PA196: Pattern Recognition

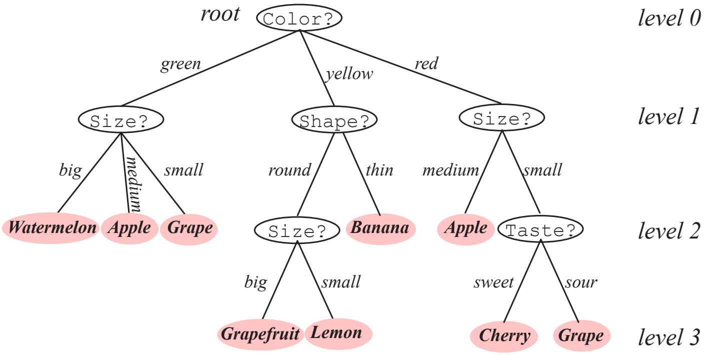
07. Decision trees
08. Multiple classifier systems

Dr. Vlad Popovici
popovici@recetox.muni.cz

RECETOX
Masaryk University, Brno

Outline

Outline


root
Color?
level 0
green
yellow
red
Size?
Shape?
Size?
level 1
big
medium
small
round
thin
medium
small
Watermelon
Apple
Grape
Size?
Banana
Apple
Taste?
level 2
big
small
sweet
sour
Grapefruit
Lemon
Cherry
Grape
level 3

[DHS - Fig.8.1]

- attributes can be continuous or nominal/categorical
- there is no need to have a metric
- the interpretation is simple and can be written as a logical proposition
- natural handling of multi-class problems
- different equivalent trees...
- feature selection embedded into the algorithm
- what if there are tens of thousands of features?
- how many levels?

Outline

Classification And Regression Trees - CART

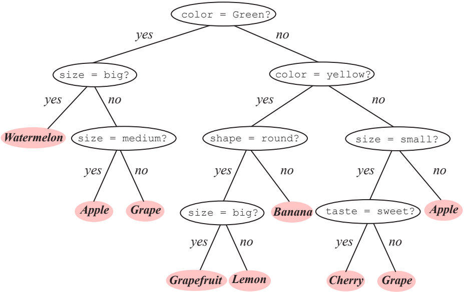
- we are given a training set $S = \{(\mathbf{x}_i, y_i) | i = 1, \ldots, n\}$ where y_i codes the class g_i and $\mathbf{x}_i$ are some *ordered collection of attributes*
- a tree splits the training set into subsets
- the objective is to "grow" a tree such that the leaves are *pure*: all elements in a subset belong to the same class

Issues:

- binary or multi-valued decision in the nodes? (i.e. how many splits?)
- which attribute should be tested?
- when should a node be declared a leaf?
- pruning strategies?
- if a leaf is impure, what's its label?
- how to handle missing values?

Number of splits

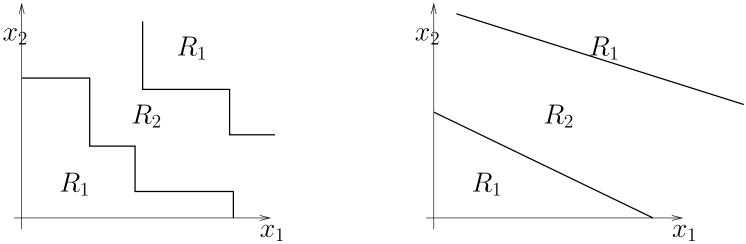
- design decision: what's the *branching factor B* of a node?
- $B = 2$: binary trees
- any tree can be transformed into a binary tree



[DHS - Fig.8.2]

Attribute (variable) selection

Decision boundaries: single variable binary decisions lead to boundaries that are (by portions) orthogonal to the axes. Oblique boundaries can only be approximated (by large trees).



- for a node N we search for that attribute T that would make the descendant nodes as pure as possible
- *impurity* $i(N)$: 0 if all elements belong to the same class, "large" if the classes are equally represented
- *entropy impurity*:

$$i(N) = -\sum_i P(g_i) \log_2 P(g_i)$$

- (for binary classification) *variance impurity*

$$i(N) = P(g_1)P(g_2)$$

- *Gini impurity* (generalized variance impurity):

$$i(N) = \sum_{i \neq j} P(g_i)P(g_j) = 1 - \sum_i [P(g_i)]^2$$

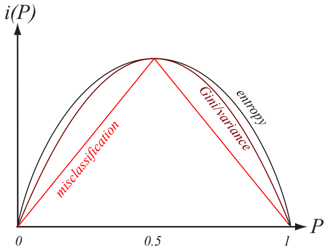
(interpretation: expected error rate if the label is randomly selected from the class distribution present at N)

- *misclassification impurity*

$$i(N) = 1 - \max_i P(g_i)$$

(interpretation: minimum probability of a misclassification)

Comparison of various impurity measures for the two-class case



[DHS-Fig 8.4]

How to choose the test at the node N?

