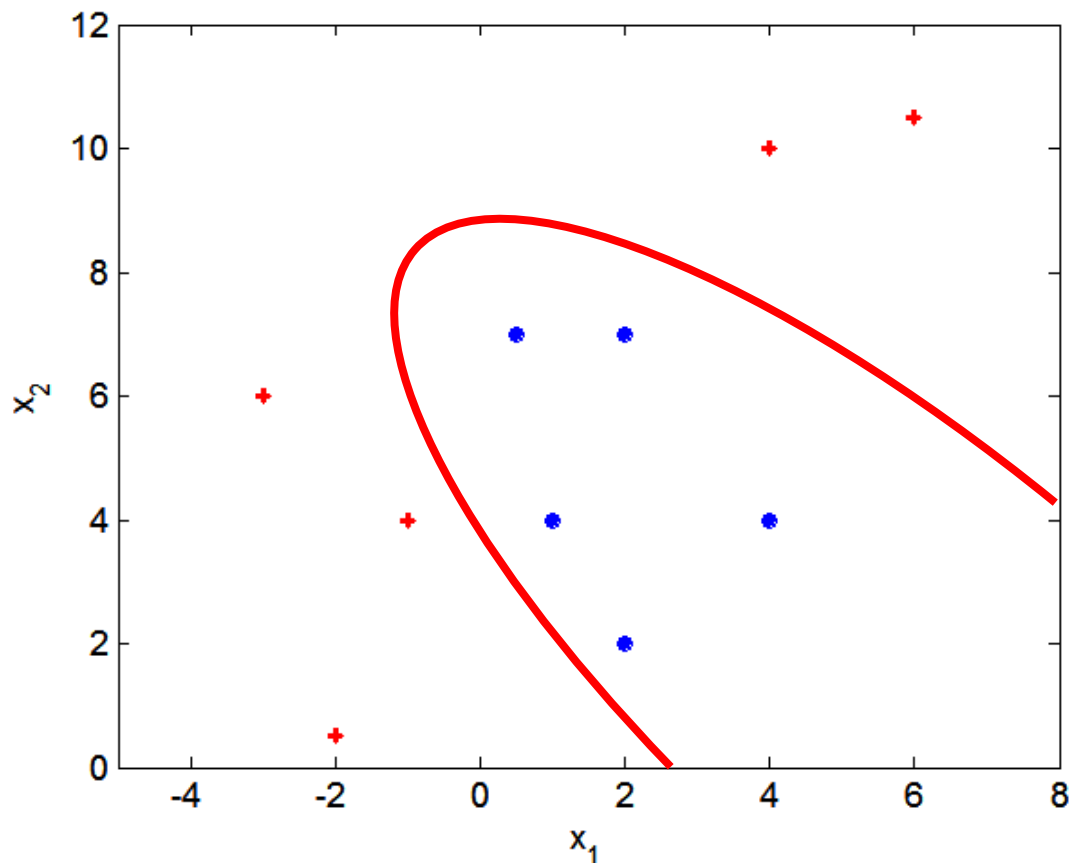


- Find hyperplane **maximizes** the margin  $\Rightarrow$  B1 is better than B2



# Nonlinear Support Vector Machines

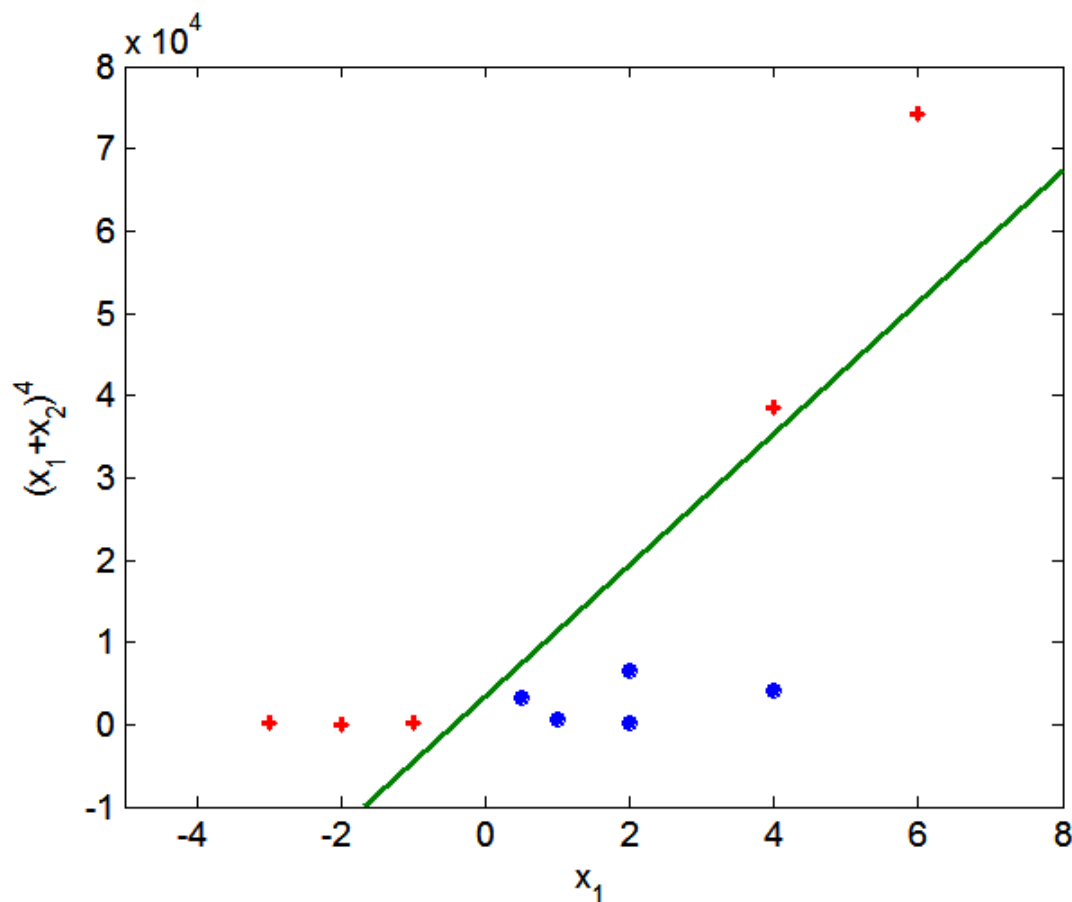
- What if decision boundary is not linear?





# Nonlinear Support Vector Machines

- Transform data into higher dimensional space





# Evaluation of Support Vector Machines

- Accuracy
  - Among best performers
- Interpretability
  - Model and prediction are not interpretable
    - Black box model
- Incrementality
  - Not incremental
- Efficiency
  - Model building requires significant parameter tuning
  - Very fast classification
- Scalability
  - Medium scalable both in training set size and attribute number
- Robustness
  - Robust to noise and outliers

# Artificial Neural Networks



Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis  
*Politecnico di Torino*

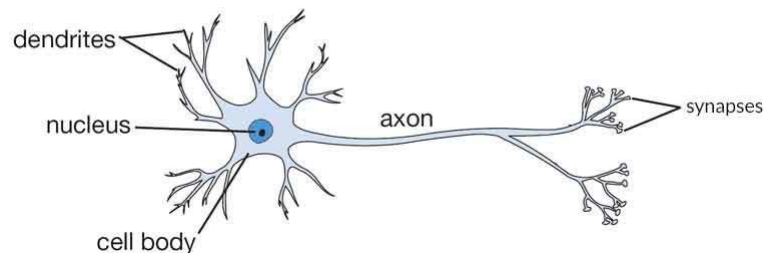


# Artificial Neural Networks

- Inspired to the structure of the human brain
  - Neurons as elaboration units
  - Synapses as connection network



Biological Neuron

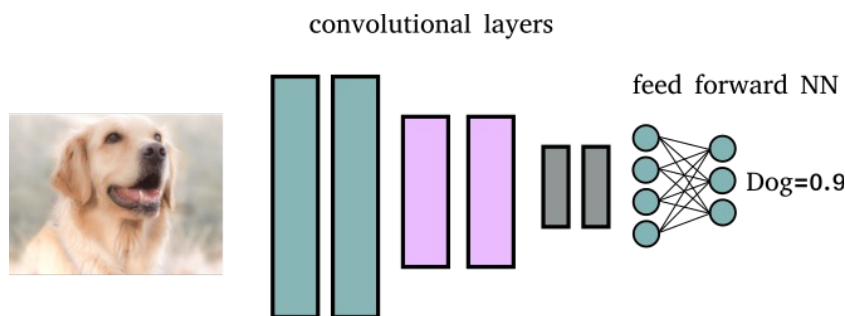




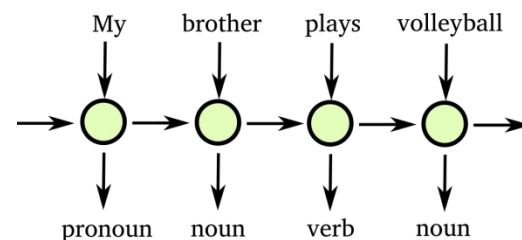
# Artificial Neural Networks

## ■ Different tasks, different architectures

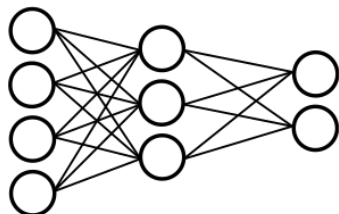
image understanding: convolutional NN (CNN)



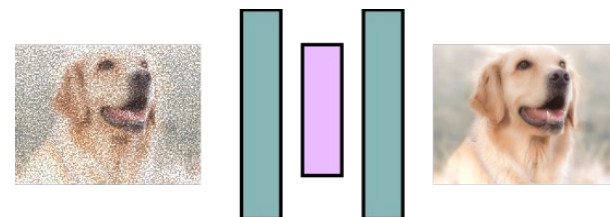
time series analysis: recurrent NN (RNN)



numerical vectors classification: feed forward NN (FFNN)

