Classification fundamentals

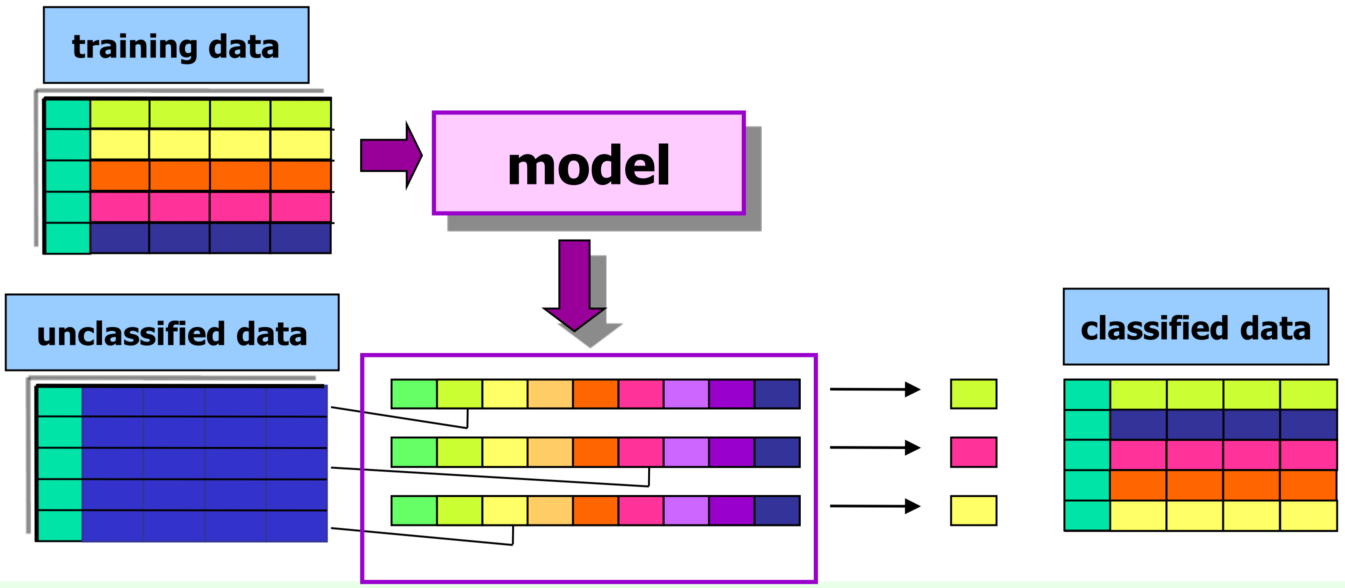
Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis

Politecnico di Torino

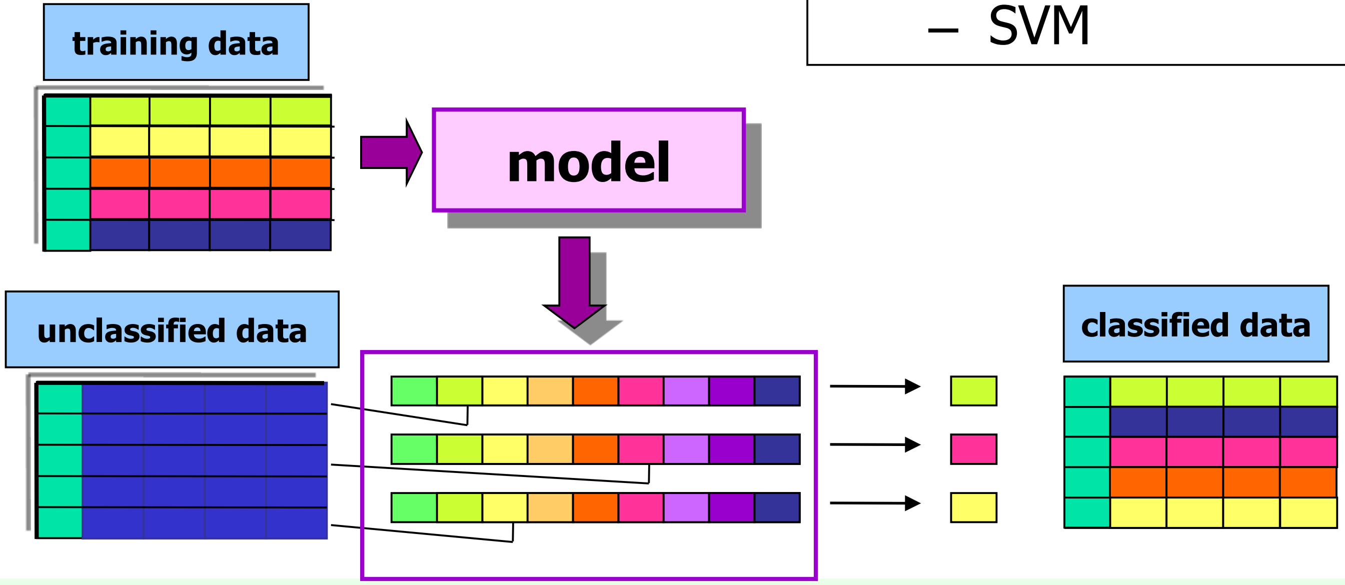
Classification

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



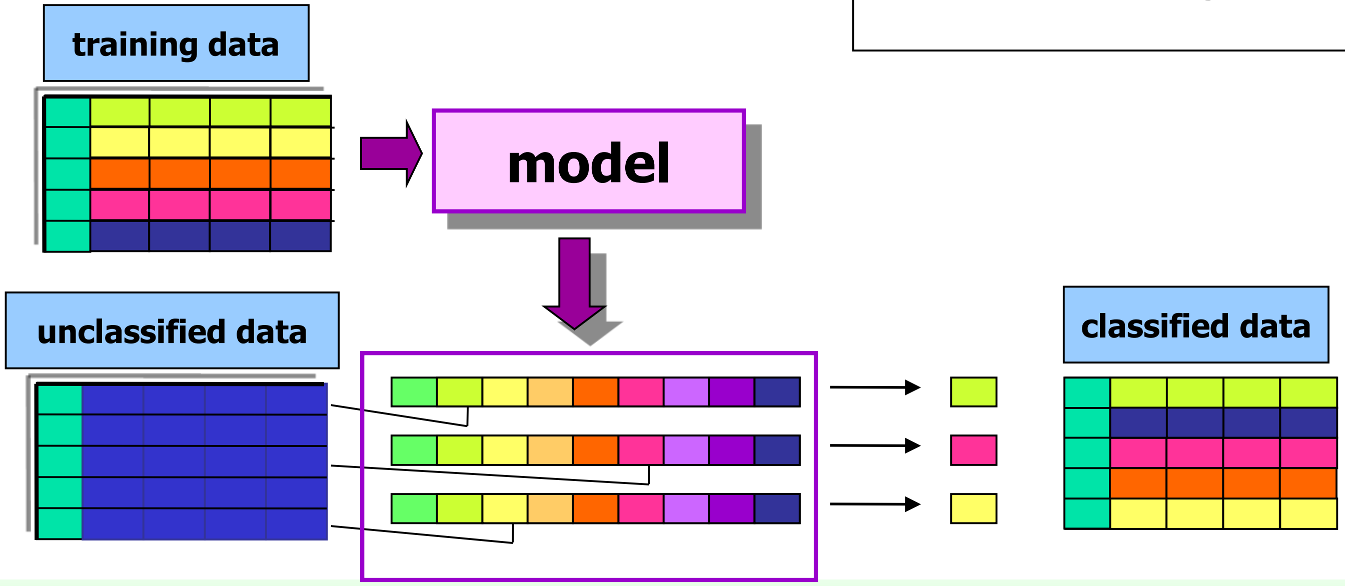
Classification

- Approaches
  - decision trees
  - bayesian classification
  - classification rules
  - neural networks
  - k-nearest neighbours
  - SVM



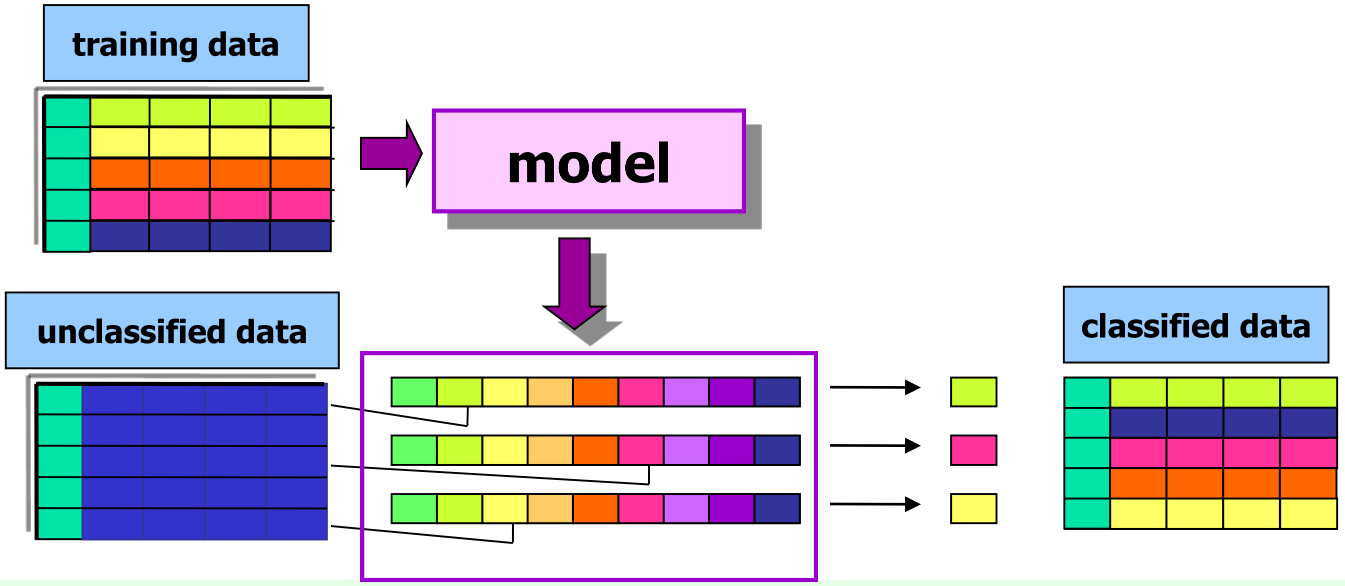
Classification

- Requirements
  - accuracy
  - interpretability
  - scalability
  - noise and outlier management



Classification

- Applications
  - detection of customer propension 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
- Efficiency
  - model building time
  - classification time
- Scalability
  - training set size
  - attribute number
- Robustness
  - noise, missing data
- Interpretability
  - model interpretability
  - model compactness

Decision trees

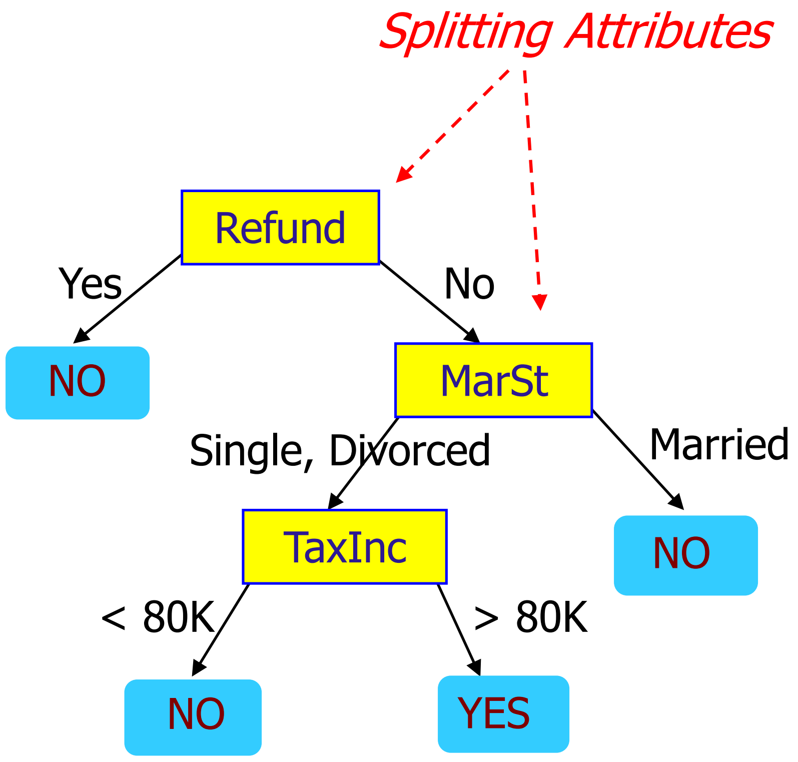
Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis

Politecnico di Torino

Example of decision tree

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



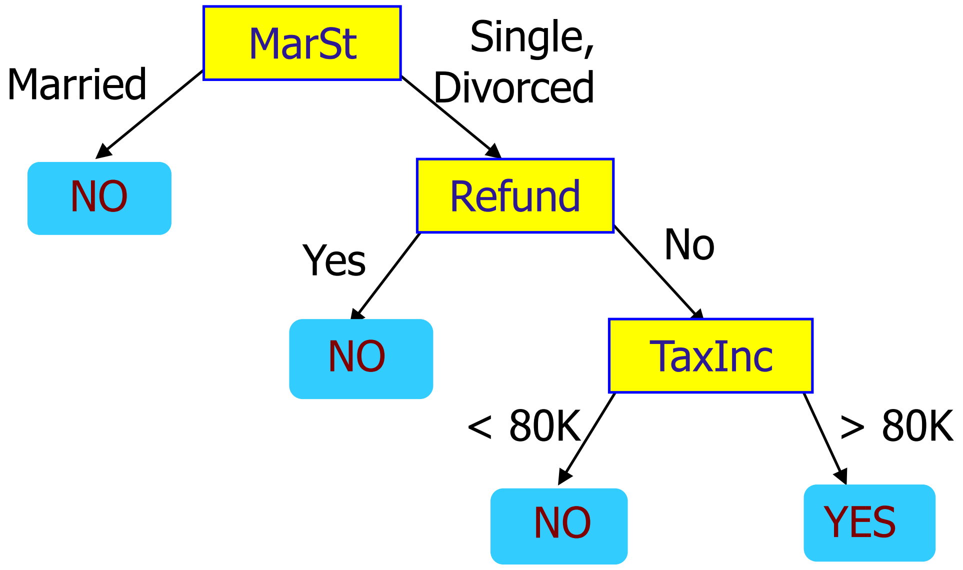
Training Data

Model: Decision Tree

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

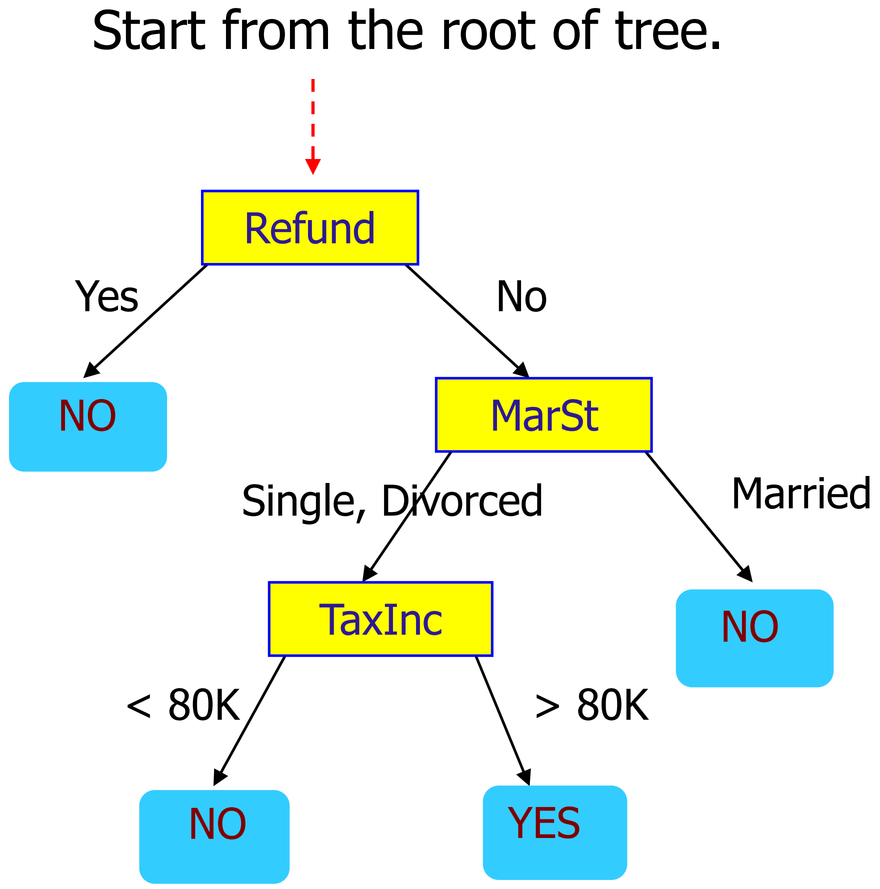
Another example of decision tree

Tid	Refund	Marital Status	Taxable Income	Cheat
	categorical	categorical	continuous	class
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



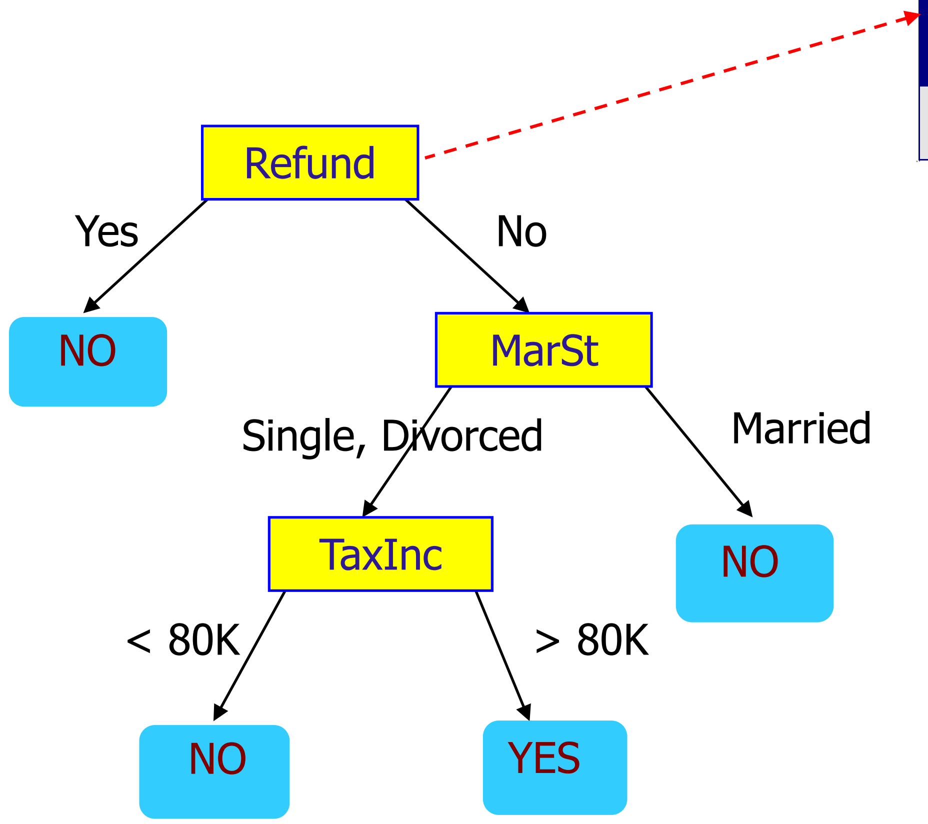
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

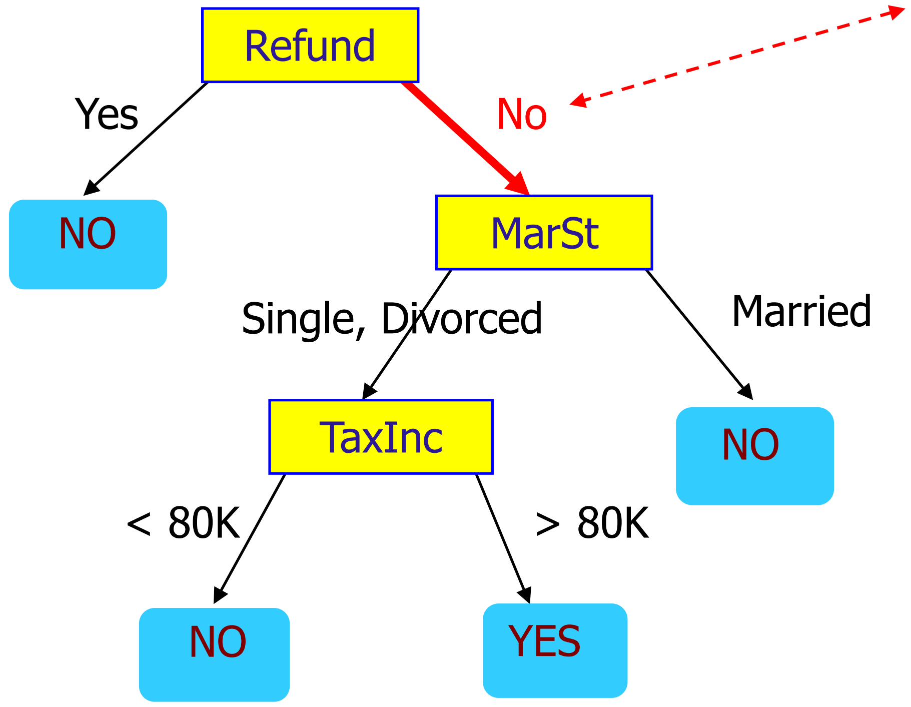
Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data



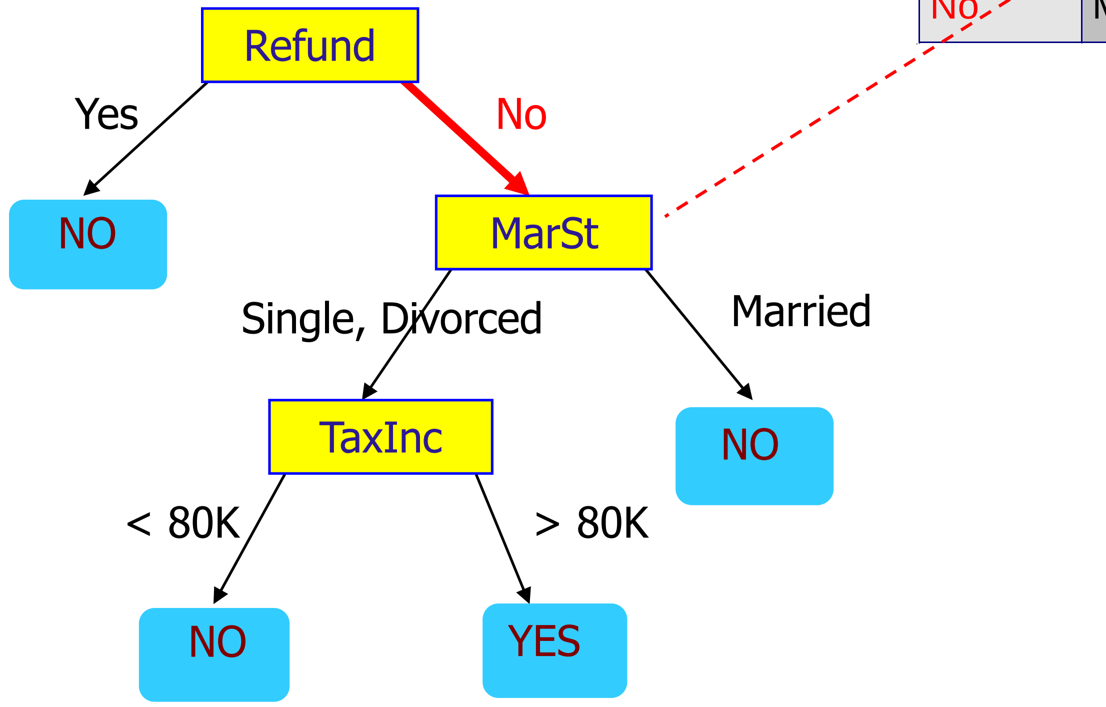
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

