Supervised Learning
- Part 1 -

Road Map

- What is supervised learning
- Evaluation of classifiers
- Decision trees. ID3 and C4.5
- Rule induction systems
- Summary

Objectives

- Supervised learning is one of the most studied subdomain of Data Mining
- It is also part of Artificial Intelligence (subdomain of Machine learning)
- Means that a new model can ge built starting from past experiences (data)

Definitions

- Supervised learning includes:
  - Classification: results are discrete values
  - Regression: results are continuous or ordered values

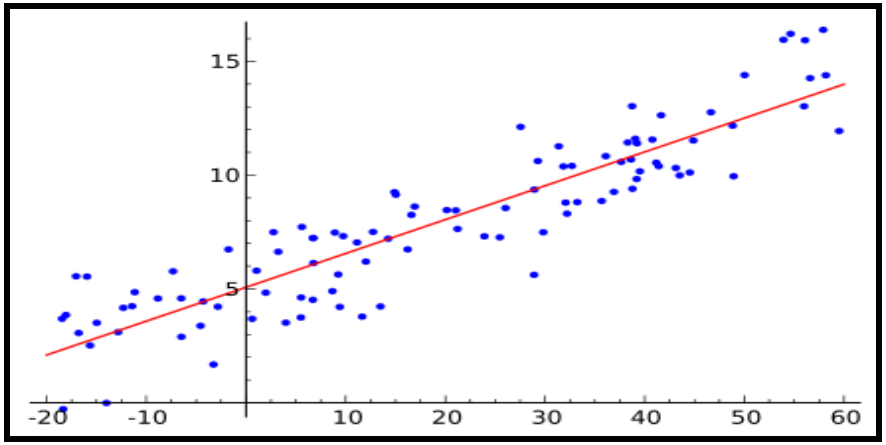
Regression

- Regression comes from statistics.
- Meaning: predicting a value of a given continuous valued variable based on the values of other variables, assuming a linear or nonlinear model of dependency ([Tan, Steinbach, Kumar 06]).
- Used in prediction and forecasting - its use overlaps machine learning.
- Regression analysis is also used to understand relationship between independent variables and dependent variable and can be used to infer causal relationships between them.

Example

Linear regression example

(from http://en.wikipedia.org/wiki/File:Linear_regression.svg)



In this example: for new values on Ox axis the Oy value can be predicted using the regression function.

Classification

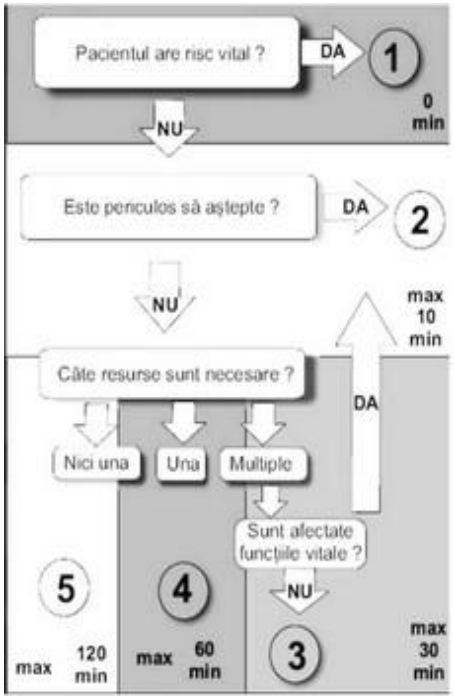
Input:

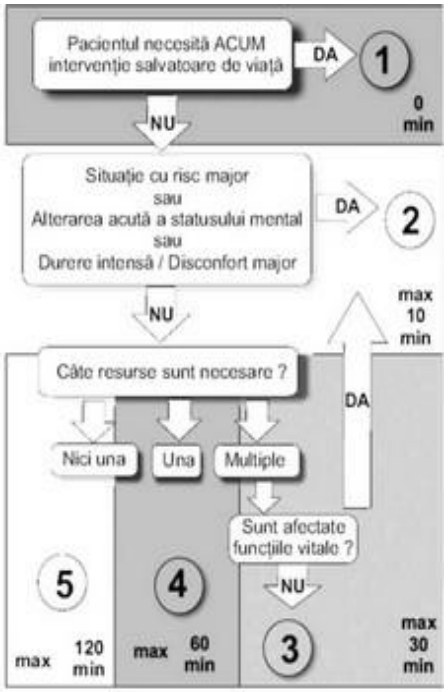
- A set of k classes $C = \{c_1, c_2, \ldots, c_k\}$
- A set of n labeled items $D = \{(d_1, c_{i1}), (d_2, c_{i2}), \ldots, (d_{kn} c_{in})\}$. The items are $d_1, \ldots, d_n$, each item dj being labeled with class $c_j \in C$. D is called the **training set**.
- For some algorithms calibration, a **validation set** is required. This validation set contains also labeled items not included in the training set.