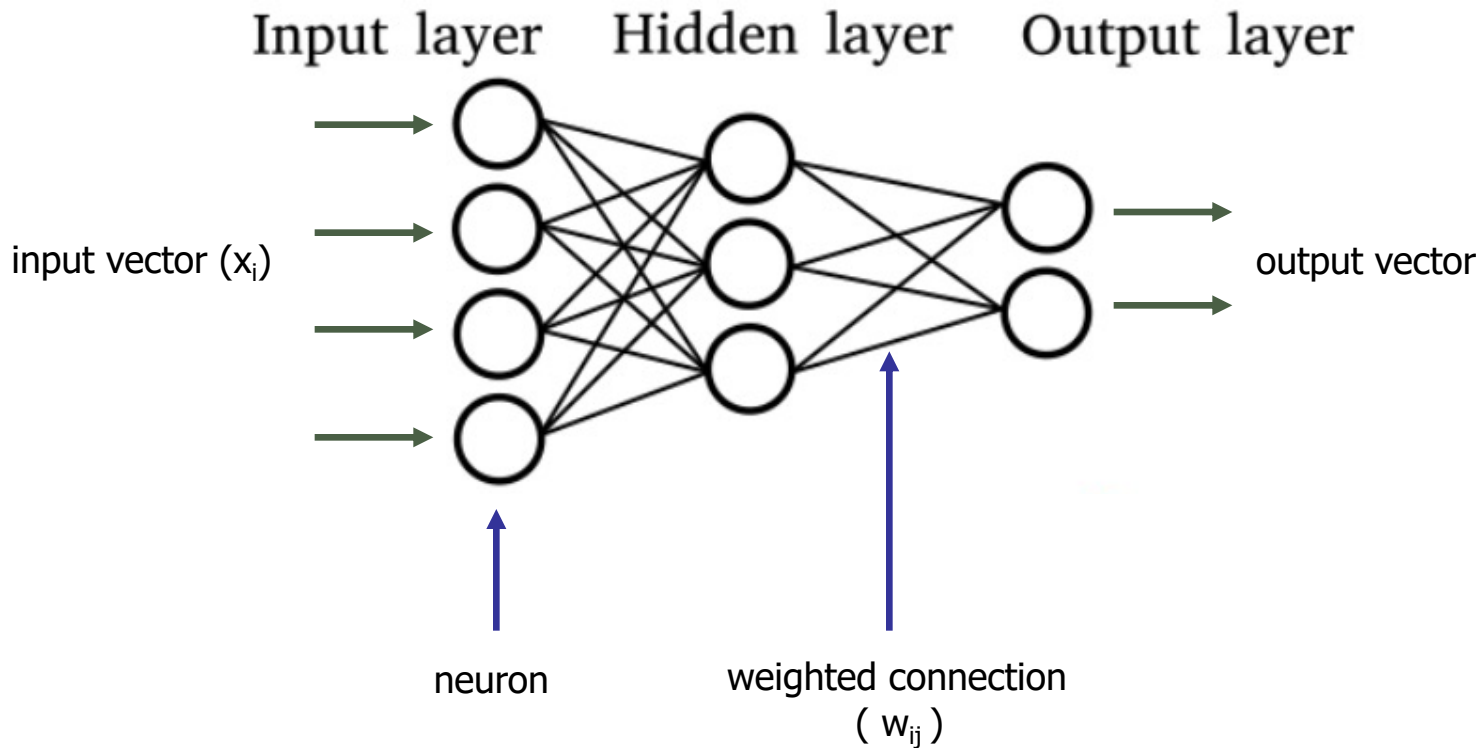


denoising: auto-encoders



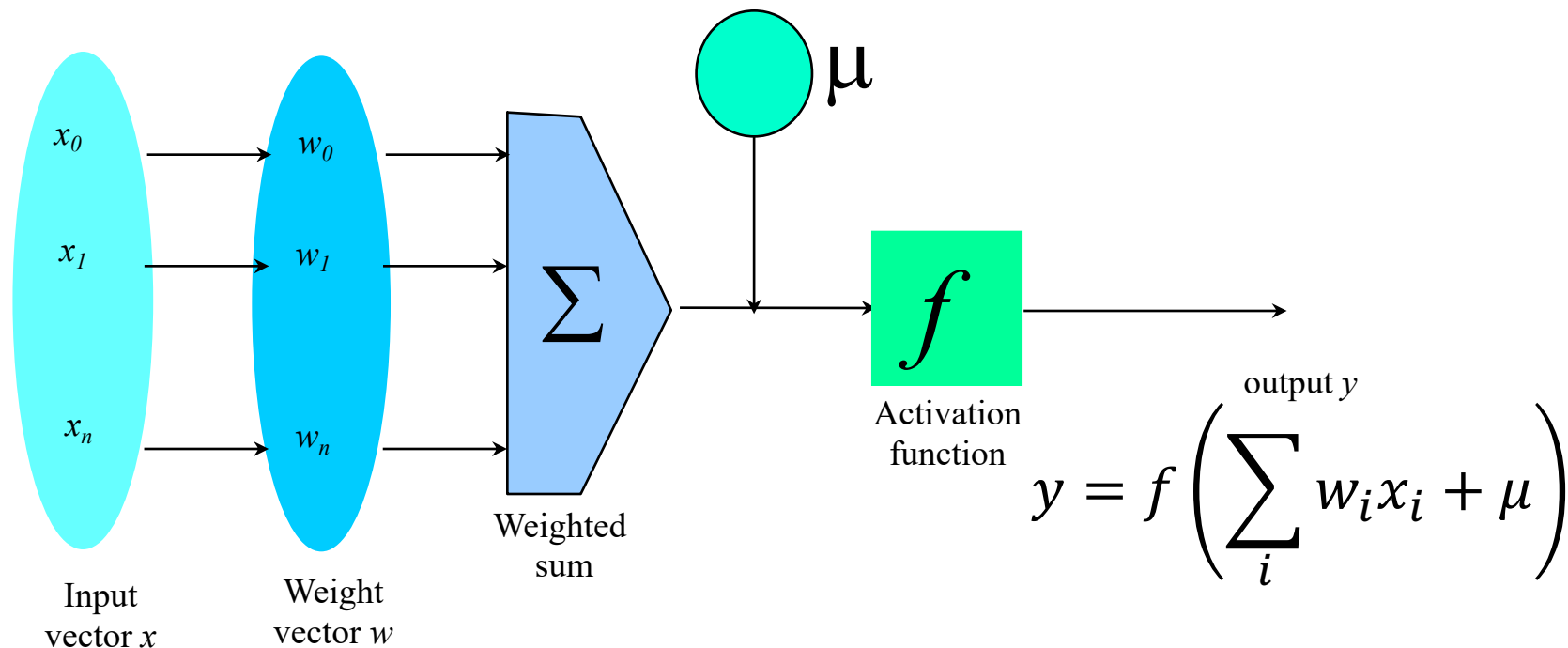


# Feed Forward Neural Network





# Structure of a neuron

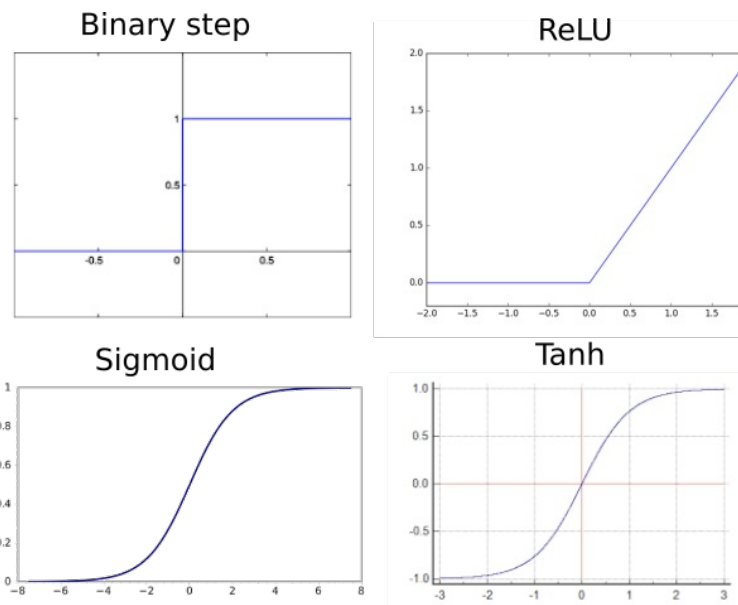




# Activation Functions

## ■ Activation

- simulates biological activation to input stymula
- provides non-linearity to the computation
- may help to saturate neuron outputs in fixed ranges

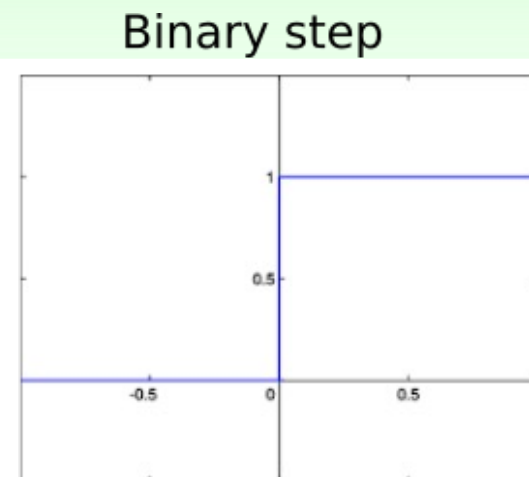




# Activation Functions

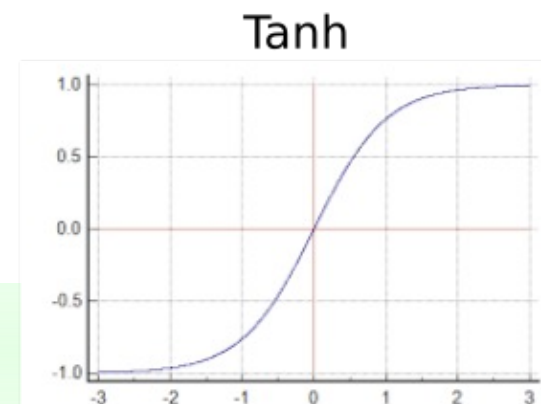
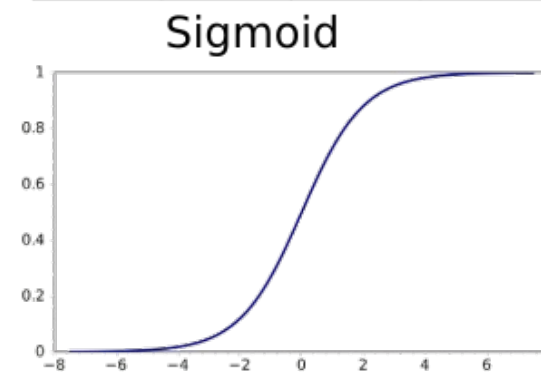
## ■ Binary Step

- outputs 1 when input is non-zero
- useful for binary outputs
- **issues:** not appropriate for gradient descent
  - derivative not defined in  $x=0$
  - derivative equal to 0 in every other position



## ■ Sigmoid, tanh

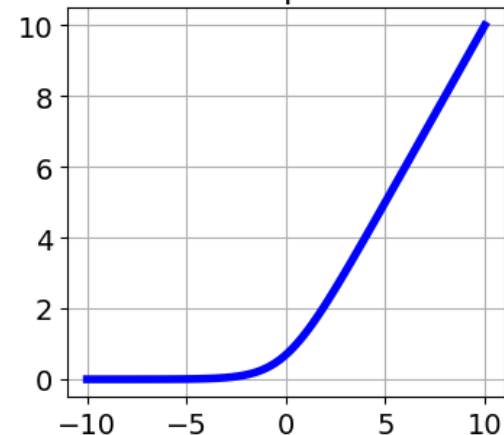
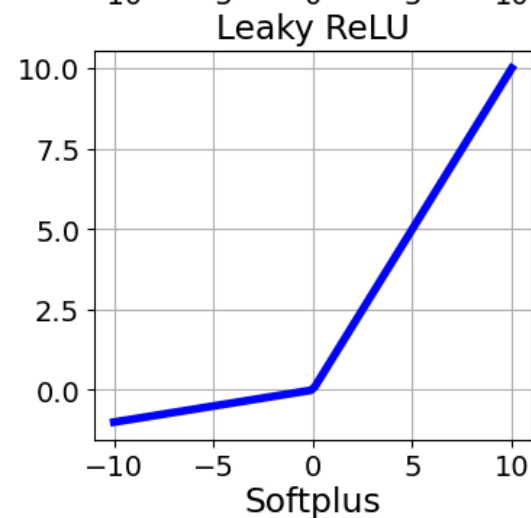
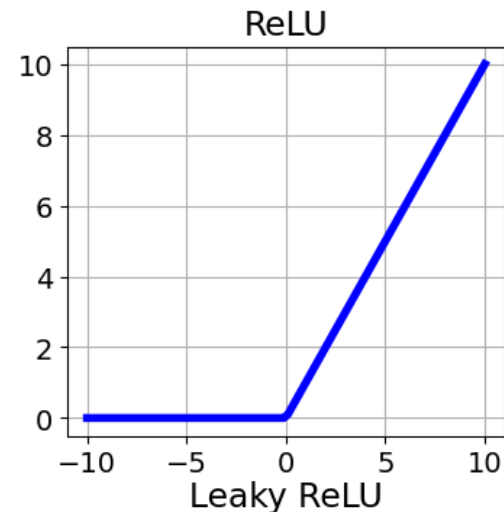
- Differentiable versions of the binary step
- saturates input value in a fixed range
- Used to enforce constraints
  - E.g. *sigmoid* in output layers allows generating values between 0 and 1
  - (useful when outputs are interpreted as probabilities)





# Activation Functions

- ReLU (Rectified Linear Unit)
  - used in deep networks (e.g. CNNs)
    - avoids vanishing gradient
    - does not saturate
  - neurons activate linearly only for positive input
  - $f(x) = x$  if  $x > 0$ , 0 otherwise
- ReLU variants
  - E.g., Leaky ReLU
    - $f(x) = x$  if  $x > 0$ ,  $\alpha x$  otherwise ( $\alpha$  is fixed)
  - Parametric ReLU (PreLU)
    - $f(x) = x$  if  $x > 0$ ,  $\alpha x$  otherwise ( $\alpha$  is learned)
  - Softplus
    - $f(x) = \ln(1 + e^x)$



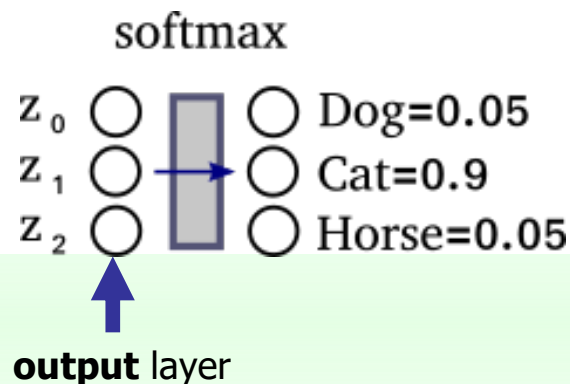


# Activation Functions

## ■ Softmax

- Used to enforce properties expected from discrete probability distributions
  - $p(x) \in [0, 1]$
  - $\sum_{x \in \mathcal{X}} p(x) = 1$
- Differently to other activation functions
  - it is applied only to the **output** layer
  - works by considering **all the neurons** in the layer
- after softmax, the output vector can be interpreted as a discrete **distribution of probabilities**
  - e.g. the probabilities for the input pattern of belonging to each class

$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_i e^{z_i}}$$





# Building a FFNN

- For each node, definition of

- set of weights
- offset value

providing the highest accuracy on the training data

- Iterative approach on training data instances



# Building a FFNN

## ■ Base algorithm

- Initially assign random values to weights and offsets
- Process instances in the training set one at a time
  - For each neuron, compute the result when applying weights, offset and activation function for the instance
  - Forward propagation until the output is computed
  - Compare the computed output with the expected output, and evaluate error
  - Backpropagation of the error, by updating weights and offset for each neuron
- The process ends when
  - % of accuracy above a given threshold
  - % of parameter variation (error) below a given threshold
  - The maximum number of epochs is reached



