Machine Learning

Artificial Intelligence

What is ML?

Definition

- Arthur Samuel (1959)
  - "field of study that gives computers the ability to learn without being explicity programmed"
- Herbert Simon (1970)
  - "Learning is any process by which a system improves performance from experience."
- Tom Mitchell (1998)
  - "a well-posed learning problem is defined as follows: He says that a computer program is set to learn from an experience E with respect to some task T and some performance measure P if its performance on T as measured by P improves with experience E"
- Ethem Alpaydin (2010)
  - Programming computers to optimize a performance criterion using example data or past experience.

Necessity

- Better computational systems
  - To difficult or to expensive to be constructed manually
    - Systems that automatically adapt
      - Spam filters
    - Systems that discovers information in large database → data mining
      - Financial analysis
      - Text/image analyses
- Understanding the biological systems

How to design a ML?

- Improve of task **T**
  - Establish the goal (what has to be learned) – ***objective function*** – and its ***representation***
  - Select a learning algorithm to perform the inference of the goal based on experience
- Respect a performance metric **P**
  - Evaluation of the algorithm's performances
- Based on experience **E**
  - Select an experience database

Examples:

- T: playing checkers
- P: percent of winning games
- E: playing the game

- T: handwritten recognition
- P: percent of correct recognized words
- E: database of images with different words

- T: separate the spams
- P: percent of correct classified emails
- E: databases with annotated emails

Objective Function

What is the function that must be learned?

Ex.: checkers game → a function that:

Selects the next move

Evaluates a move

In order to identify the best next move

Ex. for checkers game

A linear combination of *#white_pieces*, *#black_pieces*, *#white_compromised_pieces*, *#black_compromised_pieces*

Representation of objective function

Different representations:

- Table
- Symbolic rules
- Numeric functions
- Probabilistic functions

There is a trade-off between

- How expressive is meaning of the representation
- Easy of learning

Objective function computation

- Polynomial time
- Non-polynomial time

The Learning Algorithm

Selection

- According to the training data
- Induce the hypothesis definition that
    - Match the data
    - Generalize the unseen data
- Main principle
    - Error minimisation (cost function – loss function)

Evaluation

Experimental

- By comparing different methods on different data (cross-validation)
- Collect data based on performances
    - Accuracy, training time, testing time
- Statistical analyse of the differences

Theoretic

- Mathematical analyse of algorithms and theorem proving
    - Computational complexity
    - Ability to match the training data
    - Complexity of the most relevant sample for learning

Comparison between 2 algorithms

Performance measures

- Parameters of a statistic series
- Proportion (percent) computed for a statistical series (ex. Accuracy)

Comparing based on confidence intervals

- For a problem and 2 solving algorithms with performances p_1 and p_2
- Confidence intervals

$$I_1=[p_1-\Delta_1,p_1+\Delta_1] \text{ and } I_2=[p_2-\Delta_2,p_2+\Delta_2]$$

- If $I_1 \cap I_2=\emptyset \rightarrow$ algorithm 1 works better than algorithm 2 (for the given problem)
- if $I_1 \cap I_2 \neq \emptyset \rightarrow$ impossible to decide

Confidence interval for the mean (average)

- For a statistical series of n data, with computed mean m and dispersion σ, determine the confidence interval of the mean μ
- $P(-z \leq (m-\mu)/(\sigma/\sqrt{n}) \leq z) = 1-\alpha \rightarrow \mu \in [m-z\sigma/\sqrt{n},\ m+z\sigma/\sqrt{n}]$
- $P = 95\% \rightarrow z = 1.96$

Confidence interval for accuracy

- For an accuracy p computed for n data, determine the confidence interval of accuracy
- $p \in [p-z(p(1-p)/n)^{1/2},\ p+z(p(1-p)/n)^{1/2}]$
- $P = 95\% \rightarrow z = 1.96$

$P=1-\alpha$	z
99.9%	3.3
99.0%	2.577
98.5%	2.43
97.5%	2.243
95.0%	1.96
90.0%	1.645
85.0%	1.439
75.0%	1.151

Training database

choose the training database based on:

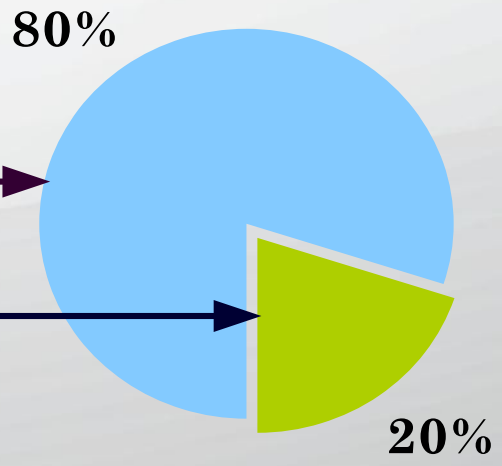
- direct experience
  - pairs (in, out) that are useful for the objective function
  - eg. checkers game → board game annotated by correct or incorrect move
- indirect experience
  - useful feedback (unlike i/o pairs) for the objective function
  - eg. checkers game → sequences of moves and the final score of the game

data sources

- random generated examples
  - positive and negative examples
- positive examples collected by a learner
- real examples

content

- **training data**
- **test data**



characteristics

- independent data
  - otherwise → collective learning
- training and testing data must respect the same distribution law
  - otherwise → *transfer learning/inductive transfer*
    - vehicle recognition → truck recognition
    - text analyses
    - spam filters

Training database

characteristics extracted (attributes) from raw data

- quantitative characteristics → nominal or rational scale
  - continuous values → weight
  - discrete values → # of computers
  - range values → event times
- qualitative characteristics
  - nominal → colour
  - ordinal → sound intensity (low, medium, high)
- structured
  - trees – root is a generalisation of children (vehicle → car, bus, tractor, truck)

data transformation

- standardisation → numerical attributes
  - remove the scale effect (different scale and units)
  - raw values are transformed in z scores
    - $Z_{ij} = (x_{ij} - \mu_j)/\sigma_j$, where x_{ij} – value of j^{th} attribute of i^{th} instance, μ_j (σ_j) is the mean (standard deviation) of j^{th} attribute for all instances
- selection of some attributes

ML classification – goal oriented

Intelligent systems for prediction

Aim: predict the output for a new input based on a previously learned model

Eg. predicting sales of a product for a time in the future based on price, calendar month, region, average income

Intelligent systems for classification

Aim: classify an object into one or more – known or unknown - categories based on their characteristics