Output:

- A model or method for classifying new items. The set of new items that will be classified using the model/method is called the **test set**

Example. Model: decision tree





- The result for the example:

Felix	**Yes**	**Yes**	**No**	**No**	**No**	**Yes**	**?????**

will be C0

Input data format

- In most of the cases the training set (as well as the validation set and the test set) may be represented by a **table** having a *column for every attribute* of X and the *last column contains the class label*.
- A very used example is **Play-tennis** where weather conditions are used to decide if players may or not start a new game.
- This dataset will be used also in this course.

Play tennis dataset

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Approaches

- The interest in supervised learning is a shared between:
  - statistics,
  - data mining and
  - artificial intelligence
- There are a wide range of problems solved by supervised learning techniques, the number of algorithms or methods in this category is very large.
- There are many approaches and in this course (and the next) only some of them are covered, as follows.

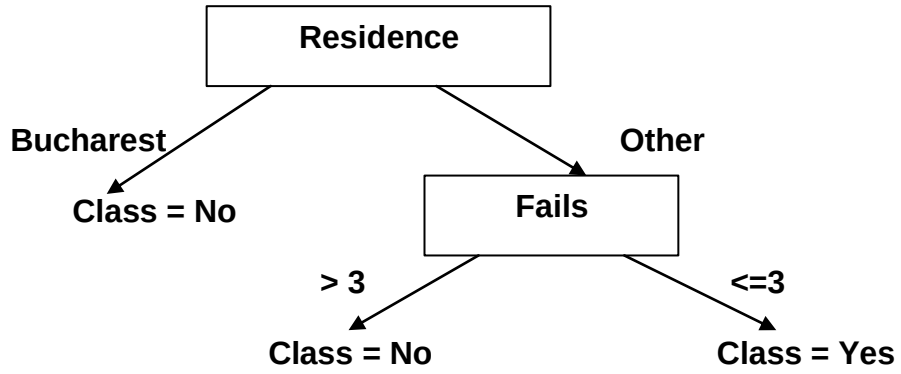
Decision trees

- Decision trees: an example is the UPU decision tree as described in the previous example.
- In a decision tree non-leaf nodes contain decisions based on the attributes of the examples (attributes of argument $\mathbf{x_i}$) and each leaf i $\mathbf{y_i}$ is a class from Y.
- ID3 and C4.5 are two well known algorithms for building decision trees and are presented in this course.

Rule induction systems

- Rule induction systems: from each decision tree a set of rules can be inferred, rules that can replace the decision tree.
- Consider the following decision tree for deciding if a student will be allowed or not in the students residence.

Rule induction systems



The decision tree can be replaced by the following set of rules (one for each path):

- Residence = Bucharest → Class = No
- Residence = Other, Fails > 3 → Class = No
- Residence = Other, Fails <= 3 → Class = Yes

Rule induction systems

- But rules can be obtained not only from decision trees but also directly from the training set.
- This course presents some methods for doing this job.

Classification using association rules

- Classification using association rules: Class association rules can be used in building classifiers.
- Some methods for performing this task are presented.

Naïve Bayesian classification

- Naïve Bayesian classification: for every example $\mathbf{x}_i$ the method computes the probability for each class $\mathbf{y}_j$ from C.
- Classification is made picking the most probable class for each example.
- The word naïve is used because some simplifying assumptions are made.

Support vector machines

- Support vector machines: this method is used for binary classification.
- Examples are classified in only two classes: either positive or negative.
- It is a very efficient method and may be used recursively to build a classifier with more than two classes.

KNN

- K-nearest neighbor: a very simple but powerful method for classifying examples based on the labels of their neighbors.

Ensemble methods

- Ensemble methods: Random Forest, Bagging and Boosting.
- In these cases more than one classifier is built and the final classification is made by aggregating their results.
- For example, a Random Forest consists in many decision trees and the output class is the mode of the classes output by individual trees.

Road Map

- What is supervised learning
- Evaluation of classifiers
- Decision trees. ID3 and C4.5
- Rule induction systems
- Summary

Accuracy and error rate

- For estimating the efficiency of a classifier several measures may be used:
- **Accuracy** (or predictive accuracy) is the proportion of the test examples set correctly classified by the model (or method):

$$Accuracy = \frac{Number\ of\ correct\ classifications}{Total\ number\ of\ test\ examples}$$

- **Error rate** is the proportion of incorrectly classified test examples:

$$Error\_rate = 1 - Accuracy$$

Other measures

- In some cases where examples are classified in only two classes (called Positive and Negative) other measures can be also defined.
- Consider the confusion matrix containing the number of correctly and incorrectly classified examples (Positive examples as well as Negative examples):

	Classified as Positive	Classified as Negative
Actual positive	TP = True Positive	FN = False Negative
Actual negative	FP = False Positive	TN = True Negative

Other measures

- TP = the number of correct classifications for Positive examples.
- TN = the number of correct classifications for Negative examples.
- FP = the number of incorrect classifications for Negative examples.
- FN = the number of incorrect classifications for Positive examples.