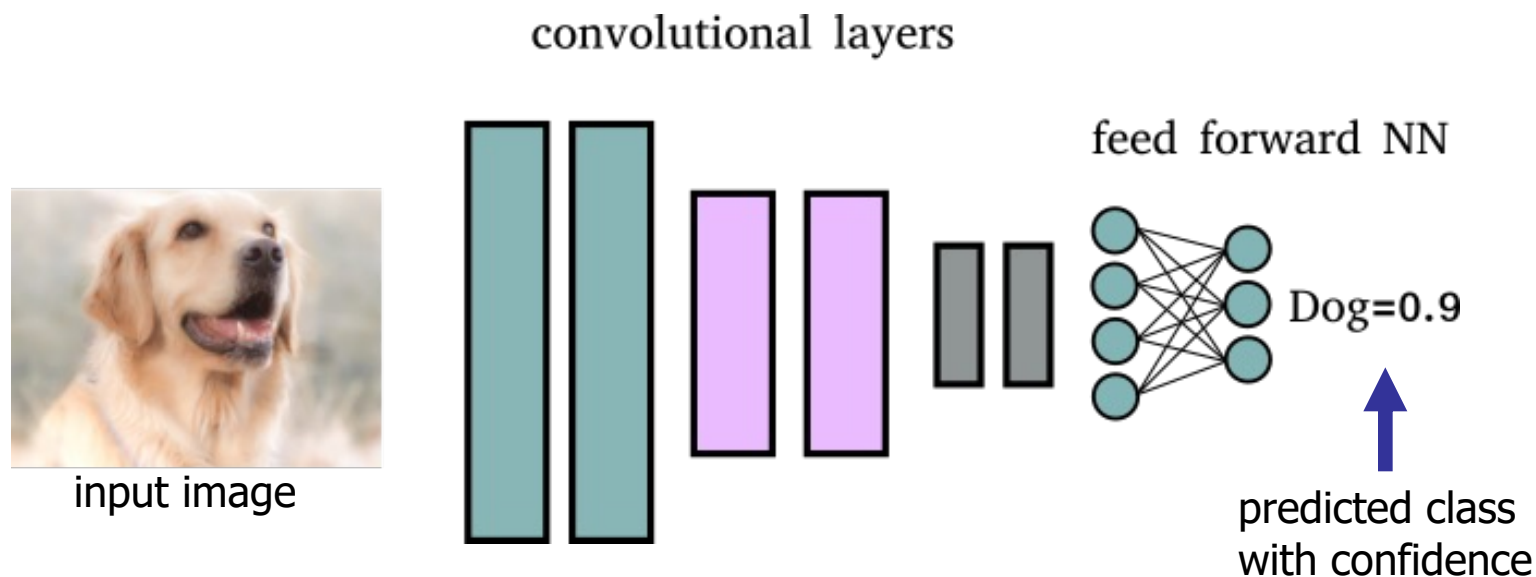
# Evaluation of Feed Forward NN

- Accuracy
  - Among best performers
- Interpretability
  - Model and prediction are not interpretable
    - Black box model
- Incrementality
  - Not incremental (but fine-tuning can be used)
- Efficiency
  - Model building requires *very complex* parameter tuning
    - It requires significant time
  - Very fast classification
- Scalability
  - Medium scalable both in training set size and attribute number
- Robustness
  - Robust to noise and outliers
  - Requires large training set
    - Otherwise unstable when tuning parameters



# Convolutional Neural Networks

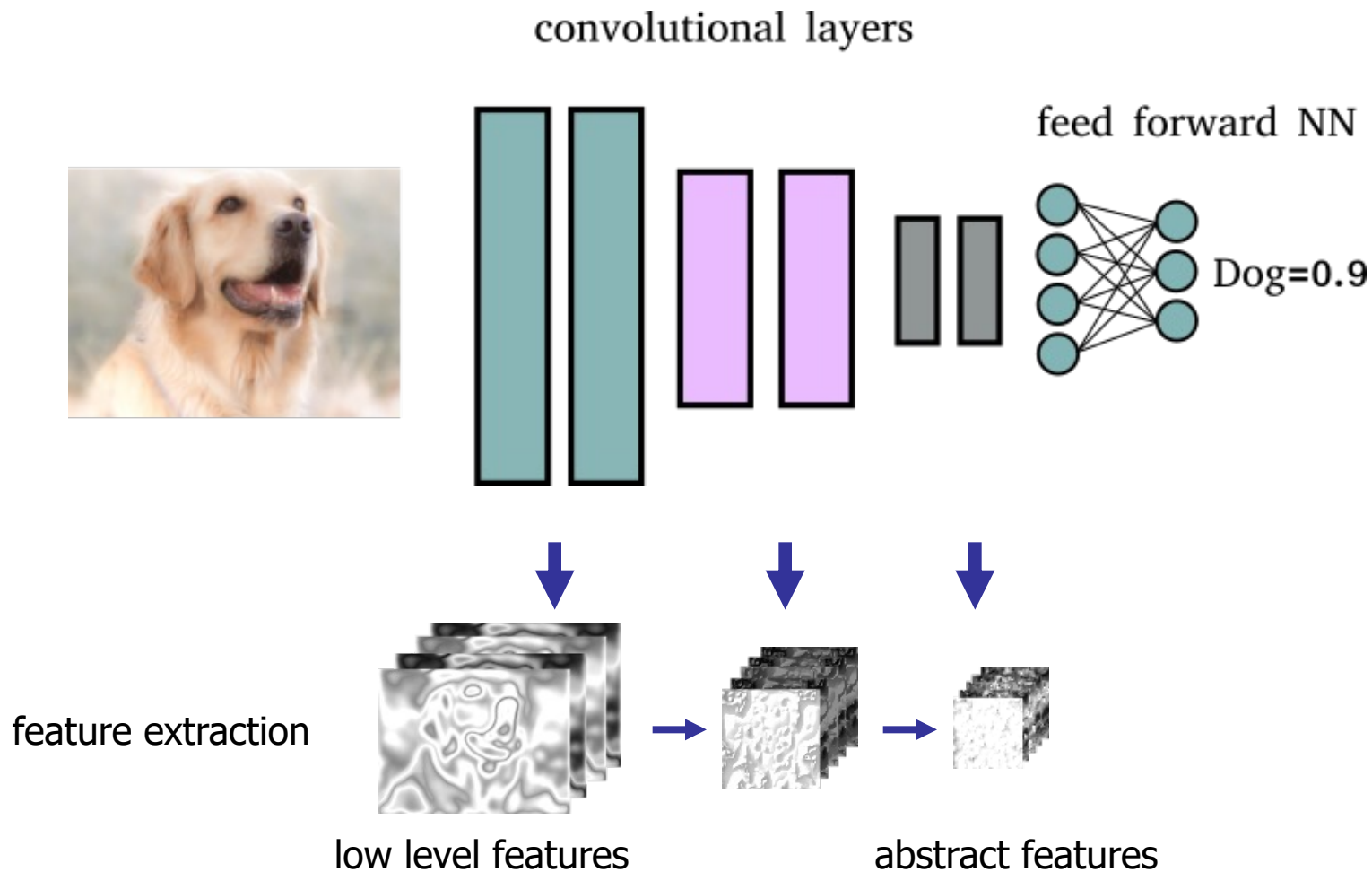
- Allow automatically extracting **features** from images and performing **classification**



Convolutional Neural Network (CNN) Architecture

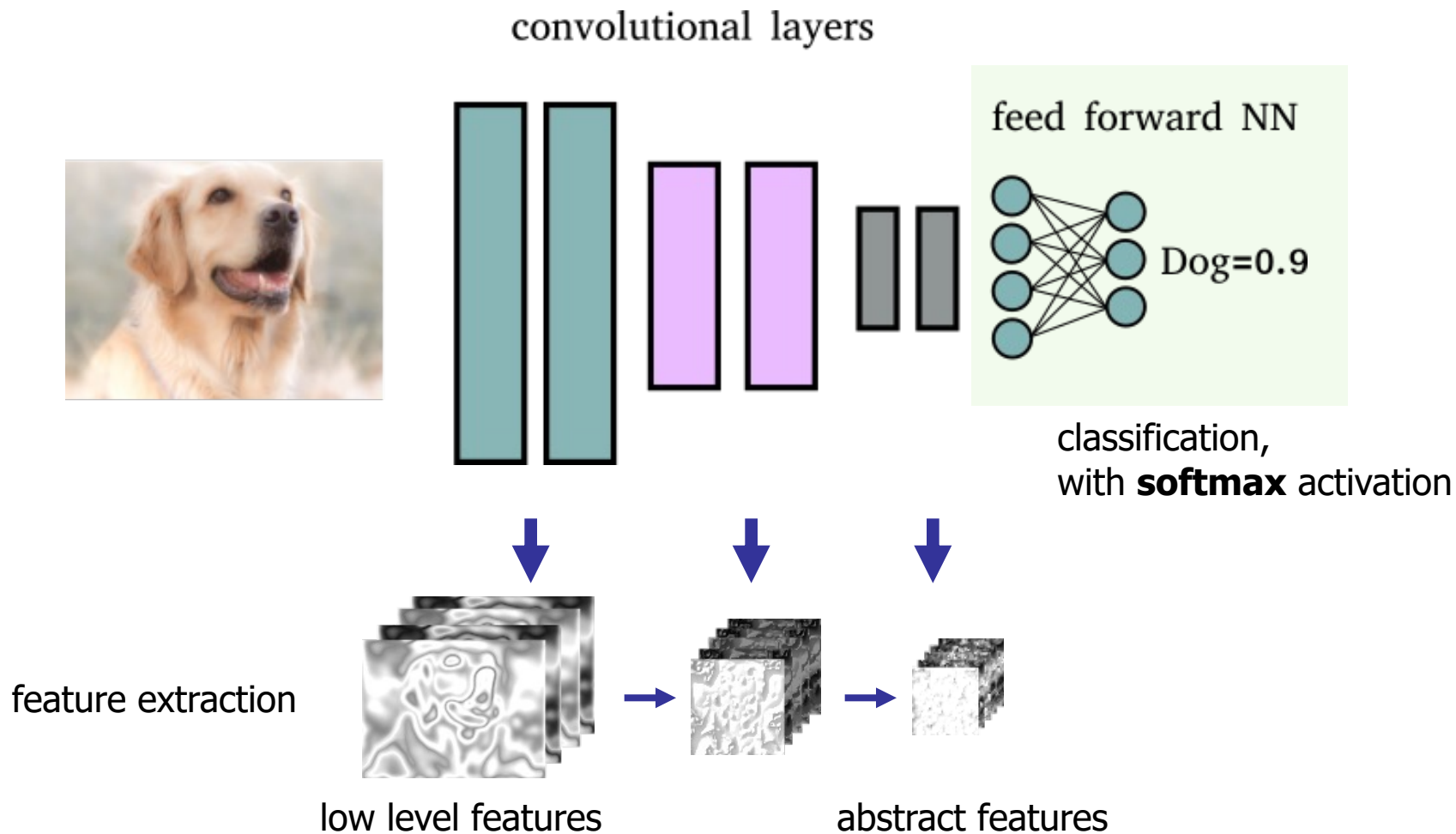


# Convolutional Neural Networks





# Convolutional Neural Networks

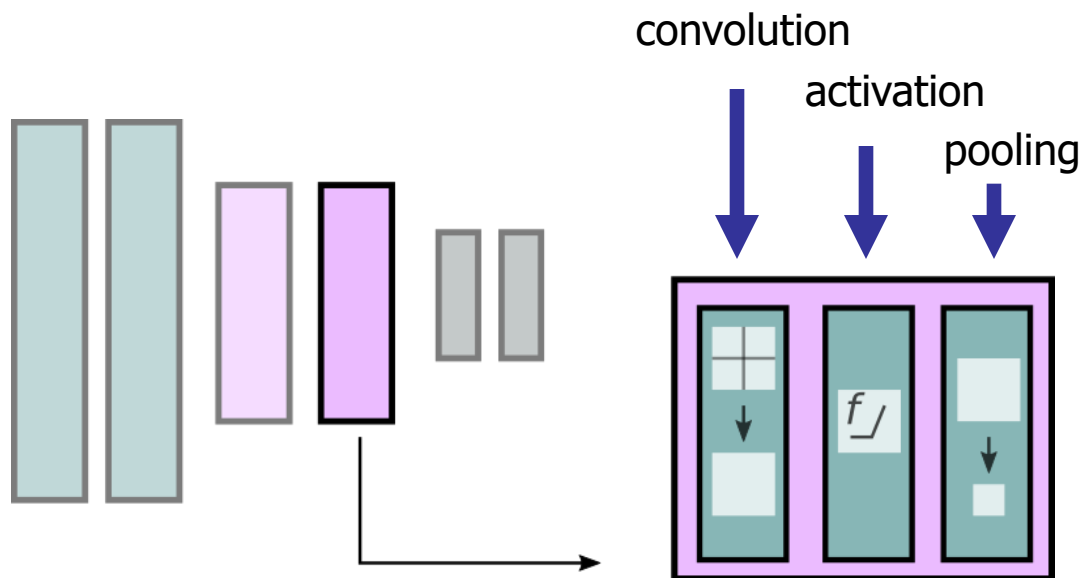




# Convolutional Neural Networks

## ■ Typical convolutional layer

- *convolution* stage: feature extraction by means of (hundreds to thousands) sliding filters
- sliding filters *activation*: apply activation functions to input tensor
- *pooling*: tensor downsampling



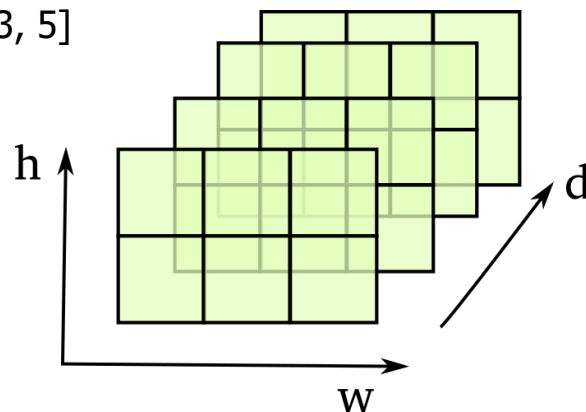


# Convolutional Neural Networks

## ■ Tensors

- data flowing through CNN layers is represented in the form of *tensors*
- *Tensor* = N-dimensional vector
- *Rank* = number of dimensions
  - scalar: rank 0
  - 1-D vector: rank 1
  - 2-D matrix: rank 2
- *Shape* = number of elements for each dimension
  - e.g. a vector of length 5 has shape [5]
  - e.g. a matrix  $w \times h$ ,  $w=5$ ,  $h=3$  has shape  $[h, w] = [3, 5]$

rank-3 tensor with shape  
 $[d, h, w] = [4, 2, 3]$

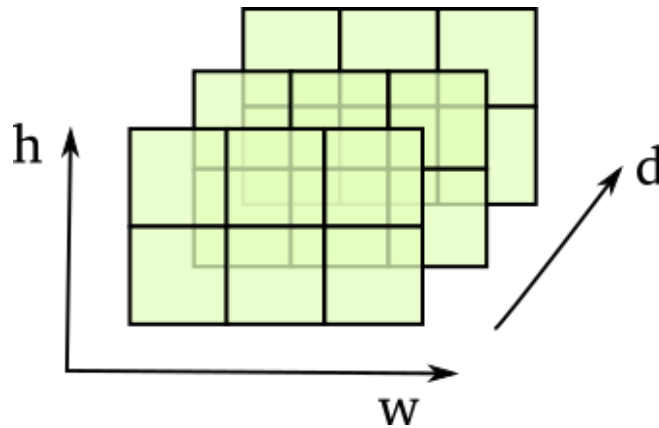




# Convolutional Neural Networks

## ■ Images

- rank-3 tensors with shape  $[d, h, w]$
- where  $h$ =height,  $w$ =width,  $d$ =image depth (1 for grayscale, 3 for RGB colors)