- heuristic (greedy approach): choose the test that maximizes the decrease in impurity of the descendent nodes:

$$\Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R)$$

  where N_L and N_R are the two (left and right) descendent nodes and P_L is the fraction of examples that go to the left subtree
- one has to find the attribute (variable) T to test and the threshold value that would maximize $\Delta i(N)$

- in general, entropy or Gini impurity functions are preferred; but the choice makes little difference to the final quality of the classifier
- finding the optimal threshold may involve an optimization process for continuous variables
- for categorical variables, the optimal value is found by exhaustive search
- the optimum is local and may not be unique
- the misclassification impurity is not always decreasing
- there are algorithms that allow multiway splits

Stopping criteria

- if too early: not enough accuracy; if too late: overfitting
- use only a part of the data for growing the tree and the rest for estimating its error rate (either single split of training set or in cross-validation manner). Grow the tree as long as the error rate (on the validation set) decreases;
- or: grow the tree as long as the reduction in impurity is above a threshold;
- or: grow the tree as long as there are more than a certain number of elements in any leaf
- or: split until a minimum of

$$\alpha \text{size} + \sum_{\text{leaf nodes}} i(N)$$

is reached (kind of MDL)

Alternative approach:

- try to assess the *statistical significance* of the reduction in impurity
- different tests (e.g. χ^2) can be used
- you can also build an empirical distribution for Δi from the nodes already in the tree (after several nodes already there)
- etc etc

Tree pruning

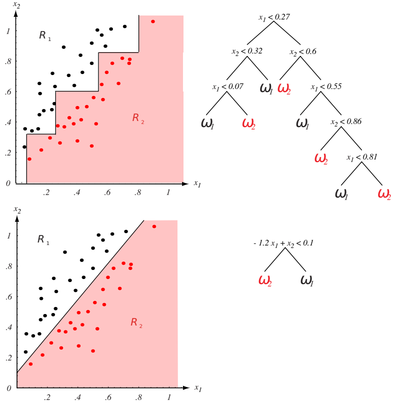
- *horizon effect*: the split decision at a node does not "see" the decisions in the descendent nodes
- *tree pruning* is the opposite strategy to early stopping
- a tree is grown to its fullest, and then leaves or even nodes are joined
- these action try to optimize a global cost function
- the approach is much more computationally expensive than early stopping
- alternative: use propositional logic to simplify the rules expressed by the tree: remove irrelevant rules and try to improve classification performance on a validation set

Label assignment for the leaves

- if a leaf is pure - the label is clear
- if $i(N) > 0$, then the majority rule is used
- pure leaves is not the most important criterion: it may indicate overfitting or over-sensitivity to small changes in training data (noise)

Other issues

- an approximation for the training complexity $\mathcal{O}(dn^2 \log n)$
- multivariable decisions



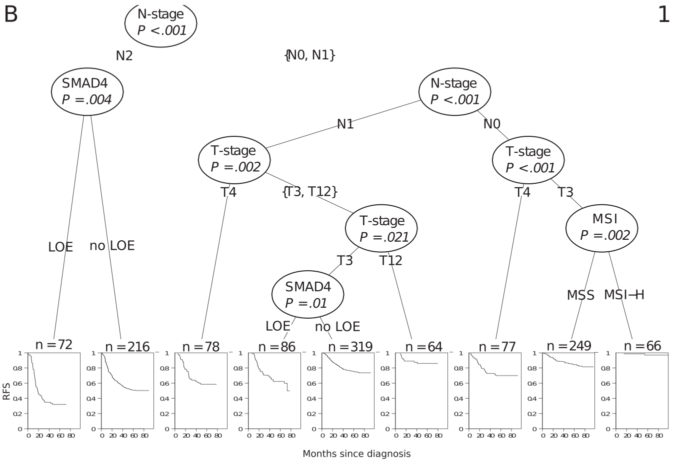
[DHS - Fig.8.5]

Outline

Other classical tree methods:

- *ID3* - interactive dichotomizer - it is intended for nominal attributes
  - the real values are quantized and used as nominal
  - the branching factor is usually > 2
- *C4.5* - successor and refinement of ID3
  - combines techniques from CART and ID3
  - real values are treated as by CART
  - nominal values generate multiple splits like in ID3
  - special method for pruning the rules

A slightly different tree...