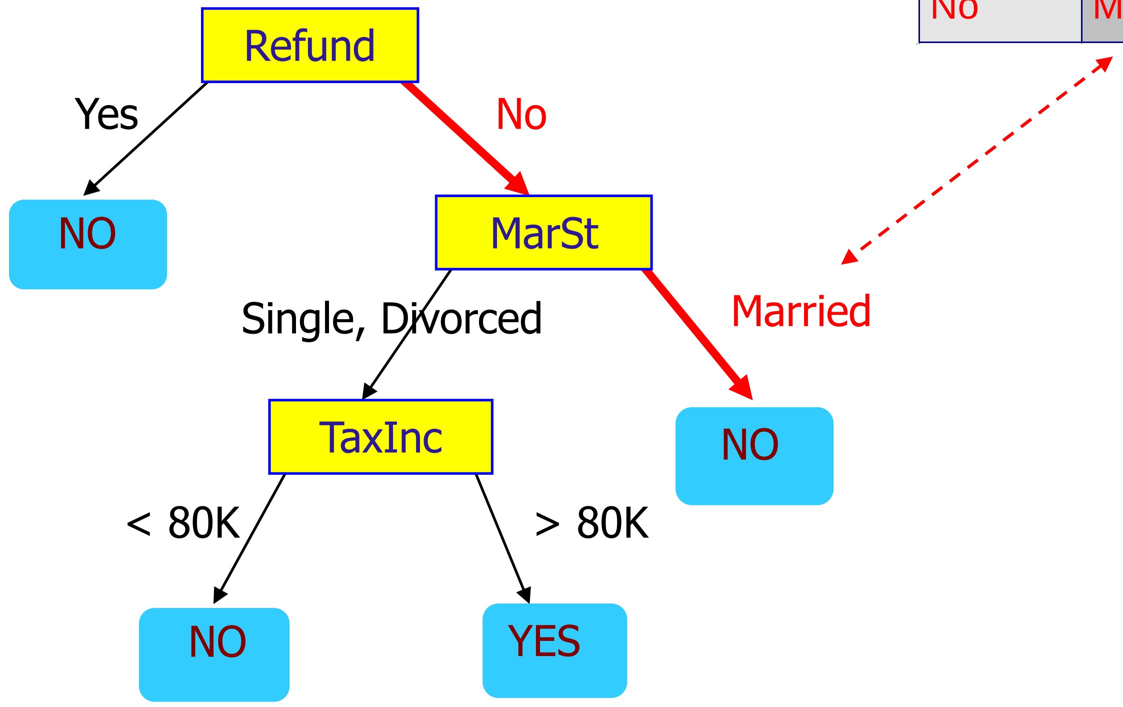
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

Apply Model to Test Data



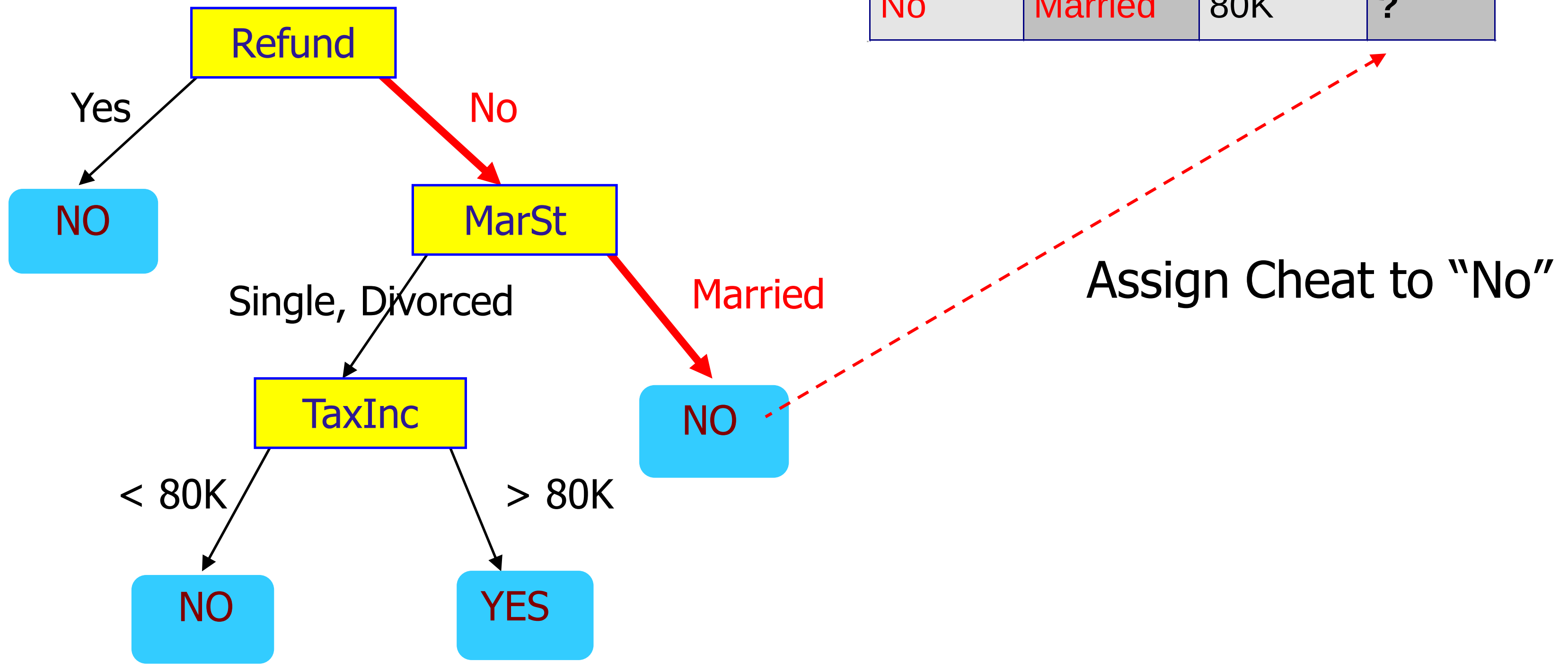
Test Data

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

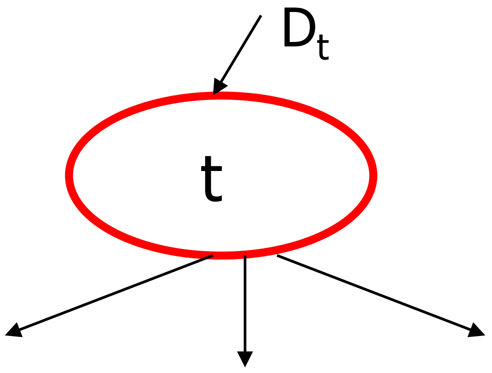
General structure of Hunt's algorithm

Basic steps

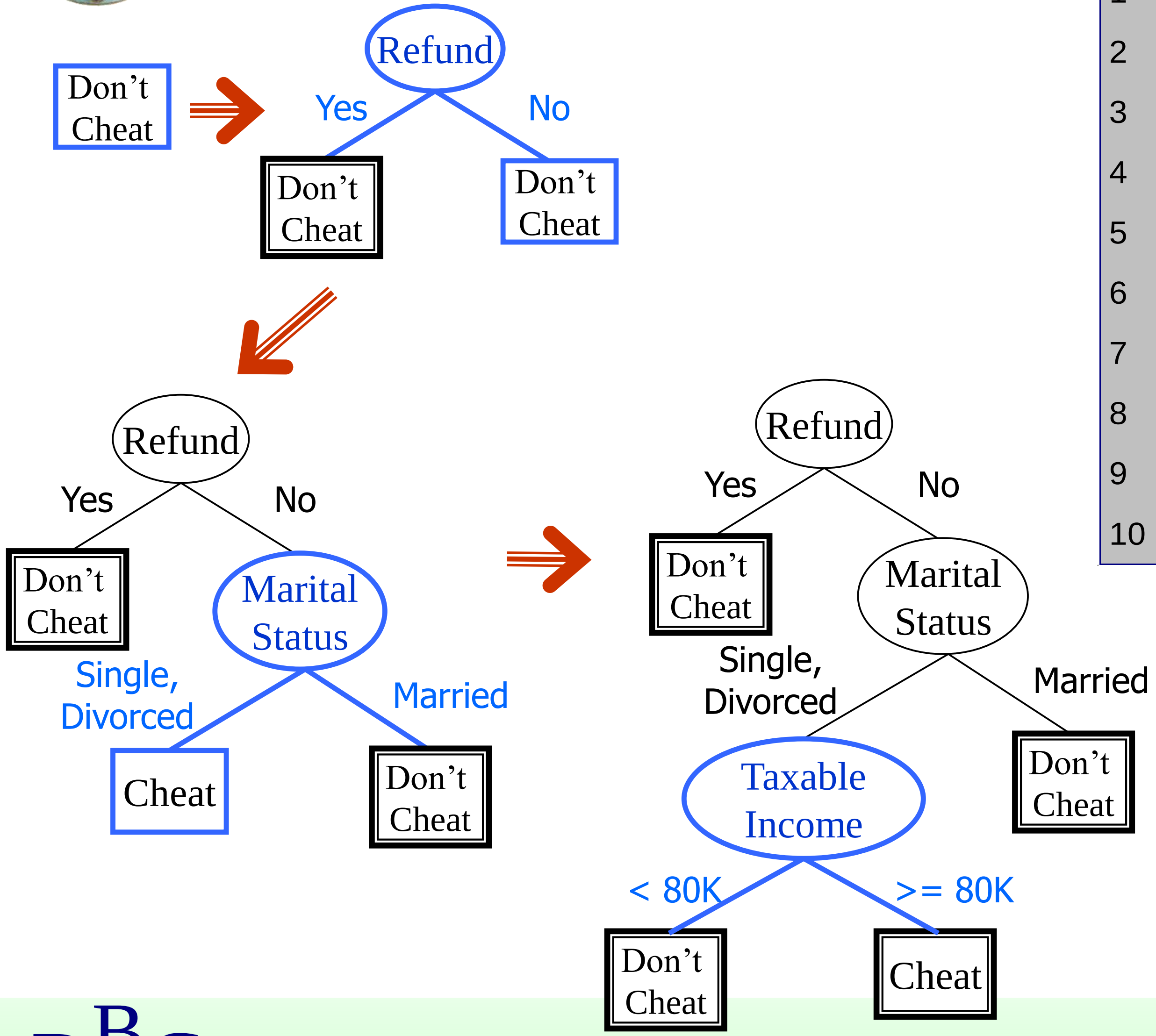
- If D_t contains records that belong to more than one class
  - select the "best" attribute A on which to split D_t and label node t as A
  - split D_t into smaller subsets and recursively apply the procedure to each subset
- If D_t contains records that belong to the same class y_t
  - then t is a 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

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

Decision tree induction

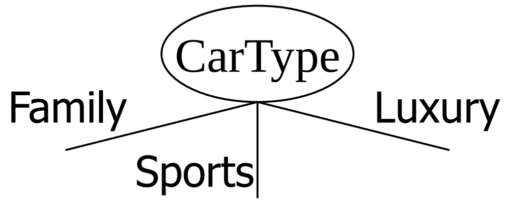
- Adopts a greedy strategy
  - "Best" attribute for the split is selected locally at each step
    - not a global optimum
- Issues
  - Structure of test condition
    - Binary split versus multiway split
  - Selection of the best attribute for the split
  - Stopping condition for the algorithm

Structure of test condition

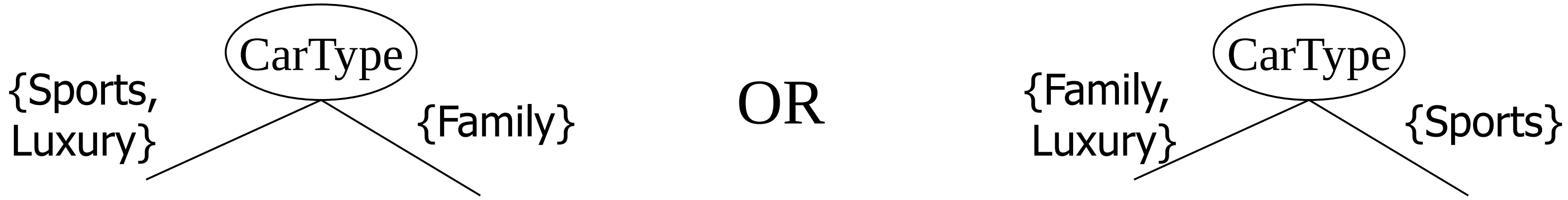
- Depends on attribute type
  - nominal
  - ordinal
  - continuous
- Depends on number of outgoing edges
  - 2-way split
  - multi-way split

Splitting on nominal attributes

- Multi-way split
  - use as many partitions as distinct values

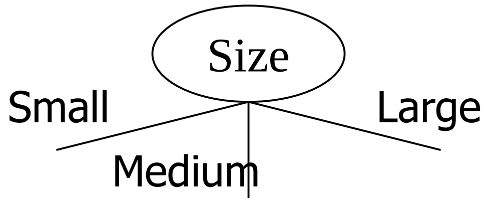


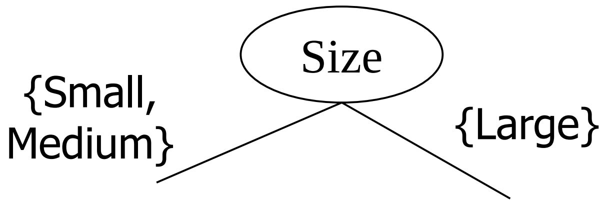
- Binary split
  - Divides values into two subsets
  - Need to find optimal partitioning



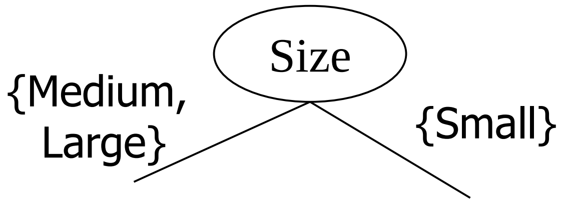
Splitting on ordinal attributes

- Multi-way split
  - use as many partitions as distinct values
- Binary split
  - Divides values into two subsets
  - Need to find optimal partitioning

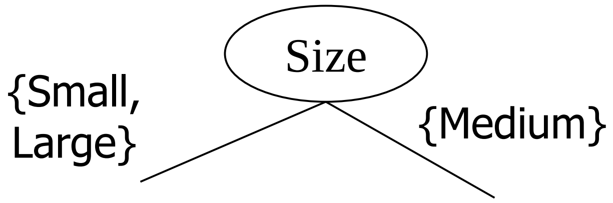




OR



What about this split?

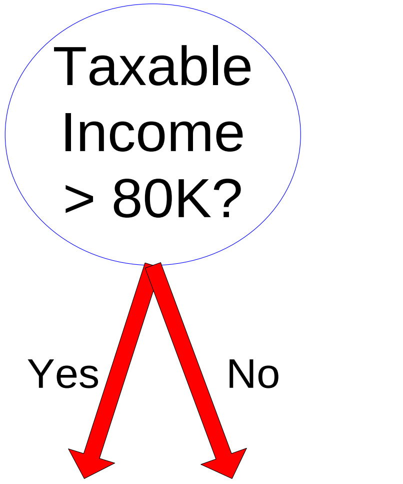


Splitting on continuous attributes

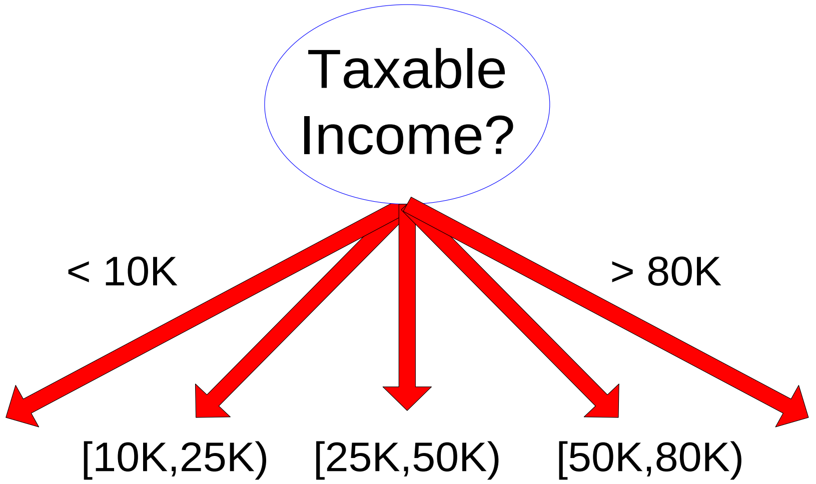
- Different techniques
  - Discretization to form an ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – discretize during tree induction

    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



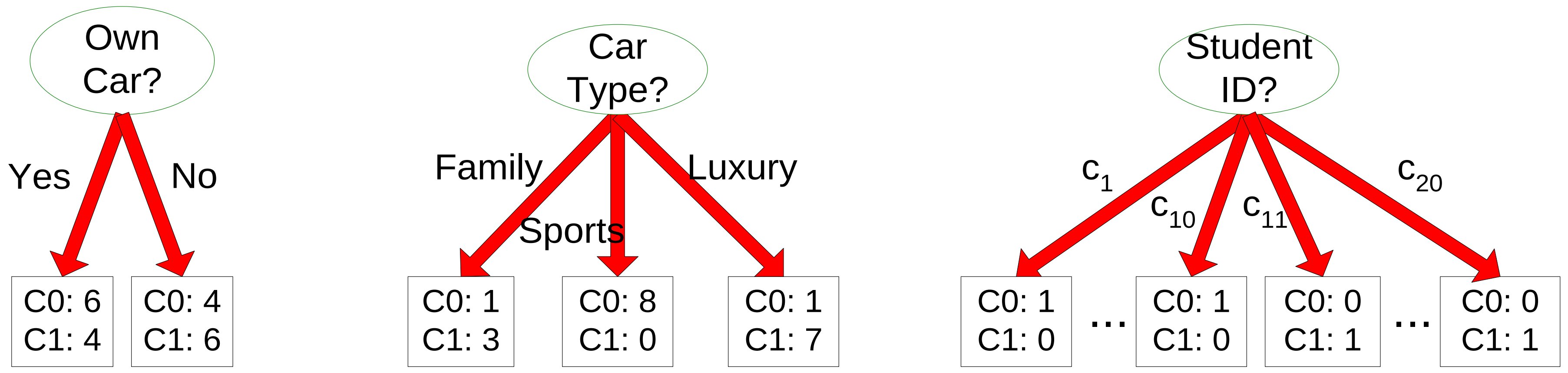
(i) Binary split



(ii) Multi-way split

Selection of the best attribute

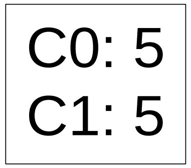
Before splitting: 10 records of class 0,
10 records of class 1



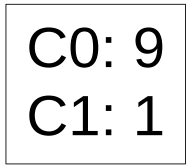
Which attribute (test condition) is the best?

Selection of the best attribute

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



Non-homogeneous, high degree of impurity



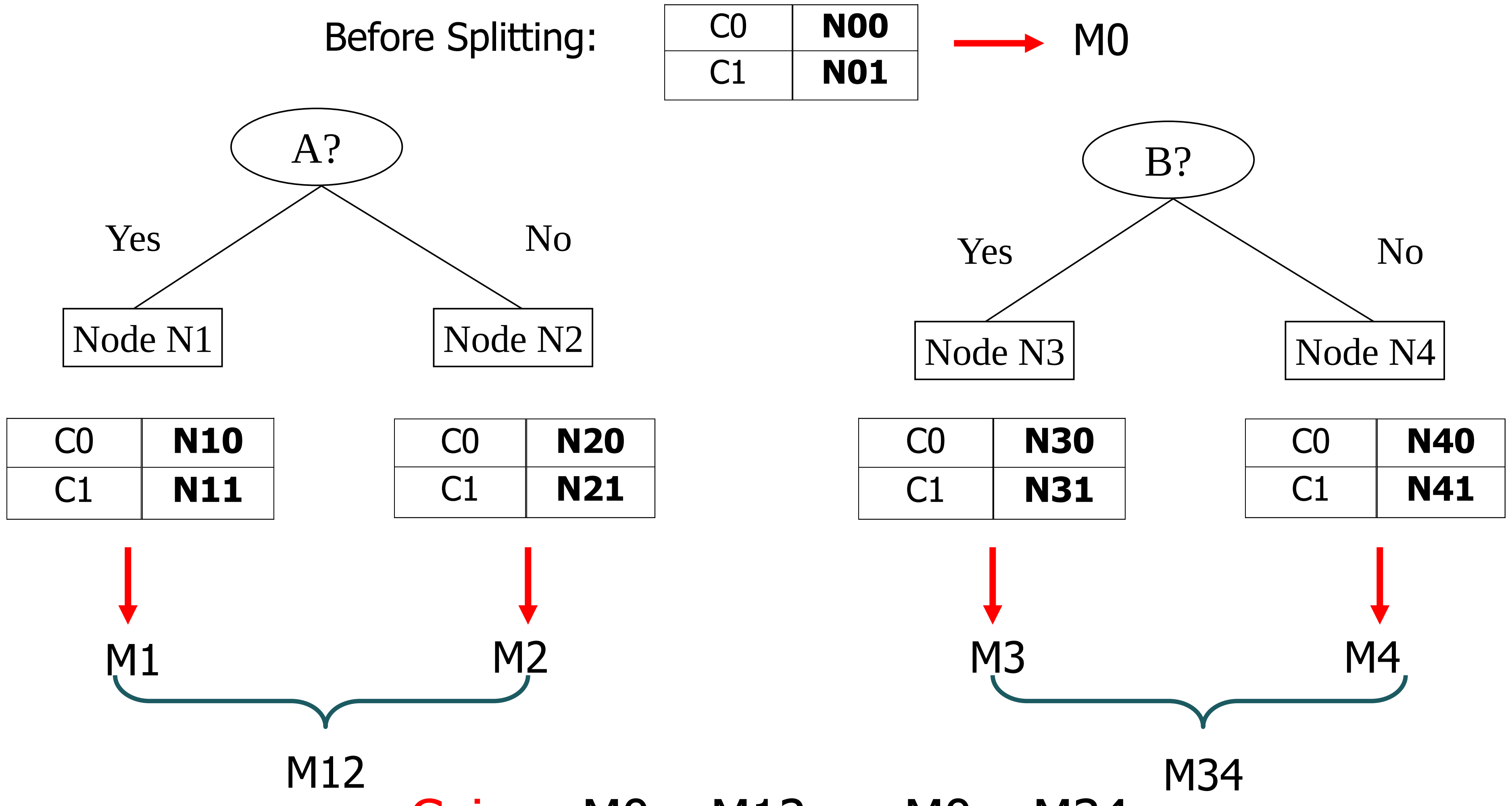
Homogeneous, low degree of impurity

Measures of node impurity

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

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

How to find the best attribute



Gain = M0 – M12 vs M0 – M34

GINI impurity measure

- Gini Index for a given node t

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

$p(j \mid t)$ 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	**0**
C2	**6**
Gini=0.000	

C1	**1**
C2	**5**
Gini=0.278	

C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

Examples for computing GINI

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

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

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

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

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

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

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

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

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

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

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

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

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

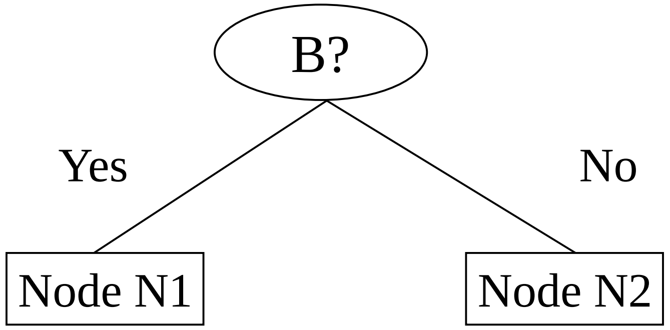
where

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**
Gini = 0.500	

Gini(N1)
$= 1 - (5/7)^2 - (2/7)^2$
$= 0.408$

Gini(N2)
$= 1 - (1/5)^2 - (4/5)^2$
$= 0.32$

	N1	N2
C1	**5**	**1**
C2	**2**	**4**
Gini=?		