Eg.: diagnostic systems for cancer: malign or benign or normal

Intelligent systems for regression

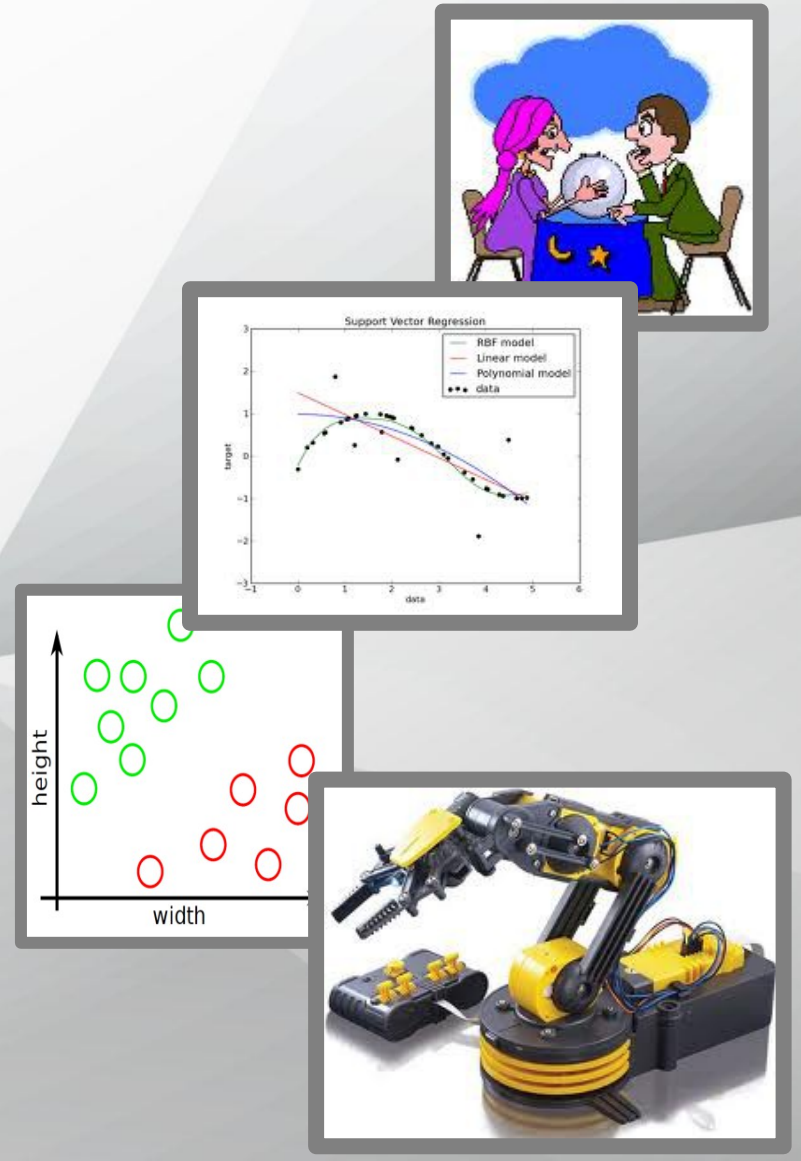
Aim: estimation of the (uni or multi variable) function's shape based on a previously learned model

Eg.: estimate the function that models the edge of a surface

Intelligent systems for planning

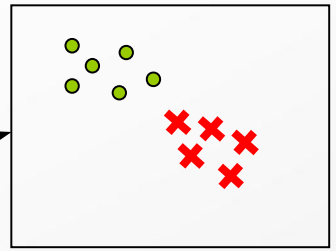
Aim: generate a sequence of optimal actions for performing a task

Eg.: planning the moves of a robot from a position to a source of energy



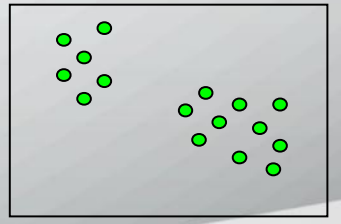
ML classification – based on learning experience

- Intelligent systems with supervised learning



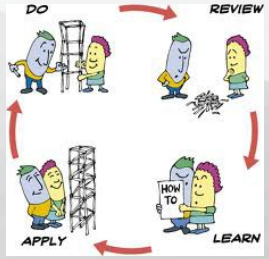
data is labeled

- Intelligent systems with unsupervised learning

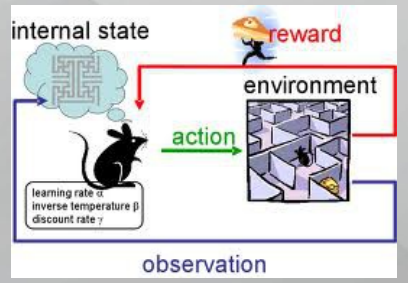


data is not labeled

- Intelligent systems with active learning



- Intelligent systems with reinforcement learning



Supervised learning

Aim

to provide a correct output for a new input

Definition

for a given data set (examples, instances, cases)

- training data – pairs $(atribute\_data_i, output_i)$, where
  - $i = 1,N$ (N = number of training data pairs)
  - $atribute\_data_i = (atr_{i1}, atr_{i2}, ..., atr_{im})$, m – number of atributes (characteristics, properties) for one data
  - $output_i$
    - a category from a predefined given set with k elements (number of classes) → classification problem
    - a real number → regression problem
- teste data – a set $(atribute\_data_i)$, $i = 1,n$ (n = number of test pairs).

determine

- a function (unknown) that maps the input atributes to the output on the training data set
- the output (class/value) associated with the test data (new) using the detected function

Other names

classification (regression), inductive learning

Process → 2 steps

- **Training**

  learning the classification model – using an algorithm
- **Testing**

  test the model with new data (not seen in the training step)

Feature

DB is labeled (for learning and sometimes also for testing)

Suitable problem types

- regression, classification

Supervised learning
learning quality

We consider a performance measure for the algorithm that is evaluated during both steps, training and testing, separately.

ex. accuracy (Acc = number of samples correct classified / total number of samples)

Evaluation Methods

for a large dataset	for a small dataset	for a very small dataset
disjunctive sets for training and testing	cross validation with *h* equal sub sets of the dataset	leave one out cross validation

Difficulties: **over-fitting** – good performance on training data but very poor on testing data

Performance measures:

- statistical, efficiency, robustness, scalability, interpret-ability, compactness, ...

Statistical metrics

Classification Accuracy

- the ratio of number of correct predictions to the total number of input samples

$$Acc = \frac{TP + TN}{N}$$

– works well only if there are equal number of samples belonging to each class.

Precision

- the number of correct positive results divided by the number of positive results predicted by the classifier.

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

Recall

- the number of correct positive results divided by the number of all relevant samples

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

Confusion Matrix

describes the complete performance of the model.

		Reality	
		pozitiv class	negativ class
computed results	pozitiv class	*True positiv (TP)*	*False positiv (FP)*
	negativ class	*False negative (FN)*	*True negative (TN)*

F1 Score

- is the Harmonic Mean between precision and recall.
- the range for F1 Score is [0, 1]

$$F1 = 2 * \frac{1}{\frac{1}{Prec} + \frac{1}{Recall}}$$

– how precise your classifier is (how many instances it classifies correctly),

– how robust it is (it does not miss a significant number of instances).

Mean Squared Error

- is the average of the squares of the differences between the original values and the predicted values.

$$MeanSquaredError = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

– it is easier to compute the gradient

Unsupervised learning

Aim

to detect a model or a internal util structure of the data

Definition

<u>for a given data set (examples, instances, cases)</u>

- training data – a set $\{atribute\_data_i \mid i=1,N\}$ where
  - N = number of training data pairs
  - $atribute\_data_i = (atr_{i1}, atr_{i2}, ..., atr_{im})$, m – number of atributes (characteristics, properties) for one data
- teste data – a set $(atribute\_data_i)$, $i=1,n$ (n = number of test pairs).

