Introduction to Machine Learning
NPFL 054

http://ufal.mff.cuni.cz/course/npfl054

Barbora Hladká
hladka@ufal.mff.cuni.cz

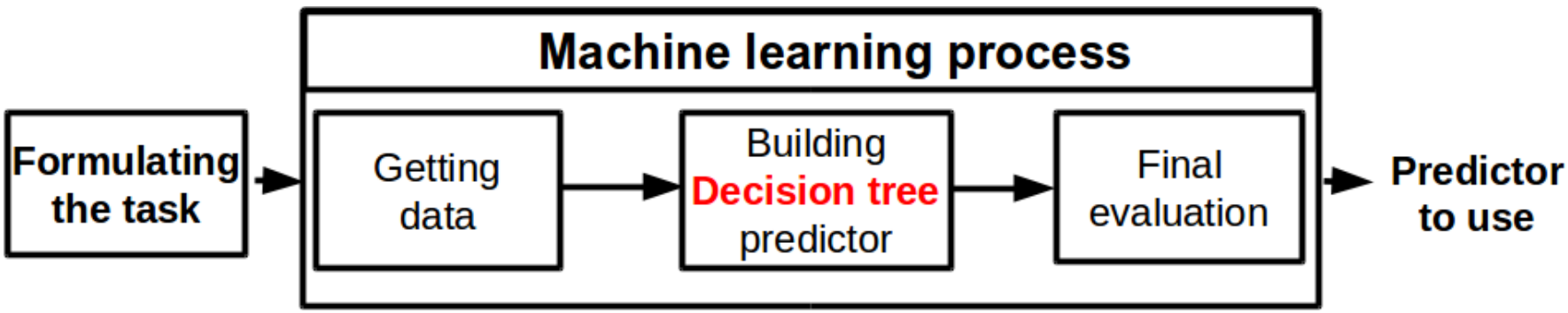
Martin Holub
holub@ufal.mff.cuni.cz

Charles University,
Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics

Lecture #3, Part II
The idea of Decision Trees and Random Forests

Lecture #3, Part II
The idea of Decision Trees and Random Forests

Decision Tree is a learning method suitable for both classification and regression tasks

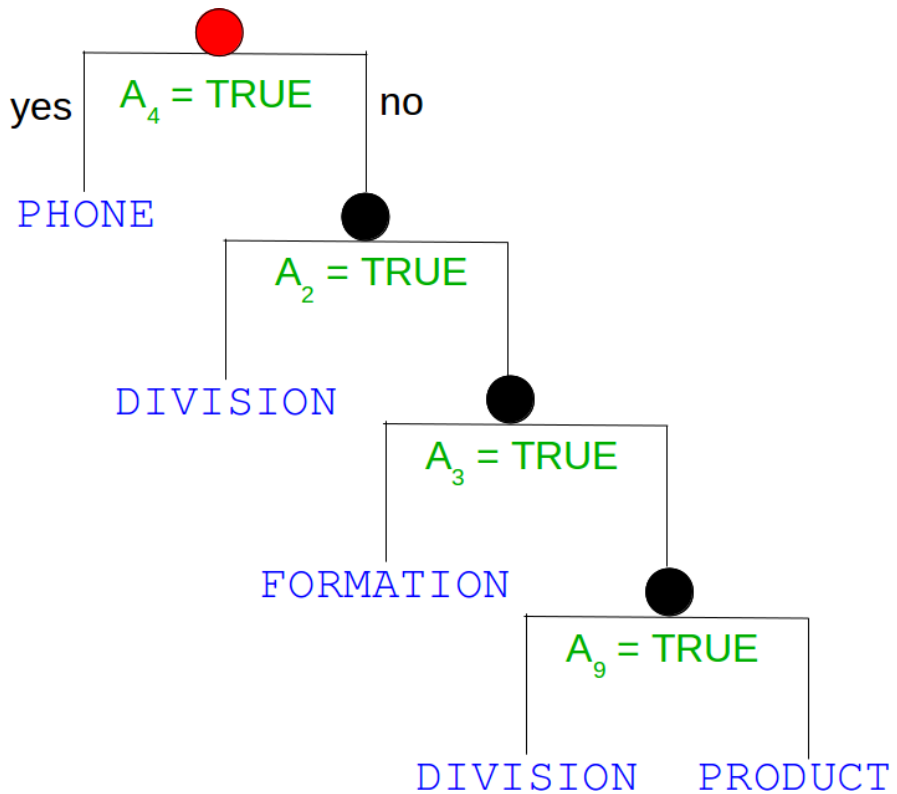


Example classification task: WSD
see the NPFL054 web page → Materials → wsd-attributes.pdf

Decision tree structure

A **decision tree** $T = (V, E)$ is a rooted tree where V is composed of internal **decision nodes** and terminal **leaf nodes**.