Gini(split on B)
$= 7/12 * 0.408 +$
$5/12 * 0.32$
$= 0.371$

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)

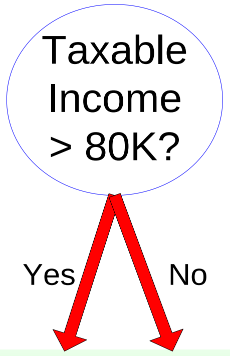
	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	**0.400**	

	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 ≥ v

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



From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

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

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
	Taxable Income									
Sorted Values →	60	70	75	85	90	95	100	120	125	220

Split Positions →	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	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	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		***0.300***		0.343		0.375		0.400		0.420	

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

Entropy impurity measure (INFO)

- Entropy at a given node t

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

$p(j \mid t)$ 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	**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) $\log_2$ (5/6) = 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

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

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

Classification error impurity measure

- Classification error at a node t

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

- Measures misclassification error made by a node
  - Maximum (1 - 1/n_c) when records are equally distributed among all classes, implying least interesting information
  - Minimum (0.0) when all records belong to one class, implying most interesting information

Examples for computing error

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

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

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

Error = 1 – max (0, 1) = 1 – 1 = 0

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

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

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

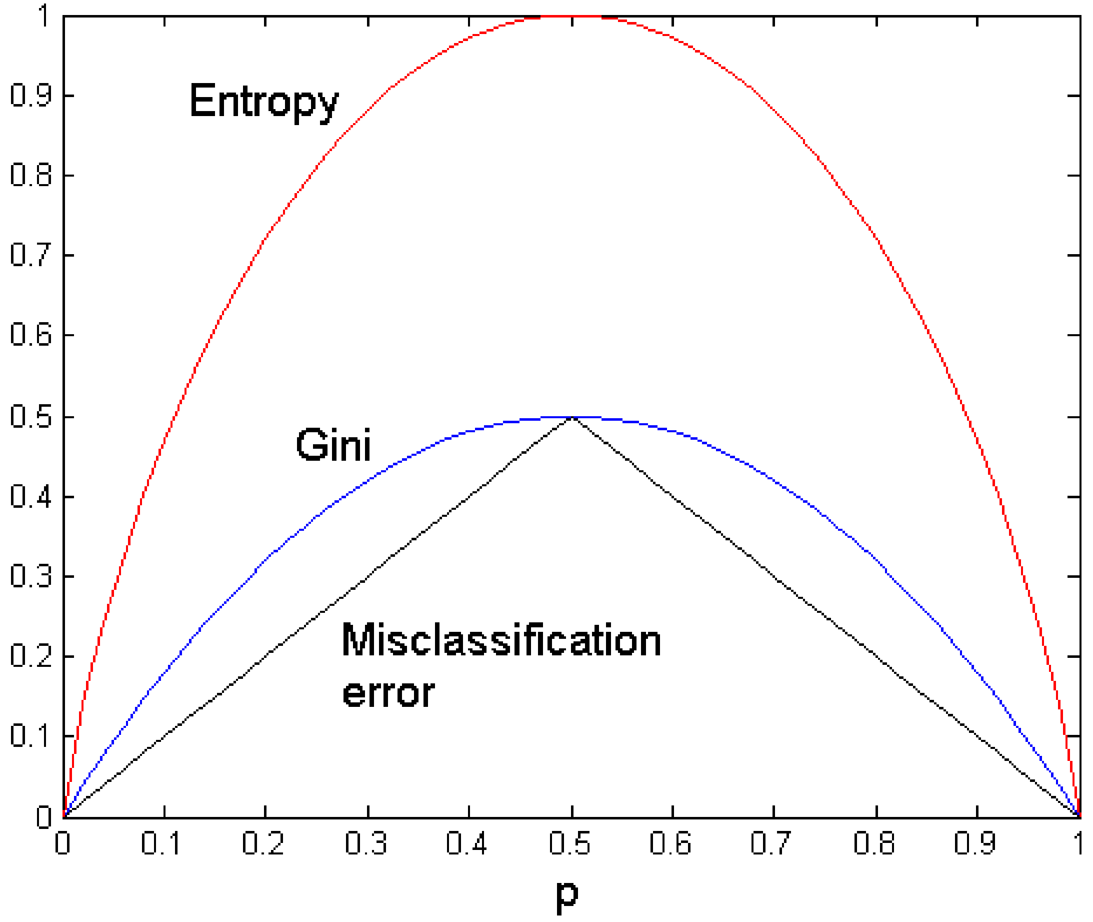
C1	**2**
C2	**4**

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

Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Comparison among splitting criteria

For a 2-class problem

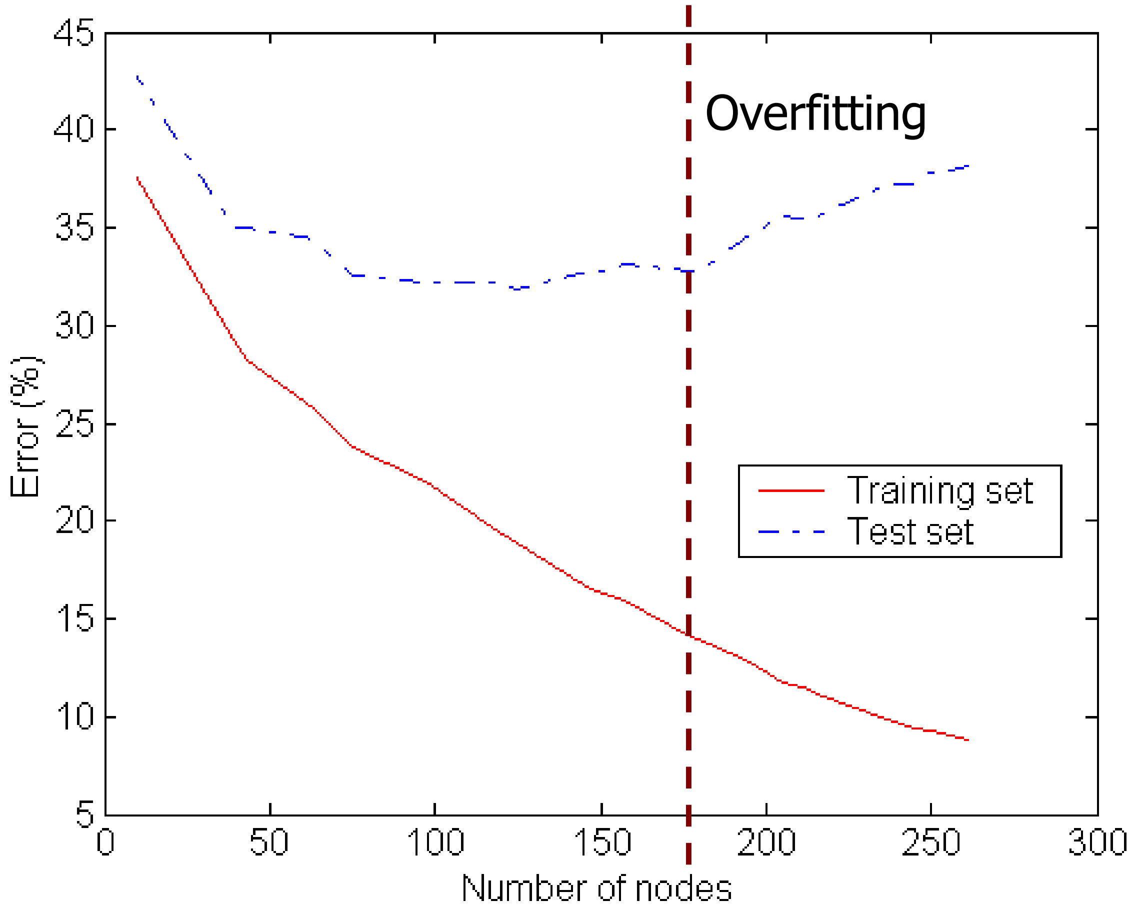


From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

Stopping Criteria for Tree Induction

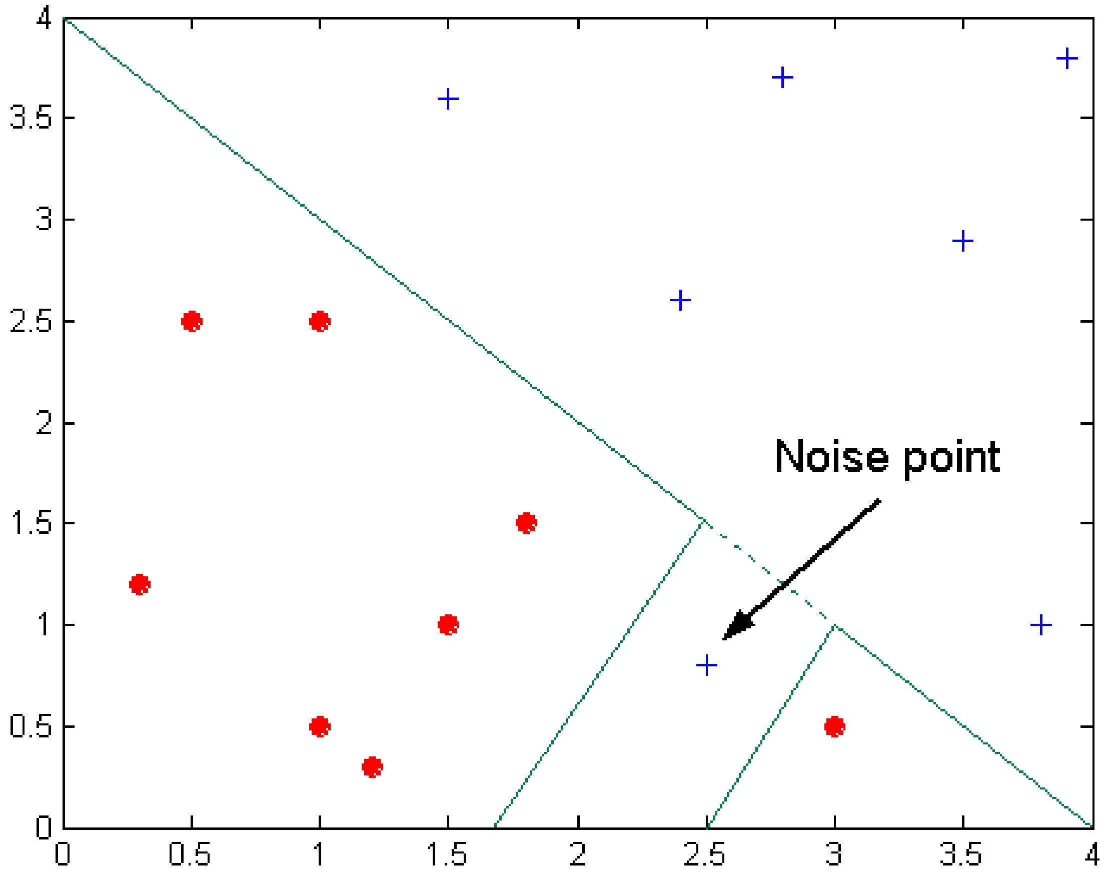
- Stop expanding a node when all the records belong to the same class
- Stop expanding a node when all the records have similar attribute values
- Early termination
  - Pre-pruning
  - Post-pruning

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 χ^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 instance with missing value to child nodes
  - Affect how a test instance with missing value is classified

Other issues

- Data Fragmentation
- Search Strategy
- Expressiveness
- Tree Replication

Search strategy

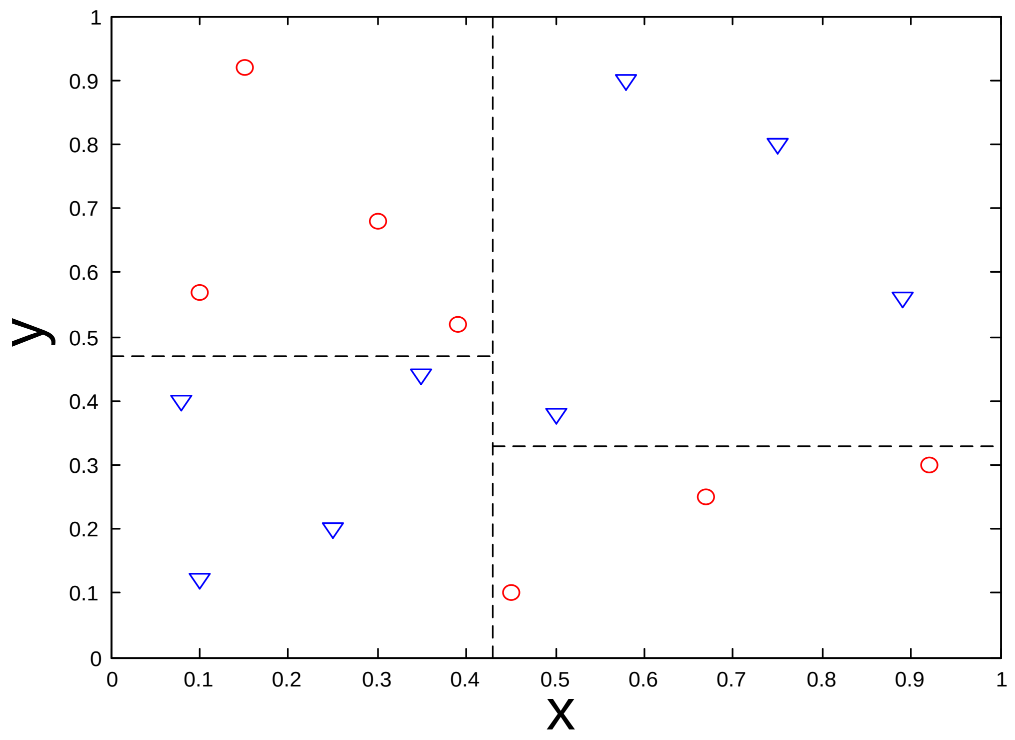
- Finding an optimal decision tree is NP-hard
- The algorithm presented so far uses a greedy, top-down, recursive partitioning strategy to induce a reasonable solution
- Other strategies?
  - Bottom-up
  - Bi-directional

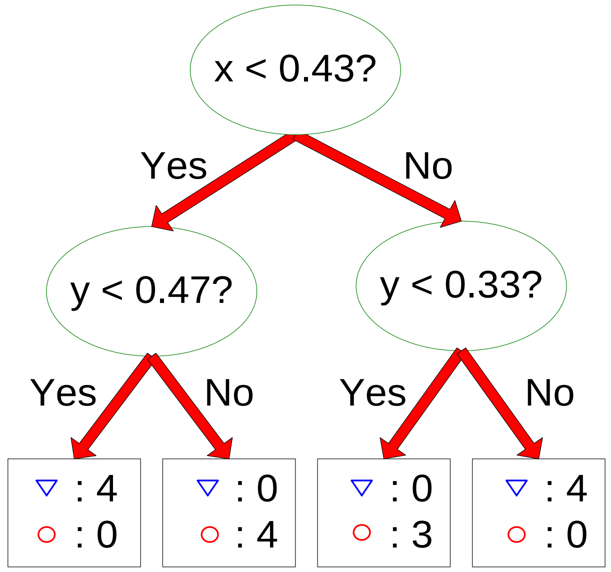
Expressiveness

- Decision tree provides expressive representation for learning discrete-valued function
  - But they do not generalize well to certain types of Boolean functions
    - Example: parity function:
      - Class = 1 if there is an even number of Boolean attributes with truth value = True
      - Class = 0 if there is an odd number of Boolean attributes with truth value = True
    - For accurate modeling, must have a complete tree

- Not expressive enough for modeling continuous variables
  - Particularly when test condition involves only a single attribute at-a-time

Decision boundary

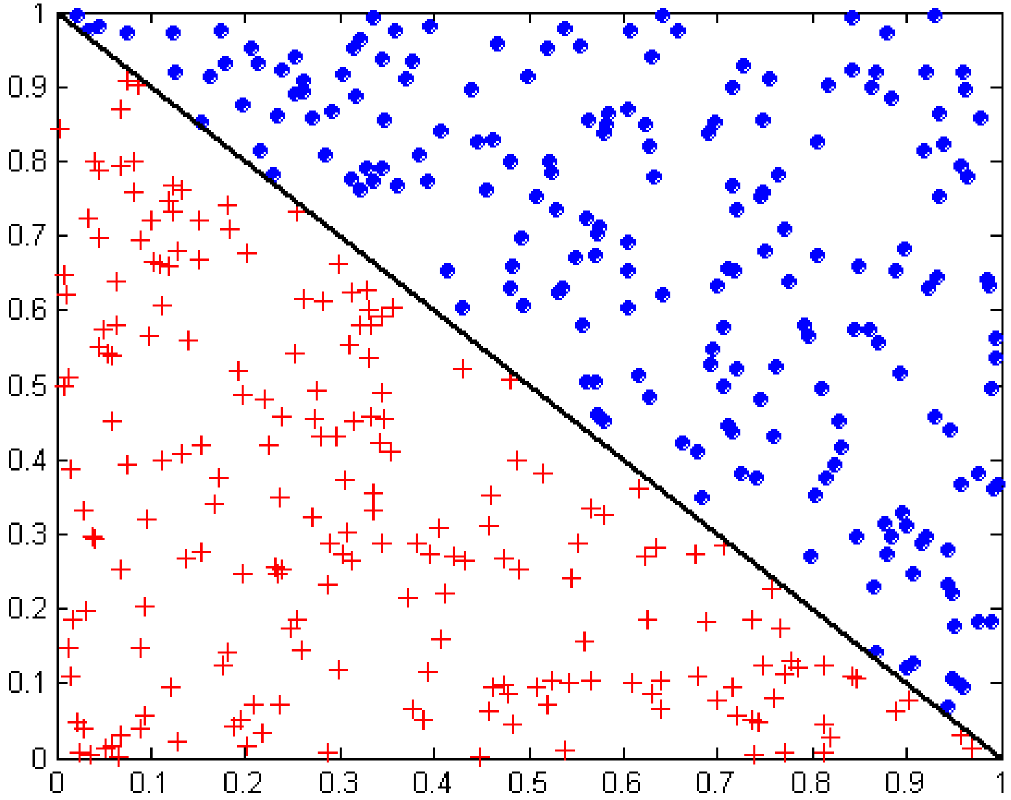


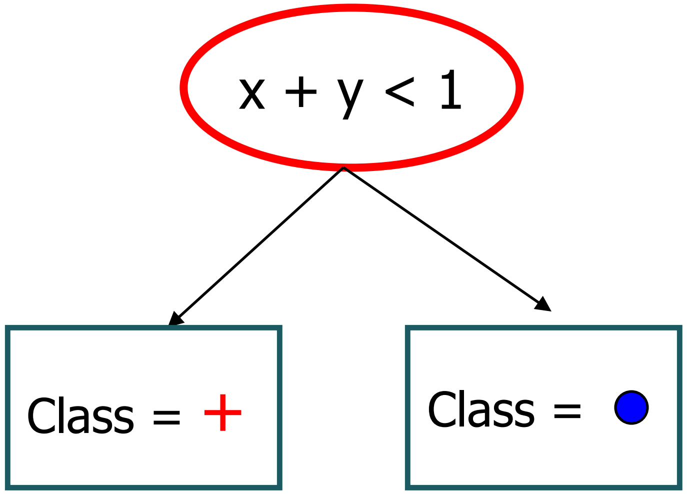


- Border line between two neighboring regions of different classes is known as decision boundary
- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

Oblique decision trees





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

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

Decision Tree Based Classification

- Advantages
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Accuracy is comparable to other classification techniques for many simple data sets
- Disadvantages
  - accuracy may be affected by missing data

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

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

Bibliography: Hastie, Tibshirani, Friedman, The Elements of Statistical Learning, Springer, 2009

Random Forest

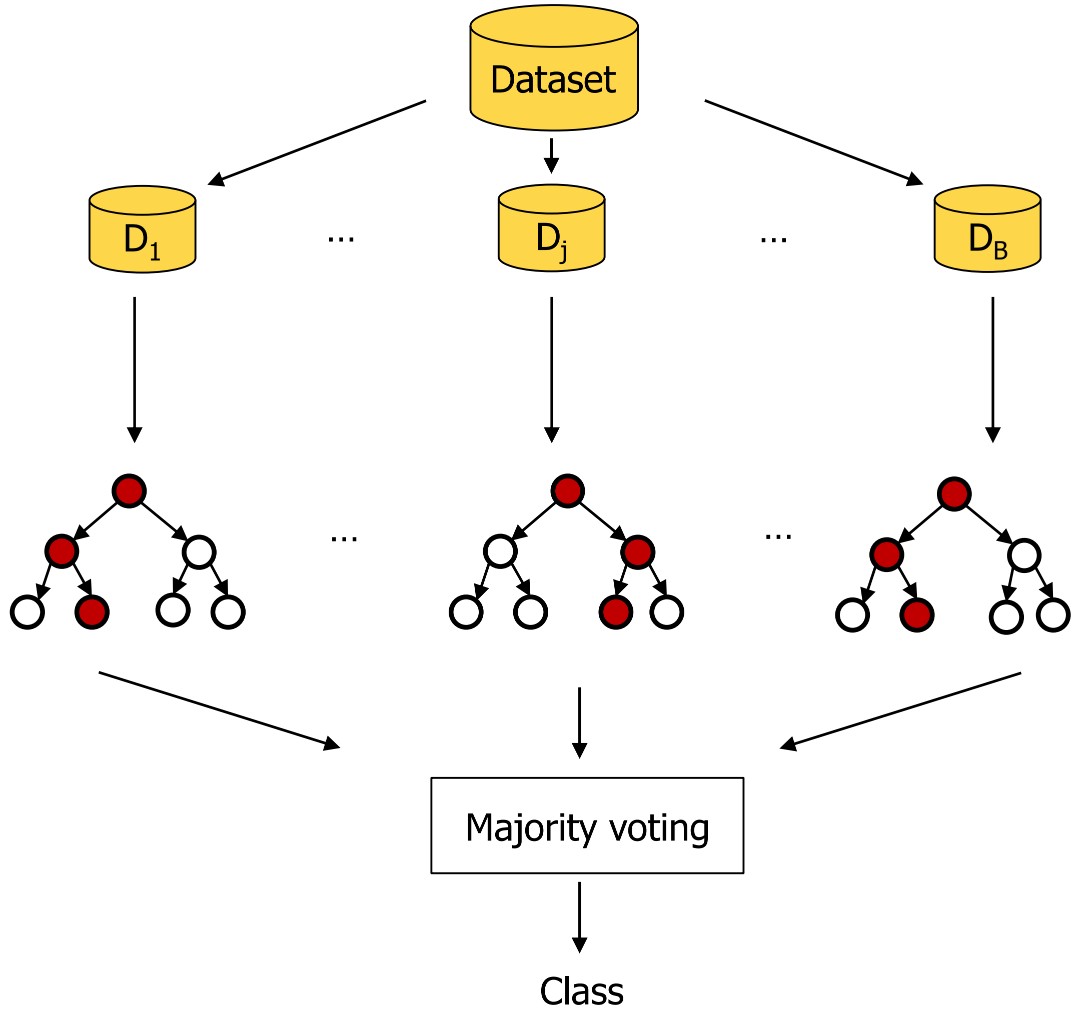
Original Training data

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, ..., 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 could be selected as best attribute for the split

Random Forest – Algorithm Recap

- Given a training set *D* of *n* instances with p features
- For b = 1, ..., 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 ≈ $\sqrt{p}$)
      - the best split is computed among these m features
- Class is assigned by majority voting among the B predictions

Random Forest

- Strong points
  - higher accuracy than decision trees
  - fast training phase
  - robust to noise and outliers
  - provides global feature importance, i.e. an estimate of which features are important in the classification
- Weak points
  - results can be difficult to interpret
    - A prediction is given by hundreds of trees
      - but at least we have an indication through feature importance

Rule-based classification

Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis

Politecnico di Torino

Model evaluation

Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis

Politecnico di Torino

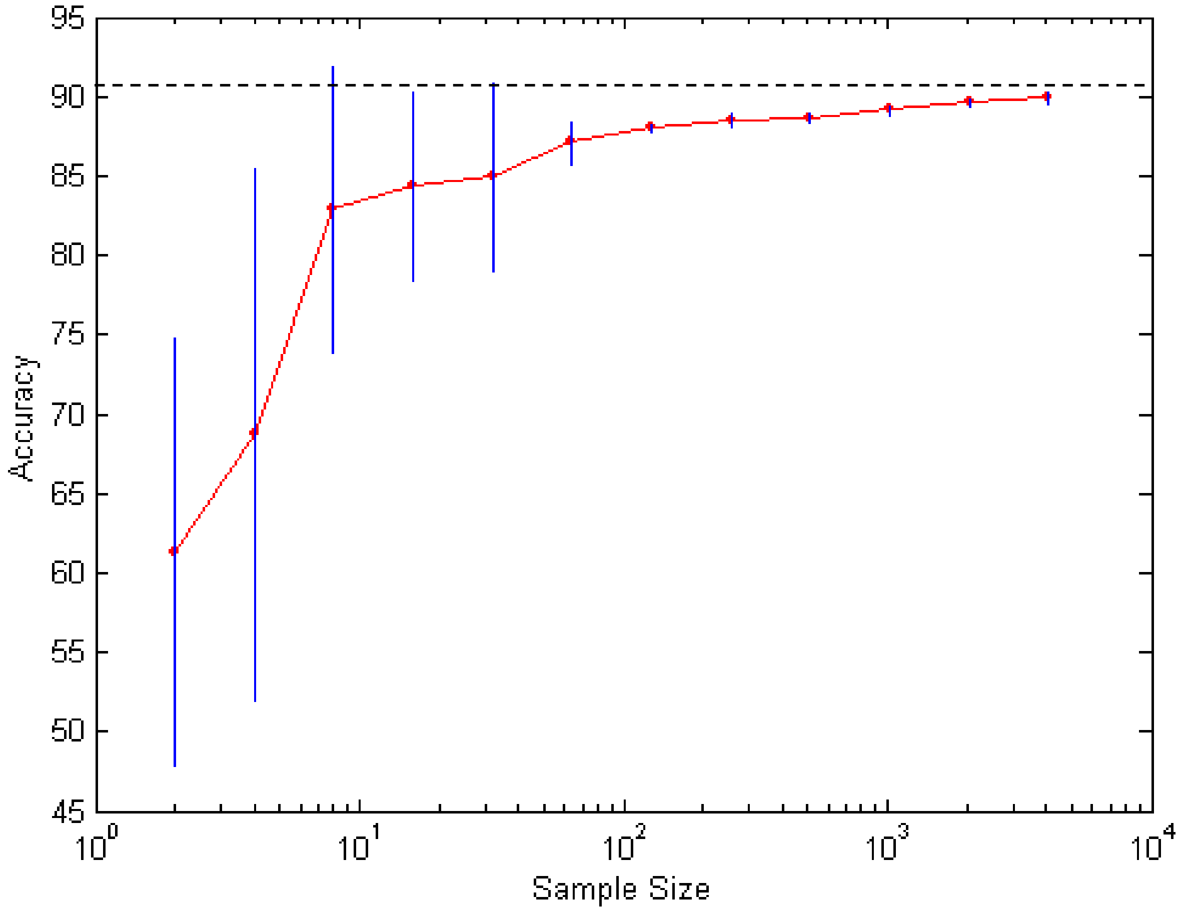
Model evaluation

- 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 sample size
- Requires a sampling schedule for creating learning curve:
  - Arithmetic sampling (Langley, et al)
  - Geometric sampling (Provost et al)

Effect of small sample size:

- Bias in the estimate
- Variance of estimate

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

Methods of estimation

- Partitioning labeled data in
  - training set for model building
  - test set for model evaluation
- Several partitioning techniques
  - holdout
  - cross validation
- Stratified sampling to generate partitions
  - without replacement
- Bootstrap
  - Sampling with replacement