<u>determine</u>

- an unknown function that groups the training data in several classes
  - the number of classes k can be predefined or unknown
  - the data from one class are similar
- the output is the class the new input data (new) using the detected function

Examples of distances

consider two points p and q from R^m

- Euclidean $d(p,q)=\sqrt{\sum_{j=1}^{m}(p_j-q_j)^2}$
- Manhattan $d(p,q)=\sum_{j=1}^{m}|p_j-q_j|$
- Mahalanobis – measures the distance between a point p and a distribution Q
- Internal product $d(p,q)=\sum_{j=1}^{m}p_j q_j$
- Cosine $d(p,q)=\sum_{j=1}^{m}p_j q_j / (\sqrt{\sum_{j=1}^{m}p_j^2}\sqrt{\sum_{j=1}^{m}q_j^2})$
- Hamming – number of differences between p and q
- Levenshtein – the minimum number of transformations necessary to chage p in q

Distance vs. Similarity

Distance → min Similarity → max

Unsupervised learning

Other names
- clustering

Process → 2 steps
- Training
  - learning (determine), using an algorithm, the existing clusters
- Testing
  - test the model with new data (not seen in the training step)

Feature
- DB is not labeled (for learning and sometimes also for testing)

Suitable problem types
- Identifying some classes (clusters)
  - Genome analysis
  - Image processing
  - Social network analysis
  - Market segmentation
  - Astronomic data analysis
  - Computer clustering
- Dimensional reduction
- Identifying data properties (causes, explanations, etc.)
- Modeling data density

Learning quality

Internal criteria – high similarity within a cluster and reduced one between clusters
- distance inside the cluster
- distance between the clusters
- Davies-Bouldin index
- Dunn Index

External criteria – using benchmarks made from already grouped data sets
- Comparison with known data – almost impossible in practice
- Precision
- Recall
- F1-measure

Active learning

The algorithm can receive supplementary information during the learning step in order to improve it's performance.

Example: what is the subset of the training data that will facilitate the learning process.

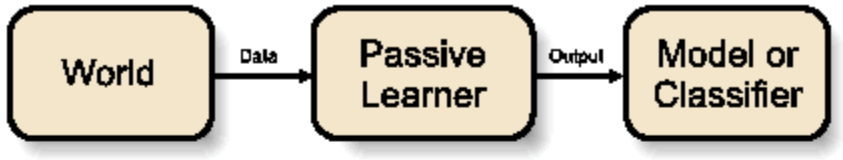


Figure 1.1: General schema for a passive learner.

Figure 1.2: General schema for an active learner.

Reinforcement learning

Aim

Learning, over a period of time, a course of action (behavior) that maximizes long-term rewards (earnings)

Problems' type

Ex. Training a Dog (Good and Bad Dog)

Characteristic

Interaction with the environment (actions → rewards)

Decision sequence

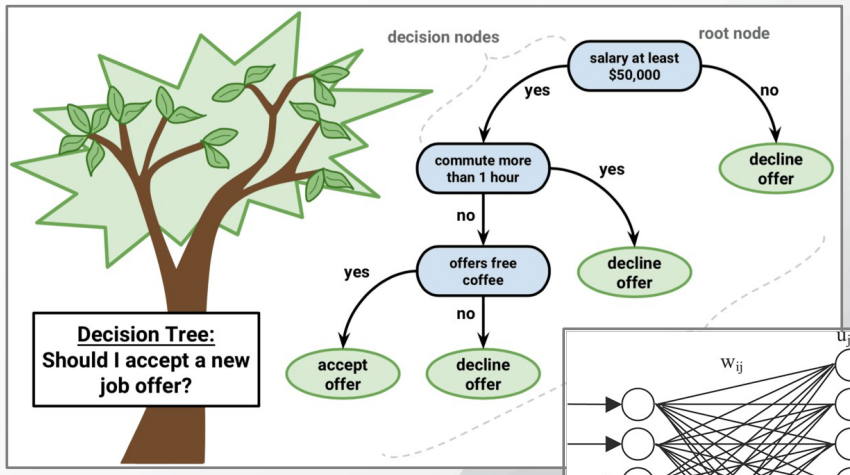
Supervised learning

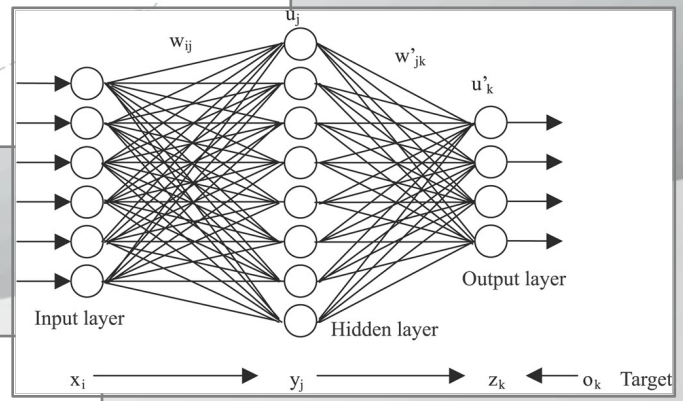
Decision → consequence (malignant or benign cancer)

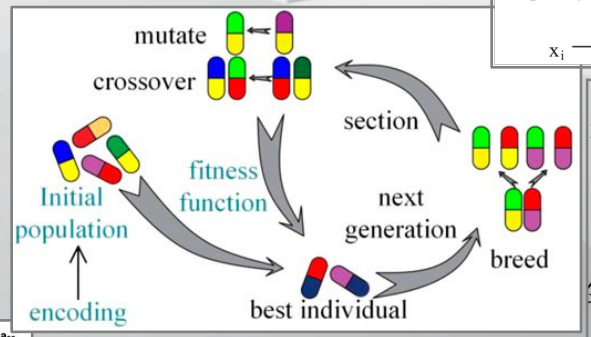
Intelligent Systems

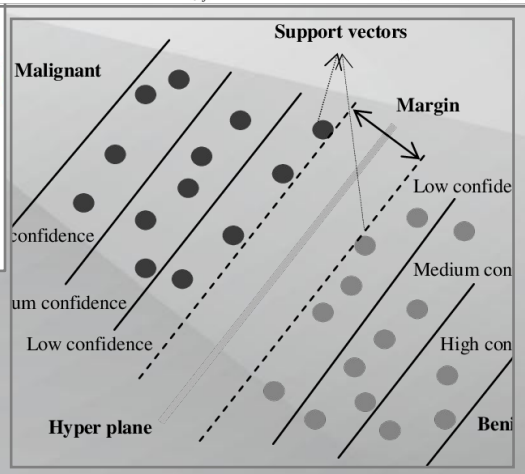
Based on algorithm

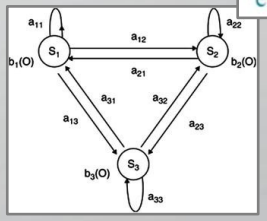
- **Decision trees**
- **Artificial Neural Networks**
- **Evolutionary algorithms**
- **Support Vector Machines**
- **Hidden Markov Models**











Decision Trees (DT)