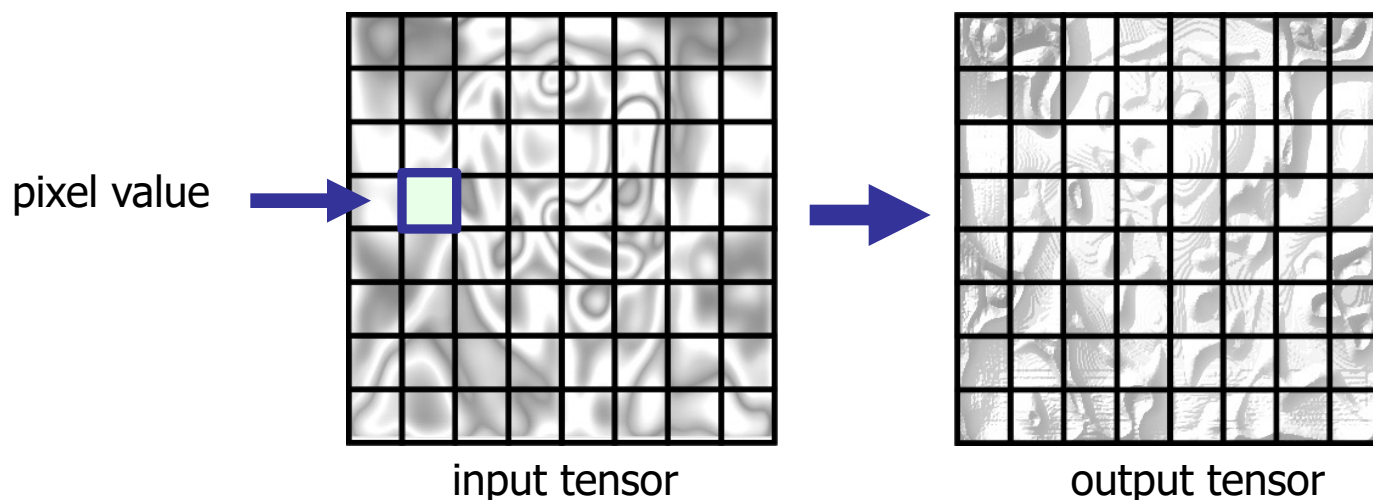




# Convolutional Neural Networks

## ■ Convolution

- processes data in form of *tensors* (multi-dimensional matrices)
- **input**: input image or intermediate features (tensor)
- **output**: a tensor with the extracted features

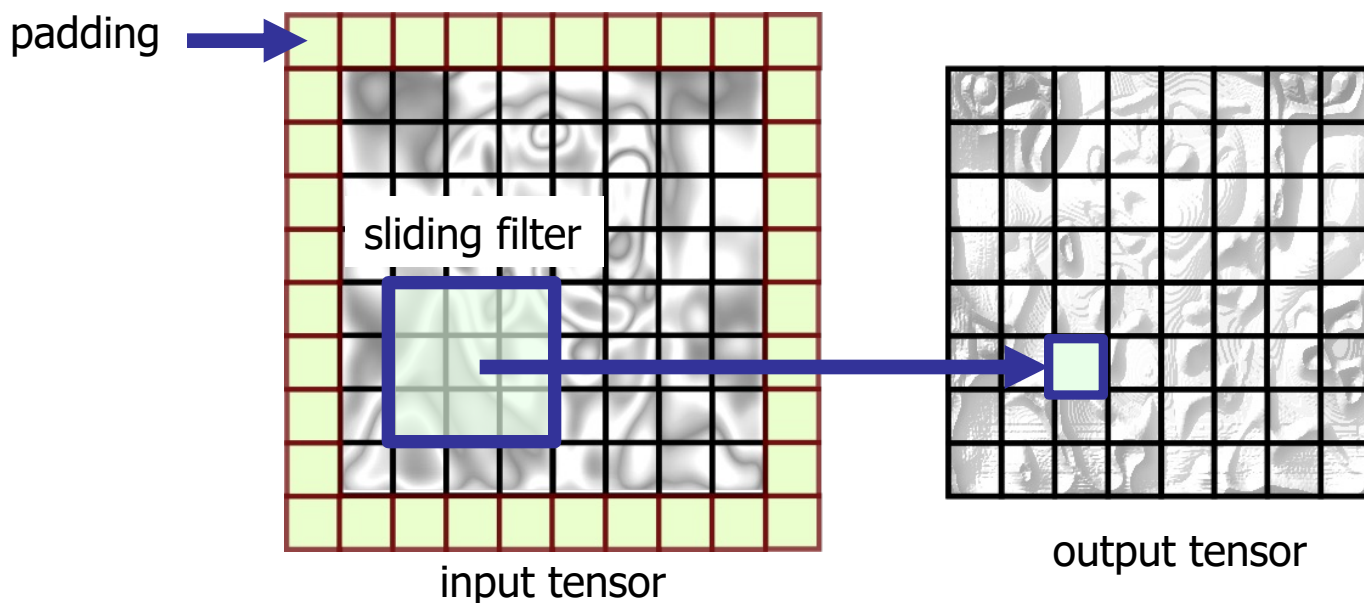




# Convolutional Neural Networks

## ■ Convolution

- a *sliding filter* produces the values of the output tensor
- sliding filters contain the trainable *weights* of the neural network
- each convolutional layer contains many (hundreds) filters

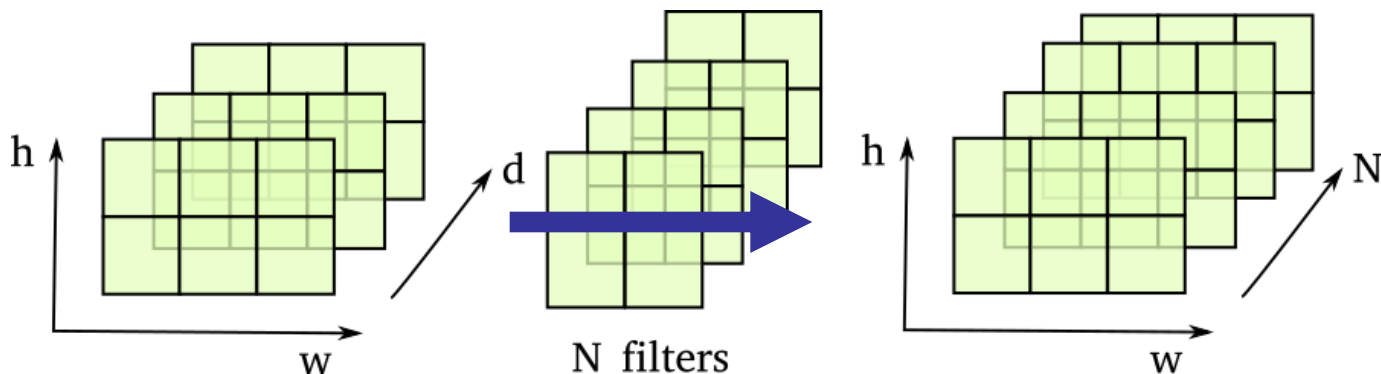




# Convolutional Neural Networks

## ■ Convolution

- images are transformed into features by convolutional filters
- after convolving a tensor  $[d, h, w]$  with  $N$  filters we obtain
  - a rank-3 tensor with shape  $[N, h, w]$
  - hence, each filter generates a layer in the depth of the output tensor

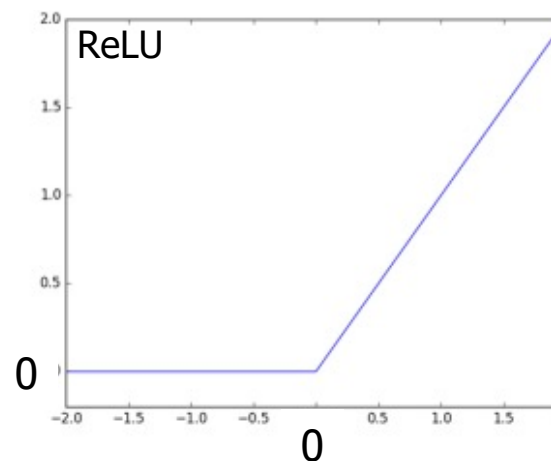
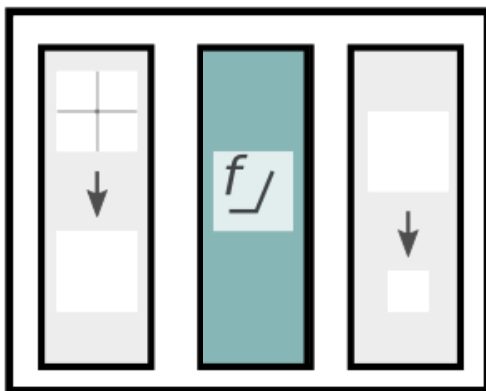




# Convolutional Neural Networks

## ■ Activation

- simulates biological activation to input stimuli
- provides non-linearity to the computation
- ReLU is typically used for CNNs
  - faster training (no vanishing gradients)
  - does not saturate
  - faster computation of derivatives for backpropagation

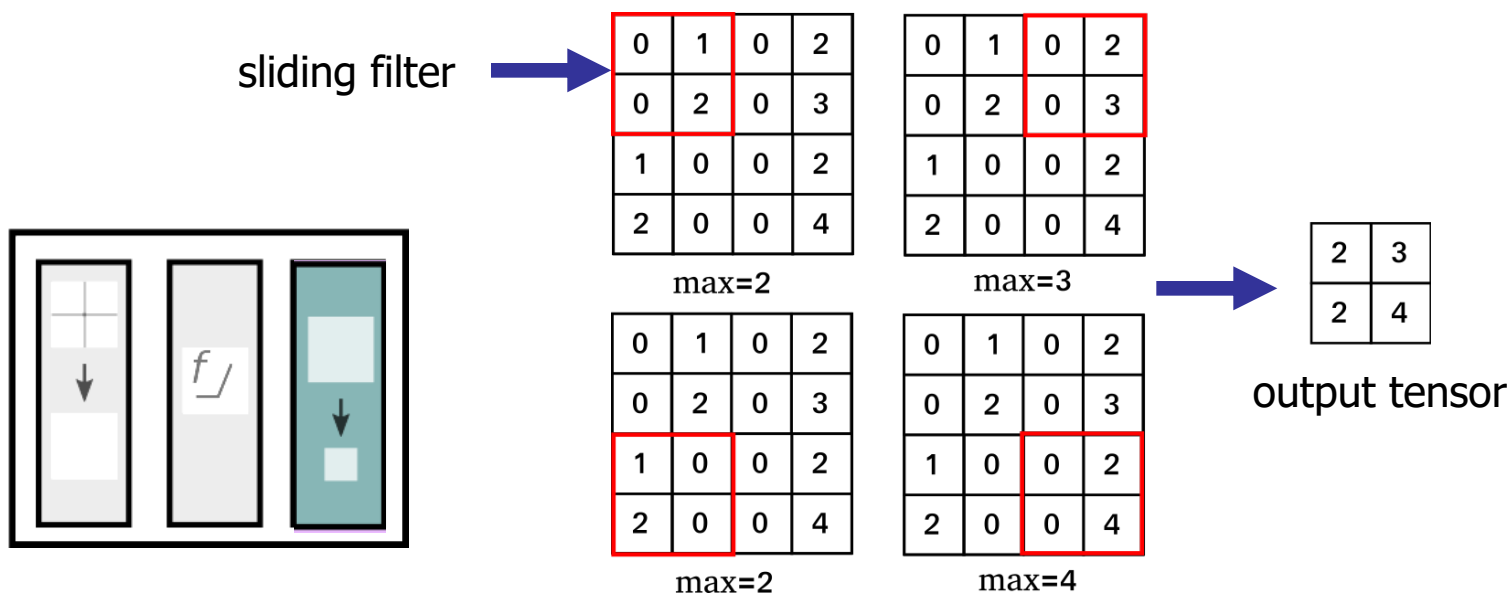




# Convolutional Neural Networks

## ■ Pooling

- performs tensor *downsampling*
- *sliding filter* which replaces tensor values with a *summary* statistic of the nearby outputs
- *maxpool* is the most common: computes the maximum value as statistic