Holdout

- Fixed partitioning
  - reserve 2/3 for training and 1/3 for testing
- 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

Metrics for model evaluation

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

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

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

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

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

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

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

- Maximize

$$\text{F - measure (F)} = \frac{2rp}{r + p}$$

Class specific measures

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

$$\text{Precision (p)} = \frac{a}{a+c}$$

$$\text{Recall (r)} = \frac{a}{a+b}$$

$$\text{F-measure (F)} = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$

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)

    TPR = TP/(TP+FN)

  against
  - FPR, False Positive Rate (on the x-axis)

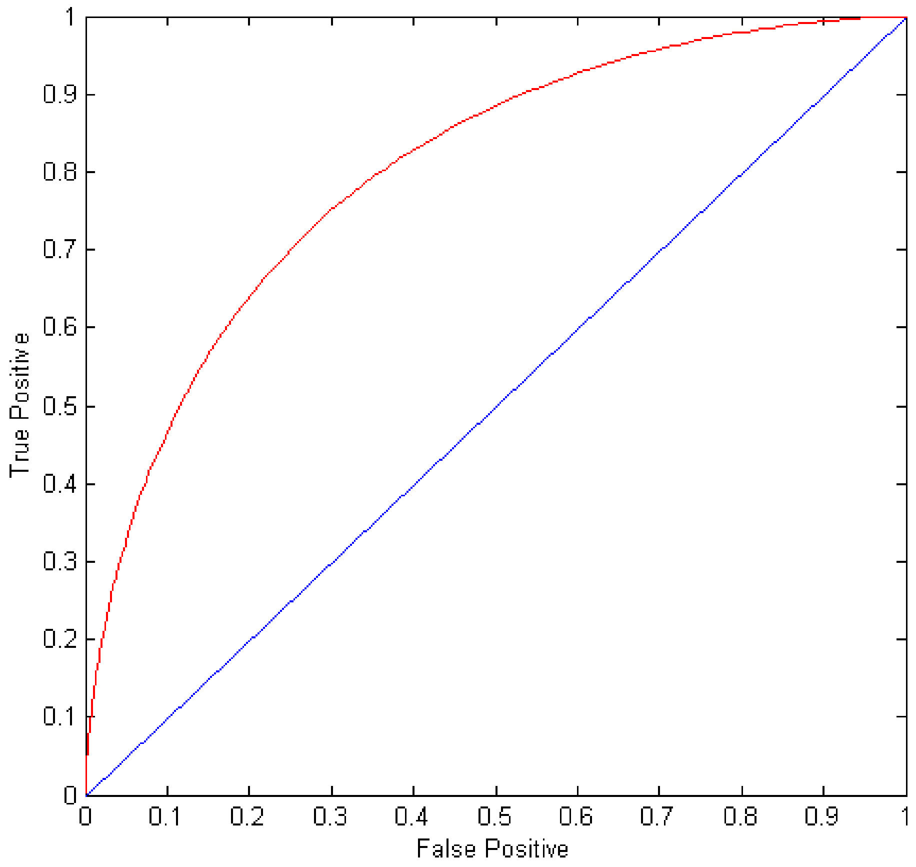
    FPR = FP/(FP + TN)

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



From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

How to build a ROC curve

Instance	P(+|A)	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(+|A)
- Sort the instances according to P(+|A) in decreasing order
- Apply threshold at each unique value of P(+|A)
- Count the number of TP, FP, TN, FN at each threshold
  - TP rate

    TPR = TP/(TP+FN)
  - FP rate

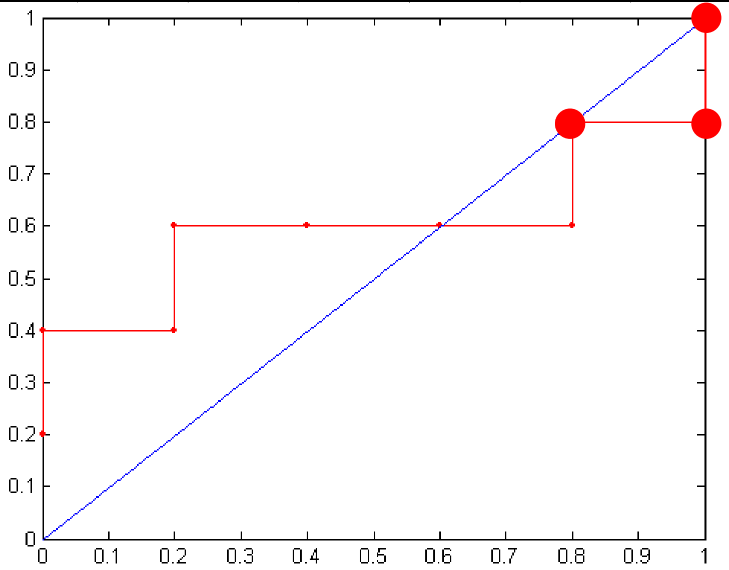
    FPR = FP/(FP + TN)

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

How to build a ROC curve

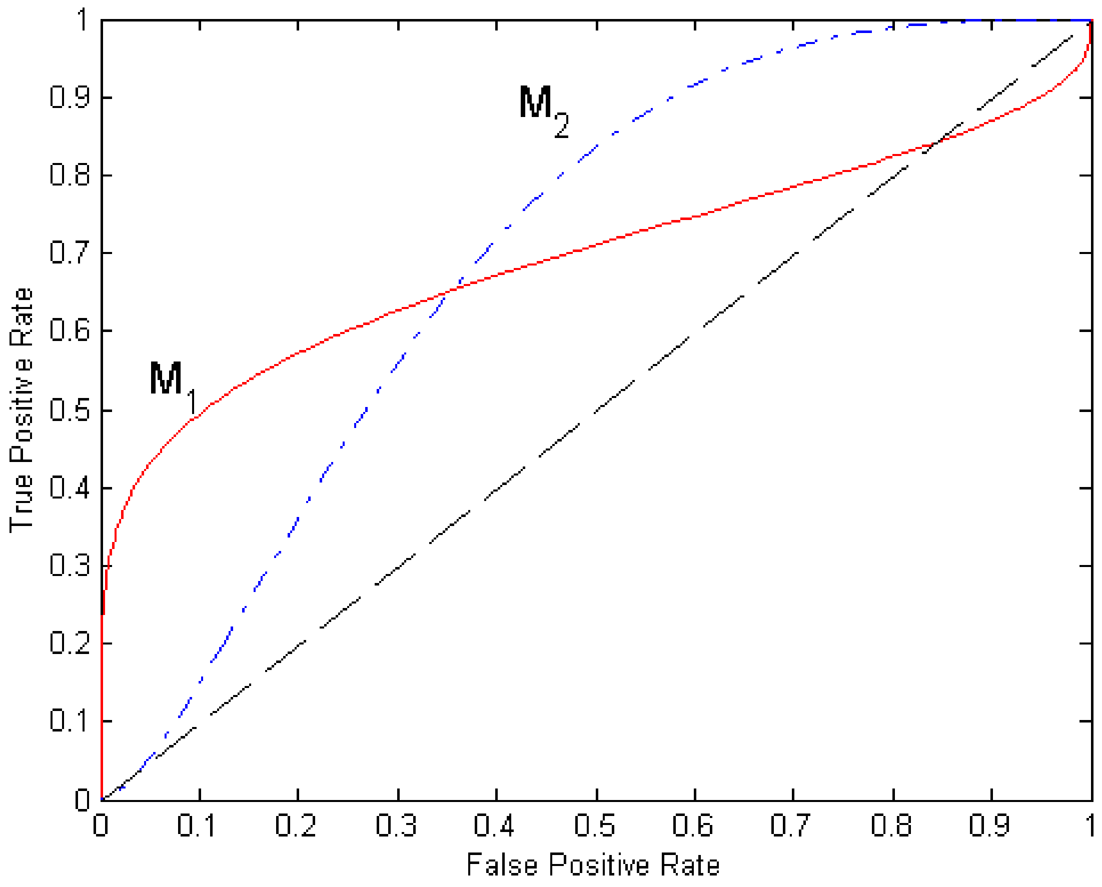
Class	+	-	+	-	-	-	+	-	+	+	
P(+|A)	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

ROC Curve



From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

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

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

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 attributes
    - y is the class label
  - *LHS*: rule antecedent or condition
  - *RHS*: rule consequent
- Examples of classification rules
  - (Blood Type=Warm) $\wedge$ (Lay Eggs=Yes) $\rightarrow$ Birds
  - (Taxable Income < 50K) $\wedge$ (Refund=Yes) $\rightarrow$ Cheat=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

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

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 => Bird

Rule R3 covers the grizzly bear => 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

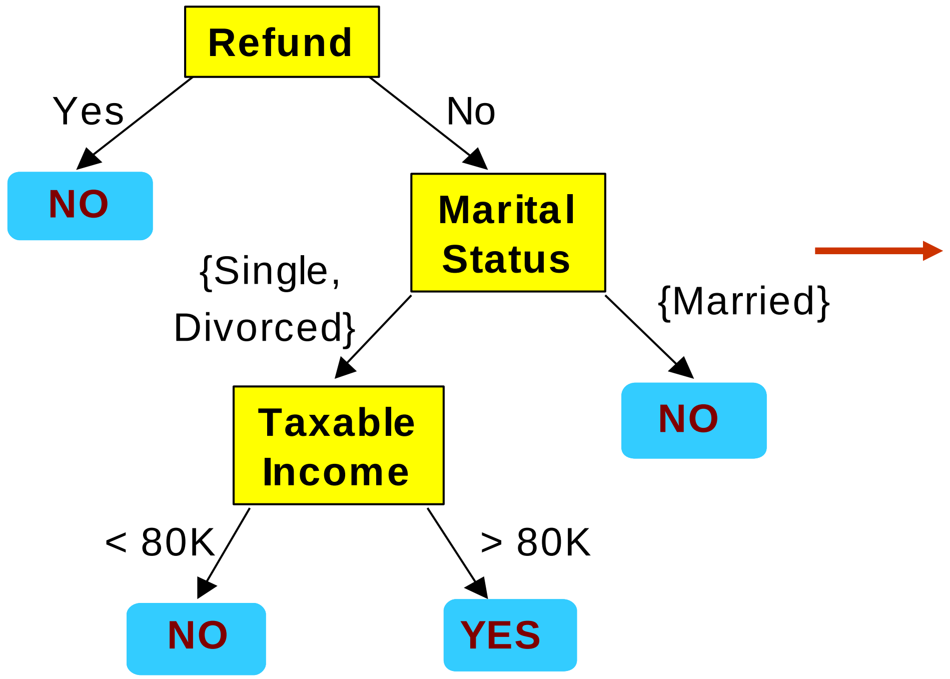
A turtle triggers both R4 and R5

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

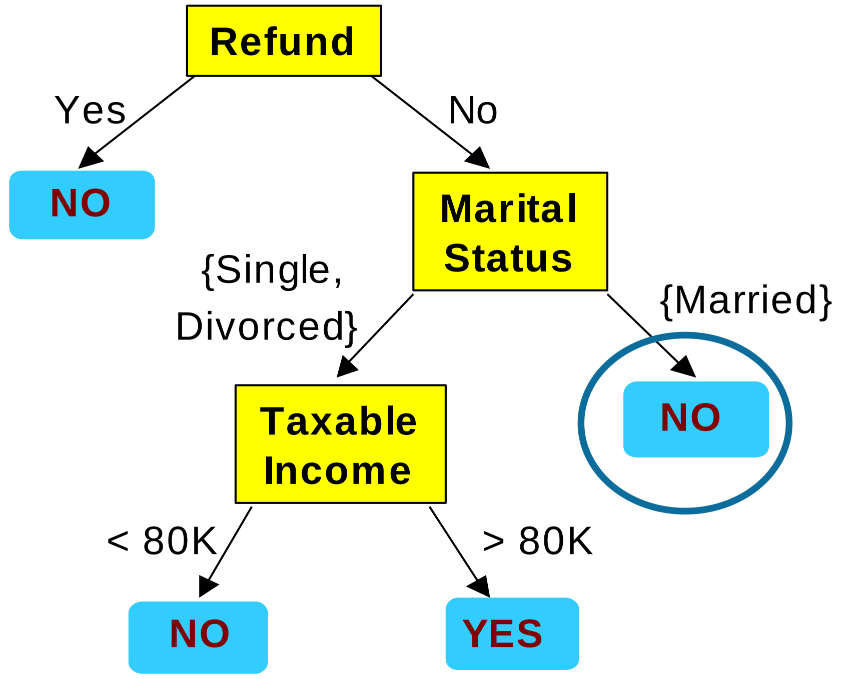
(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes

(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: (Refund=No) $\wedge$ (Status=Married) $\rightarrow$ No

Simplified Rule: (Status=Married) $\rightarrow$ 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	?

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

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

Advantages of rule-based classifiers

- As highly expressive as decision trees
- Easy to interpret
- Easy to generate
- Can classify new instances rapidly
- Performance comparable to decision trees

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

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

  - 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

Associative classification

- Strong points
  - interpretable model
  - higher accuracy than decision trees
    - correlation among attributes is considered
  - efficient classification
  - unaffected by missing data
  - good scalability in the training set size
- Weak points
  - rule generation may be slow
    - it depends on support threshold
  - reduced scalability in the number of attributes
    - rule generation may become unfeasible

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 = $<x_1,\ldots,x_k>$ 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. $P(x_1,\ldots,x_k|C)$?
- Naïve hypothesis

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

  - *statistical independence* of attributes $x_1,\ldots,x_k$
  - 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

From: Han, Kamber,"Data mining; Concepts and Techniques", Morgan Kaufmann 2006

Bayesian classification: Example

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

P(p) = 9/14
P(n) = 5/14

From: Han, Kamber,"Data mining; Concepts and Techniques", Morgan Kaufmann 2006

Bayesian classification: Example

- Data to be labeled

  X = <rain, hot, high, false>

- For class p

  P(X|p)·P(p) =
  = P(rain|p)·P(hot|p)·P(high|p)·P(false|p)·P(p)
  = 3/9·2/9·3/9·6/9·9/14 = 0.010582

- For class n

  P(X|n)·P(n) =
  = P(rain|n)·P(hot|n)·P(high|n)·P(false|n)·P(n)
  = 2/5·2/5·4/5·2/5·5/14 = 0.018286

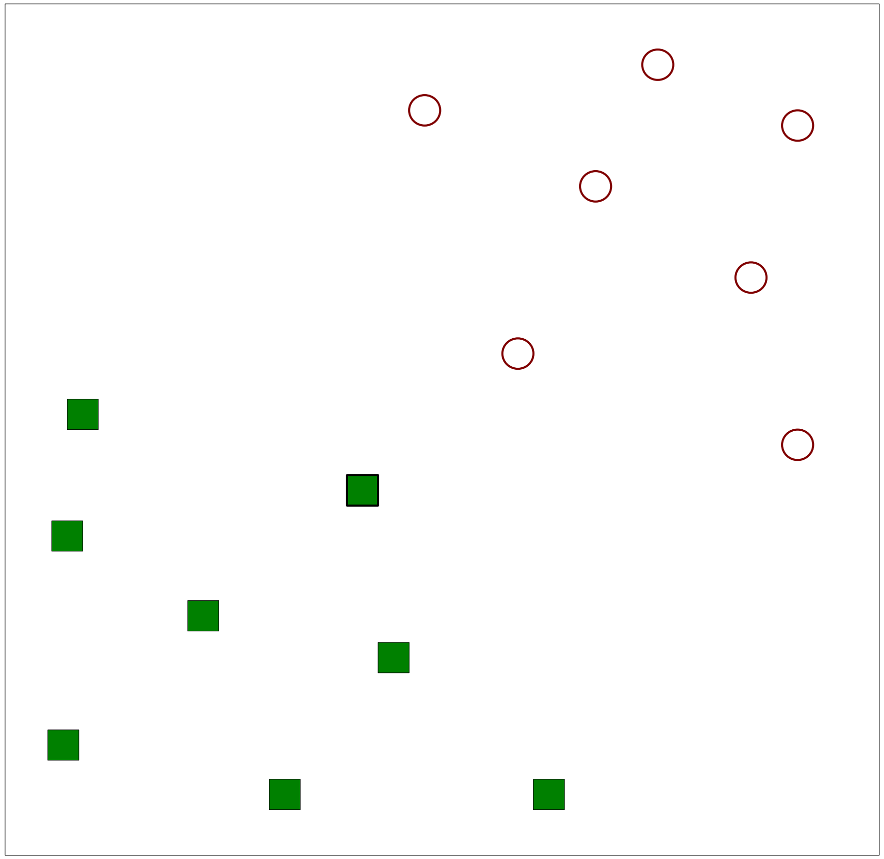
Support Vector Machines

Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis

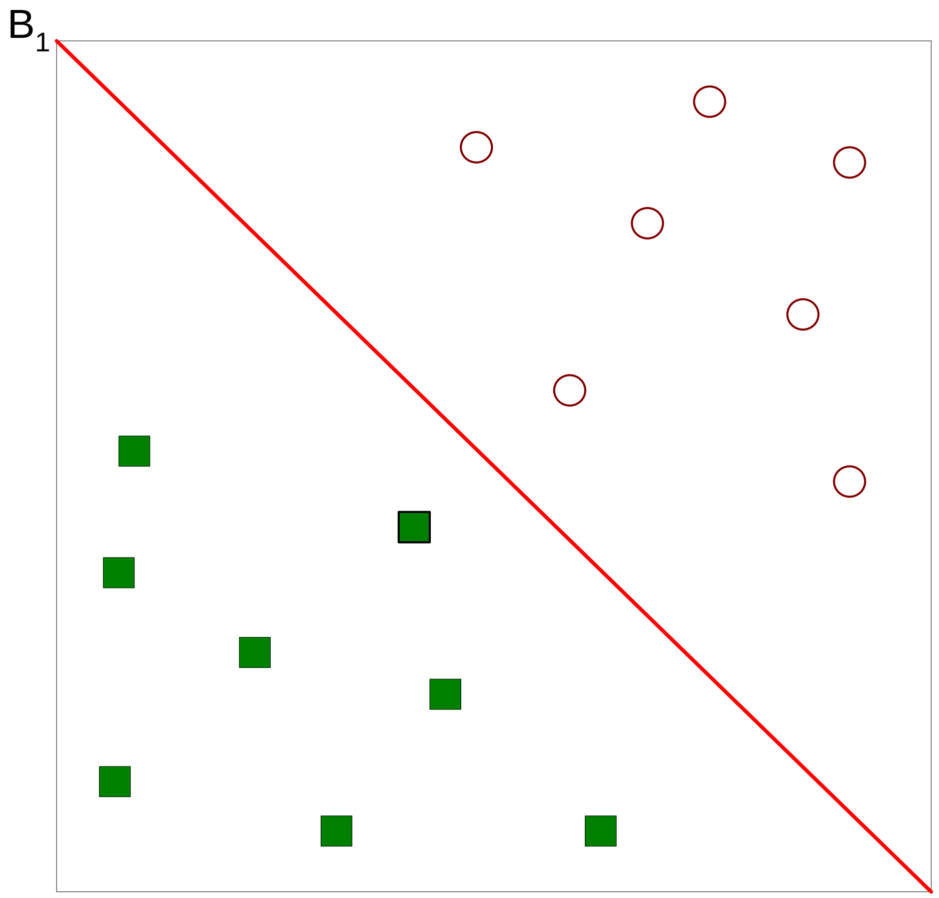
Politecnico di Torino

Support Vector Machines



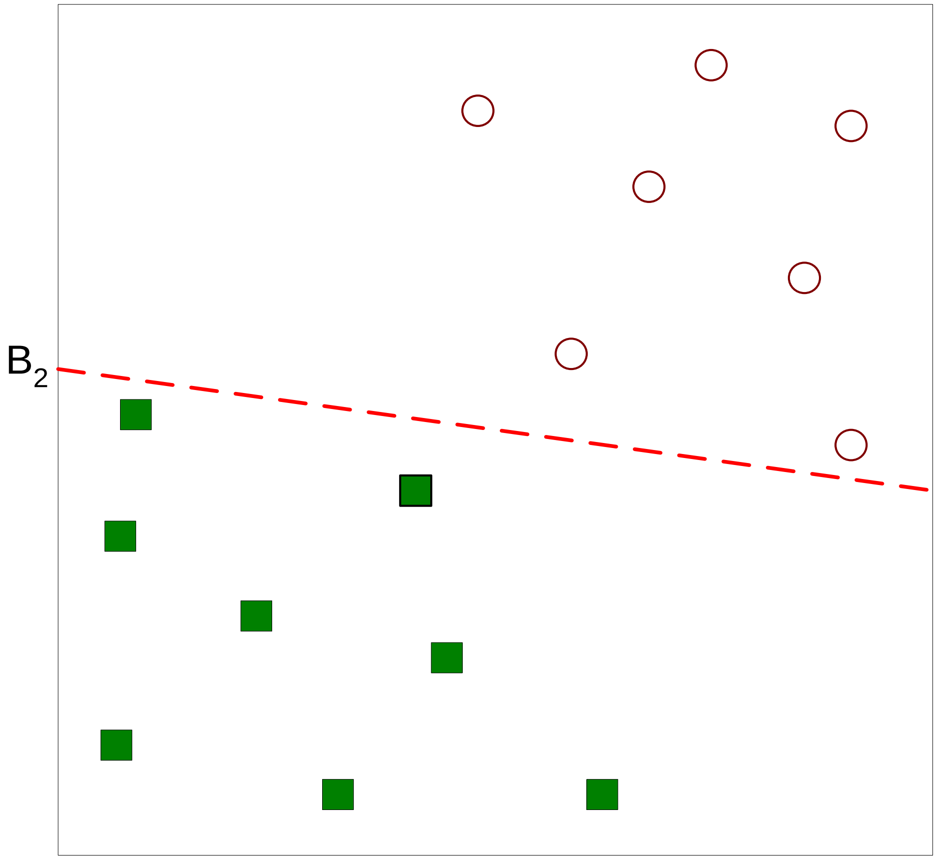
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