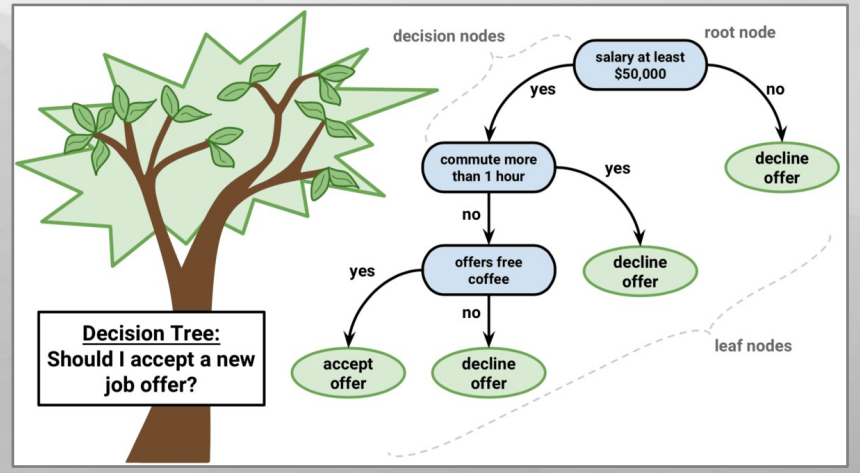
Aim

- divide a collection of articles in smaller sets by successively applying some decision rules → asking more questions

  each question is addressed based on the answer of the previous question
- elements are characterized by non-metric information

Definition

- Decision tree – a special graph (bi-color and oriented tree)
  - Contains three node types:
    - Decision nodes → possibilities of decider (a test on an attribute of item that must be classified)
    - Hazard nodes → random events outside the control of decider (exam results, therapy consequences)
    - Result nodes → final states that have a utility or a label
  - Decision and hazard nodes alternate on the tree levels
  - Result nodes → leaf (terminal nodes)
  - (oriented) edges of the tree consequences of decisions (can be probabilistic)
- Each internal node corresponds to an attribute
- Each branch under a node (attribute) corresponds to the value of that attribute
- Each leaf corresponds to a class



Problems solved with DT

Properties

- problem's instances are represented by a fixed number of attributes, each attribute having a finite number of values;
- objective function takes discrete values;
- DT represents a dis-junction of more conjunctions, each conjunction being "attribute a_i has value v_j";
- training data could contain errors;
- training data could be incomplete;
    - Some data have not all attributes.

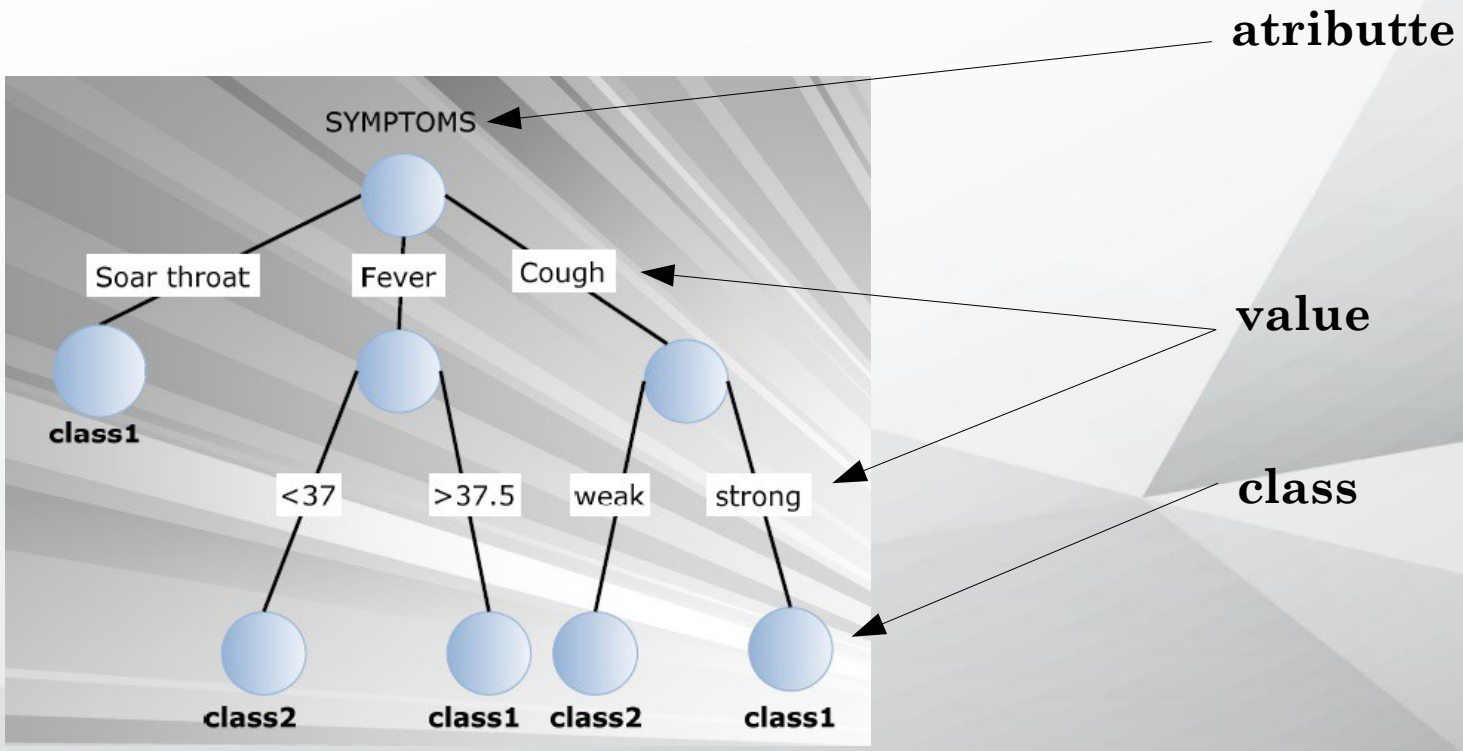
Classification problem

- Binary classifications
    - Instances are [$(attribute_{ij}, value_{ij})$, $class_i$], $i=1,2,...,n$, $j=1,2,...,m$, and $class_i$ taking 2 values;
- Multi-class (k-class)
    - Instances are [$(attribute_{ij}, value_{ij})$, $class_i$], $i=1,2,...,n$, $j=1,2,...,m$, and $class_i$ taking k values;

Regression problems

- instead to label each node by the label of a class, each node has associated a real value or a function that depends on the inputs of that node;
- input space is split in decision regions by parallel cuttings to *Ox* and *Oy;*
- discrete outputs are transformed in continuous functions;
- quality of problem solving:
    - Prediction error (square or absolute)