- **Precision** is the proportion of the correctly classified Positive examples in the set of examples classified as Positive:

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

Other measures

- **Recall** (or **sensitivity**) is the proportion of correctly classified Positive examples in the set of all Positive examples, or the rate of recognition for positive examples:

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

- **Specificity** is the rate of recognition of negative examples:

$$Sensitivity = \frac{TN}{TN + FP}$$

Other measures

- Now accuracy formula can be rewritten as:

$$Accuracy = \frac{Recall * Pos}{Pos + Neg} + \frac{Specificity * Neg}{Pos + Neg}$$

where Positive and Negative are the total number of Positive examples and Negative examples.

Other measures

- Precision and recall are usually used together because for some test examples using only one of them may lead to incorrect judgment on the performances of a classifier.
- If a set contains 100 Positive examples and 100 Negative examples and the classifier has the following result:

	Classified as Positive	Classified as Negative
Actual positive	30	70
Actual negative	0	100

Other measures

- Then precision p = 100% but recall r = 30%. Combining precision with recall by harmonic mean **F_1-score** is obtained:

$$F1\text{-}score = \frac{2}{\frac{1}{p} + \frac{1}{r}} = \frac{2pr}{p + r}$$

- For the above example F_1-score = 46%; generally F_1-score is closer to the smaller value of precision and recall.

Evaluation methods

- Evaluation methods use a data set D with labeled examples.
- This set is split in several subsets and these subsets become training/test/validation sets.

- ***Note: Evaluation refers to the classifier building method/algorithm***.

The holdout method

- In this case the data set D is split in two: a ***training set*** and a ***test set***.
- The test set is also called ***holdout set*** (from here the name of the method).
- The classifier obtained using the training set is used for classification of examples from the test set.
- Because these examples are also labeled accuracy, precision, recall and other measures can then be obtained and based on them the classifier is evaluated.

Cross validation method

There are several versions of cross validation:

1. ***k-fold cross validation***. The data set D is split in k disjoint subsets with the same size. For each subset a classifier is built and run using that subset as test set and the reunion of all k-1 remaining subsets as training set. In this way k values for accuracy are obtained (one for each classifier). The mean of these values is the final accuracy. The usual value for k is 10.
2. ***2-fold cross validation***. For k=2 the above method has the advantage of using large sets both for training and testing.

Cross validation method

3. ***Stratified cross validation***. Is a variation of k-fold cross validation. Each fold has the same distribution of the labels.

- For example, for Positive and Negative examples each fold contains roughly the same proportion of Positive examples and the same proportion of Negative examples.

4. ***Leave one out cross validation***. When D contains only a small number of examples a special k-fold cross validation may be used: each example becomes the test set and all other examples the training set.

- Accuracy for each classifier is either 100% or 0%. The mean of all these values is the final accuracy.

Bootstrap method

- Is part of resampling methods and consists in getting the training set from the data set with by sampling with replacement.
- The instances which are not picked in the training set are used as test set.
- For example, if D has 1000 labeled examples, by picking randomly an example 1000 times gives us the training set. In this training set some examples are picked more than one time.

Bootstrap method

- Statistically 63.2% of the examples in D are picked from the training set and 36.8% are not. These 36.8% becomes the test set.
- After building a classifier and run this classifier on the test set the accuracy is determined and the classifier building method may be evaluated based on its value.
- More on this method and other evaluation techniques can be found in [Sanderson 08].

Scoring and ranking

- Sometimes the user is interested only in a single class (called the Positive class for short), for example buyers of a certain type of gadgets or players of a certain game.
- If the classifier returns a probability estimate (PE) for each example in the test case to belong to the Positive class (indicating the likelihood to belong to that class) we can ***score*** each example by the value of this PE.
- After that we can ***rank*** all examples based on their PE and draw a ***lift curve***.
- The classifier method is good if the lift curve is way above the random line in the lift chart – see example.

Scoring and ranking

- The lift curve is draw by dividing the ranked examples in several bins and counting the actual Positive examples in each bin.
- This count gives the value for the lift curve.
- Remember that the evaluation of the classification methods uses a test set with labeled examples.