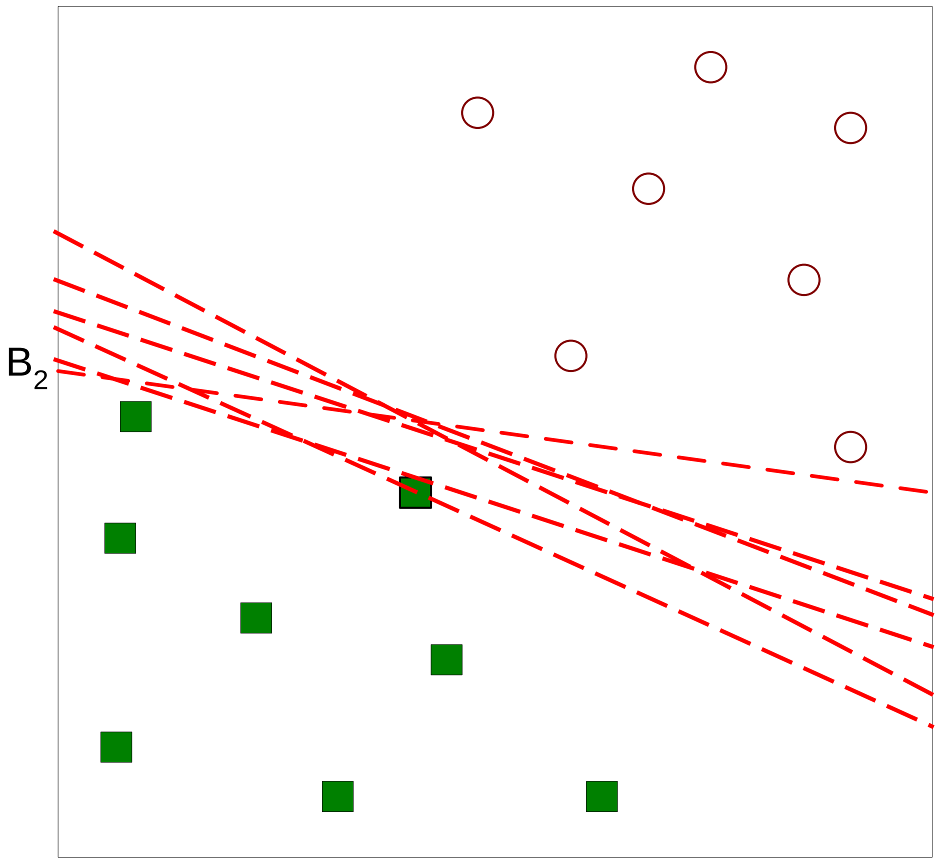
- One Possible Solution

Support Vector Machines



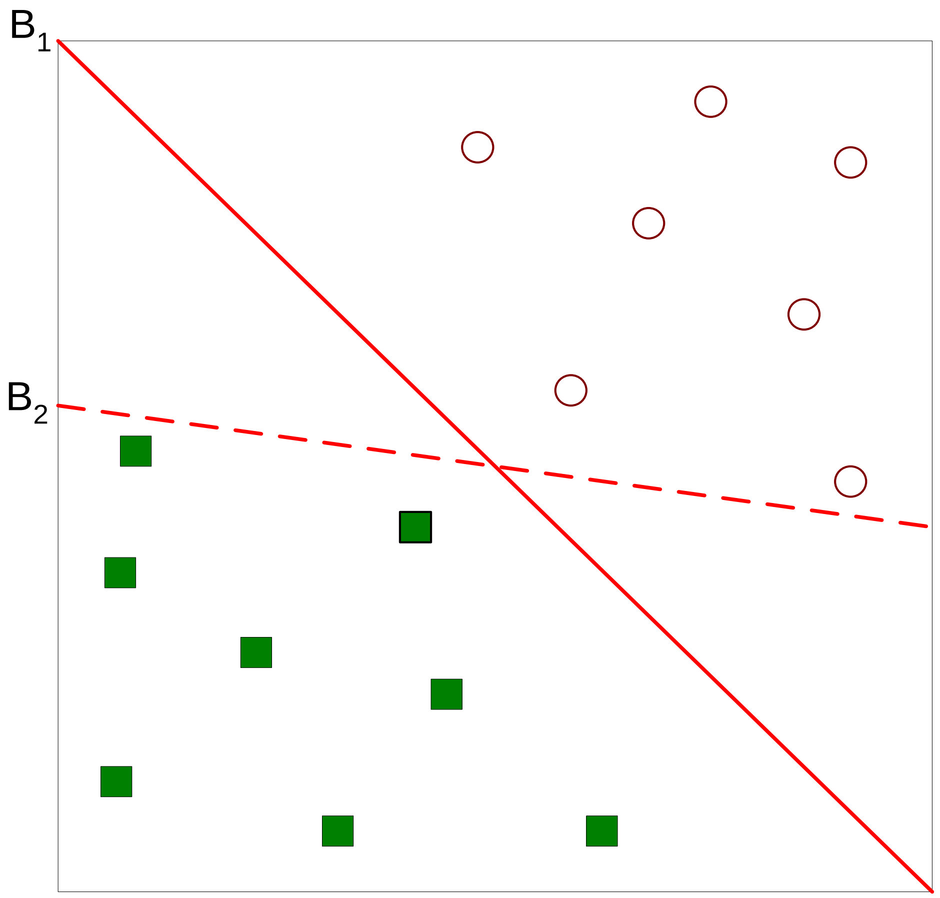
- Another possible solution

Support Vector Machines



- Other possible solutions

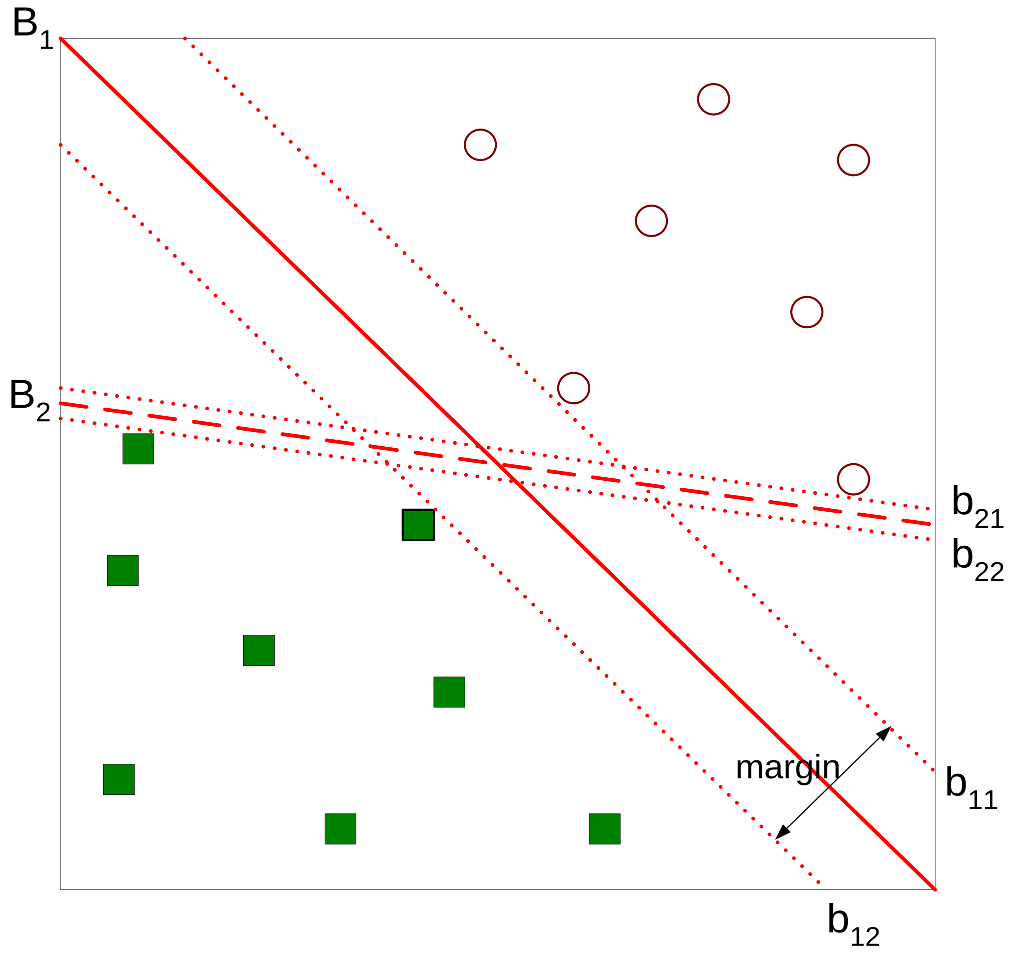
Support Vector Machines



- Which one is better? B1 or B2?
- How do you define better?

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

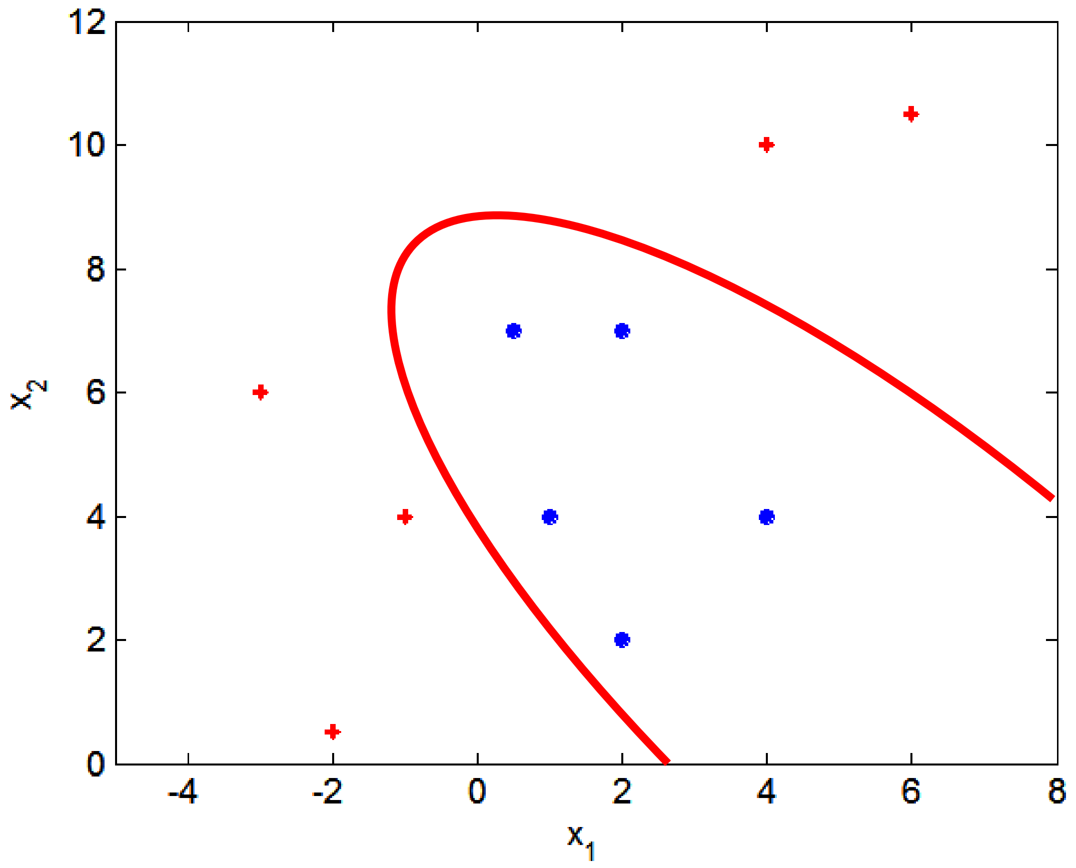
Support Vector Machines



- Find hyperplane maximizes the margin => B1 is better than B2

Nonlinear Support Vector Machines

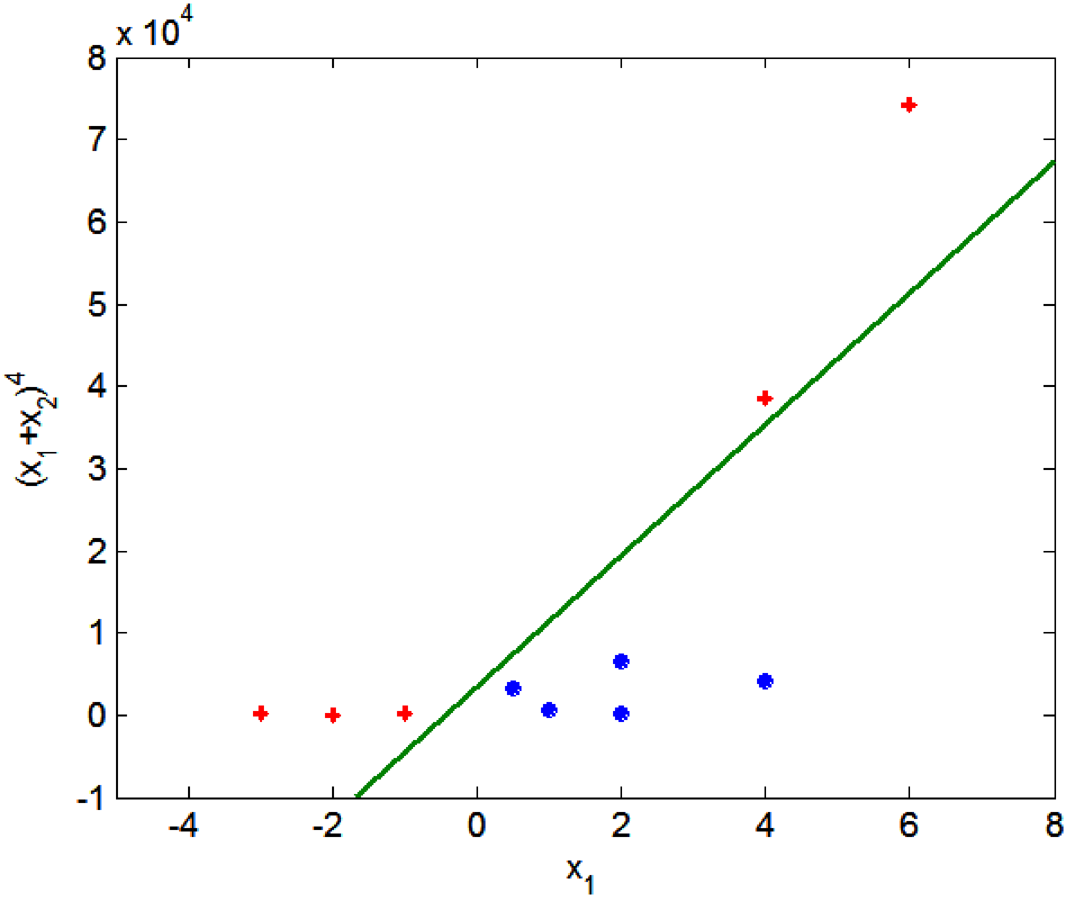
- What if decision boundary is not linear?



From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

Nonlinear Support Vector Machines

- Transform data into higher dimensional space



From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

K-Nearest Neighbor

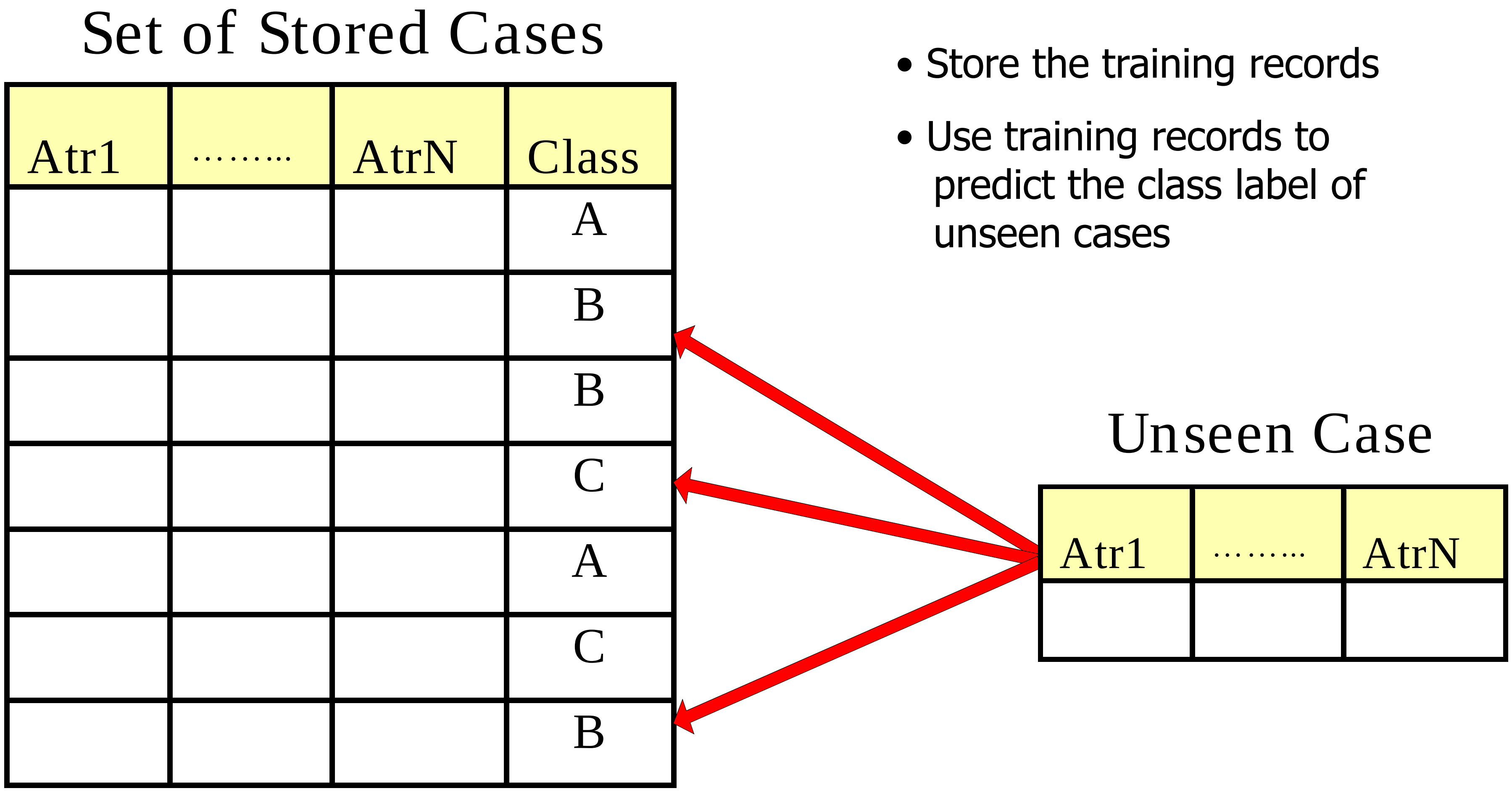
Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis

Politecnico di Torino

Instance-Based Classifiers

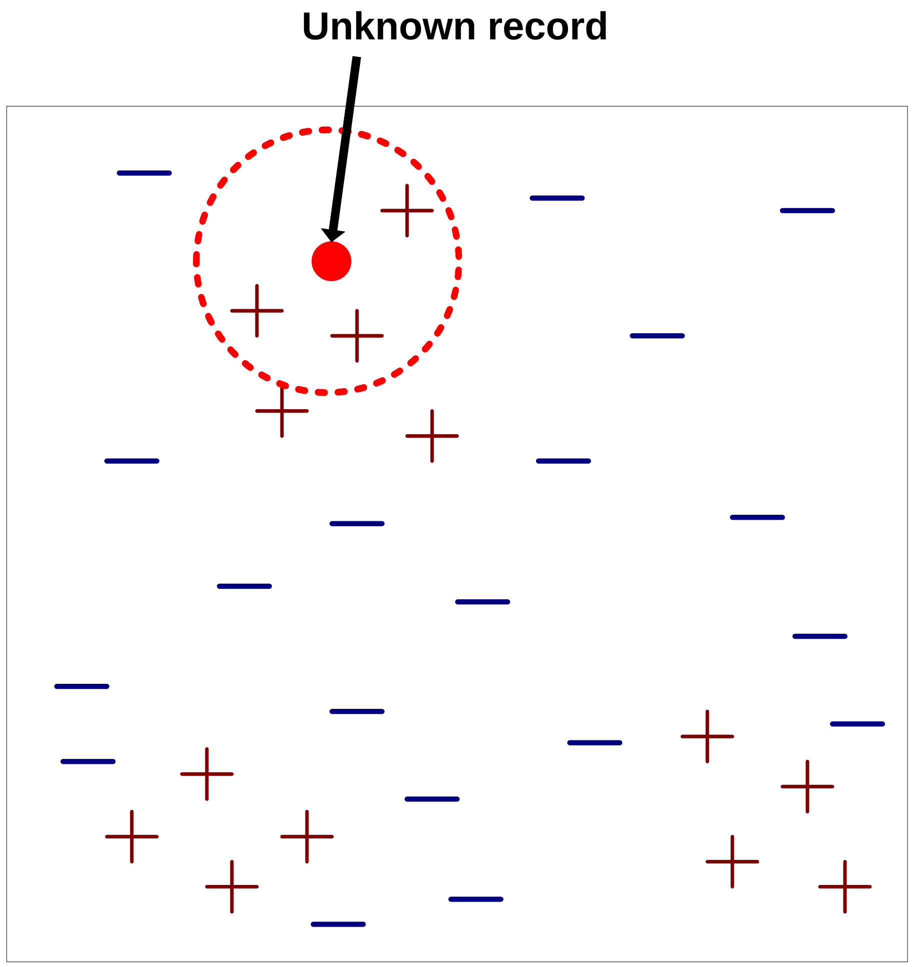
- Store the training records
- Use training records to predict the class label of unseen cases



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

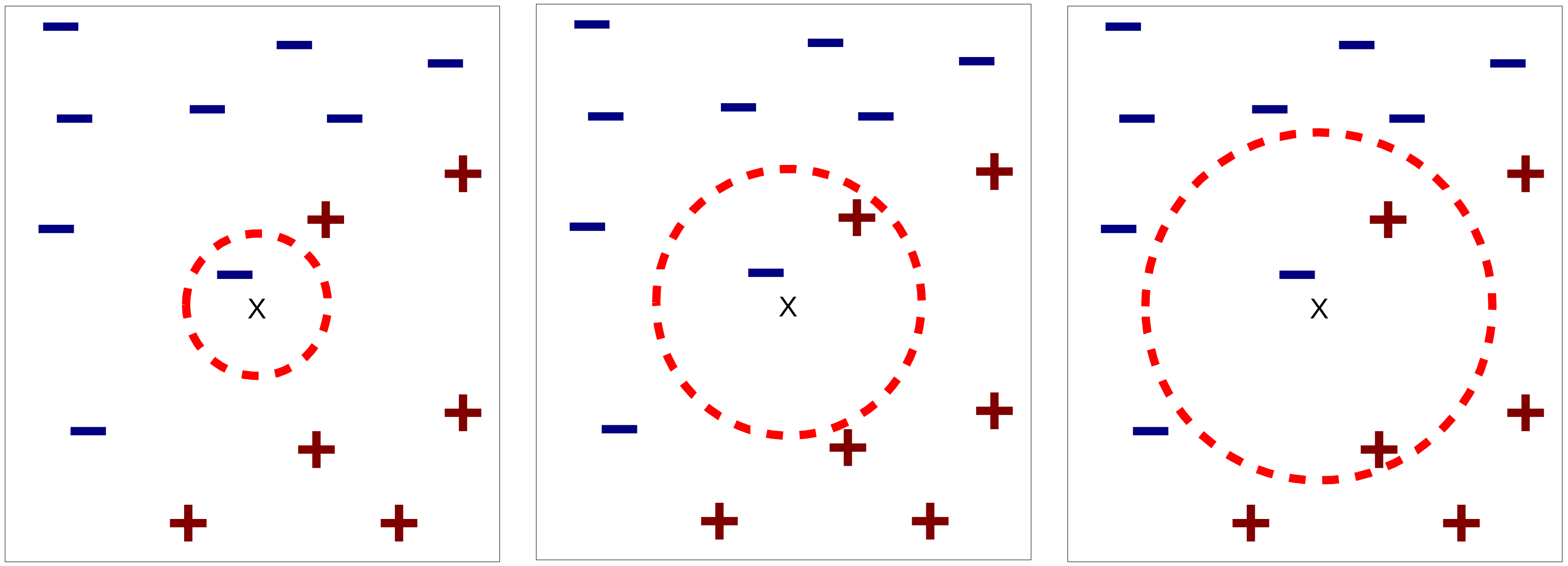
Nearest-Neighbor Classifiers



- 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)

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

Definition of Nearest Neighbor

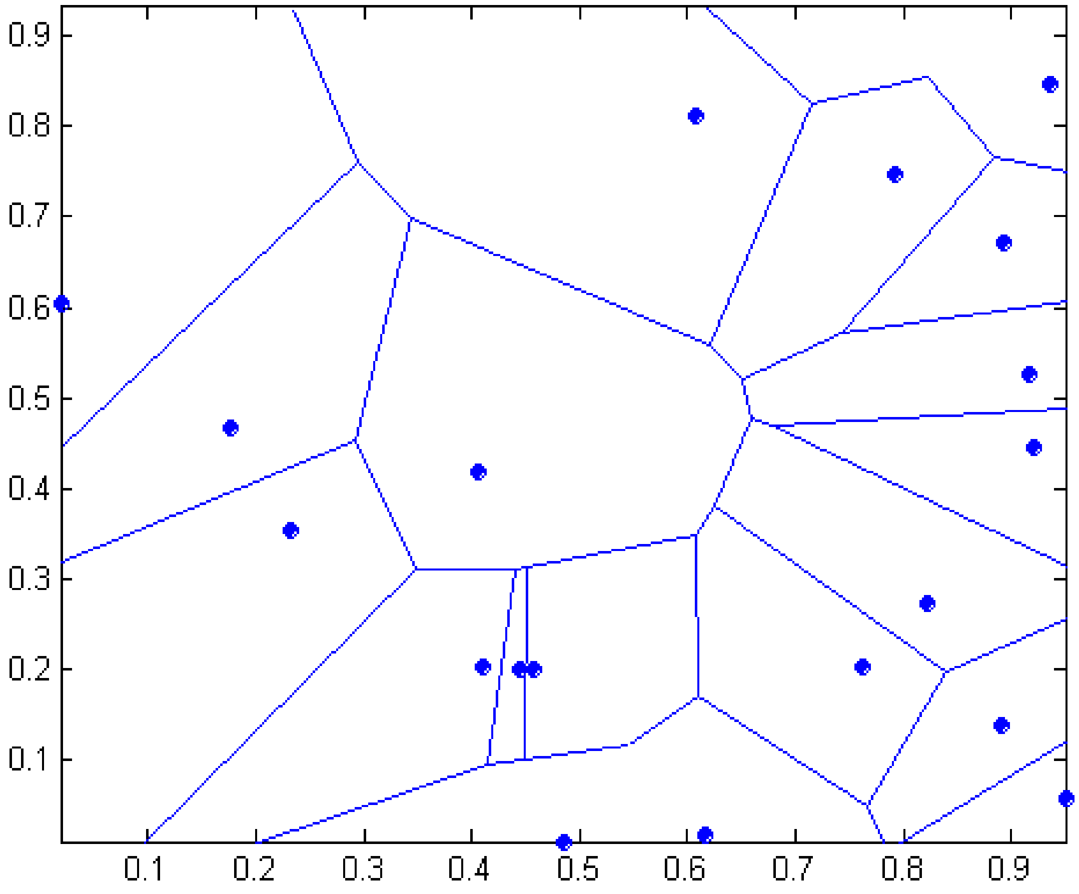


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

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

1 nearest-neighbor

Voronoi Diagram



From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

Nearest Neighbor Classification

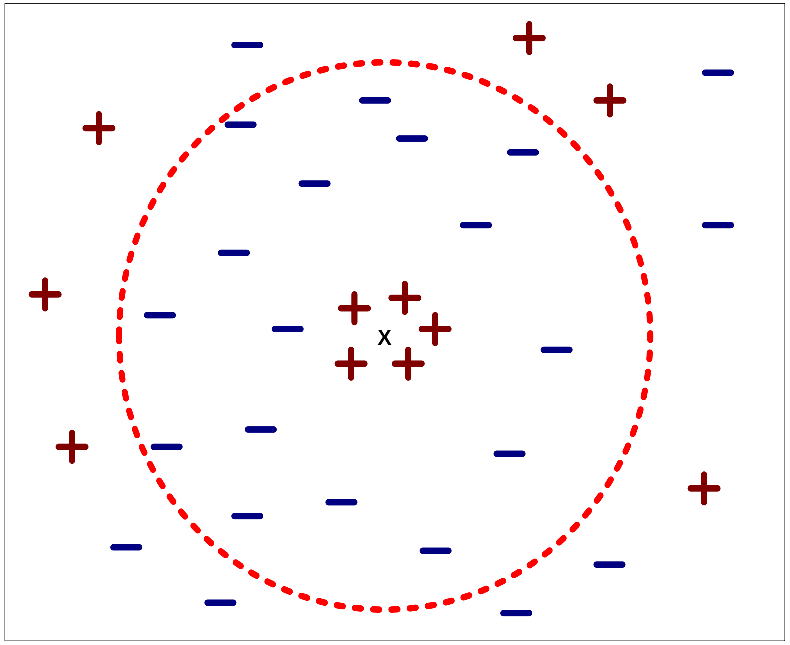
- Compute distance between two points
  - Euclidean distance