Examples of DT



Example of train data for DT

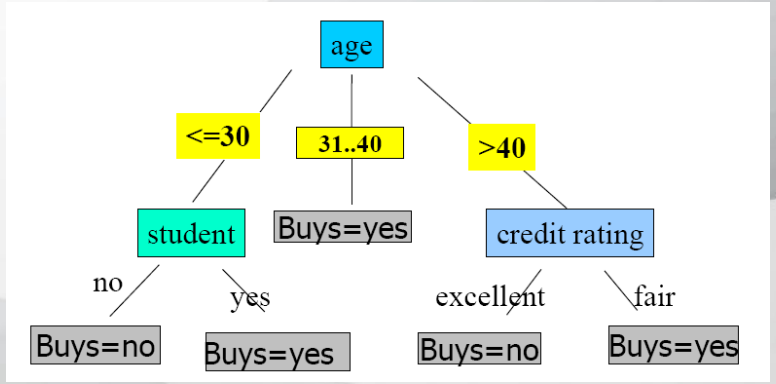
Credits

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	**No**
2	young	false	false	good	**No**
3	young	true	false	good	**Yes**
4	young	true	true	fair	**Yes**
5	young	false	false	fair	**No**
6	middle	false	false	fair	**No**
7	middle	false	false	good	**No**
8	middle	true	true	good	**Yes**
9	middle	false	true	excellent	**Yes**
10	middle	false	true	excellent	**Yes**
11	old	false	true	excellent	**Yes**
12	old	false	true	good	**Yes**
13	old	true	false	good	**Yes**
14	old	true	false	excellent	**Yes**
15	old	false	false	fair	**No**

Examples of DT

rec	Age	Income	Student	Credit_rating	Buys_computer(CLASS)
r1	<=30	High	No	Fair	No
r2	<=30	High	No	Excellent	No
r3	31…40	High	No	Fair	Yes
r4	>40	Medium	No	Fair	Yes
r5	>40	Low	Yes	Fair	Yes
r6	>40	Low	Yes	Excellent	No
r7	31…40	Low	Yes	Excellent	Yes
r8	<=30	Medium	No	Fair	No
r9	<=30	Low	Yes	Fair	Yes
r10	>40	Medium	Yes	Fair	Yes
r11	<=30	Medium	Yes	Excellent	Yes
r12	31…40	Medium	No	Excellent	Yes
r13	31…40	High	Yes	Fair	Yes
r14	>40	Medium	No	Excellent	No



Process – DT

Tree construction (induction)

Based on training data

Works bottom-up or top-down (splitting)

Using the tree as a problem solver

All decisions performed along a path from the root to a leaf form a rule

Rules from DT are used for labeling new data

Pruning

Identify and move/eliminate branches that reflect noise or exceptions

Tree construction

Split the training data into subsets based on the characteristics of data

a node – question related to a property

branches of a node – possible answers to the question of the node

initially, all examples are located in the root

- an attribute gives the root label and its values give the branches

on next levels, examples are partitioned based on their attributes → order of attributes

- for each node, an attribute is (recursively) chosen – its values → branches

splitting – greedy decision making

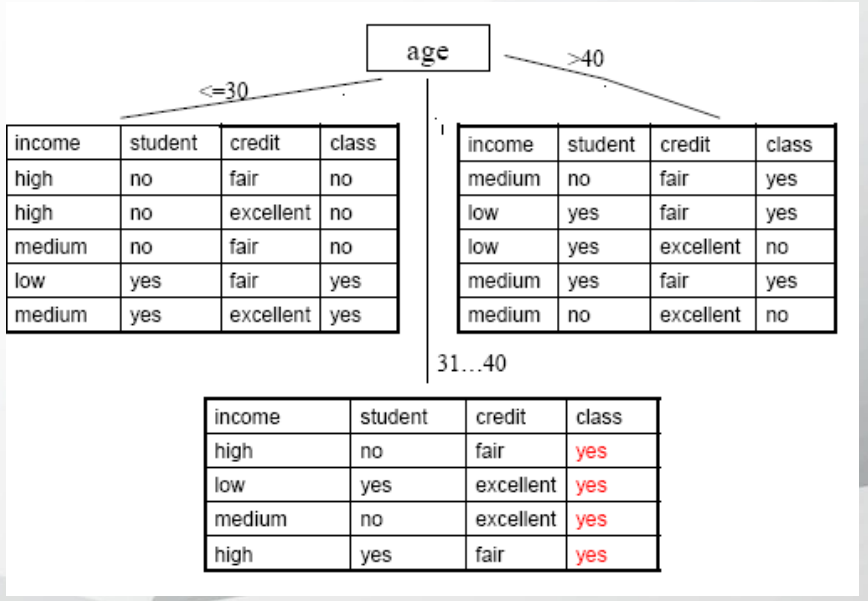
Iterative process

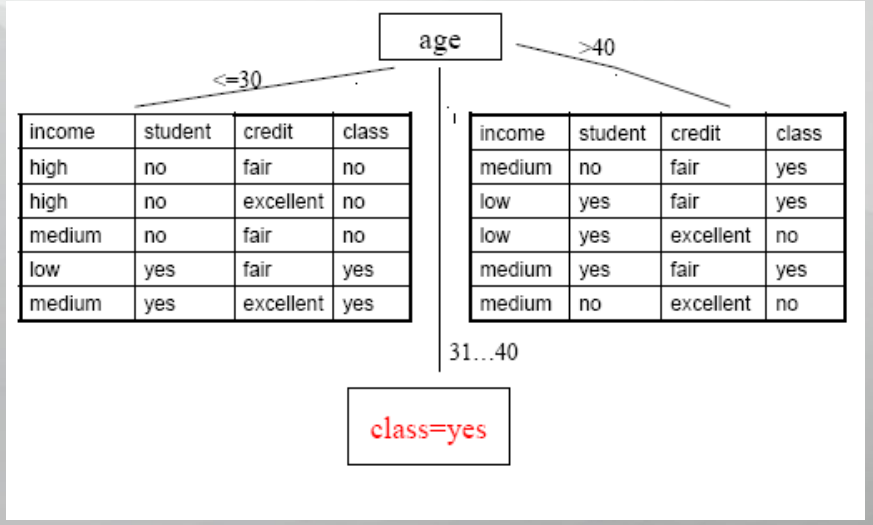
stop conditions:

- all examples from a node belong to the same class → node is a leaf and is labeled by $class_i$
- there are no left examples → node becomes a leaf and is labeled by the majority class of training data
- there are no attributes

Example – tree construction

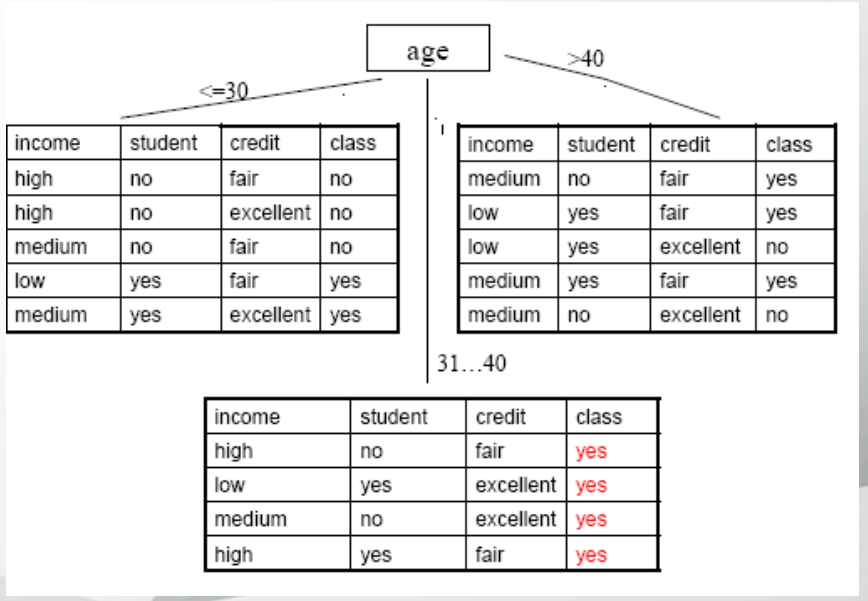
rec	Age	Income	Student	Credit_rating	Buys_computer(CLASS)
r1	<=30	High	No	Fair	No
r2	<=30	High	No	Excellent	No
r3	31…40	High	No	Fair	Yes
r4	>40	Medium	No	Fair	Yes
r5	>40	Low	Yes	Fair	Yes
r6	>40	Low	Yes	Excellent	No
r7	31…40	Low	Yes	Excellent	Yes
r8	<=30	Medium	No	Fair	No
r9	<=30	Low	Yes	Fair	Yes
r10	>40	Medium	Yes	Fair	Yes
r11	<=30	Medium	Yes	Excellent	Yes
r12	31…40	Medium	No	Excellent	Yes
r13	31…40	High	Yes	Fair	Yes
r14	>40	Medium	No	Excellent	No