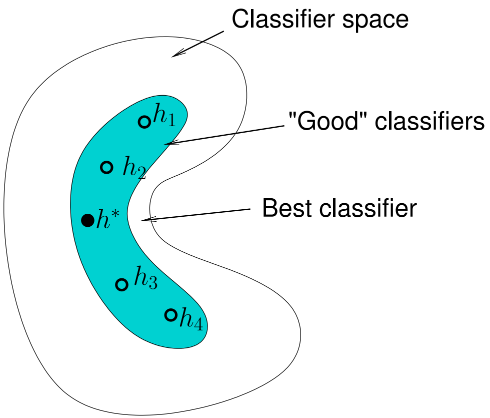
Outline

Outline

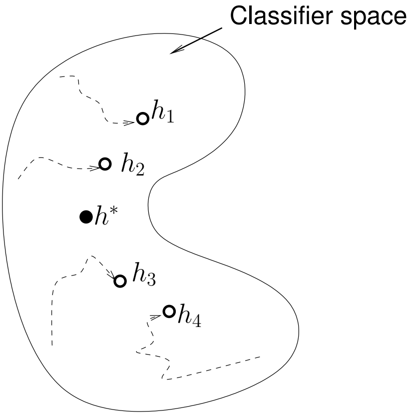
Why combining classifiers?

- obviously, in the hope of improving the overall accuracy
- instead of looking for the "best" classifier, we are looking for how "best" to combine some "reasonable" classifiers
- but, there are some other reasons: statistical, computational and representational

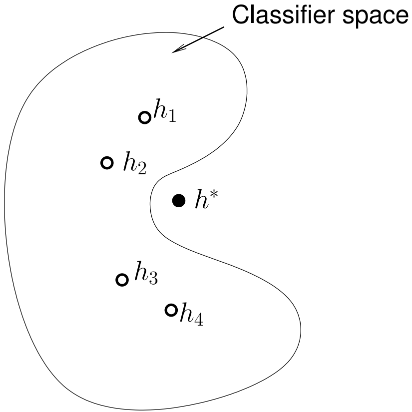
Statistical perspective: aggregating several "estimates" (classifiers) may be closer to the best classifier for the problems at hand:



Computational perspective: the various classifiers may represent only local optima from the classifier space hence, their combination may give a better approximation of the global optimum.

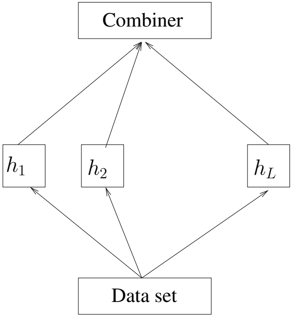


Representational perspective: maybe the space of classifiers chosen when modeling the problem does not contain the best classifier.



Different levels of combining classifiers

- data level: different subsets of the training set are used in training the *base classifiers*
- feature level: different subsets of features are used for base classifiers
- classifier level: use different base classifiers
- combiner level: use various combiners
- others: ECOC - error correcting codes: change the labels of the examples...



Classifier fusion vs selection

- c. fusion: each base learner has knowledge of the whole feature space
- c. selection: the base learners have different domains of compentencies (set of features)
- fusion: combiners based on majority vote or weighted means, etc.
- cascades of classifiers: a special case of c. selection

Decision optimization vs coverage optimization

- decision optimization: optimize the combiner for a fixed set of base learners
- coverage optimization: fix the combiner and find the best set of base learners

Trainable vs non-trainable ensembles

- non-trainable combiners: e.g. majority vote
- trainable combiners: may take into account, for example, the reliability of the base learners
- or build the combiner as the ensemble is developed (e.g. AdaBoost)

Outline

We consider a set of classifiers $h_1, \ldots, h_L : \mathbb{R}^d \to \mathcal{G}$. The goal is to construct a combiner (classifier)

$$H : \mathcal{G}^L \to \mathcal{G}$$

The space $\mathcal{G}^L$ is called *intermediate feature space*.