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

- 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

Artificial Neural Networks

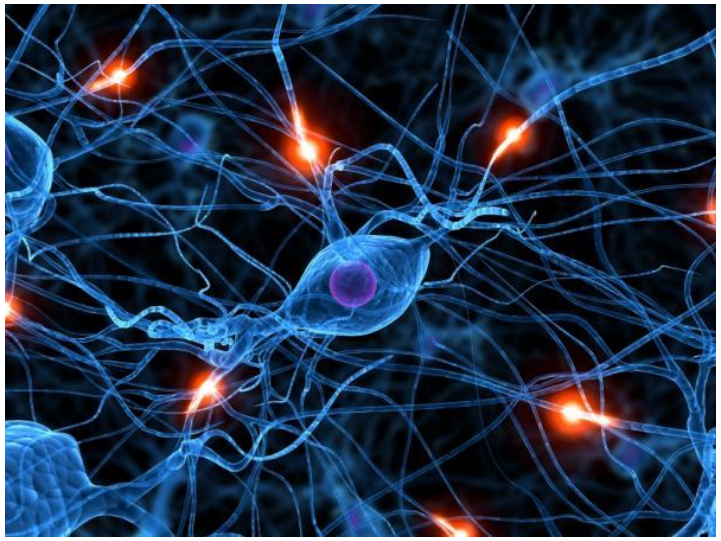
Data Base and Data Mining Group of Politecnico di Torino

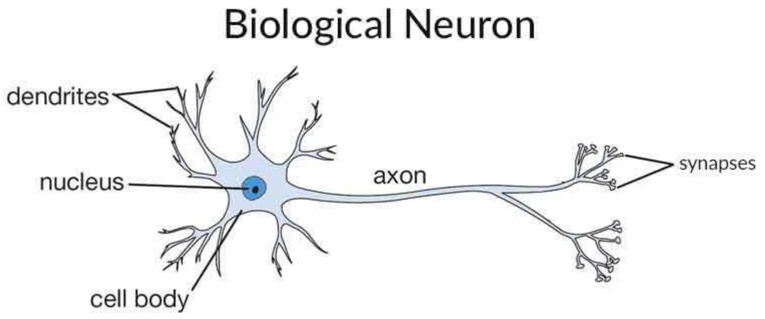
Elena Baralis

Politecnico di Torino

Artificial Neural Networks

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

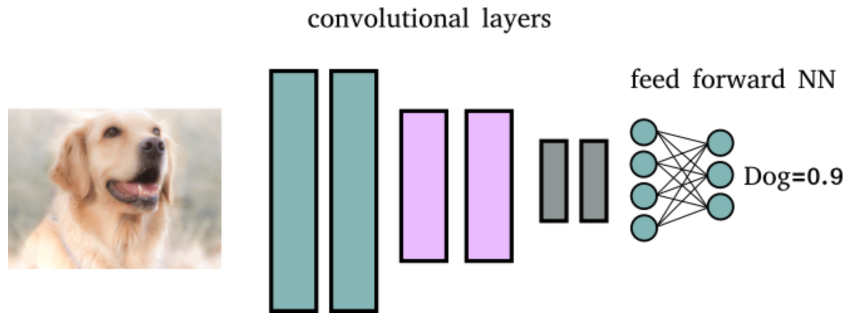




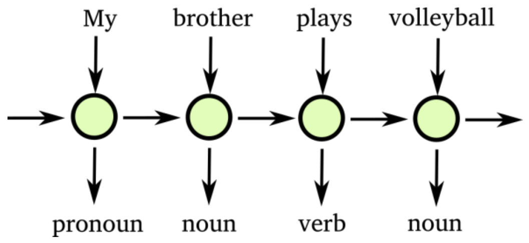
Artificial Neural Networks

- Different tasks, different architectures

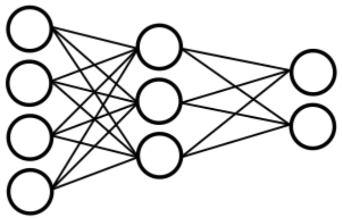
image understanding: convolutional NN (CNN)



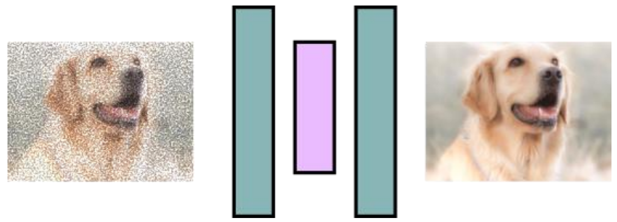
time series analysis: recurrent NN (RNN)



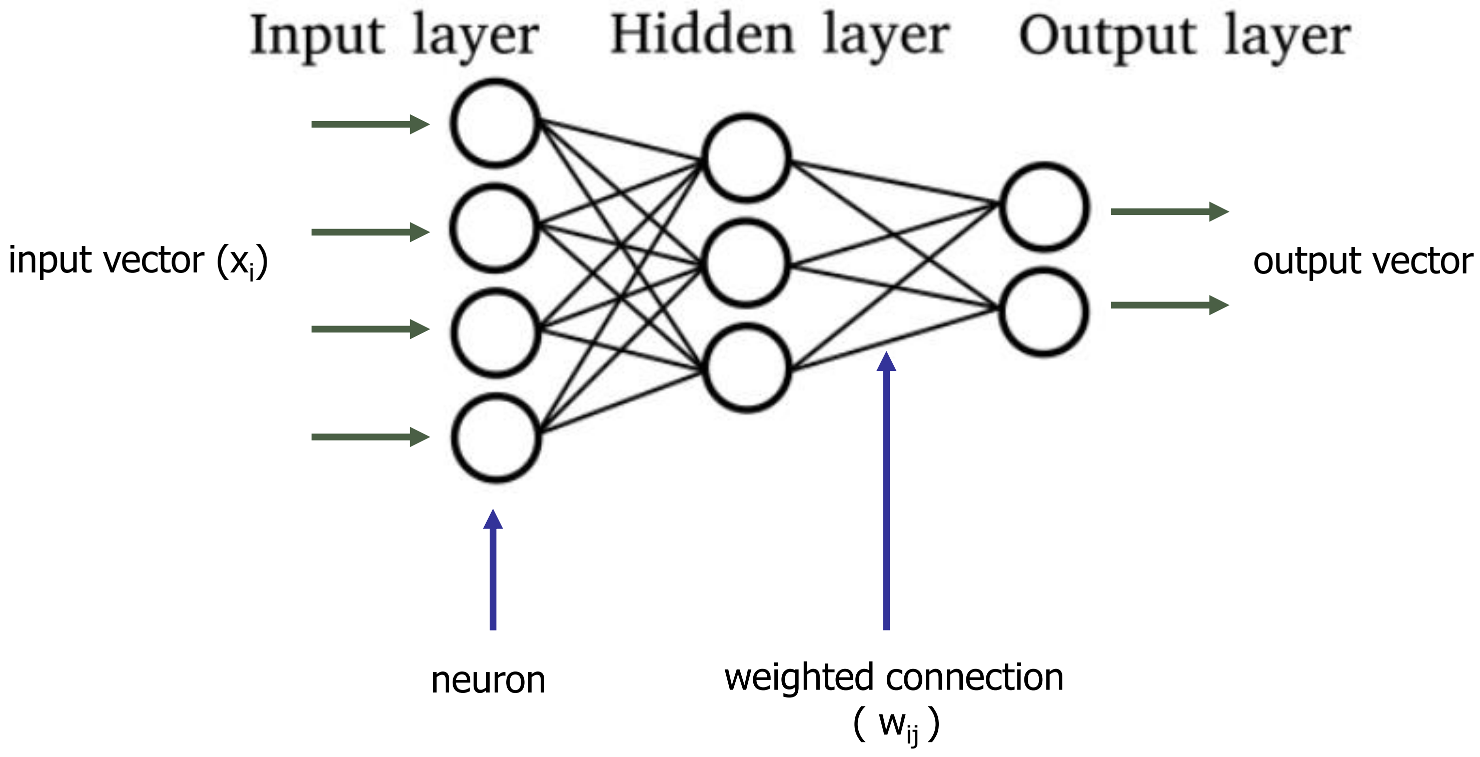
numerical vectors classification: feed forward NN (FFNN)



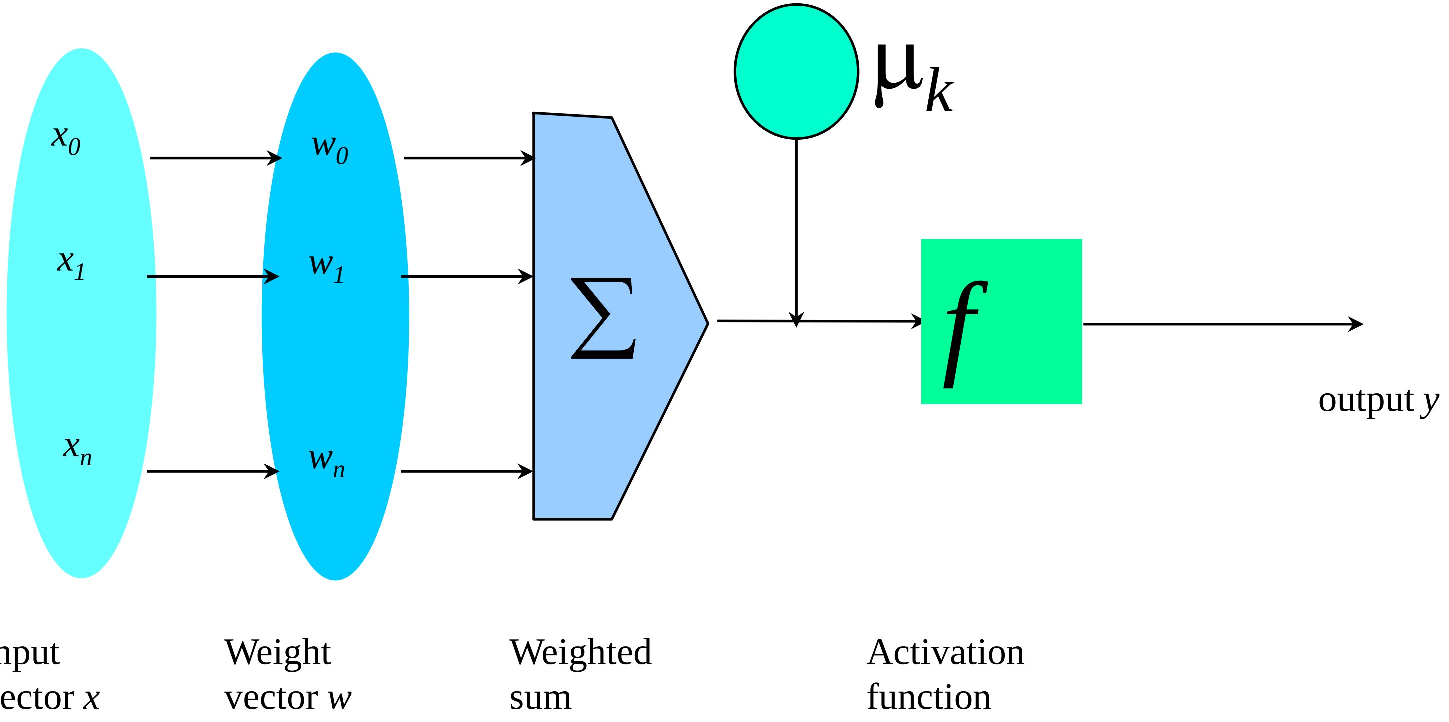
denoising: auto-encoders



Feed Forward Neural Network

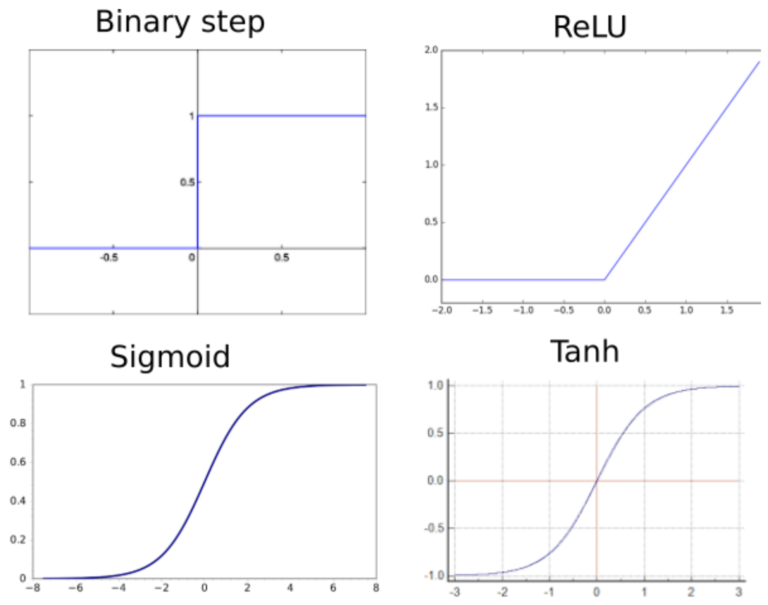


Structure of a neuron



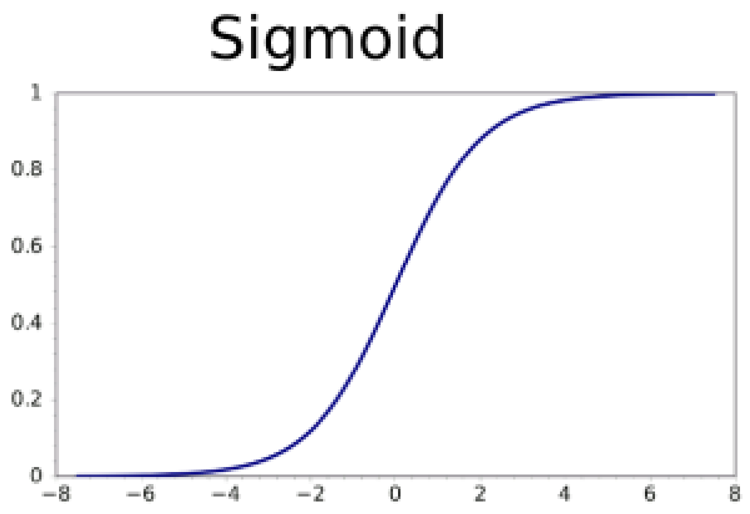
Activation Functions

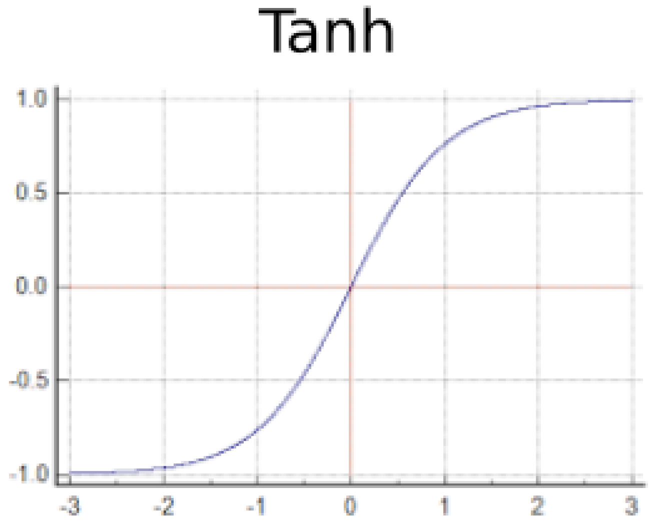
- Activation
  - simulates biological activation to input stymula
  - provides non-linearity to the computation
  - may help to saturate neuron outputs in fixed ranges



Activation Functions

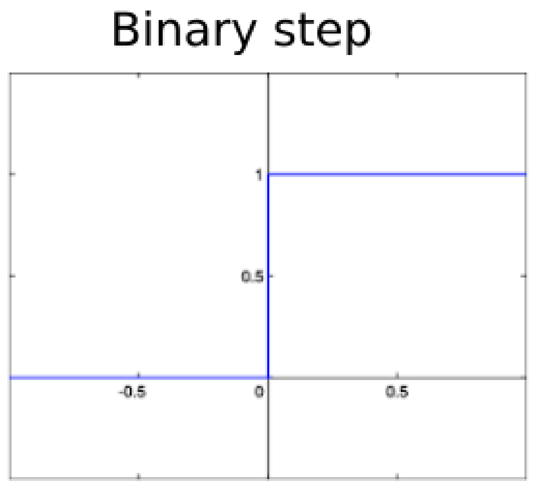
- Sigmoid, tanh
  - saturate input value in a fixed range
  - non linear for all the input scale
  - typically used by FFNNs for both hidden and output layers
    - E.g. *sigmoid* in output layers allows generating values between 0 and 1 (useful when output must be interpreted as likelihood)



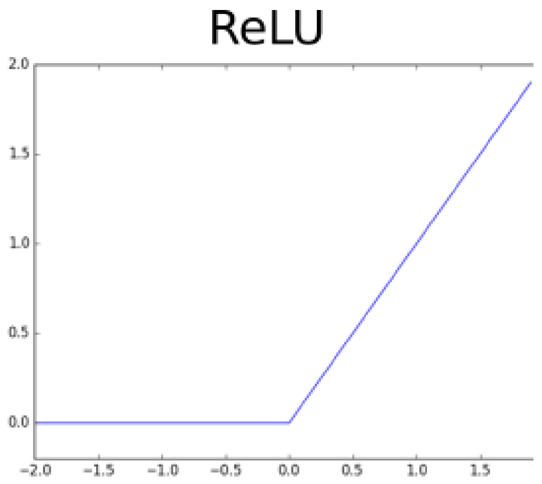


Activation Functions

- 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



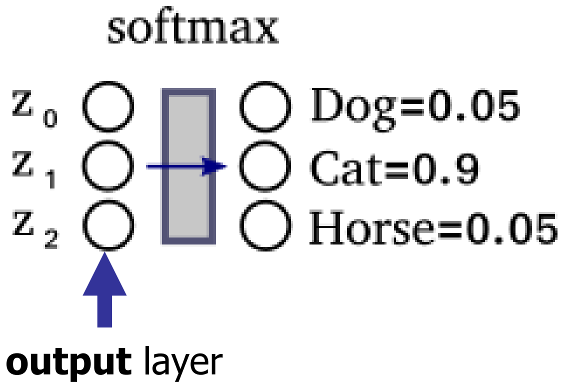
- ReLU (Rectified Linear Unit)
  - used in deep networks (e.g. CNNs)
    - avoids vanishing gradient
    - does not saturate
  - neurons activate linearly only for positive input



Activation Functions

- Softmax
  - 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

$$softmax(z_j) = \frac{e^{z_j}}{\sum_{i=0}^{N-1} e^{z_i}}$$



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

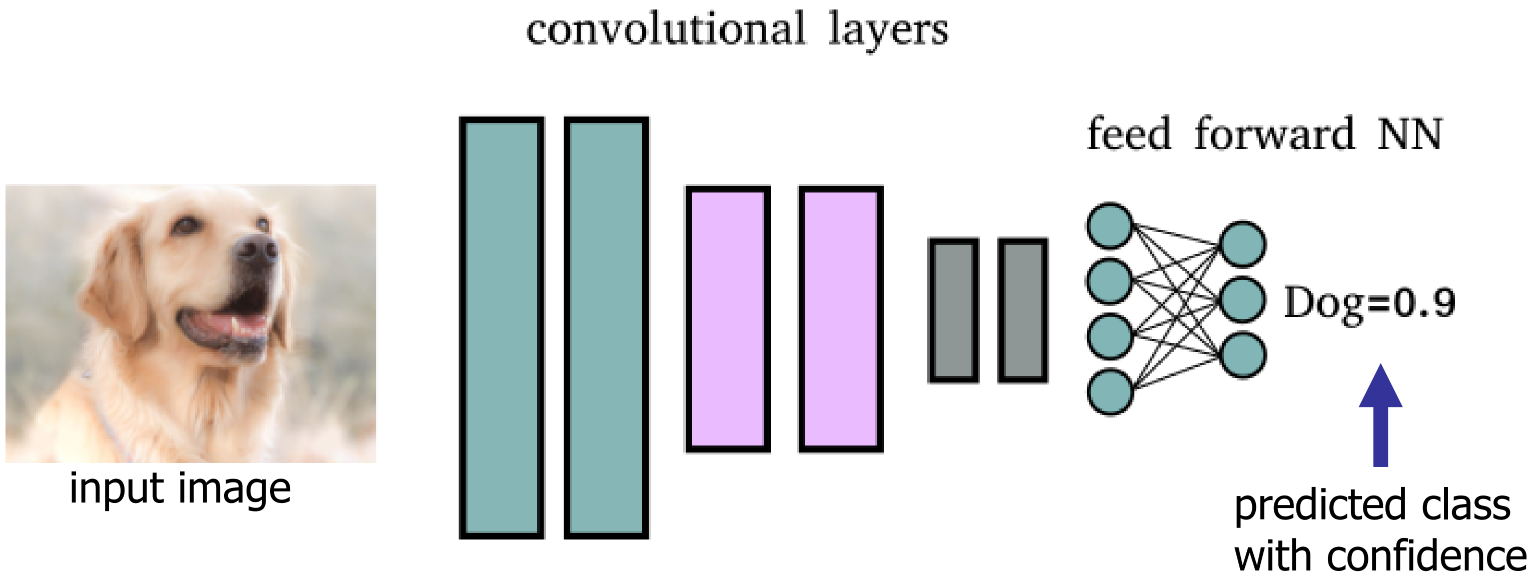
- Base algorithm
  - Initially assign random values to weights and offsets
  - Process instances in the training set one at a time
    - For each neuron, compute the result when applying weights, offset and activation function for the instance
    - Forward propagation until the output is computed
    - 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

Feed Forward Neural Networks

- Strong points
  - High accuracy
  - Robust to noise and outliers
  - Supports both discrete and continuous output
  - Efficient during classification
- Weak points
  - Long training time
    - weakly scalable in training data size
    - complex configuration
  - Not interpretable model
    - application domain knowledge cannot be exploited in the model