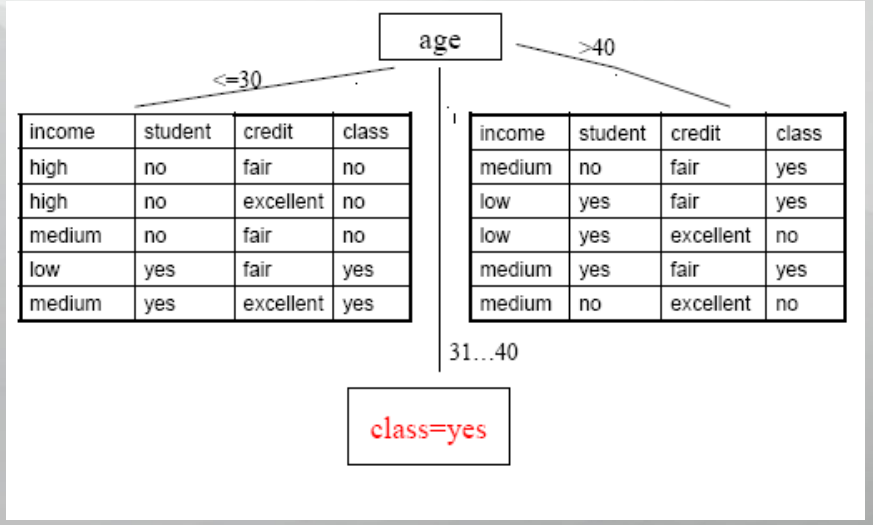




Example – tree construction

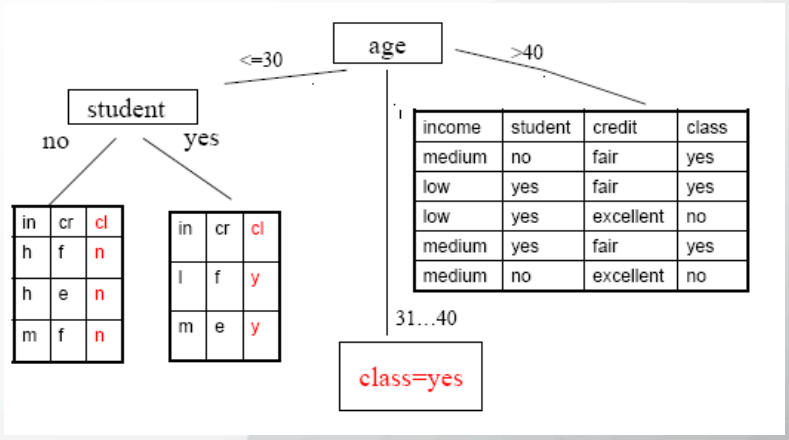
rec	Age	Income	Student	Credit_rating	Buys_computer(CLASS)
r1	<=30	High	No	Fair	No
r2	<=30	High	No	Excellent	No
r3	31…40	High	No	Fair	Yes
r4	>40	Medium	No	Fair	Yes
r5	>40	Low	Yes	Fair	Yes
r6	>40	Low	Yes	Excellent	No
r7	31…40	Low	Yes	Excellent	Yes
r8	<=30	Medium	No	Fair	No
r9	<=30	Low	Yes	Fair	Yes
r10	>40	Medium	Yes	Fair	Yes
r11	<=30	Medium	Yes	Excellent	Yes
r12	31…40	Medium	No	Excellent	Yes
r13	31…40	High	Yes	Fair	Yes
r14	>40	Medium	No	Excellent	No

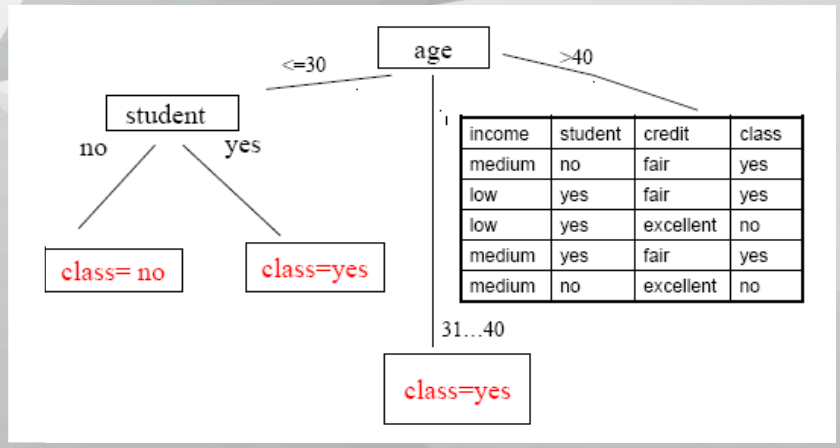




Example – tree construction

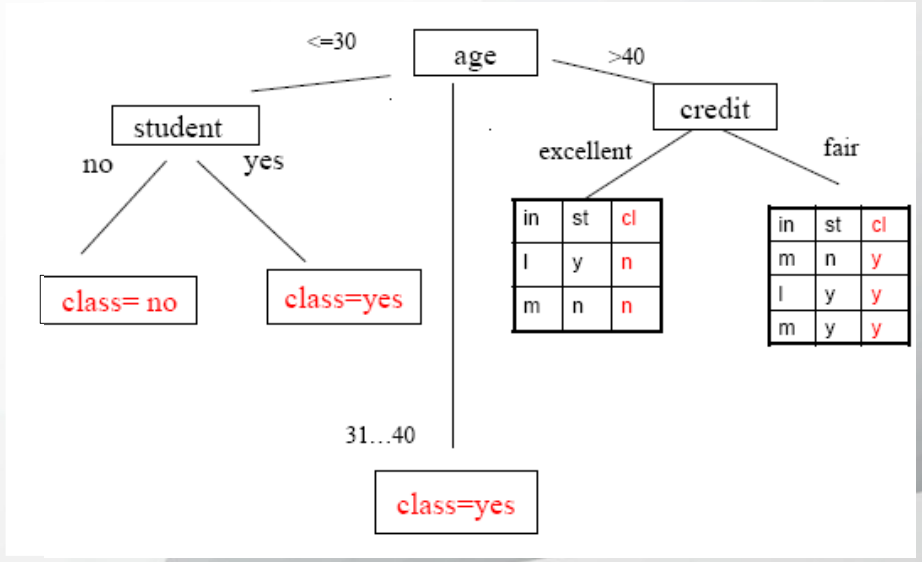
rec	Age	Income	Student	Credit_rating	Buys_computer(CLASS)
r1	<=30	High	No	Fair	No
r2	<=30	High	No	Excellent	No
r3	31…40	High	No	Fair	Yes
r4	>40	Medium	No	Fair	Yes
r5	>40	Low	Yes	Fair	Yes
r6	>40	Low	Yes	Excellent	No
r7	31…40	Low	Yes	Excellent	Yes
r8	<=30	Medium	No	Fair	No
r9	<=30	Low	Yes	Fair	Yes
r10	>40	Medium	Yes	Fair	Yes
r11	<=30	Medium	Yes	Excellent	Yes
r12	31…40	Medium	No	Excellent	Yes
r13	31…40	High	Yes	Fair	Yes
r14	>40	Medium	No	Excellent	No