Example (from [Microsoft])

- "The marketing department at Adventure Works Cycles wants to create a targeted mailing campaign. From past campaigns, they know that a 10 percent response rate is typical. They have a list of 10,000 potential customers stored in a table in the database. Therefore, based on the typical response rate, they can expect 1,000 of the potential customers to respond.
- However, the money budgeted for the project is not enough to reach all 10,000 customers in the database. Based on the budget, they can afford to mail an advertisement to only 5,000 customers. The marketing department has two choices:
- Randomly select 5,000 customers to target

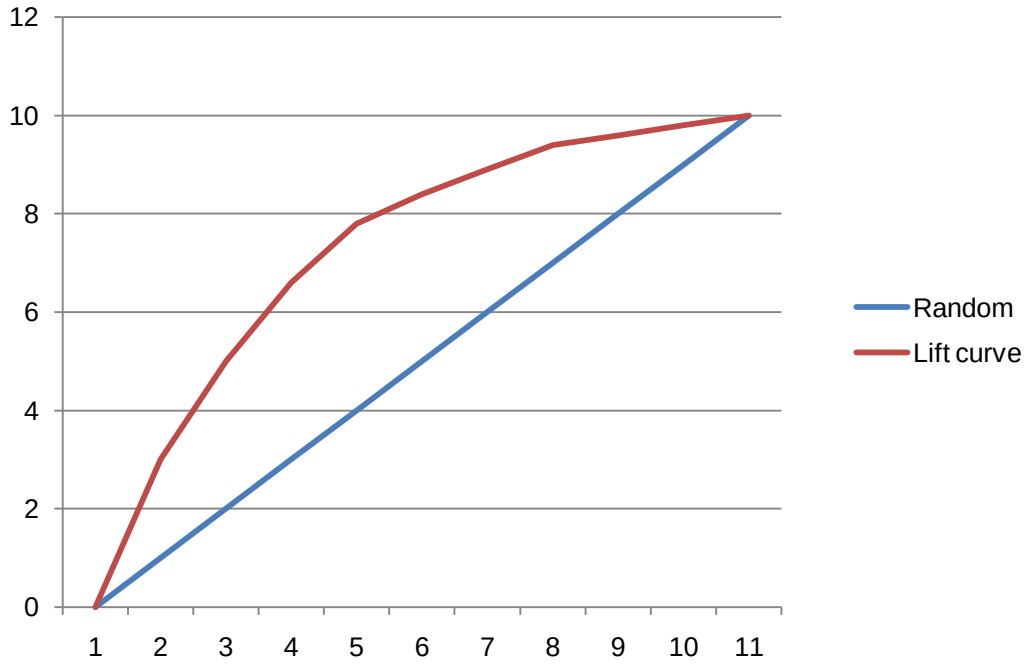
Example (from [Microsoft])

- Use a mining model to target the 5,000 customers who are most likely to respond
- If the company randomly selects 5,000 customers, they can expect to receive only 500 responses, based on the typical response rate. This scenario is what the random line in the lift chart represents.
- However, if the marketing department uses a mining model to target their mailing, they can expect a larger response rate because they can target those customers who are most likely to respond. If the model is perfect, it means that the model creates predictions that are never wrong, and the company could expect to receive 1,000 responses by mailing to the 1,000 potential customers recommended by the model.

Example (from [Microsoft])

- This scenario is what the ideal line in the lift chart represents.
- The reality is that the mining model most likely falls between these two extremes; between a random guess and a perfect prediction.
- Any improvement from the random guess is considered to be lift.”