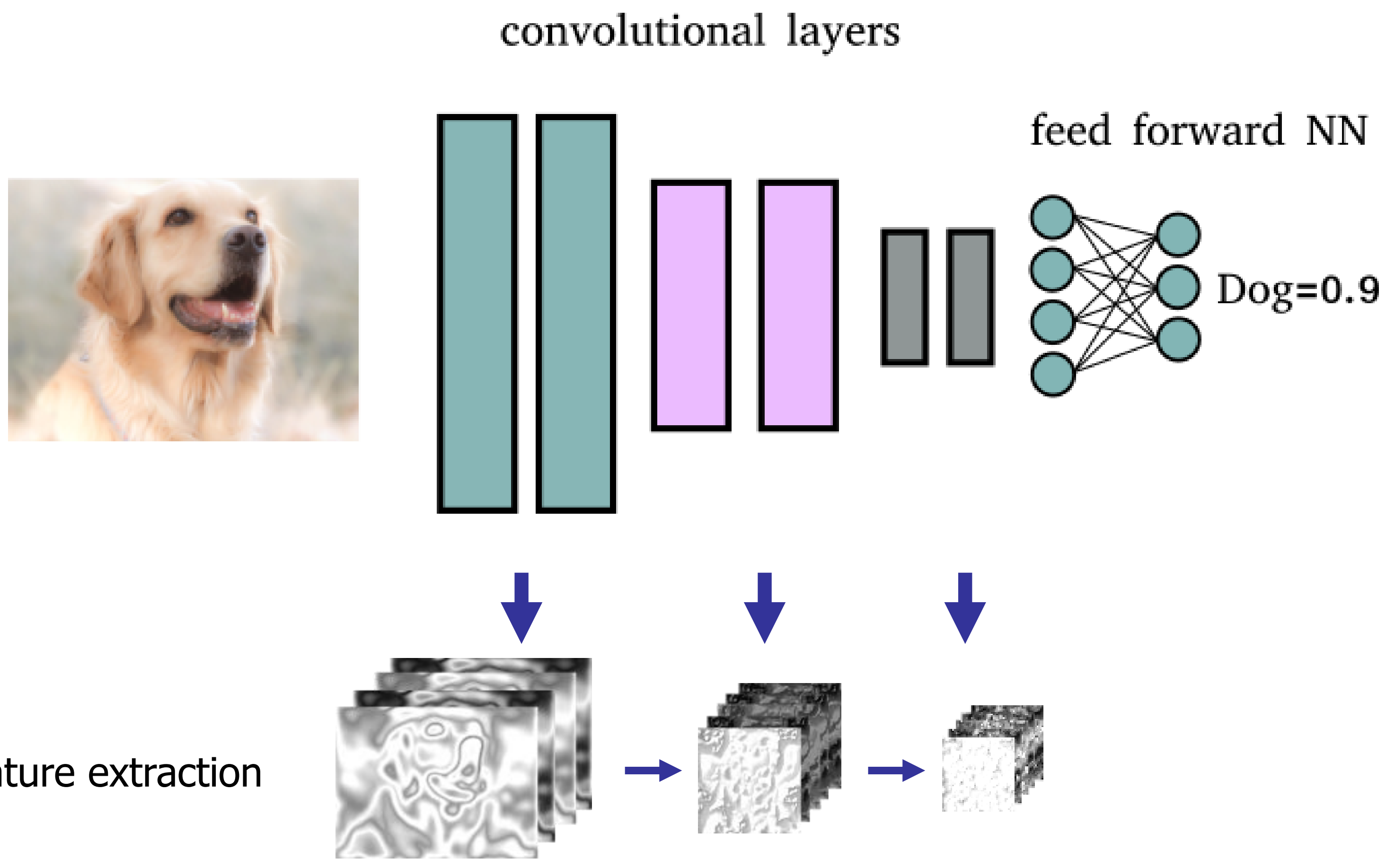
Convolutional Neural Networks

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



Convolutional Neural Network (CNN) Architecture

Convolutional Neural Networks

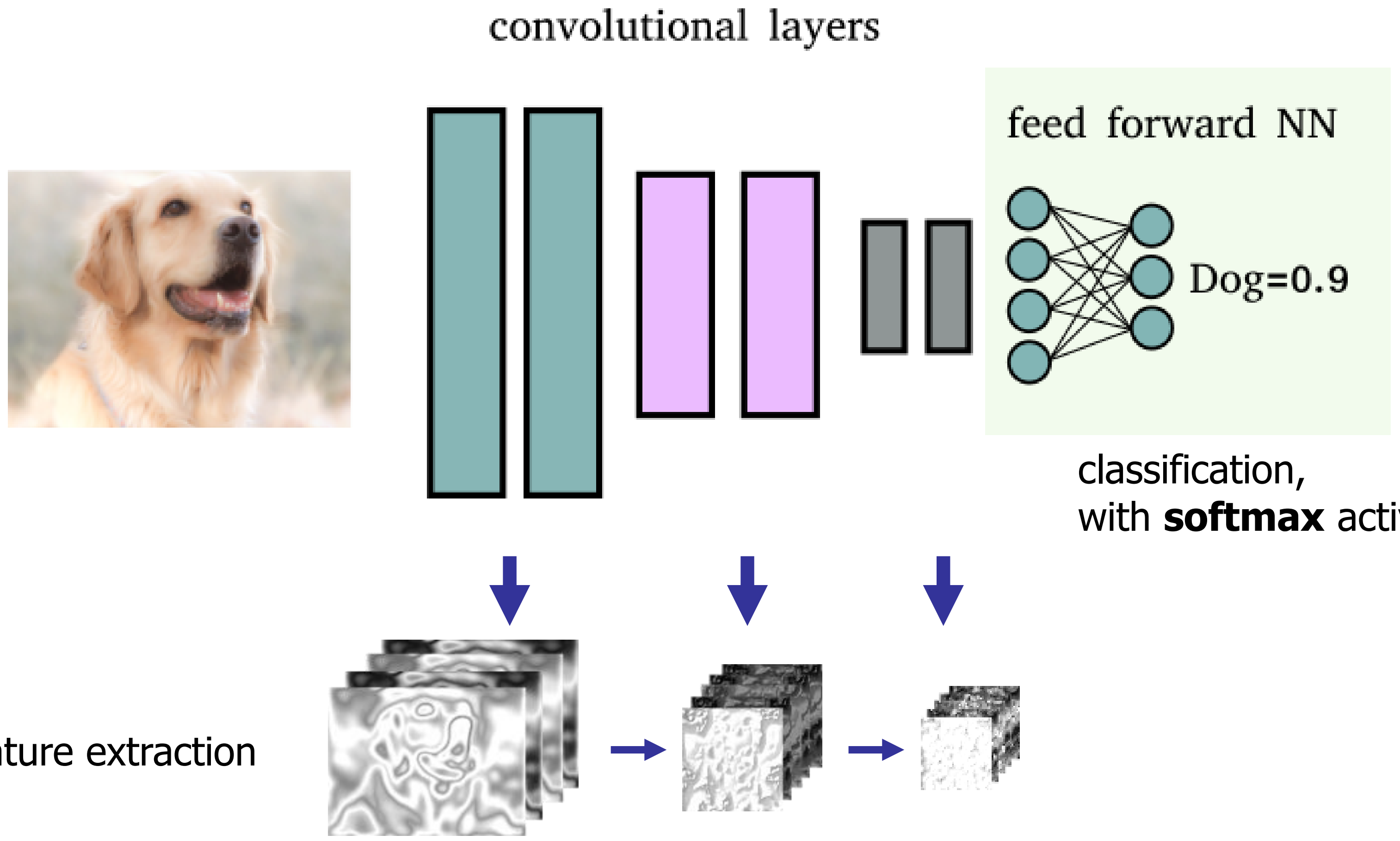


feature extraction

low level features

abstract features

Convolutional Neural Networks



classification,
with **softmax** activation

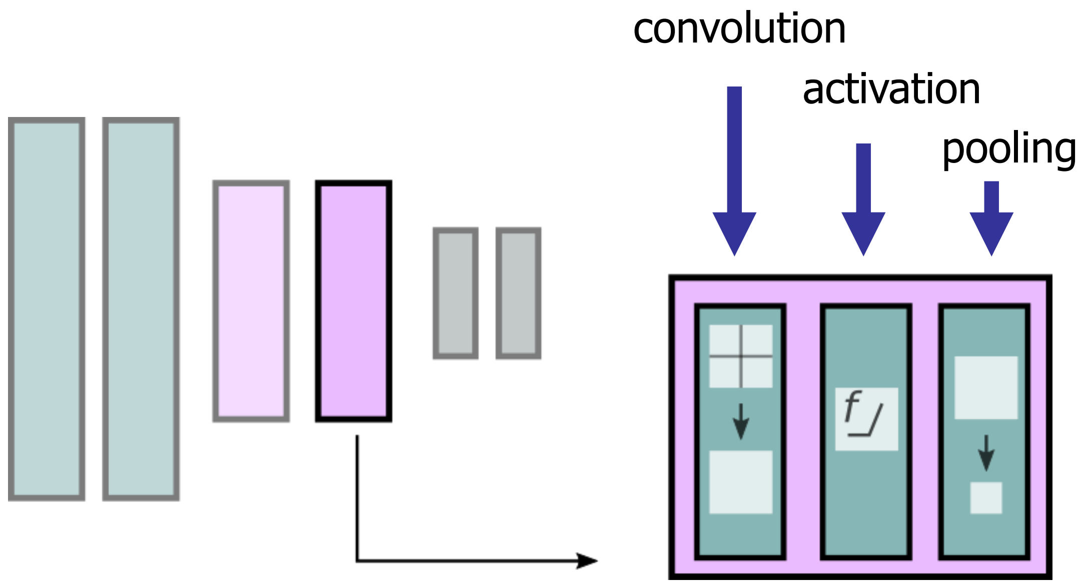
feature extraction

low level features

abstract features

Convolutional Neural Networks

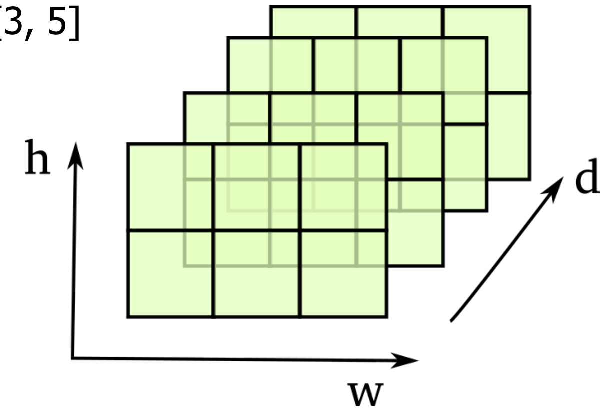
- Typical convolutional layer
  - *convolution* stage: feature extraction by means of (hundreds to thousands) sliding filters
  - sliding filters *activation*: apply activation functions to input tensor
  - *pooling*: tensor downsampling



Convolutional Neural Networks

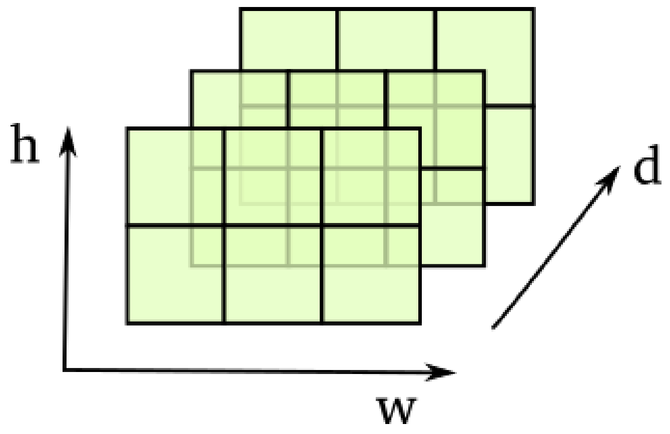
- Tensors
  - data flowing through CNN layers is represented in the form of *tensors*
  - *Tensor* = N-dimensional vector
  - *Rank* = number of dimensions
    - scalar: rank 0
    - 1-D vector: rank 1
    - 2-D matrix: rank 2
  - *Shape* = number of elements for each dimension
    - e.g. a vector of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

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



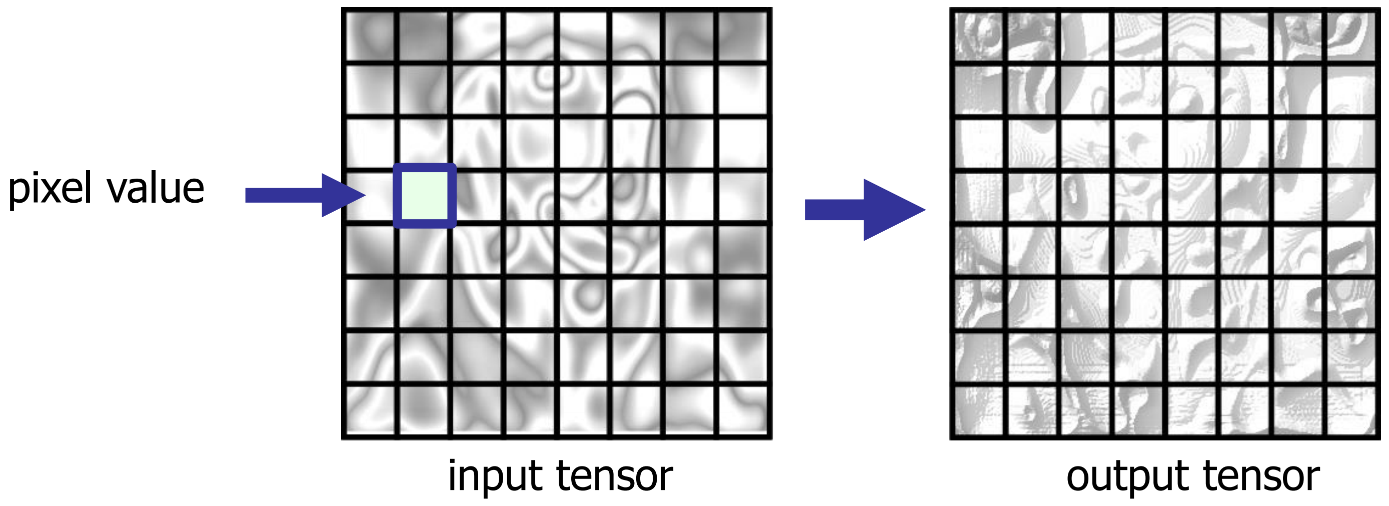
Convolutional Neural Networks

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



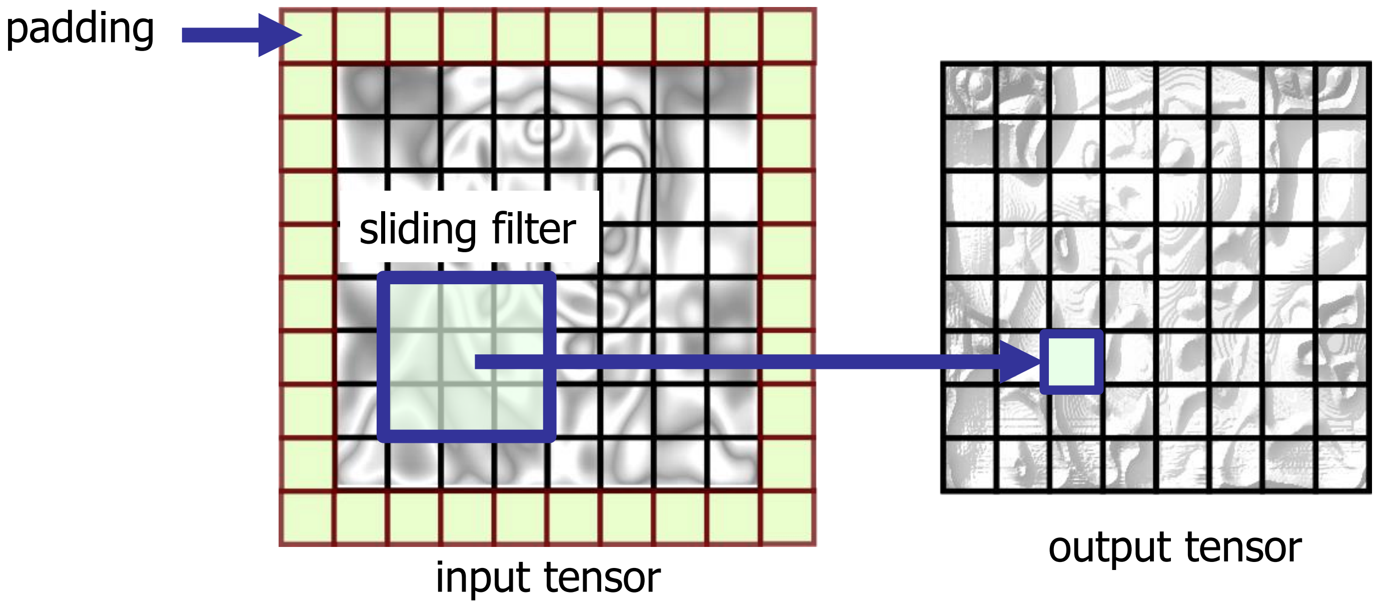
Convolutional Neural Networks

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



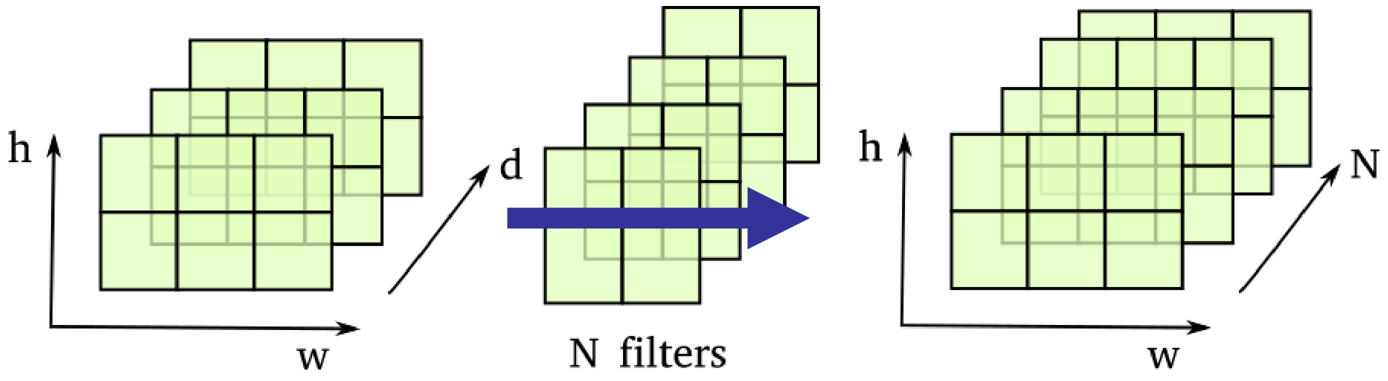
Convolutional Neural Networks

- 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



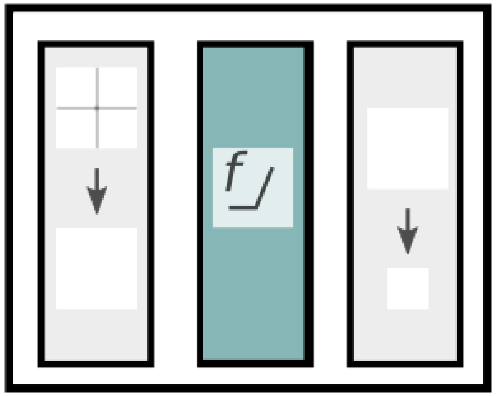
Convolutional Neural Networks

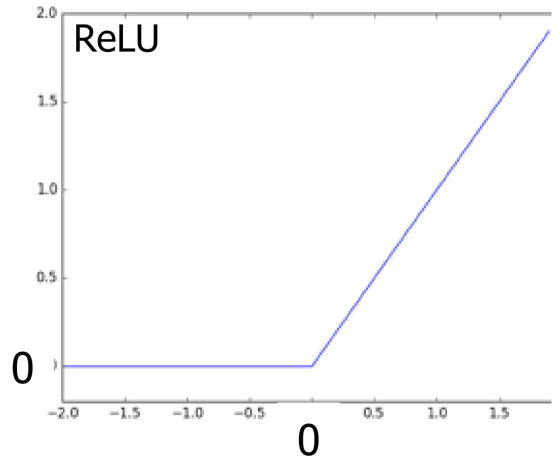
- Convolution
  - images are transformed into features by convolutional filters
  - after convolving a tensor [d,h,w] with *N filters* we obtain
    - a rank-3 tensor with shape [N,h,w]
    - hence, each filter generates a layer in the depth of the output tensor



Convolutional Neural Networks

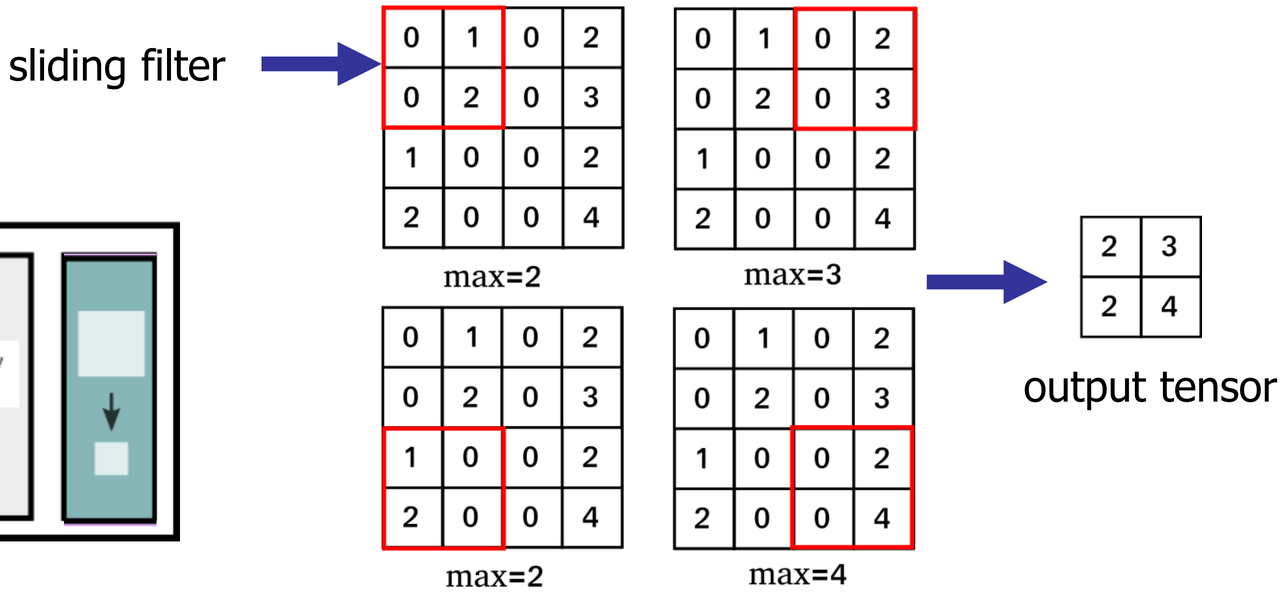
- Activation
  - symulates biological activation to input stymula
  - provides non-linearity to the computation
  - ReLU is typically used for CNNs
    - faster training (no vanishing gradients)
    - does not saturate
    - faster computation of derivatives for backpropagation

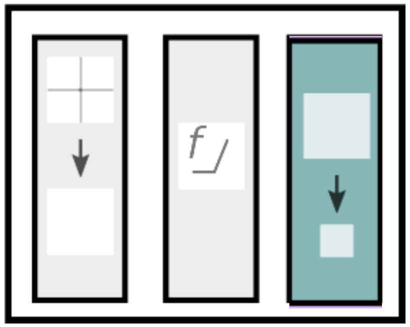




Convolutional Neural Networks

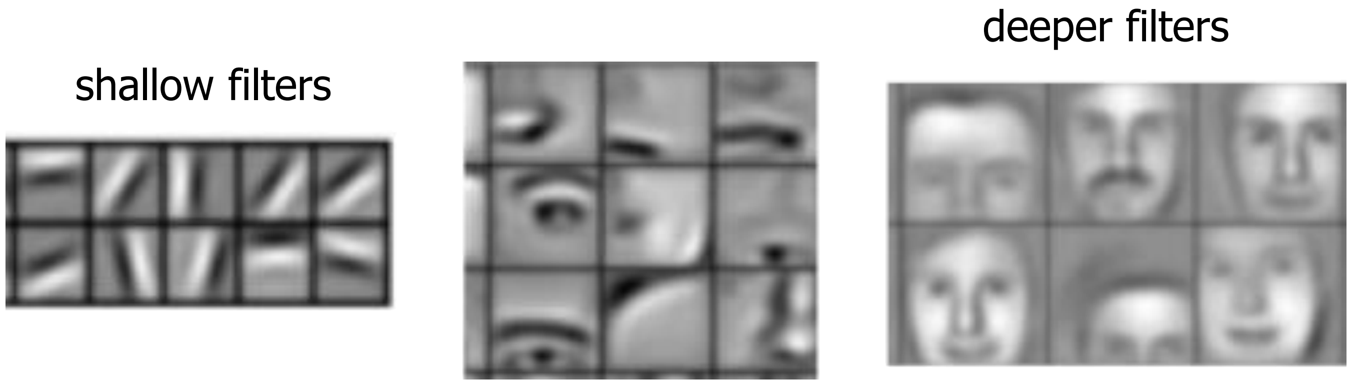
- Pooling
  - performs tensor *downsampling*
  - *sliding filter* which replaces tensor values with a *summary* statistic of the nearby outputs
  - *maxpool* is the most common: computes the maximum value as statistic





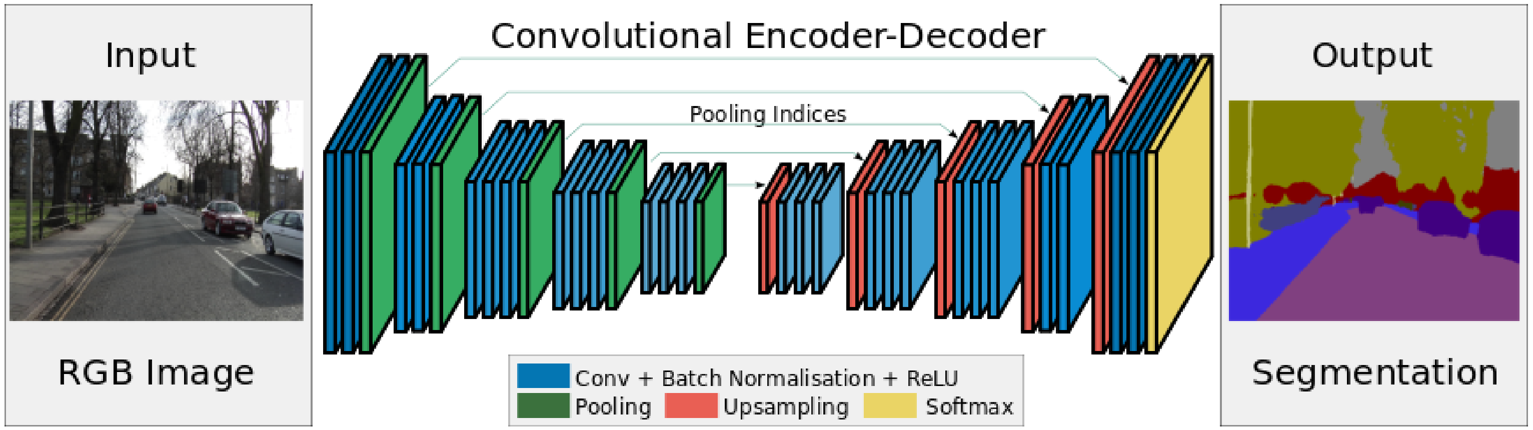
Convolutional Neural Networks

- 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)