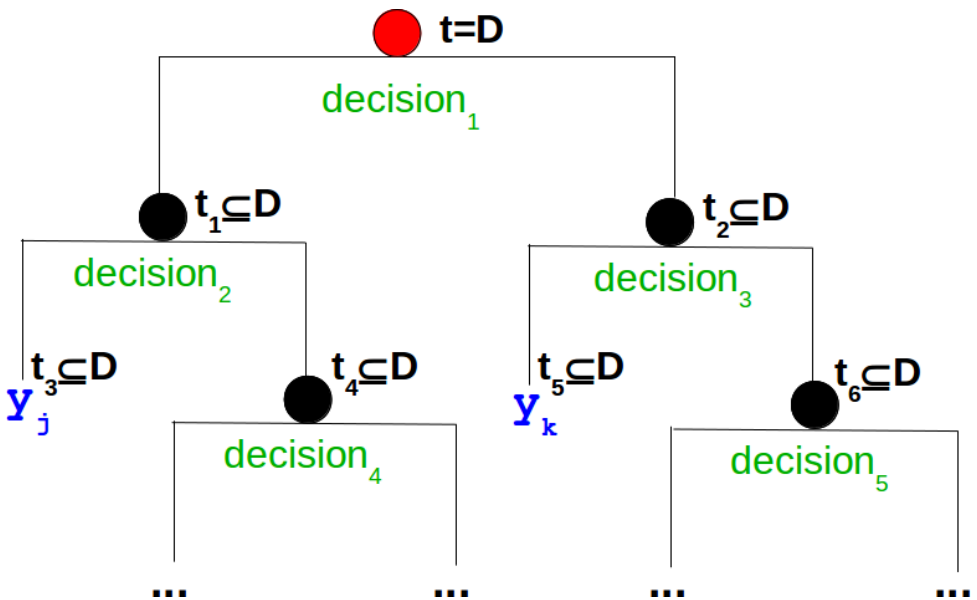
- Nodes
  - Root node
  - Internal nodes with conditions on selected features
  - Leaf nodes with `TARGET OUTPUT VALUES`
- Decisions



Decision trees — learning from training data

Decision tree learning

- Building a decision tree $T_D = (V, E)$ is based on a training data set $D = \{\langle \mathbf{x}, y \rangle : \mathbf{x} \in X, y \in Y\}$.
- Each node is associated with a set t, $t \subseteq D$. The root node is associated with $t = D$.
- Each leaf node is asociated with a fixed output value.

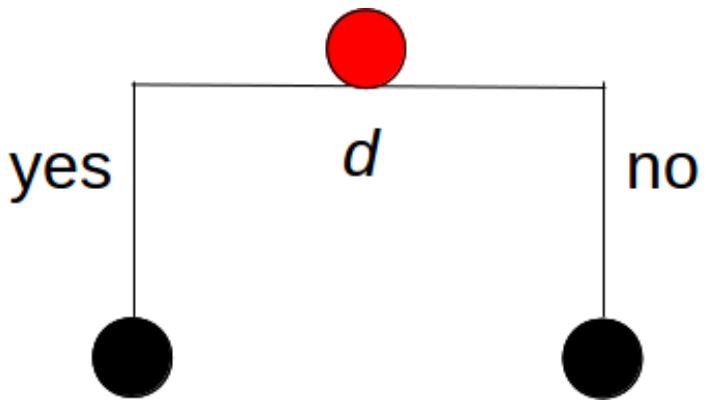


Building a decision tree from training data

A very basic idea: Assume binary decisions.

- **Step 1** Create a root node.

- **Step 2** Select decision d and add child nodes to an existing node.



Step 2 is then applied recursively.

Building a decision tree from training data

The learning process, i.e. building the tree starts from the root node and continues top-down. The root node is associated with the whole training set.