Lift curve example



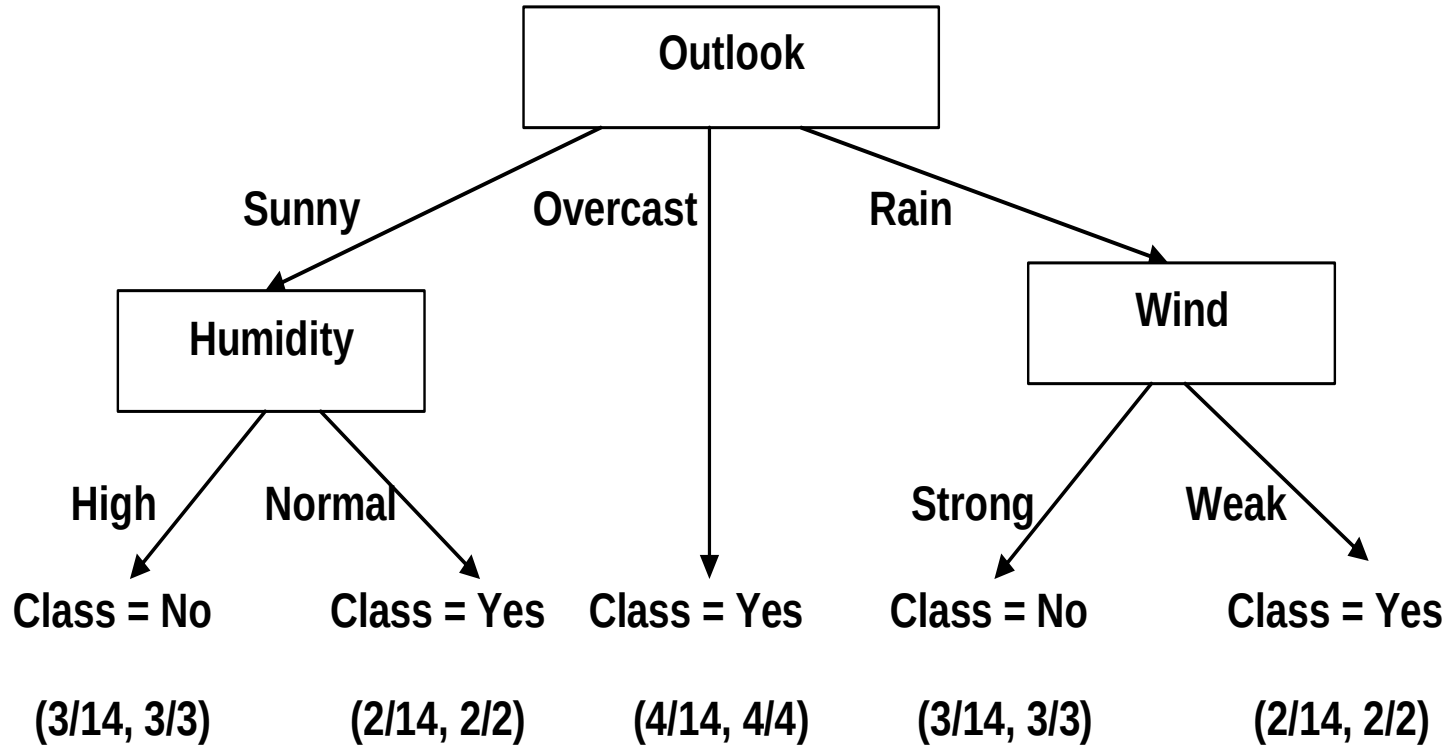
Road Map

- What is supervised learning
- Evaluation of classifiers
- Decision trees. ID3 and C4.5
- Rule induction systems
- Summary

What is a decision tree

- A very common way to represent a classification model or algorithm is a decision tree. Having a training set D and a set of example attributes A, each labeled example in D is like: ($a_1 = v_1$, $a_2 = v_2$, …, $a_n = v_n$). Based on these attributes a decision tree can be built having:
  - Internal nodes are attributes (with no path containing twice the same attribute).
  - Branches refer to discrete values (one or more) or intervals for these attributes. Sometimes more complex conditions may be used for branching.
  - Leafs are labeled with classes. For each leaf a support and a confidence may be computed: support is the proportion of examples matching the path from root to that leaf and confidence is the classification accuracy for examples matching that path. When passing from decision trees to rules, each rule has the same support and confidence as the leaf from where it comes.
  - Any example match a single path of the tree (so a single leaf = class).

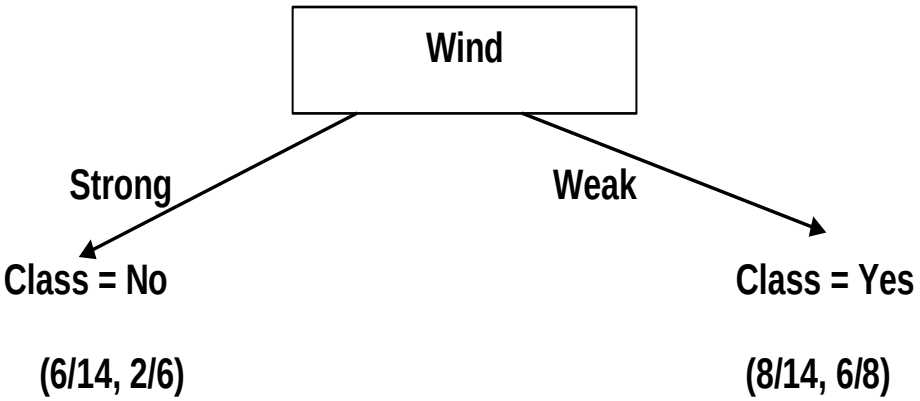
Example


Outlook
Sunny
Overcast
Rain
Humidity
Wind
High
Normal
Strong
Weak
Class = No
Class = Yes
Class = Yes
Class = No
Class = Yes
(3/14, 3/3)
(2/14, 2/2)
(4/14, 4/4)
(3/14, 3/3)
(2/14, 2/2)

Decision trees

- Numbers on the last line are the support and the confidence associated with each leaf.
- For the same data set more than one decision tree may be built.
- For example another Play tennis decision tree is in the next figure (with less confidence than previous tree):

Decision trees



ID3

- ID3 stands for Iterative Dichotomiser 3 and is an algorithm for building decision trees introduced by Ross Quinlan in 1986 (see [Quinlan 86]).
- The algorithm constructs the decision tree in a top-down manner choosing at each node the ‘best’ attribute for branching:
- First a root attribute is chosen, building a separate branch for each different value of the attribute.