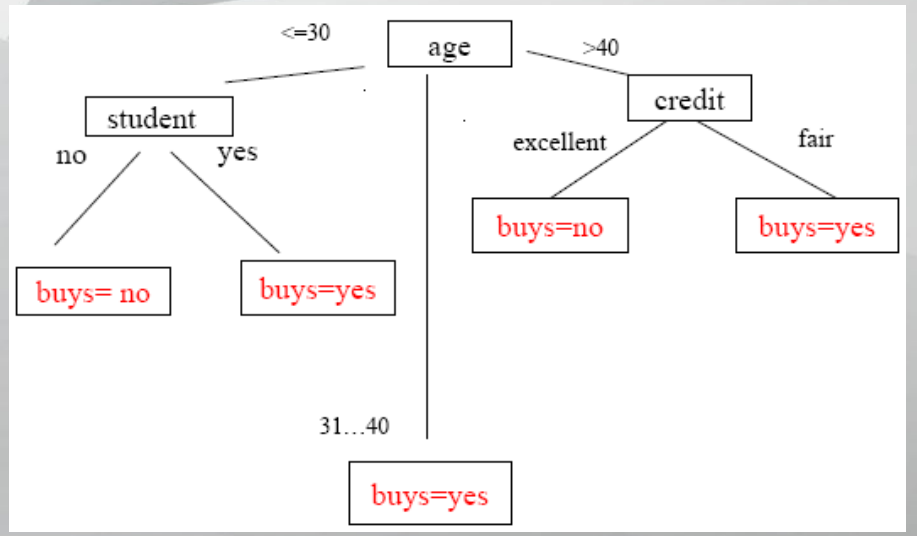




Example – tree construction

rec	Age	Income	Student	Credit_rating	Buys_computer(CLASS)
r1	<=30	High	No	Fair	No
r2	<=30	High	No	Excellent	No
r3	31…40	High	No	Fair	Yes
r4	>40	Medium	No	Fair	Yes
r5	>40	Low	Yes	Fair	Yes
r6	>40	Low	Yes	Excellent	No
r7	31…40	Low	Yes	Excellent	Yes
r8	<=30	Medium	No	Fair	No
r9	<=30	Low	Yes	Fair	Yes
r10	>40	Medium	Yes	Fair	Yes
r11	<=30	Medium	Yes	Excellent	Yes
r12	31…40	Medium	No	Excellent	Yes
r13	31…40	High	Yes	Fair	Yes
r14	>40	Medium	No	Excellent	No





ID3/C4.5 algorithm

```
generate(D, A){      //D – a partitioning of training data, A – list of attributes
    create a new node N
    if examples from D belong to a single class C then
        node N becomes a leaf and is labeled by C
        return node N
    else
        if A=∅ then
            node N becomes a leaf and is labeled by majority class of D
            return node N
        else
            separation_attribute = AttributeSelection(D, A)
            label node N by separation_attribute
            for all possible values vj of separation_attribute
                let Dj – set of examples from D that have separation_attribute=vj
                if Dj =∅ then
                    add a leaf (to node N) labeled by majority class of D
                else
                    add a node (to node N) return by generate(Dj, A–separation_attribute)
            return node N
}
```

Greedy, recursive, top-down, divide-and-conquer

AttributeSelection(D,A)

selects the attribute that corresponds to a node (root or internal node)

- Random
- Attribute with the fewest/most values
- Based on a pr-established order:
- Information gain
  - Gain rate
  - Distance between partitions created by the attribute

Information gain

An impurity measure

- 0 (minim) – if all examples belong to the same class
- 1 (maxim) – if examples are uniform distributed over classes

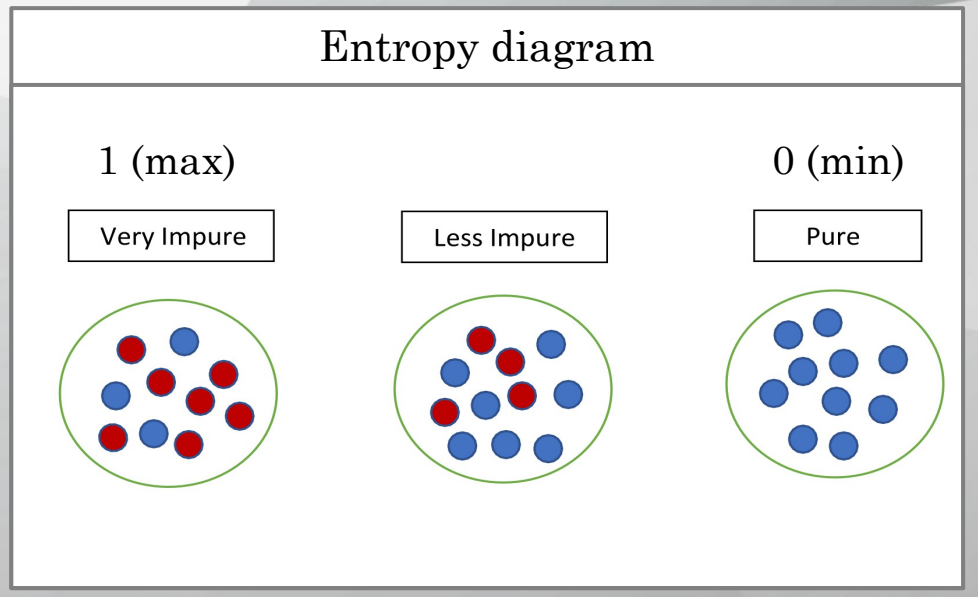
Based on data entropy

- Expected number of bits required by coding the class of an element from data
- Binary classification (2 classes): $E(S) = - p_+ log_2 p_+ - p_- log_2 p_-$ where
  - p_+ - proportion of positive examples in dataset S
  - p_- - proportion of negative examples in dataset S
- Multi-class classification: $E(S) = \sum_{i=1, 2, ..., k} - p_i log_2 p_i$

  – data entropy related to target attribute (output attribute), where
  - p_i – proportion of examples from class i in dataset S

Information gain of an attribute

- How the elimination of attribute a reduces the dataset's entropy
  - $Gain(S, a) = E(S) - \sum_{v \in valori(a)} |S_v| / |S| E(S_v)$
  - $\sum_{v \in values(a)} |S_v| / |S| E(S_v)$ – expected information