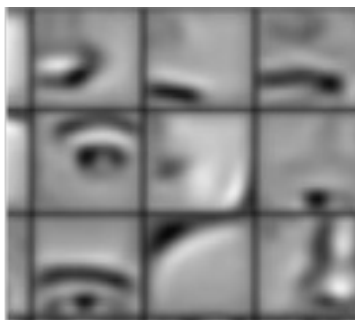


# Convolutional Neural Networks

## ■ Convolutional layers training

- during training each sliding filter learns to recognize a particular *pattern* in the input tensor
- filters in *shallow layers* recognize textures and edges
- filters in *deeper layers* can recognize objects and parts (e.g. eye, ear or even faces)

shallow filters



deeper filters

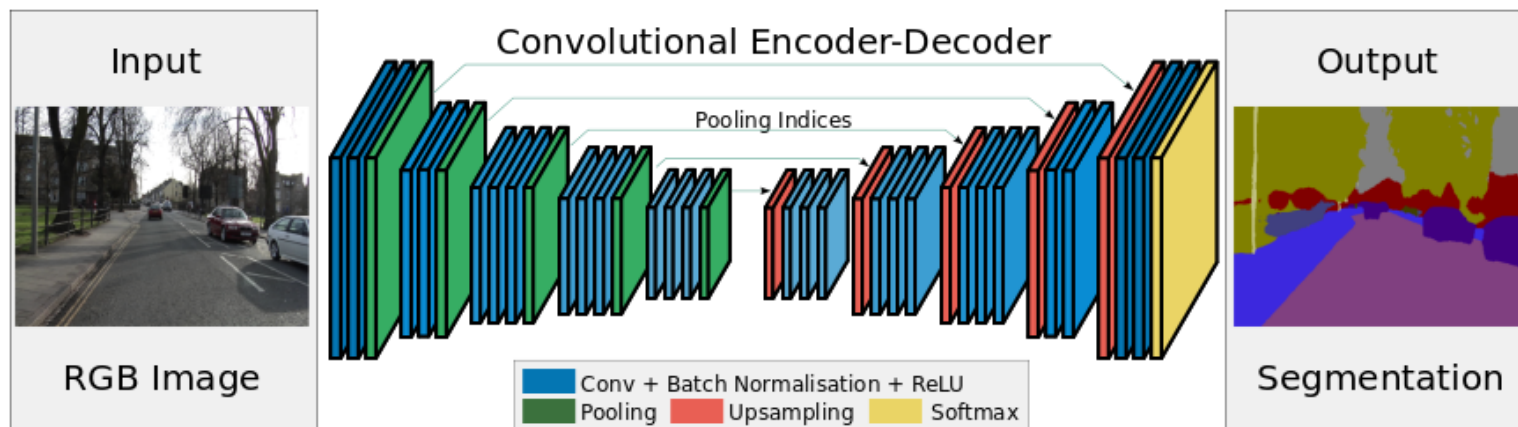




# Convolutional Neural Networks

## ■ Semantic segmentation CNNs

- allow assigning a class to each pixel of the input image
- composed of 2 parts
  - **encoder network**: convolutional layers to extract abstract features
  - **decoder network**: deconvolutional layers to obtain the output image from the extracted features

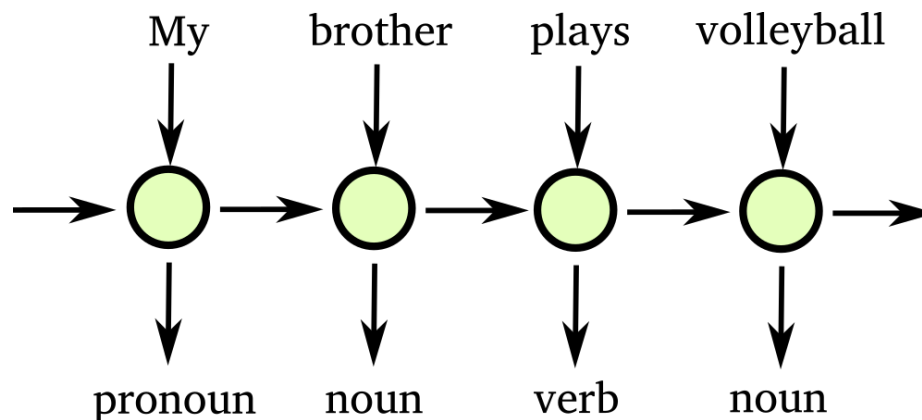


SegNet neural network



# Recurrent Neural Networks

- Allow processing *sequential* data  $x(t)$
- Differently from normal FFNN they are able to keep a *state* which evolves during time
- Applications
  - machine translation
  - time series prediction
  - speech recognition
  - part of speech (POS) tagging