ID3

- The training set is also divided, each branch inheriting the examples matching the attribute value of the branch.
- Process repeats for each descendant until all examples have the same class (in that case the node becomes a leaf labeled with that class) or all attributes have been used (the node also become a leaf labeled with the mode value – the majority class).
- An attribute cannot be chosen twice on the same path; from the moment it was chosen for a node it will never be tested again for the descendants of that node.

Best attribute

- The essence of the ID3 is how the ‘best’ attribute is discovered. The algorithm uses information theory trying to increase the purity of the datasets from the father node to the descendants.
- Let us consider a dataset $D = \{e_1, e_2, \ldots, e_m\}$ with examples labeled with classes from $C = \{c_1, c_2, \ldots, c_n\}$. Examples attributes are $A_1, A_2, \ldots, A_p$. The entropy of D can be computed as:

Entropy

$$entropy(D) = -\sum_{i=1}^{n} \Pr(c_i) \log_2 \Pr(c_i)$$

- If attribute A_k having r distinct values is considered for branching, it will partition D in r disjoint subsets D_1, D_2, …, D_r.
- The combined entropy of these subsets, computed as a weighted average of these entropies is:

$$entropy(D, A_k) = \sum_{i=1}^{r} \frac{count(D_i)}{count(D)} * entropy(D_i)$$

- All probabilities involved in the above equations are determined by counting!

Information gain

- Because the purity of the datasets is increasing, *entropy(D)* is bigger than *entropy(D, A_k)*. The difference between them is called the ***information gain***:

$$gain(D, A_k) = entropy(D) - entropy(D, A_k)$$

- ***The 'best' attribute is determined by the highest gain.***

Example

- For Play tennis dataset there are four attributes for the root of the decision tree: Outlook, Temperature, Humidity and Wind.
- The entropy of the whole dataset and the weighted values for dividing using the four attributes are:

$$entropy(D) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$$

For each attribute

$$entropy(D, Humidity) = \frac{7}{14}\left(-\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7}\right) + \frac{7}{14}\left(-\frac{6}{7}\log_2\frac{6}{7} - \frac{6}{7}\log_2\frac{6}{7}\right) = 0.79$$

- In the same way:

$$entropy(D, Wind) = 0.89$$

$$entropy(D, Temperature) = 0.91$$

$$entropy(D, Outlook) = 0.69$$

Best attribute: Outlook

- The next table contains the values for entropy and gain.
- The best attribute for the root node is Outlook, with a maximum gain of 0.25:

Attribute	entropy	gain
Humidity	0.79	0.15
Wind	0.89	0.05
Temperature	0.91	0.03
Outlook	0.69	0.25

Notes on and extensions of ID3

1. Because is a greedy algorithm it leads to a local optimum.
2. Attributes with many values leads to a higher gain. For solving this problem the gain may be replaced with the ***gain ratio***:

$$gain\text{–}ratio(D, A_k) = \frac{gain(D, A_k)}{entropy(D, A_k)}$$

Notes on and extensions of ID3

3. Sometimes (when only few examples are associated with leaves) the tree overfits the training data and does not work well on test examples.

- To avoid overfitting the tree may be simplified by pruning:
  - Pre-pruning: growing is stopped before normal end. The leaves are not 100% pure and are labeled with the majority class (the mode value).
  - Post-pruning: after running the algorithm some sub-trees are replaced by leaves. Also in this case the labels are mode values for the matching training examples. Port-pruning is better because in pre-pruning is hard to estimate when to stop.

Notes on and extensions of ID3

4. Some attribute A may be continuous. Values for A may be partitioned in two intervals:

$$A \leq t \text{ and } A > t.$$

The value of t may be selected as follows:

- Sort examples upon A
- Pick the average of two consecutive values where the class changes as candidate.
- For each candidates found in previous step compute the gain if partitioning is made using that value. The candidate with the maximum gain is considered for partitioning.

Notes on and extensions of ID3

- In this way the continuous attribute is replaced with a discrete one (two values, one for each interval).
- This attribute competes with the remaining attributes for ‘best’ attribute.
- The process repeats for each node because the partitioning value may change from a node to another.