Example

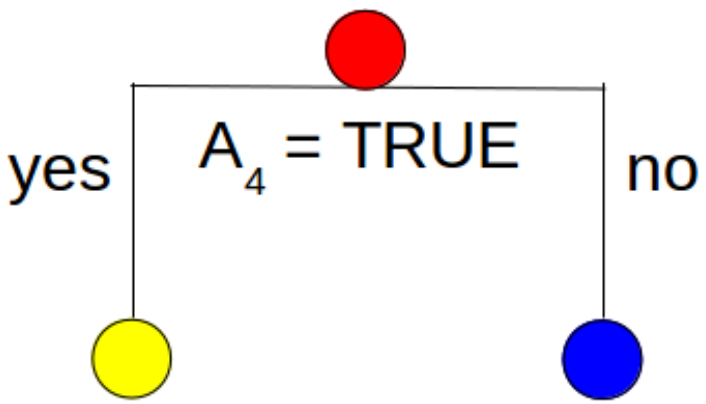
1. Assume decision
   `if` $\mathrm{A}_4 = \mathit{TRUE}$.
2. Split the training set t according to this decision into two subsets – "yellow" and "blue".

t

SENSE	...	A4	...
FORMATION		TRUE	
FORMATION		FALSE	
PHONE		TRUE	
CORD		TRUE	
DIVISION		FALSE	
...	...	...	

Building a decision tree from training data

3. Add two child nodes, "yellow" and "blue", to the root. Associate each of them with the corresponding subsets t_L and t_R, respectively. The subsets are always disjoint.



t_L

SENSE	...	A4	...
FORMATION		TRUE	
CORD		TRUE	
PHONE		TRUE	
...	...	...	

t_R

SENSE	...	A4	...
FORMATION		FALSE	
DIVISION		FALSE	
...	...	...	

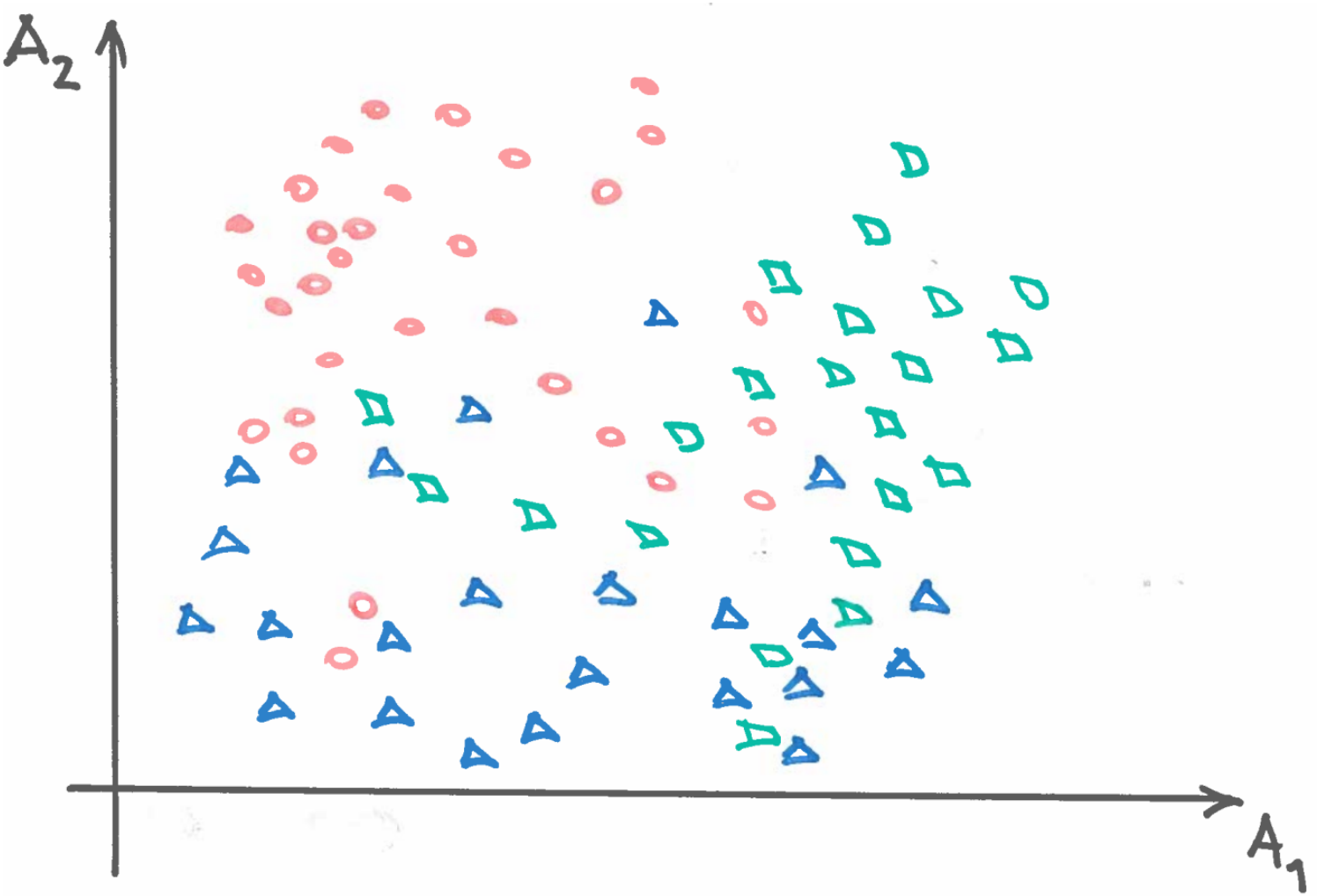
Building a decision tree from training data

- **Step 4** Repeat recursively steps (2) and (3) for both child nodes and their associated training subsets.