Convolutional Neural Networks

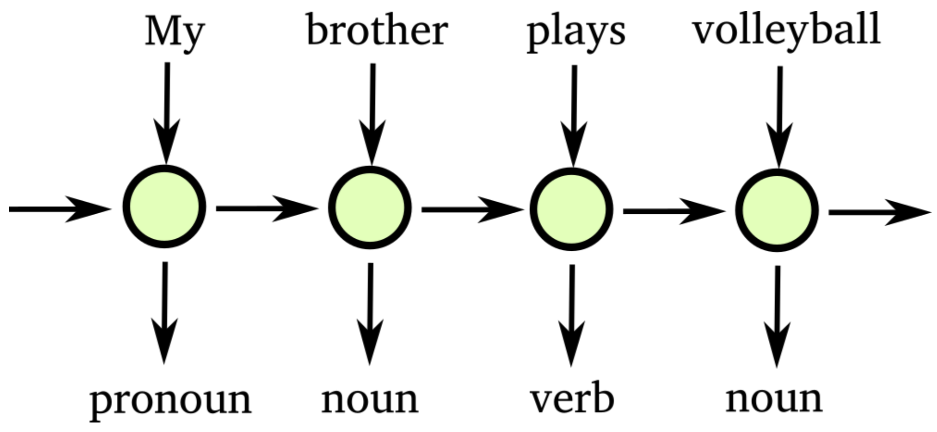
- Semantic segmentation CNNs
  - allow assigning a class to each pixel of the input image
  - composed of 2 parts
    - **encoder network**: convolutional layers to extract abstract features
    - **decoder network**: deconvolutional layers to obtain the output image from the extracted features



SegNet neural network

Recurrent Neural Networks

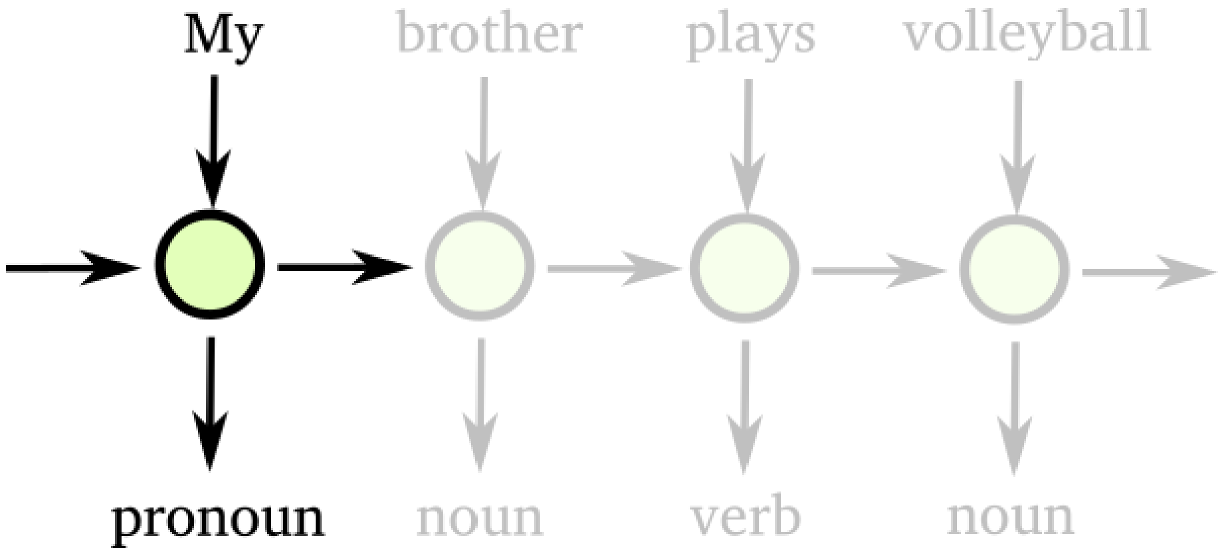
- Allow processing *sequential* data x(t)
- Differently from normal FFNN they are able to keep a *state* which evolves during time
- Applications
  - machine translation
  - time series prediction
  - speech recognition
  - part of speech (POS) tagging



Recurrent Neural Networks

- RNN execution during time

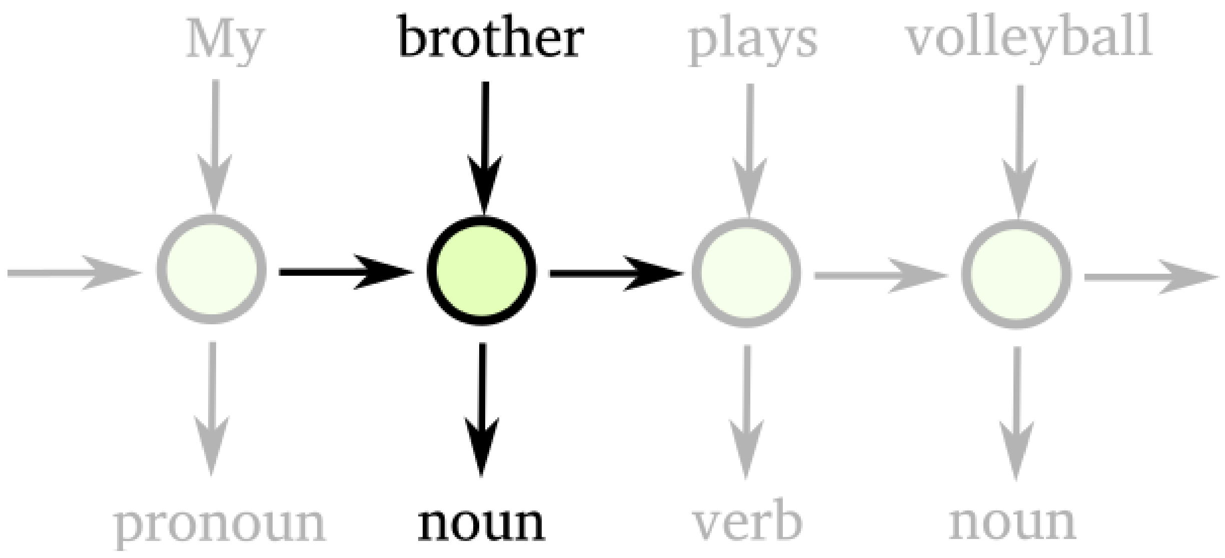
instance of the RNN at time t1



Recurrent Neural Networks

- RNN execution during time

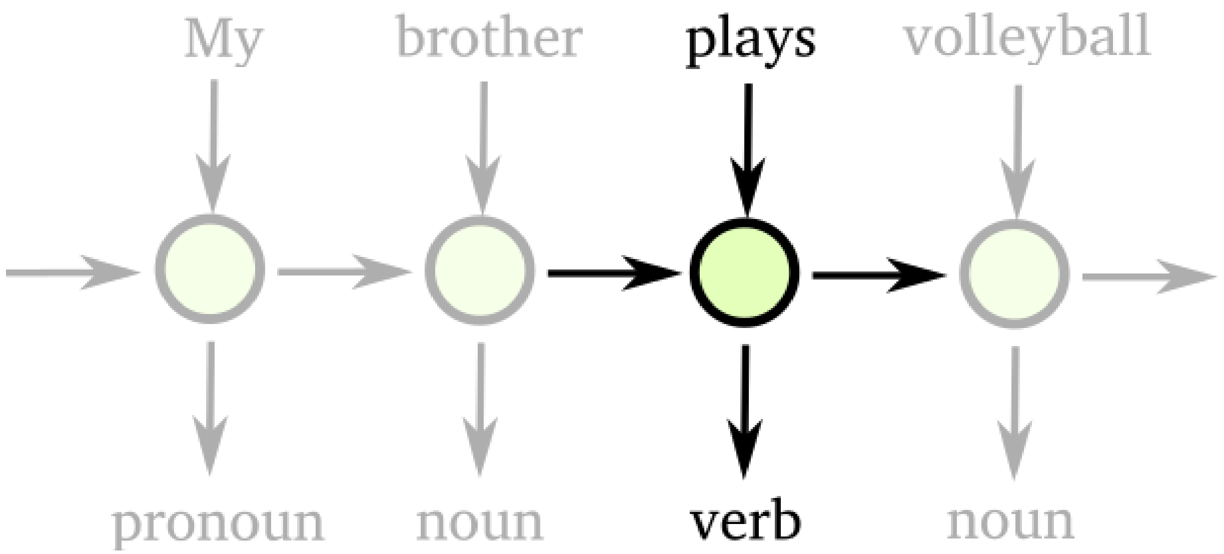
instance of the RNN at time t2



Recurrent Neural Networks

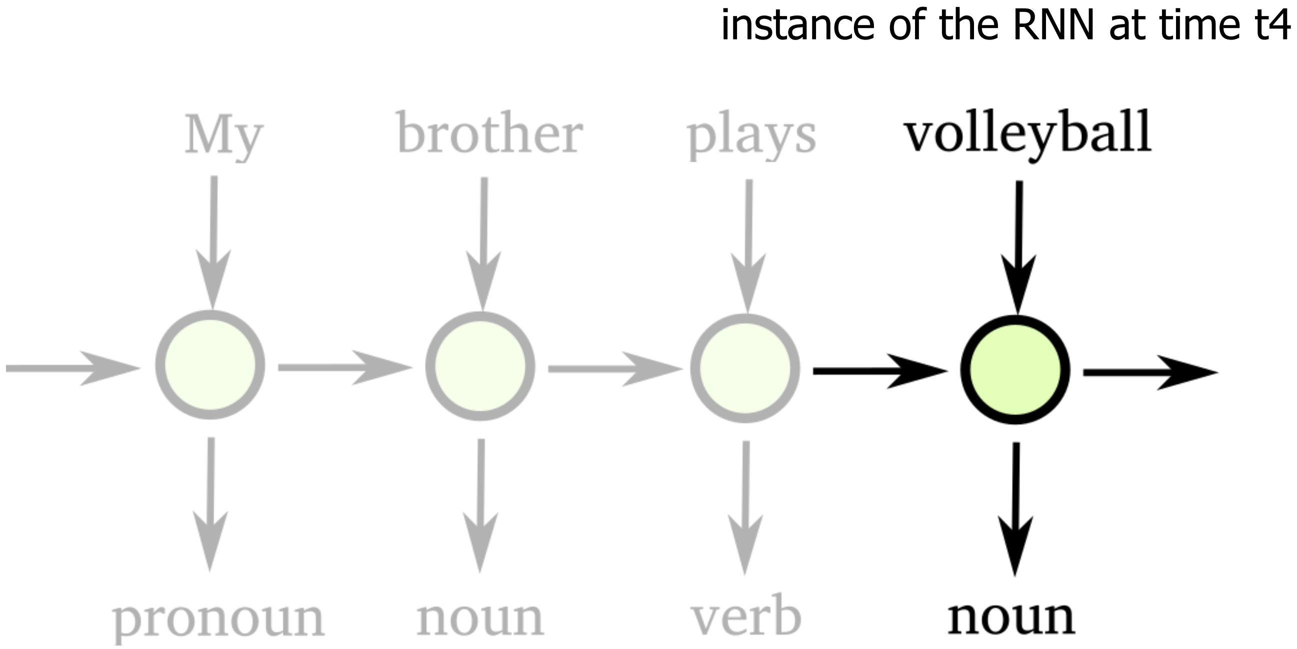
- RNN execution during time

instance of the RNN at time t3



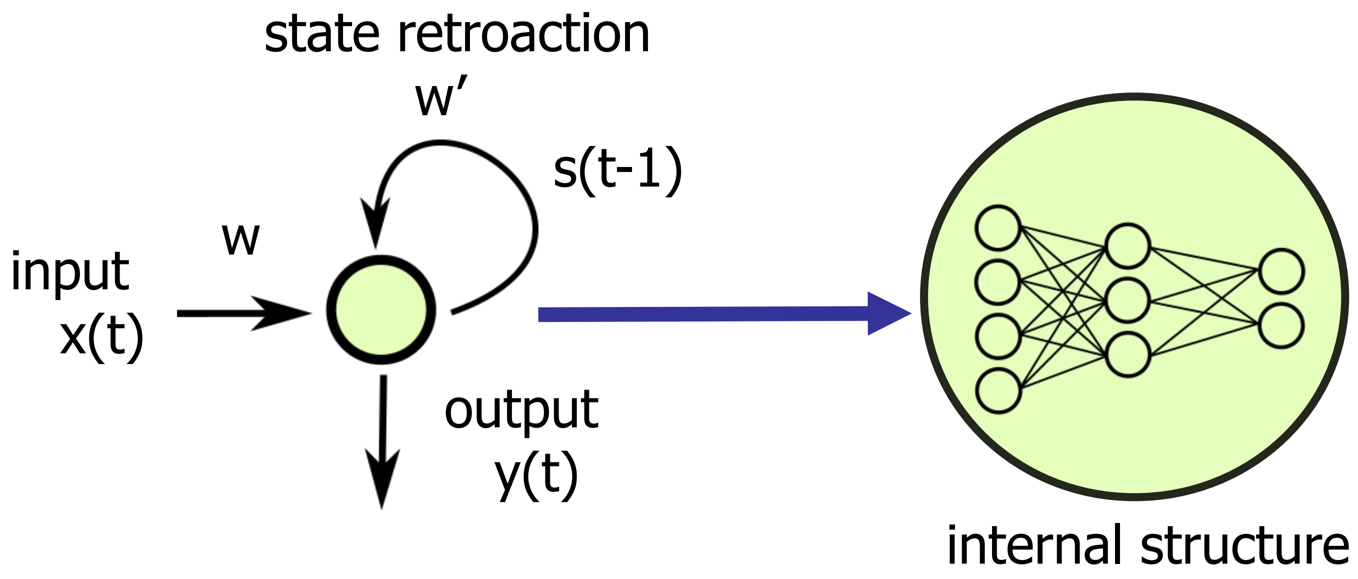
Recurrent Neural Networks

- RNN execution during time



Recurrent Neural Networks

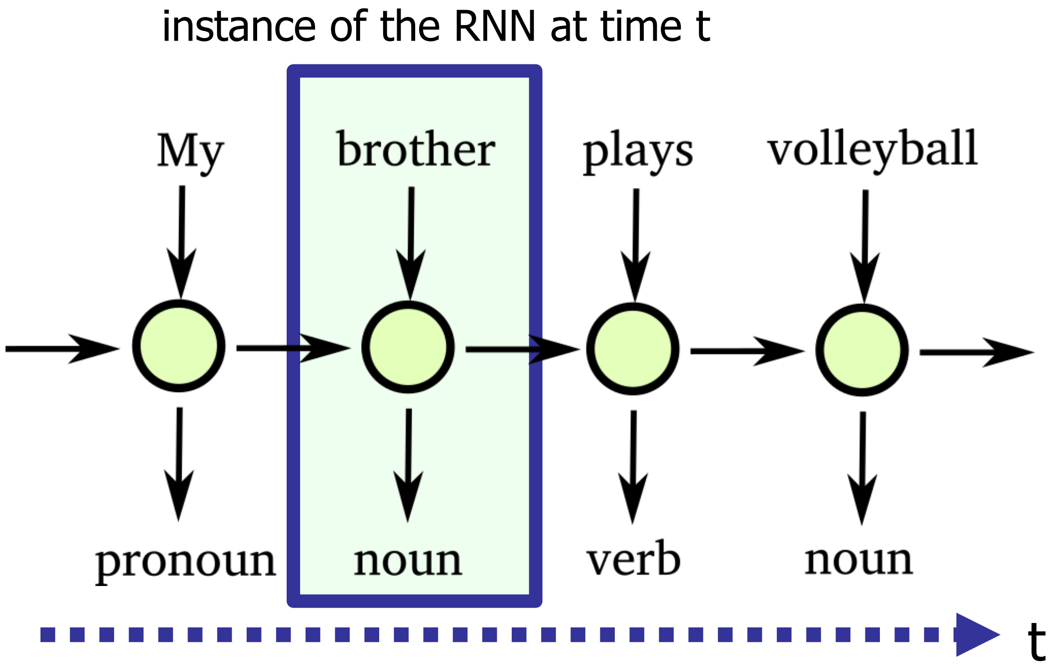
- A RNN receives as input a vector x(t) and the state at previous time step s(t-1)
- A RNN typically contains many *neurons organized in different layers*



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

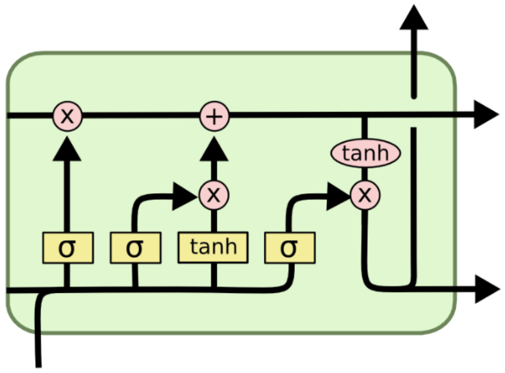
unrolled RNN diagram: shows the **same** neural network at different time steps



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

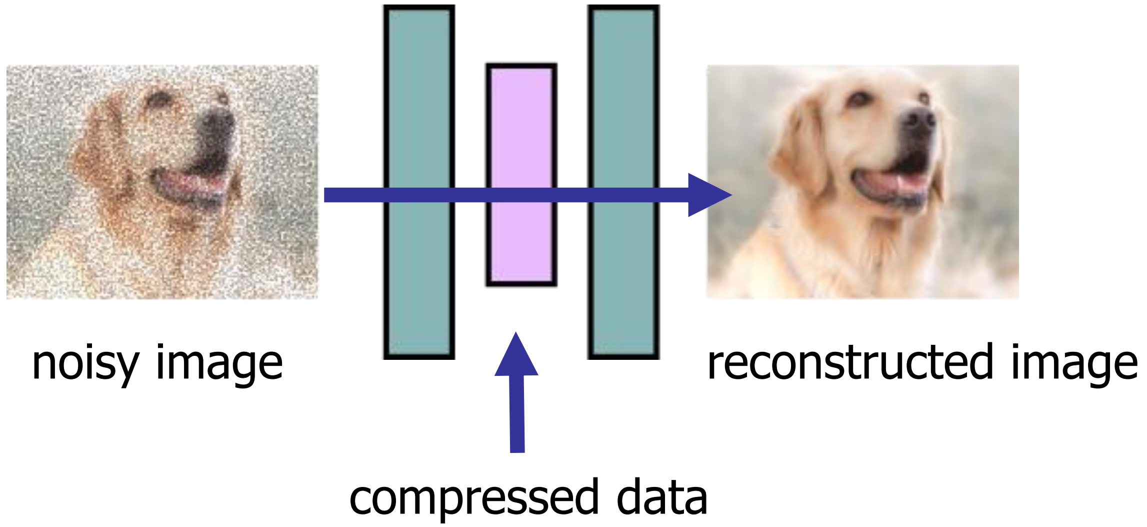
- Solution: *LSTM* (Long Short Term Memories)
  - RNN with "gates" which encourage the state information to flow through long time intervals



LSTM stage

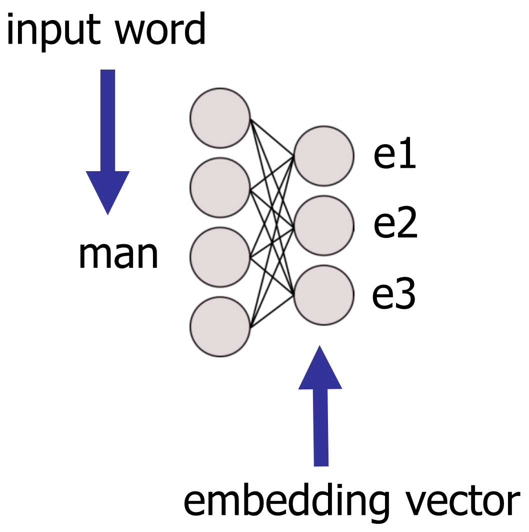
Autoencoders

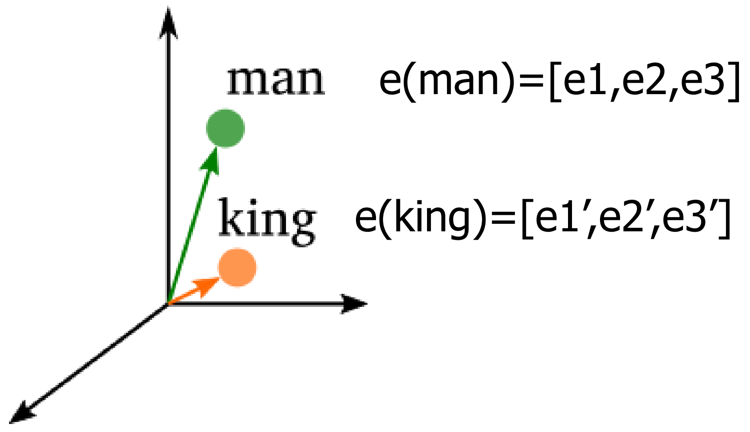
- Autoencoders allow *compressing* input data by means of compact representations and from them *reconstruct* the initial input
  - for feature extraction: the compressed representation can be used as significant set of features representing input data
  - for image (or signal) *denoising*: the image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



Word Embeddings (Word2Vec)

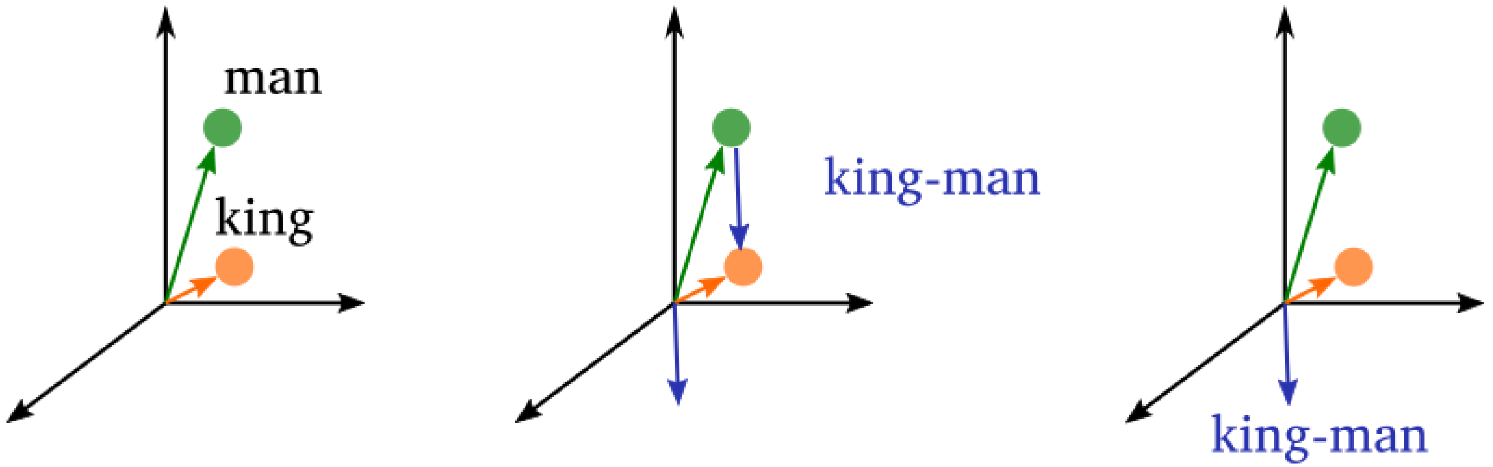
- Word *embeddings* associate words to n-dimensional vectors
  - trained on big text collections to model the word distributions in different sentences and contexts
  - able to capture the *semantic* information of each word
  - words with similar *meaning* share vectors with similar characteristics





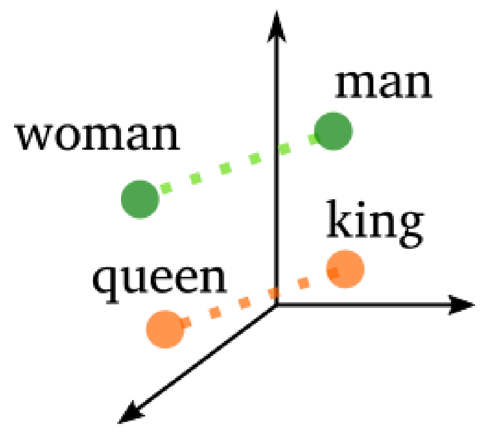
Word Embeddings (Word2Vec)

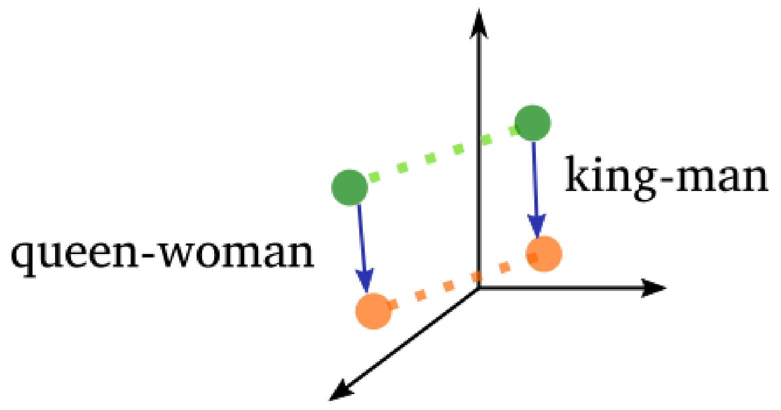
- Since each word is represented with a vector, operations among words (e.g. difference, addition) are allowed



Word Embeddings (Word2Vec)

- Semantic relationships among words are captured by vector positions





king - man = queen - woman
king - man + woman = queen