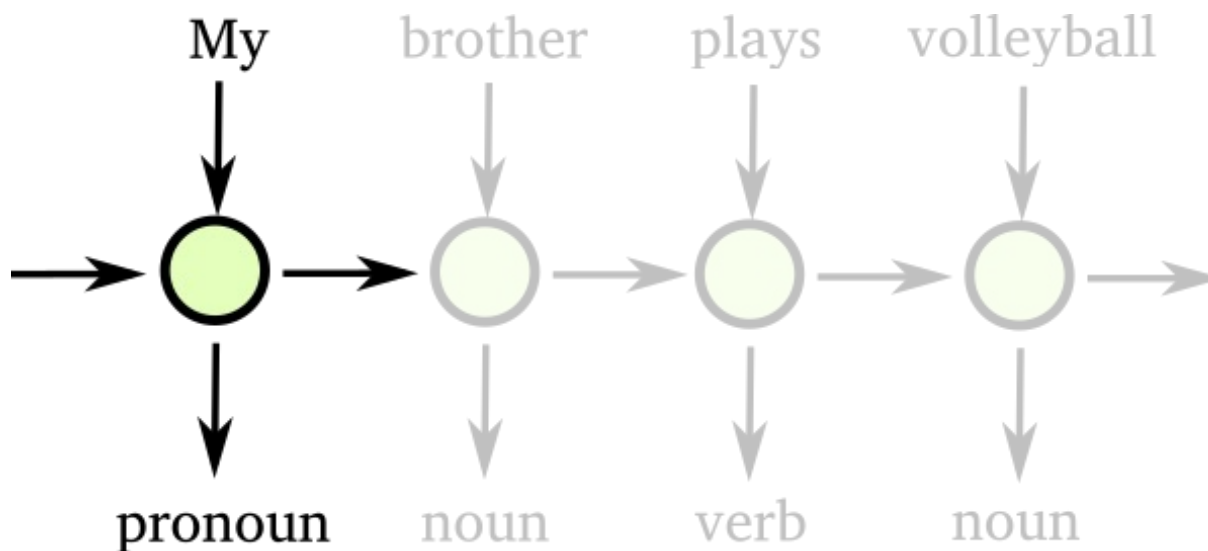




# Recurrent Neural Networks

- RNN execution during time

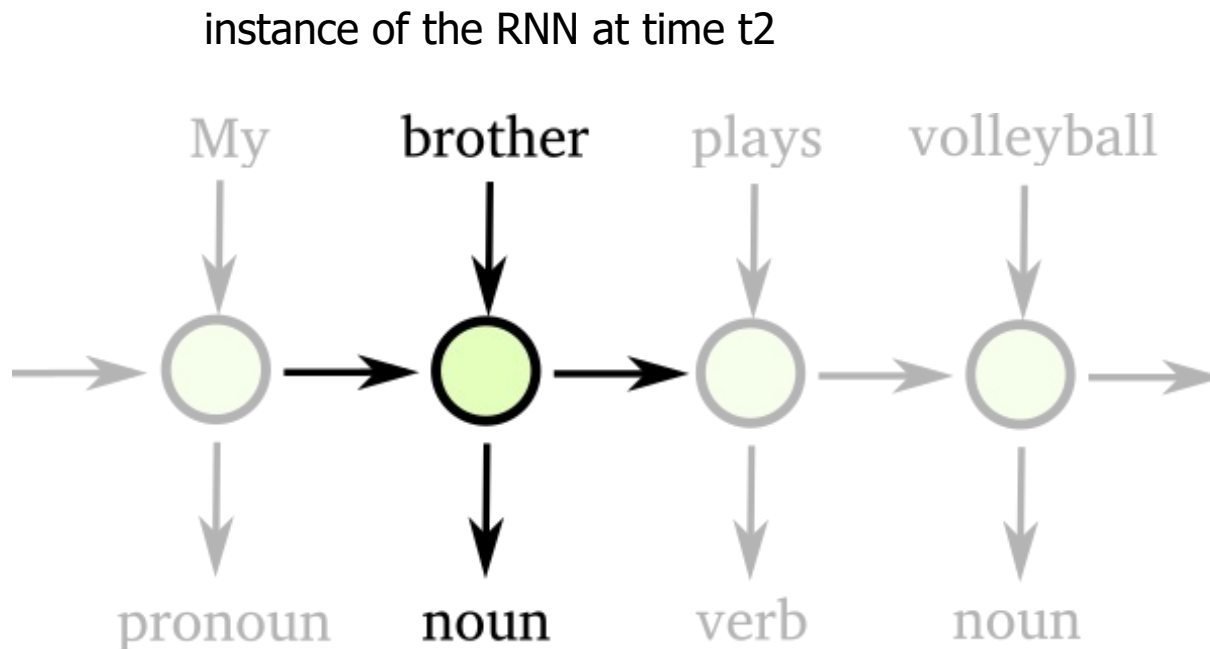
instance of the RNN at time  $t_1$





# Recurrent Neural Networks

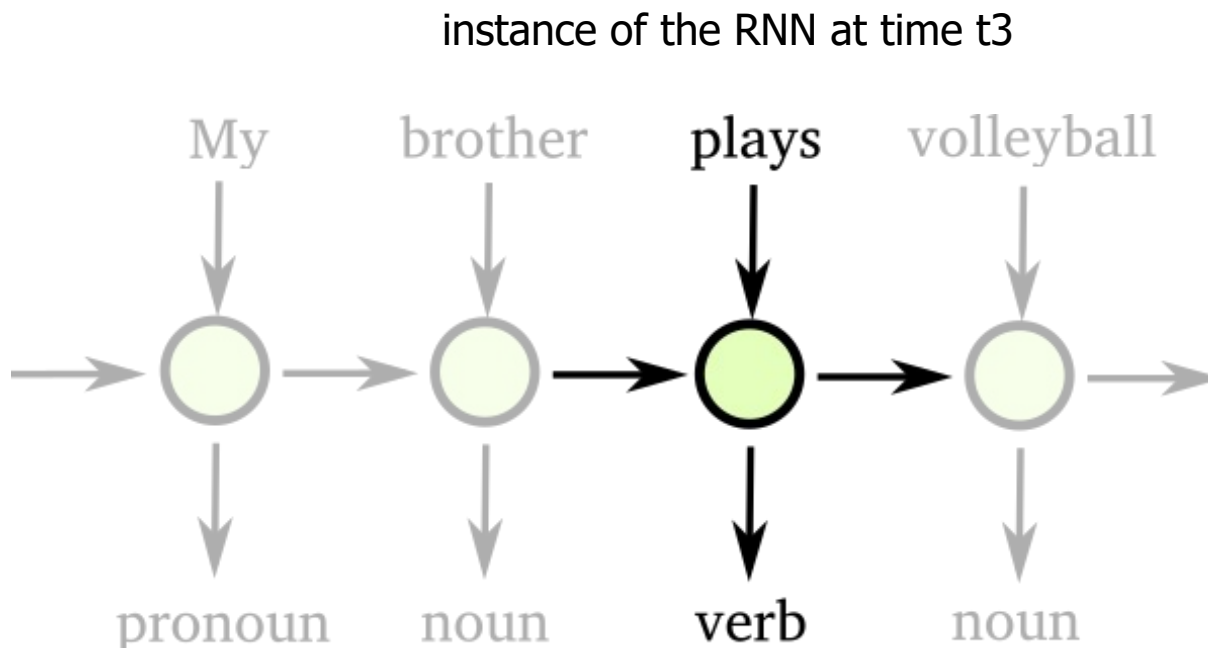
- RNN execution during time





# Recurrent Neural Networks

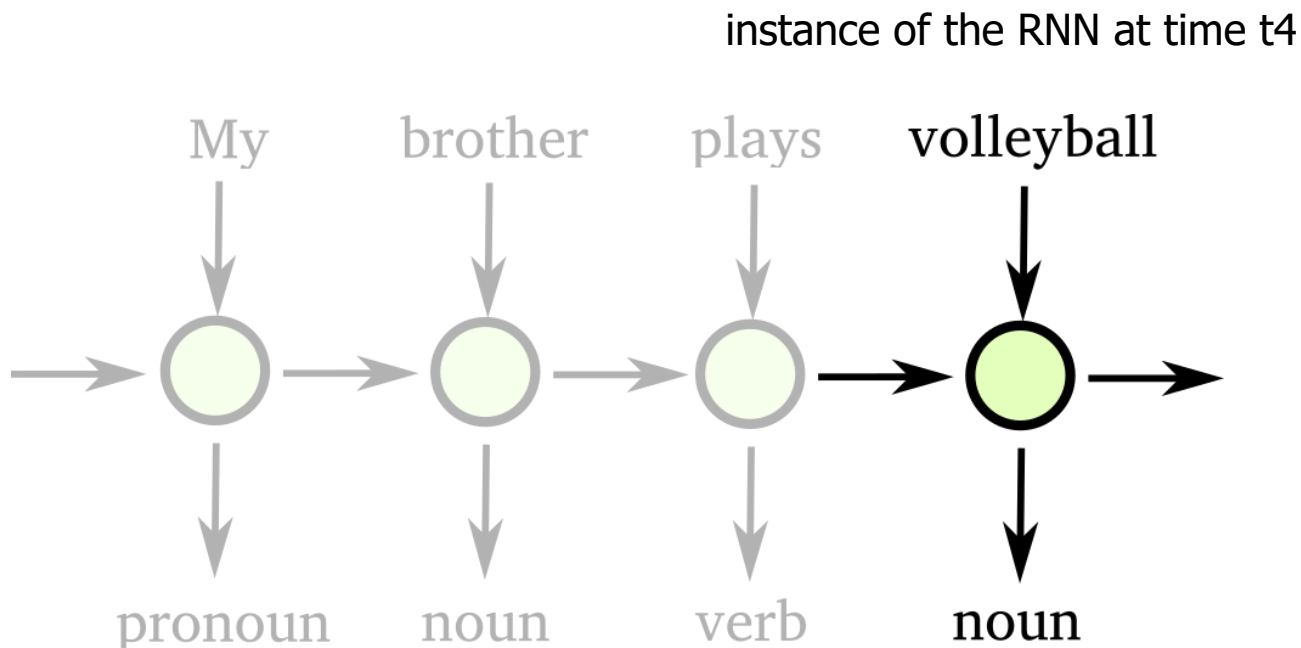
- RNN execution during time





# Recurrent Neural Networks

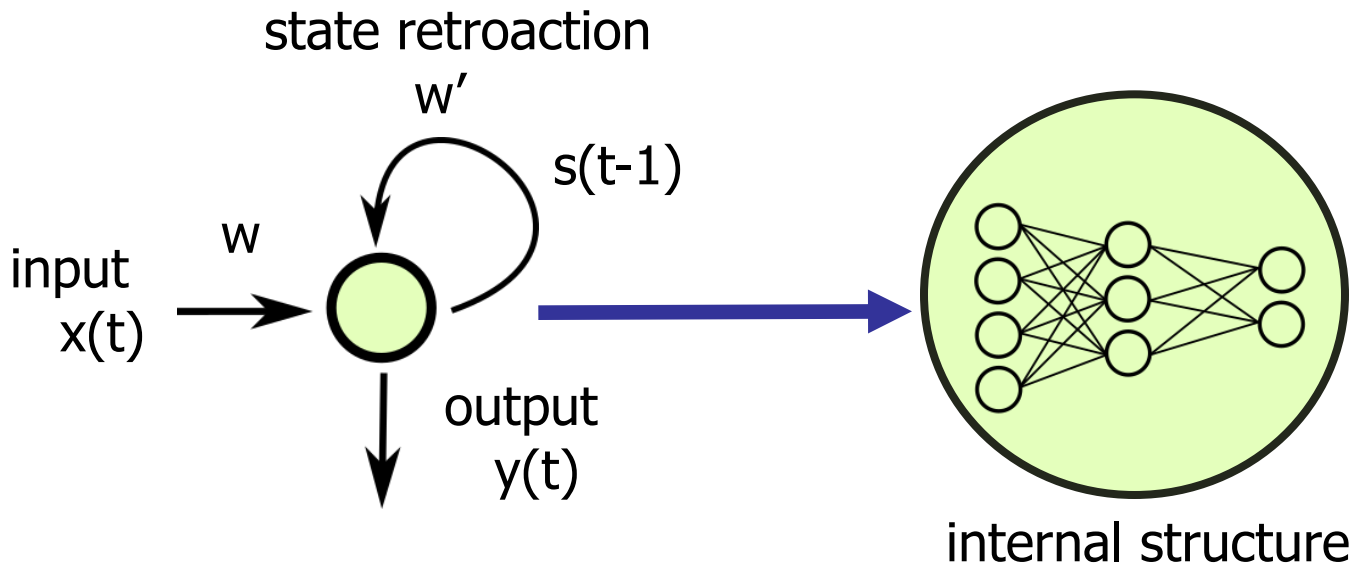
- RNN execution during time





# Recurrent Neural Networks

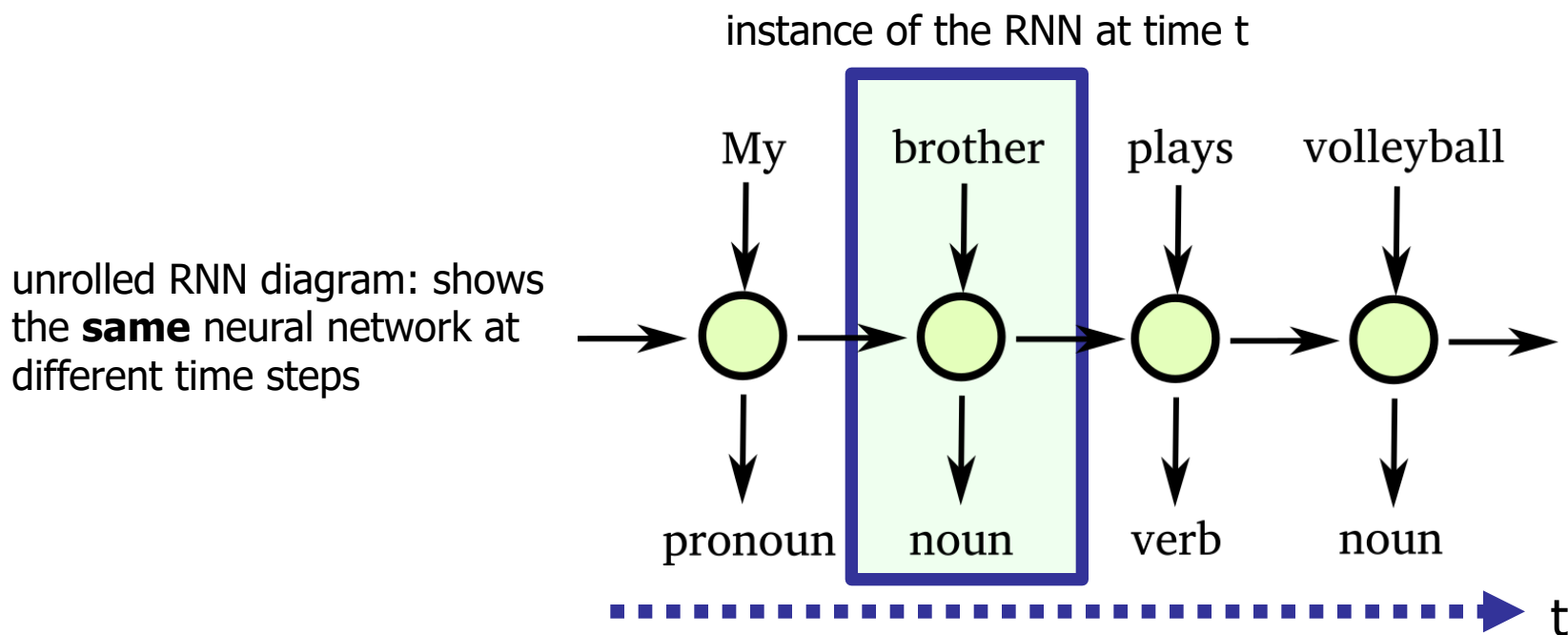
- A RNN receives as input a vector  $x(t)$  and the state at previous time step  $s(t-1)$
- A RNN typically contains many *neurons organized in different layers*





# Recurrent Neural Networks

- Training is performed with *Backpropagation Through Time*
- Given a pair training sequence  $x(t)$  and expected output  $y(t)$ 
  - error is propagated through time
  - weights are updated to minimize the error across all the time steps





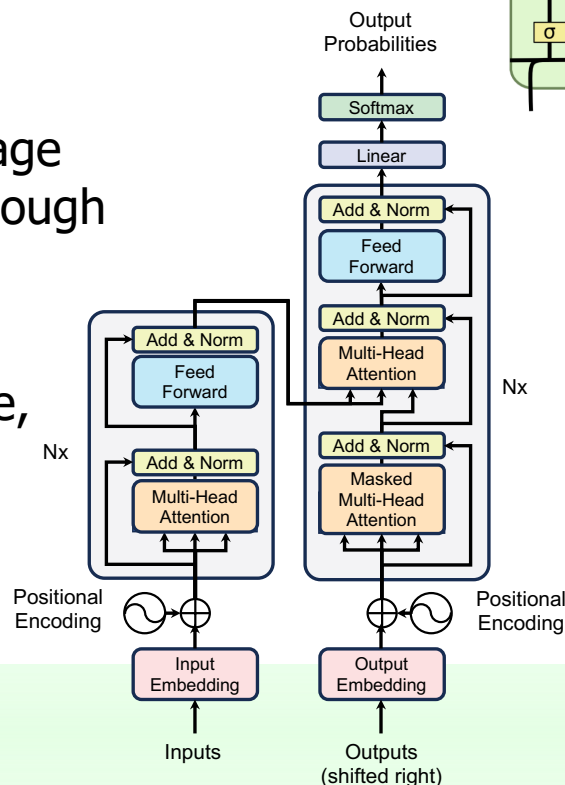
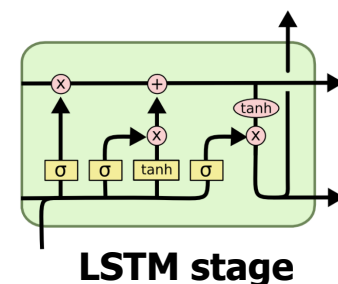
# Recurrent Neural Networks

## ■ Issues

- *vanishing gradient*: error gradient decreases rapidly over time, weights are not properly updated
- this makes harder having RNN with *long-term* memories

## ■ Solutions

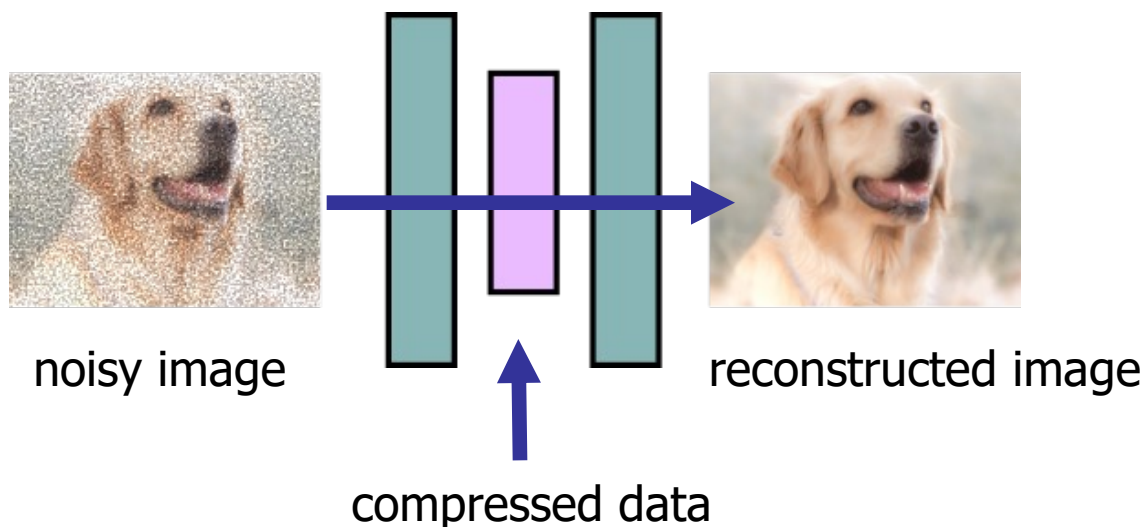
- *LSTM* (Long Short Term Memories)
  - RNN with “gates” which encourage the state information to flow through long time intervals
- *Transformers*
  - Completely different architecture, processes the entire sequence simultaneously





# Autoencoders

- Autoencoders allow *compressing* input data by means of compact representations and from them *reconstructing* the initial input
  - for feature extraction: the compressed representation can be used as significant set of features representing input data
  - for image (or signal) *denoising*: the image reconstructed from the abstract representation is denoised with respect to the original one

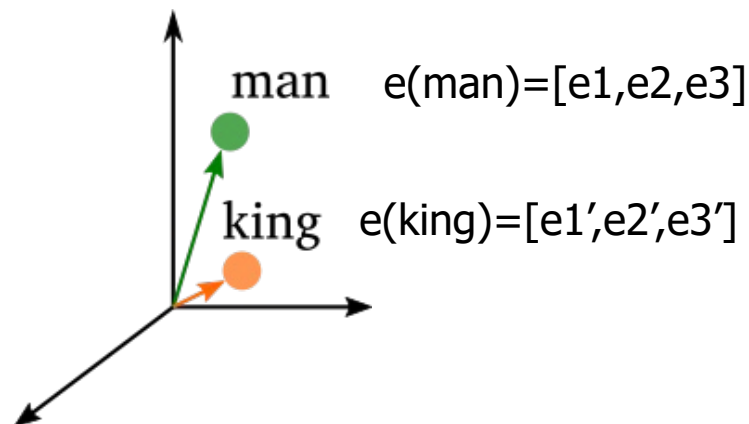
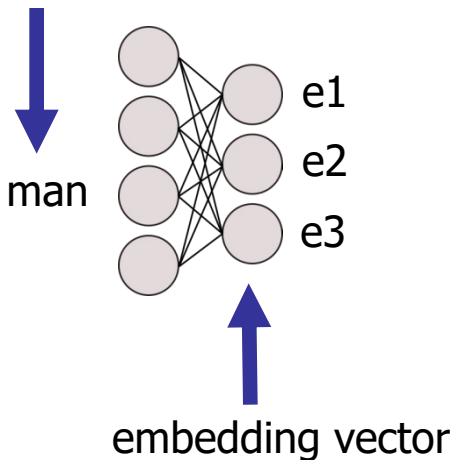




# Word Embeddings (Word2Vec)

- Word *embeddings* associate words to n-dimensional vectors
  - trained on big text collections to model the word distributions in different sentences and contexts
  - able to capture the *semantic* information of each word
  - words with similar *meaning* share vectors with similar characteristics