Types of classifier outputs:

- *type 0*: the only information about the output of classifier h_i is that it is correct or false. For a data set S the classifier h_i produces an output vector (one element for each point $\mathbf{x}_k \in S$): $[y_{ik}]$ such that

$$y_{ik} = \begin{cases} 1 & \text{if } h_i \text{ classifies correctly } \mathbf{x}_k \\ 0 & \text{otherwise} \end{cases}$$

- *type 1*: the classifier h_i produces a class label g_j for any input vector $\mathbf{x}$

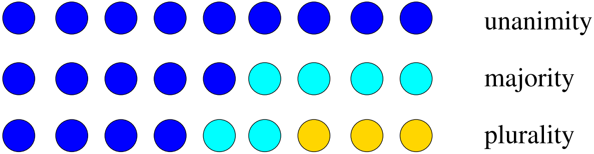
Types of classifier outputs (cont'd):

- *type 2*: each classifier produces an ordered list of possible class labels (a subset of $\mathcal{G}$), from most plausible to the least plausible
- *type 3*: each classifier outputs a vector $[f_1, \ldots, f_C] \in \mathbb{R}^C$ with values indicating the support for the hypothesis that $\mathbf{x}$ belongs to each of the $C = |\mathcal{G}|$ classes

Majority vote

- types: unanimity, majority and plurality
- let $[c_{i1}, \ldots, c_{iC}] \in \{0, 1\}^C$ be a vector associated with classifier h_i: c_{ik} is 1 if h_i assigns $\mathbf{x}$ to class g_k
- the *plurality vote* can be written as: assign $\mathbf{x}$ to g_k if

$$\sum_{i=1}^{L} c_k = \max_{j=1,\ldots,C} \sum_{i=1}^{L} c_{ij}$$



- depending on the *patterns of success and failures* of individual classifiers, the majority vote can improve significantly the overall performance...
- ...as it can decrease it with respect to the performance of the best base classifier

Weighted majority vote

- idea: give more weight to the better base classifiers
- the discriminant function for class g_i has the form

$$H_i(\mathbf{x}) = \sum_{j=1}^{L} b_j h_i(\mathbf{x})$$

- if the L base classifiers are independent with individual accuracies $p_1, \ldots, p_L$, then the accuracy of the ensemble is maximized if the weights are chosen as

$$b_i \propto \ln \frac{p_i}{1 - p_i}$$

Other methods for combining labels

- Naive Bayes: assumes conditional independence of the classifiers and tries to produce an estimate of the posterior probability based on the probabilities of assignment from each individual classifier
- multinomial methods try to estimate the posterior probability for each possible combination of labels produced by the base classifiers
- combination based on SVD uses correspondence analysis in the intermediate feature space
- etc etc

Outline

Fusion of continuous outputs

- the output of the base classifier is interpreted as a degree of confidence in the class assignment: either a confidence or a posterior probability
- the combiner tries to estimate the support (evidence) for each class
- let *DP* be the *decision profile matrix*:

$$DP(\mathbf{x}) = \begin{bmatrix} h_{11}(\mathbf{x}) & \dots & h_{1j}(\mathbf{x}) & \dots & h_{1C}(\mathbf{x}) \\ & & \ddots & & \\ h_{i1}(\mathbf{x}) & \dots & h_{ij}(\mathbf{x}) & \dots & h_{iC}(\mathbf{x}) \\ & & \dots & & \end{bmatrix}$$

with the i–th row corresponding to h_i output and the j–th column showing the evidence from all classifiers in favor of class g_j.

Class-conscious combiners:

- non-trainable (i.e. there are no parameters to optimize) compute the support μ_j for class g_j as:
  - average over $DP(\mathbf{x})_{\cdot,j}$ (j–th column)
  - minimum/maximum/median over $DP(\mathbf{x})_{\cdot,j}$
  - trimmed mean, product, some other mean, over $DP(\mathbf{x})_{\cdot,j}$
- trainable:
  - various weighted means
  - fuzzy integral: measures also the "strength" of all subsets of classifiers

Class-indifferent combiners:

- *decision templates*: build a "typical" (template) decision profile for each class and then compare the current decision profile with the template
- the comparison can use Euclidean, Minkowski, city-block etc metrics
- you can try a k–NN in the intermediate feature space

Outline

Classifier selection

- idea: build "expert" classifiers for some subdomains (subspace of the original space) and find a way to identify which base classifier should take the decision for each new input $\mathbf{x}$
- there are either dynamic or static estimation of regions of competence for base classifiers
- different ways of combining their outputs: fusion or selection
- stochastic selection: select the label for $\mathbf{x}$ by sampling from from $\{h_1, \ldots, h_L\}$ according to some distribution $p_1(\mathbf{x}), \ldots, p_L(\mathbf{x})$
- or, choose the classifier with highest $p_i(\mathbf{x})$
- or, weighted average...