Example – Information gain

	a1	a2	a3	Class
d1	big	red	circle	class 1
d2	small	red	square	class 2
d3	small	red	circle	class 1
d4	big	blue	circle	class 2

$$S=\{d_1,d_2,d_3,d_4\}\Rightarrow p_+=\frac{2}{4},\ p_-=\frac{2}{4}\Rightarrow E(S)=-p_+\log_2 p_+-p_-\log_2 p_-=1$$

$$S_{v=\text{big}}=\{d_1,d_4\}\Rightarrow p_+=\frac{1}{2},\ p_-=\frac{1}{2}\Rightarrow E(S_{v=\text{big}})=1$$

$$S_{v=\text{small}}=\{d_2,d_3\}\Rightarrow p_+=\frac{1}{2},\ p_-=\frac{1}{2}\Rightarrow E(S_{v=\text{small}})=1$$

$$S_{v=\text{red}}=\{d_1,d_2,d_3\}\Rightarrow p_+=\frac{2}{3},\ p_{minus}=\frac{1}{3}\Rightarrow E(S_{v=\text{red}})=0.923$$

$$S_{v=\text{blue}}=\{d_4\}\Rightarrow p_+=0,\ p_-=1\Rightarrow E(S_{v=\text{blue}})=0$$

$$S_{v=\text{circle}}=\{d_1,d_3,d_4\}\Rightarrow p_+=\frac{2}{3},\ p_{minus}=\frac{1}{3}\Rightarrow E(S_{v=\text{cicle}})=0.923$$

$$S_{v=\text{square}}=\{d_4\}\Rightarrow p_+=0,\ p_-=1\Rightarrow E(S_{v=\text{square}})=0$$

Formula for information gain

$$Gain(S,a)=E(S)-\sum_{v\in values(a)}\frac{|S_v|}{|S|}E(S_v)$$

$$Gain(S,a_1)=1-(\frac{|S_{v=\text{big}}|}{|S|}E(S_{v=\text{big}})+\frac{|S_{v=\text{small}}|}{|S|}E(S_{v=\text{small}}))=1-(\frac{2}{4}1+\frac{2}{4}1)=0$$

$$Gain(S,a_2)=1-(\frac{|S_{v=\text{red}}|}{|S|}E(S_{v=\text{red}})+\frac{|S_{v=\text{blue}}|}{|S|}E(S_{v=\text{blue}}))=1-(\frac{3}{4}0.923+\frac{1}{4}0)=0.307$$

$$Gain(S,a_2)=1-(\frac{|S_{v=\text{circle}}|}{|S|}E(S_{v=\text{circle}})+\frac{|S_{v=\text{square}}|}{|S|}E(S_{v=\text{square}}))=1-(\frac{3}{4}0.923+\frac{1}{4}0)=0.307$$

Gain rate

Penalises an attribute by integrating a new term that depends on spreading degree and on uniformity degree of separation – *Split information*

Split information

- entropy related to possible values of attribute a
- describes the potential worth of splitting a branch from a node

$$SplitInformation(S,a) = -\sum_{v=value(a)} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{S}$$

where S_v is the proportion of examples from dataset S that have attribute a with value v

Information gain ratio is the ratio between the *information gain* and the *split information* value:

$$IGT(S,a) = \frac{gain(S,a)}{SplitInformation(S,a)}$$

aims to reduce a bias towards multi-valued attributes!

Using a tree as a solver

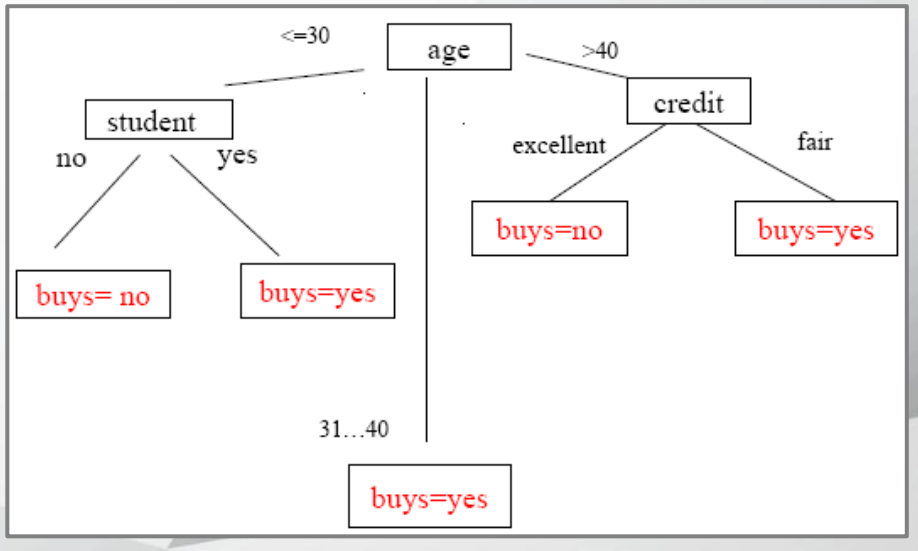
Extract the rules from the constructed tree

- IF *age* = *"<=30"* AND *student* = *"no"* THEN *buys_computer* = *"no"*
- IF *age* = *"<=30"* AND *student* = *"yes"* THEN *buys_computer* = *"yes"*
- IF *age* = *"31…40"* THEN *buys_computer* = *"yes"*
- IF *age* = *">40"* AND *credit_rating* = *"excellent"* THEN *buys_computer* = *"no"*
- IF *age* = *">40"* AND *credit_rating* = *"fair"* THEN *buys_computer* = *"yes"*

Use the rules for classifying the test data (new data)

- for *x* (a data without class) – rules can be written as predicates

- IF *age(x, <=30)* AND *student(x, no)* THEN *buys_computer (x, no)*
- IF *age(x, <=30)* AND *student (x, yes)* THEN *buys_computer (x, yes)*



Difficulties

- *Underfitting* – DT constructed on training data is to simple → large classification error during training and testing
- *Overfitting* – DT constructed on training data match the training data, but it ca not generalize new data

Solutions

- Pruning – remove some branches (not useful, redundant) → smaller tree
- Cross-validation

Pruning a tree

pre-pruning

Increasing the tree is stopped during construction by stopping the division of nodes that become leaf labeled by majority class of examples from that node

post-pruning

After the DT is constructed, eliminate the branches of some nodes that become leaf → classification error reduces (on testing data)

Motivation – simplifying the tree

- After the DT is constructed, classification rules are extracted in order to represent the knowledge as *if-then* rules (easy to understand)
- A rule is create by traversing the DT from root to a leaf
- Each pair (*attribute, value*) ↔ (*node, edge*) – is a conjunction in the premise of the rule (*if* part), except the last node of the path that is a leaf and represents the consequence (output, *then* part) of the rule

Conclusion

Advantages

- Easy to understand and interpret
- Can use nominal or categorized data
- Decision logic can be easy followed (rules are visible)
- Works better with large data

Disadvantages

- Instability due some change the training data
- Complexity due the representation
- Difficult to use
- The DT construction is expensive
- The DT construction requires a lot of information

Tool example:

WEKA J48

THANK YOU !