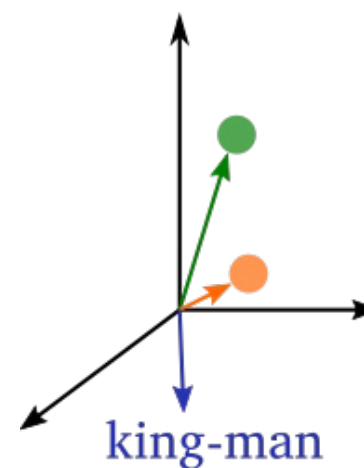
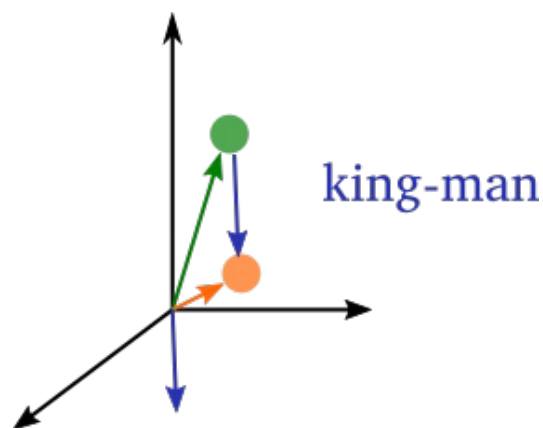
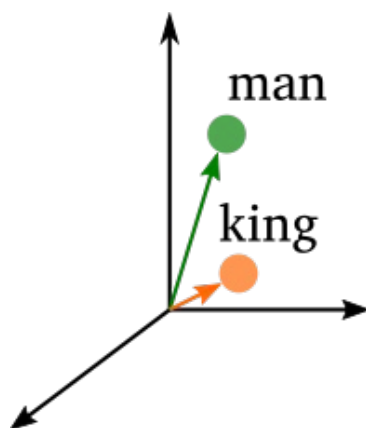
input word





# Word Embeddings (Word2Vec)

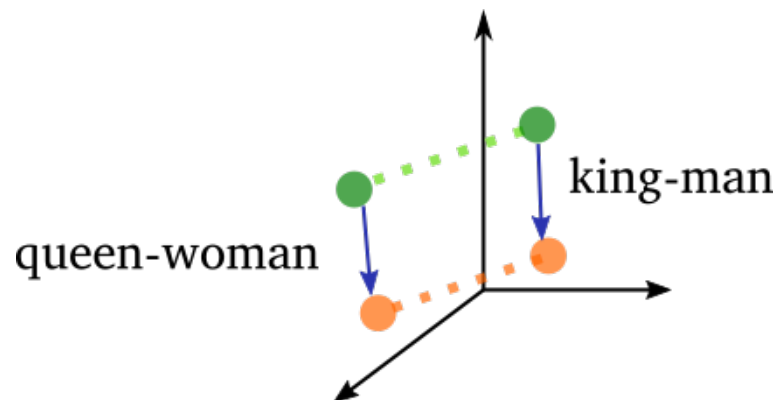
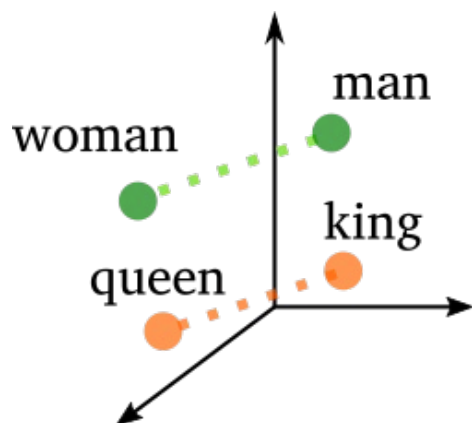
- Since each word is represented with a vector, operations among words (e.g. difference, addition) are allowed





# Word Embeddings (Word2Vec)

- Semantic relationships among words are captured by vector positions



$\text{king} - \text{man} = \text{queen} - \text{woman}$   
 $\text{king} - \text{man} + \text{woman} = \text{queen}$

# Model evaluation



Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis, Flavio Giobergia

*Politecnico di Torino*



# Model evaluation

- Methods for performance evaluation
  - Partitioning techniques for training and test sets
- Metrics for performance evaluation
  - Accuracy, other measures
- Techniques for model comparison
  - ROC curve

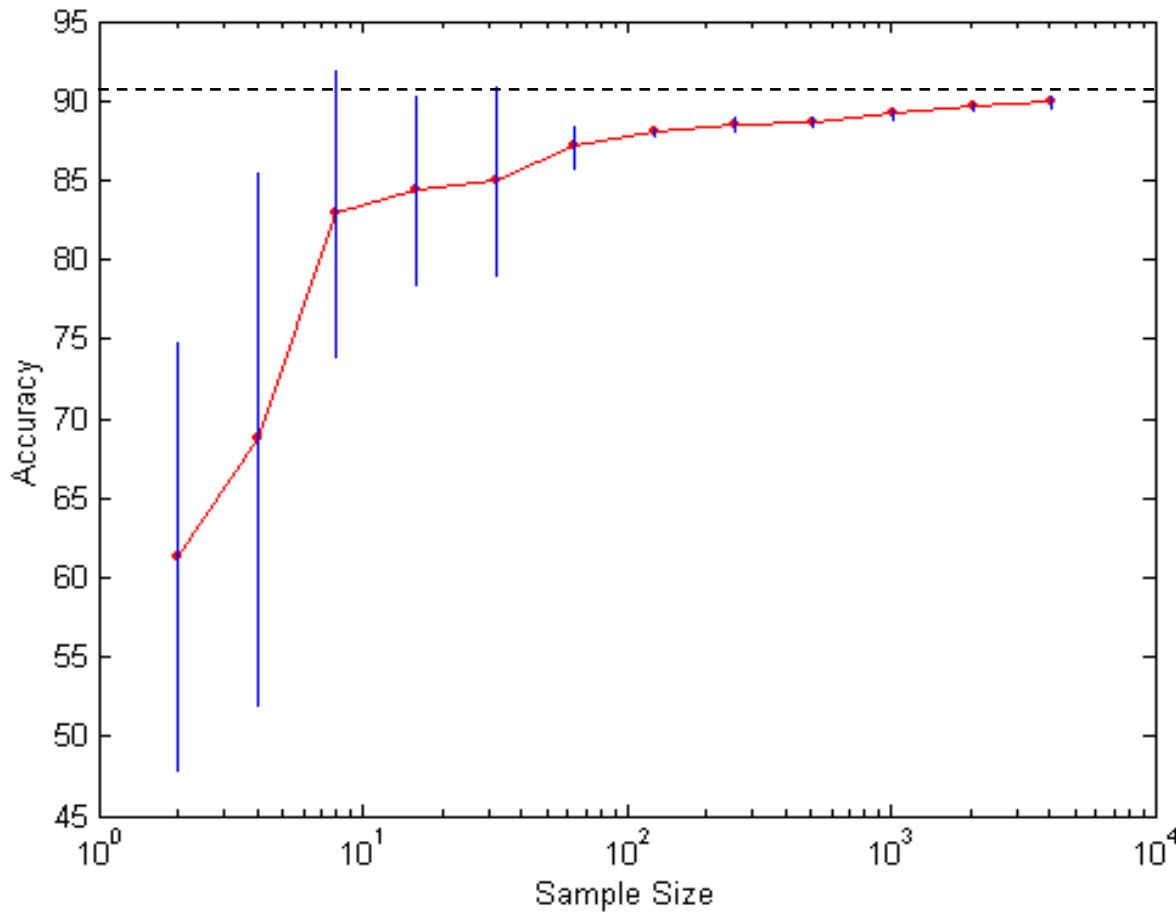


# Methods for performance evaluation

- Objective
  - reliable estimate of performance
- Performance of a model may depend on other factors besides the learning algorithm
  - Class distribution
  - Cost of misclassification
  - Size of training and test sets



# Learning curve



- Learning curve shows how accuracy changes with varying training sample size
- Requires a sampling schedule for creating learning curve:
  - Arithmetic sampling (Langley, et al)
  - Geometric sampling (Provost et al)



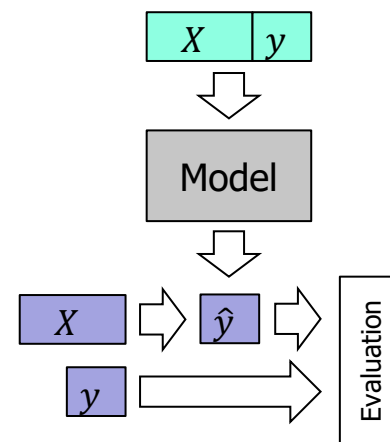
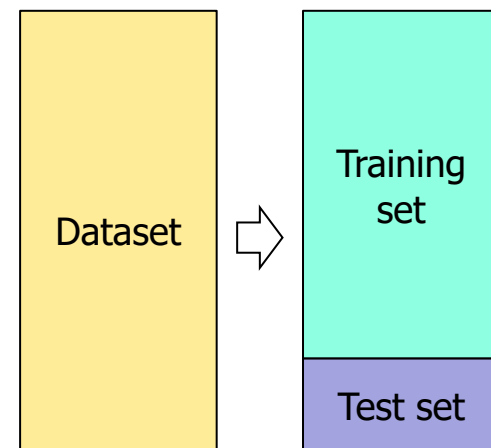
# Partitioning data

- Several partitioning techniques
  - holdout
  - cross validation
- Stratified sampling to generate partitions
  - without replacement
- Bootstrap
  - Sampling with replacement



# Holdout

- Fixed partitioning
  - For instance, 80% for **training**, 20% for **test**
  - Other proportions may be appropriate, depending on the dataset size
- Appropriate for large datasets
  - may be repeated several times
    - repeated holdout





# Cross validation

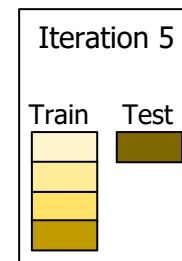
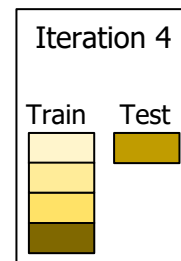
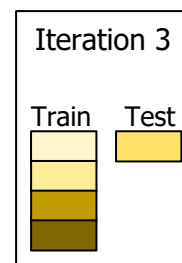
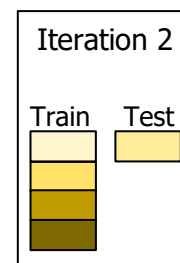
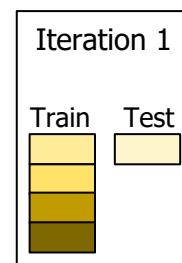
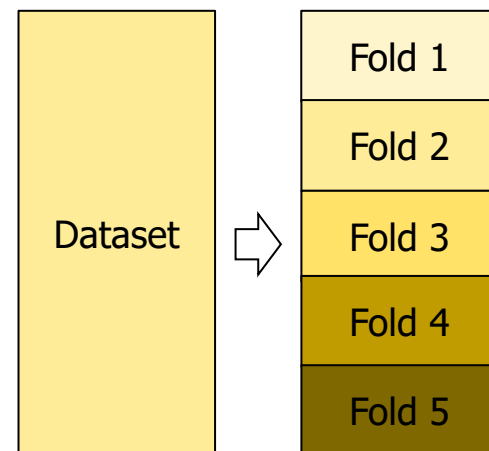
## ■ Cross validation

- partition data into  $k$  disjoint subsets (i.e., folds)
- $k$ -fold: train on  $k-1$  partitions, test on the remaining one
  - repeat for all folds
- reliable accuracy estimation, not appropriate for very large datasets

## ■ Leave-one-out

- cross validation for  $k=n$
- only appropriate for very small datasets

$k$ -fold,  $k = 5$





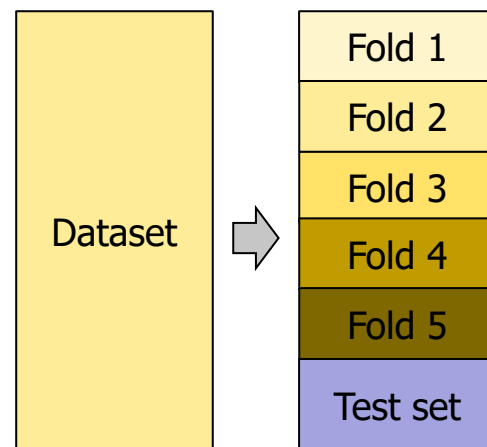
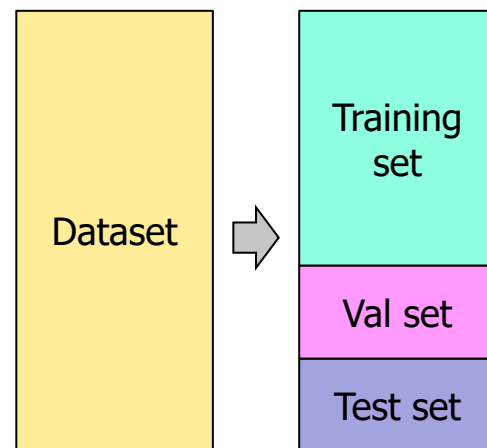
# Model performance estimation

- Model training step
  - Building a new model
- Model validation step
  - Hyperparameter tuning
  - Algorithm selection
- Model test step
  - Estimation of model performance



# Model performance estimation

- Typical dataset size
  - Training set 60% of labeled data
  - Validation set 20% of labeled data
  - Test set 20% of labeled data
- Splitting labeled data
  - Use hold-out to split in
    - training+validation
    - test
  - Use cross validation to split in
    - training
    - validation

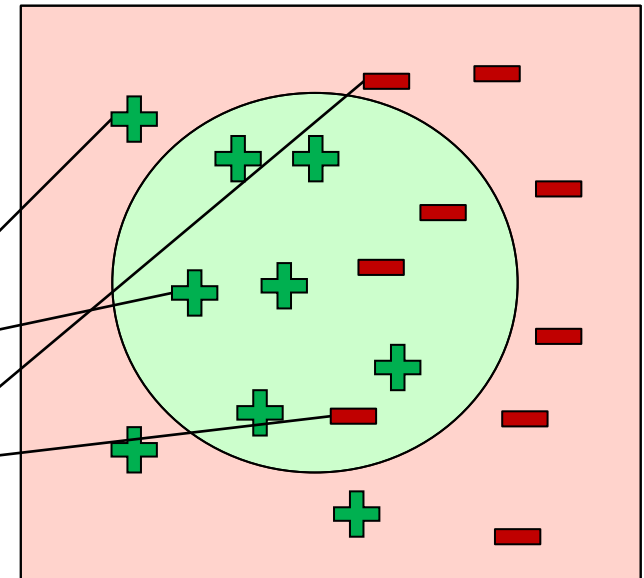




# Metrics for model evaluation

- Evaluate the predictive accuracy of a model
- Confusion matrix
  - binary classifier