Notes on and extensions of ID3

5. Attribute cost: some attributes are more expensive than others (measured not only in money).

- It is better that lower-cost attributes to be closer to the root than other attributes.
- For example, for an emergency unit it is better to test the pulse and temperature first and only when necessary perform a biopsy.
- This may be done by weighting the gain by the cost:

A.

$$weighted\text{–}gain(D, A_k) = \frac{gain(D, A_k)}{cost(A_k)}$$

C4.5

- C4.5 is the improved version of ID3, and was developed also by Ross Quinlan (as well as C5.0). Some characteristics:
  - Numeric (continuous) attributes are allowed
  - deal sensibly with missing values
  - post-pruning to deal with for noisy data
- The most important improvements from ID3 are:
  1. The attributes are chosen based on gain-ratio and not simply gain.

C4.5

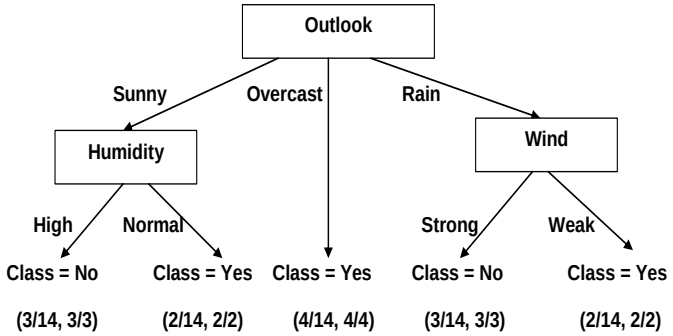
- The most important improvements from ID3 are:
  2. Post pruning is performed in order to reduce the tree size. The pruning is made only if it reduces the estimated error. There are two prune methods:
     - Sub-tree replacement: A sub-tree is replaced with a leaf but each sub-tree is considered only after all its sub-trees. This is a bottom-up approach.
     - Sub-tree raising: A node is raised and replaces a higher node. But in this case some examples must be reassigned. This method is considered less important and slower than the first.

Road Map

- What is supervised learning
- Evaluation of classifiers
- Decision trees. ID3 and C4.5
- Rule induction systems
- Summary

Rules

- Rules can easily be extracted from a decision tree: each path from the root to a leaf corresponds to a rule.
- From the decision tree in example 2 five IF THE rules can be extracted:



Rules

The rules are (one for each path):

1. IF Outlook = Sunny AND Humidity = High
   THEN Play Tennis = No;
2. IF Outlook = Sunny AND Humidity = Normal
   THEN Play Tennis = Yes;
3. IF Outlook = Overcast
   THEN Play Tennis = Yes;
4. IF Outlook = Rain AND Wind = Strong
   THEN Play Tennis = No;
5. IF Outlook = Rain AND Wind = Weak
   THEN Play Tennis = Yes;

Rule induction

- In the case of a set of rules extracted from a decision tree, rules are mutually exclusive and exhaustive.
- But rules may be obtained directly from the training data set by ***sequential covering***.
- A classifier built by sequential covering consists in an ordered or unordered list of rules (called also decision list), obtained as follows:
  - Rules are learned one at a time
  - After a rule is learned, the tuples covered by that rule are removed
  - The process repeats on the remaining tuples until some stopping criteria are met (no more training examples, the quality of a rule returned is below a user-specified threshold, …)

Fining rules

- There are many algorithms for rule induction: FOIL, AQ, CN2, RIPPER, etc.
- There are two approaches in sequential covering:
  1. Finding **ordered rules**, by first determining the conditions and then the class.
  2. Finding a set of **unordered rules** by first determining the class and then the associated condition.

Ordered rules

- The algorithm:

RuleList ← ∅
Rule ← learn-one-rule(*D*)
while *Rule* ≠ ∅ AND *D* ≠ ∅ **do**
    RuleList ← *RuleList* + *Rule* // append Rule at the end of RuleList
    D = *D* – {examples covered by *Rule*}
    Rule ← learn-one-rule(*D*)
Endwhile
// append majority class as last/default rule:
RuleList ← RuleList + {c|c is the majority class}
return *RuleList*

Learn-one-rule

- Function learn-one-rule is built considering all possible attribute-value pairs (Attribute op Value) where Value may be also an interval.
- The process tries to find the left side of a new rule which is a condition.
- At the end the rule is constructed using as right side the majority class of the examples covered by the left side condition:

Learn-one-rule

1. Start with an empty *Rule* and a set of *BestRules* containing this rule:

   $Rule \leftarrow \emptyset$

   $BestRules \leftarrow \{Rule\}$

2. For each member **b** of *BestRules* and for each possible attribute-value pair **p** ***evaluate*** the combined condition $\mathbf{b} \cup \mathbf{p}$. If this condition is better than *Rule* then it replaces the old value of *Rule*.
3. At the end of the process a best rule with an incremented dimension is found. Also in *BestRules* the best n combined conditions discovered at this step are kept (implementing a beam search).

Learn-one-rule

4. Repeat steps 2 and 3 until no more conditions in *BestRules*. Note that a condition must verify a given threshold at evaluation time, so *BestRules* may have less then n members.
5. If *Rule* is evaluated and found enough efficient (considering the given threshold) then *Rule* → c is returned otherwise an empty rule is the result. The class c is the majority class of the examples covered by *Rule*.

- The evaluation of a rule may be done using the entropy of the set containing examples covered by that rule.

Unordered rules

- The algorithm:

```
RuleList ← ∅
foreach class c ∈ C do
        D = Pos ∪ Neg          // Pos = { examples of class c from D}
                               // Neg = D - Pos
        while Pos ≠ ∅ do
          Rule ← learn-one-rule(Pos, Neg, c);
          if Rule = ∅
                then
                  Quitloop
                else
                  RuleList ← RuleList + Rule //  append Rule at the end of RuleList
                  Pos = Pos  – {examples covered by Rule}
                  Neg  = Neg  – {examples covered by Rule}
          endif
        endwhile
endfor
return RuleList
```

learn-one-rule again

- For learning a rule two steps are performed: grow a new rule and then prune it.
- Pos and Neg are split in two parts each: GrowPos, GrowNeg, PrunePos and PruneNeg.
- The first part is used for growing a new rule and the second for pruning.
- At the ***'grow' step*** a new condition/rule is build, as in the previous algorithm.
- Only the best condition is kept at each step (and not best n).
- Evaluation for the new best condition C' obtained by adding an attribute-value pair to C is made using a different gain:

learn-one-rule again

$$gain(C, C') = p_1 * \left(\log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0} \right)$$

where:

- p_0, n_0: the number of positive/negative examples covered by C in GrowPos/ GrowNeg.
- p_1, n_1: the number of positive/negative examples covered by C' in GrowPos/ GrowNeg.

- The rule maximizing this gain is returned by the 'grow' step.

learn-one-rule again

- At the ‘prune’ step sub-conditions are deleted from the rule and the deletion that maximize the function below is chosen:

$$v(Rule, PrunePos, PruneNeg) = \frac{p - n}{p + n}$$

- Where p, n are the examples number in PrunePos/PruneNeg covered by the rule after sub-condition deletion.
- Next slide: another example of building a all rules for a given class: the IREP algorithm (Incremental Reduced Error Pruning) in [Cohen 95]

IREP

```
procedure IREP(Pos, Neg)
begin
 Ruleset := ∅
 while Pos ≠ ∅ do
    // grow and prune a new rule
    split (Pos, Neg) into (GrowPos, GrowNeg) and (PrunePos, PruneNeg)
    Rule := GrowRule(GrowPos, GrowNeg)
    Rule := PruneRule(Rule, PrunePos, PruneNeg)
    if the error rate of Rule on (PrunePos, PruneNeg) exceeds 50%
          then return Ruleset
          else add Rule to Ruleset
                remove examples covered by Rule from (Pos, Neg)
    endif
 endwhile
 return Ruleset
end
```

Summary

This course presented:

- What is supervised learning: definitions, data formats and approaches.
- Evaluation of classifiers: accuracy and other error measures and evaluation methods: holdout set, cross validation, bootstrap and scoring and ranking.
- Decision trees building and two algorithms developed by Ross Quinlan (ID3 and C4.5) .
- Rule induction systems

- Next week: Supervised learning – part 2.

References

- [Liu 11] Bing Liu, 2011. Web Data Mining, Exploring Hyperlinks, Contents, and Usage Data, Second Edition, Springer, chapter 3.
- [Han, Kamber 06] Jiawei Han and Micheline Kamber, Data Mining: Concepts and Techniques, 2nd ed., The Morgan Kaufmann Series in Data Management Systems, Jim Gray, Series Editor Morgan Kaufmann Publishers, March 2006. ISBN 1-55860-901-6
- [Sanderson 08] Robert Sanderson, Data mining course notes, Dept. of Computer Science, University of Liverpool 2008, Classification: Evaluation http://www.csc.liv.ac.uk/~azaroth/courses/current/comp527/lectures/comp527-13.pdf
- [Quinlan 86] Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn. 1, 1 (Mar. 1986), 81-106, http://www.cs.nyu.edu/~roweis/csc2515-2006/readings/quinlan.pdf
- [Wikipedia] Wikipedia, the free encyclopedia, en.wikipedia.org
- [Microsoft] Lift Chart (Analysis Services - Data Mining), http://msdn.microsoft.com/en-us/library/ms175428.aspx
- [Cohen 95] William W. Cohen, Fast Effective Rule Induction, in “Machine Learning: Proceedings of the Twelfth International Conference” (ML95), http://sci2s.ugr.es/keel/pdf/algorithm/congreso/ml-95-ripper.pdf