	PREDICTED CLASS	
	Class=Yes	Class=No
	Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	Class=No
	Class=No	Class=No





# Accuracy

- Most widely-used metric for model evaluation

$$\text{Accuracy} = \frac{\text{Number of correctly classified objects}}{\text{Number of classified objects}}$$

- Not always a reliable metric



# Accuracy

- For a binary classifier

ACTUAL CLASS	PREDICTED CLASS	
	Class=Yes	Class=No
Class=Yes	True positive (TP)	False negative (FN)
Class=No	False positive (FP)	True negative (TN)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$



# Limitations of accuracy

- Consider a binary problem

- Cardinality of Class 0 = 9900
- Cardinality of Class 1 = 100

- Model

$() \rightarrow \textit{class 0}$

- Model predicts everything to be class 0
  - accuracy is  $9900/10000 = 99.0 \%$
- Accuracy is misleading because the model does not detect any class 1 object



# Limitations of accuracy

- Classes may have different importance
  - Misclassification of objects of a given class is more important
  - e.g., ill patients erroneously assigned to the healthy patients class
- Accuracy is not appropriate for
  - unbalanced class label distribution
  - different class relevance



# Class specific measures

- Evaluate separately for each class  $C$

$$Precision(C) = \frac{\text{Number of objects correctly assigned to } C}{\text{Number of objects assigned to } C}$$

$$Recall(C) = \frac{\text{Number of objects correctly assigned to } C}{\text{Number of objects belonging to } C}$$

- Maximize

$$F_1 \text{ score}(P) = \frac{2 \cdot Recall(P) \cdot Precision(P)}{Precision(P) + Recall(P)}$$



# Class specific measures

- For a binary classification problem
  - on the confusion matrix, for the positive class

$$Precision(P) = \frac{TP}{TP + FP}$$

$$Recall(P) = \frac{TP}{TP + FN}$$

ACTUAL CLASS	PREDICTED CLASS		
		Class=Yes	Class=No
	Class=Yes	True positive (TP)	False negative (FN)
	Class=No	False positive (FP)	True negative (TN)

$$F_1 \text{ score}(P) = \frac{2 \cdot Recall(P) \cdot Precision(P)}{Recall(P) + Precision(P)} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$



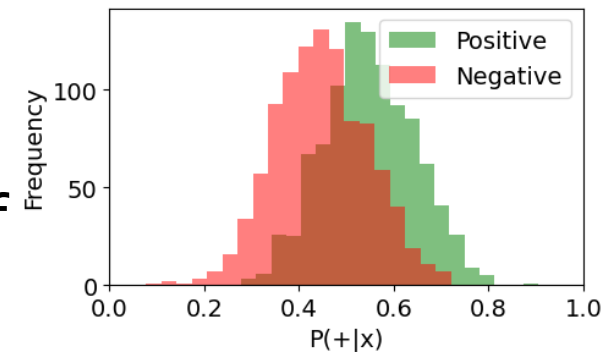
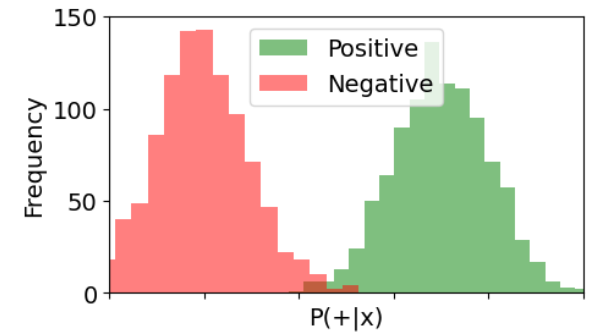
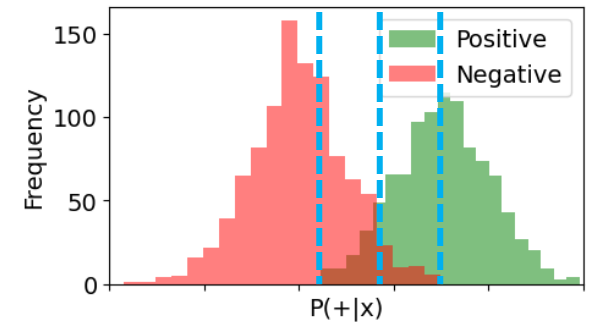
# ROC (Receiver Operating Characteristic)

- Developed in 1950s for signal detection theory to analyze noisy signals
  - characterizes the trade-off between positive hits and false alarms
- ROC curve plots
  - TPR, **True Positive Rate** (on the y-axis)  
$$\text{TPR} = \text{TP} / \text{P} = \text{TP} / (\text{TP} + \text{FN})$$
  
against
    - FPR, **False Positive Rate** (on the x-axis)  
$$\text{FPR} = \text{FP} / \text{N} = \text{FP} / (\text{FP} + \text{TN})$$



# ROC usage

- Classification models can generally assign a probability of membership to a class
  - E.g., in the binary case,  $P(+|x)$
  - $P(-|x) = 1 - P(+|x)$
- We can choose a *thresholding value*  $\tau$ 
  - If  $P(+|x) > \tau \rightarrow$  Predict "+"
  - Otherwise, predict "-"
- We can use the ROC curve to study how the choice of  $\tau$  affects the performance of the classifier

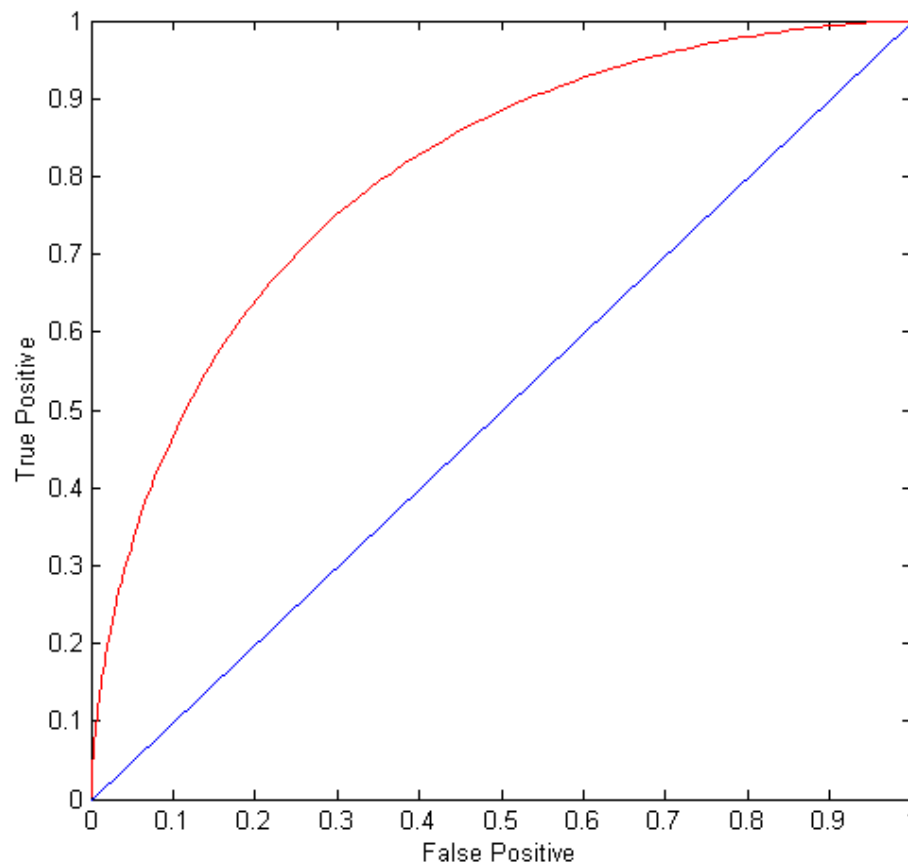




# ROC curve

(FPR, TPR)

- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (0,1): ideal
- Diagonal line
  - Random guessing
  - Below diagonal line
    - prediction is opposite of the true class





# How to build a ROC curve

Instance	$P(+ x)$	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

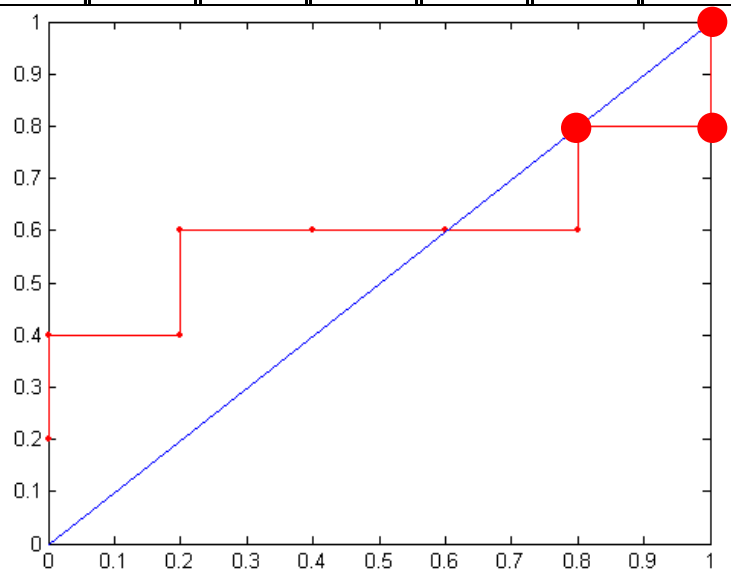
- Use classifier that produces posterior probability for each test instance  $P(+|x)$
- Sort the instances according to  $P(+|x)$  in decreasing order
- Apply threshold at each unique value of  $P(+|x)$
- Count the number of TP, FP, TN, FN at each threshold
  - TP rate
$$TPR = TP / (TP + FN)$$
  - FP rate
$$FPR = FP / (FP + TN)$$



# How to build a ROC curve

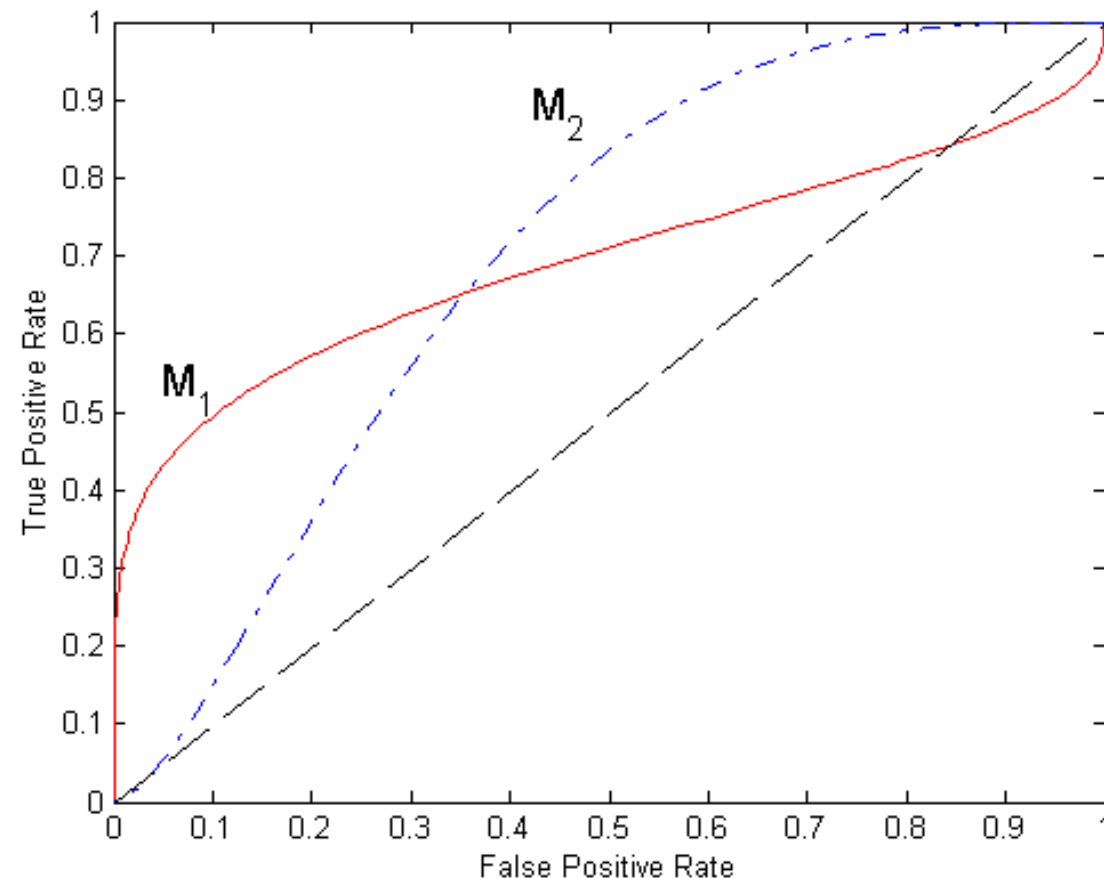
Class	+	-	+	-	-	-	+	-	+	+	
P(+ A)	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

ROC Curve





# Using ROC for Model Comparison



- No model consistently outperforms the other
  - $M_1$  is better for small FPR
  - $M_2$  is better for large FPR
- Area under ROC curve
  - Ideal  
Area = 1.0
  - Random guess  
Area = 0.5