- **Step 5** Stop recursion for a node if a stopping criterion is fulfilled. Then create a leaf node with an output value.

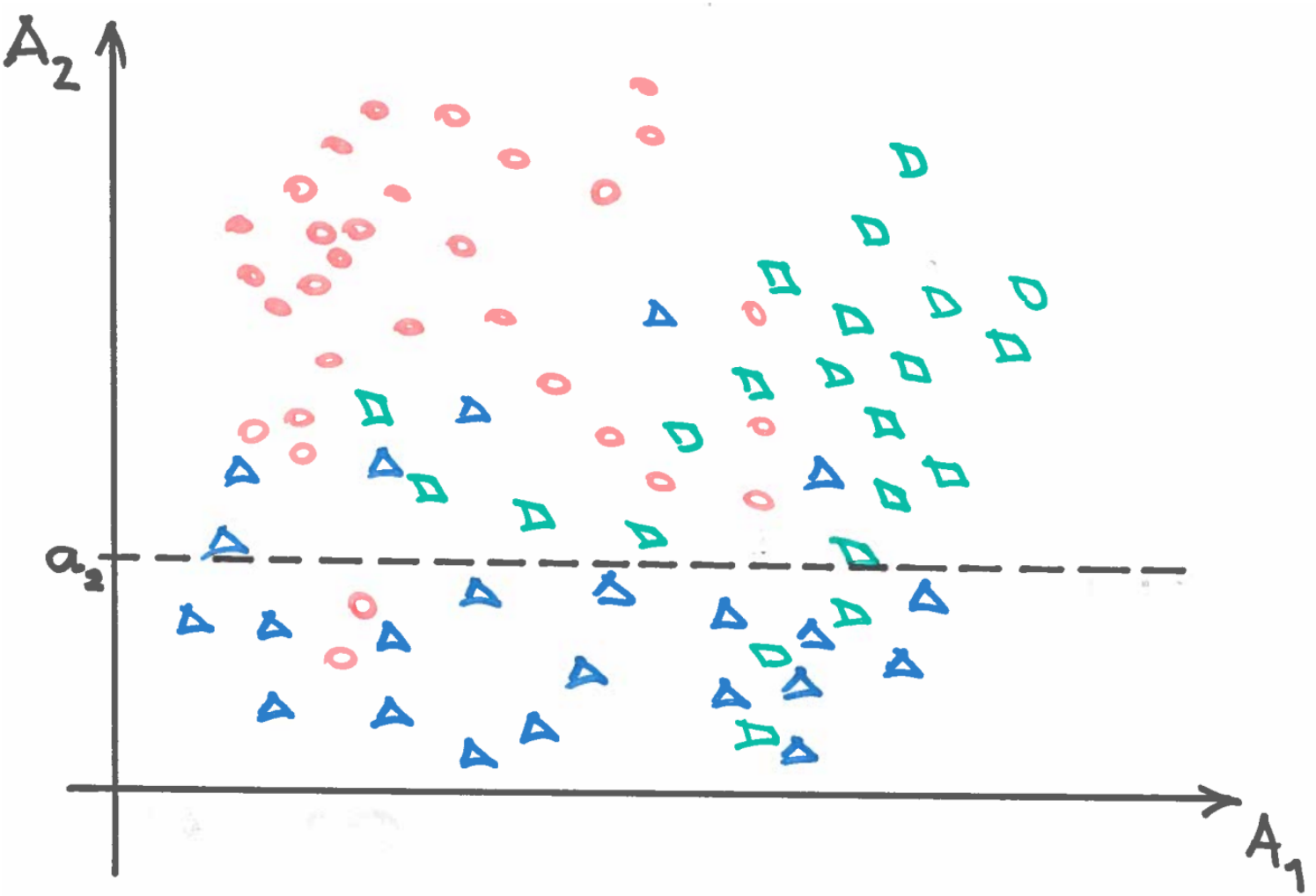
Learning decision tree – example training data

Two continuous features A_1 and A_2, and three target classes



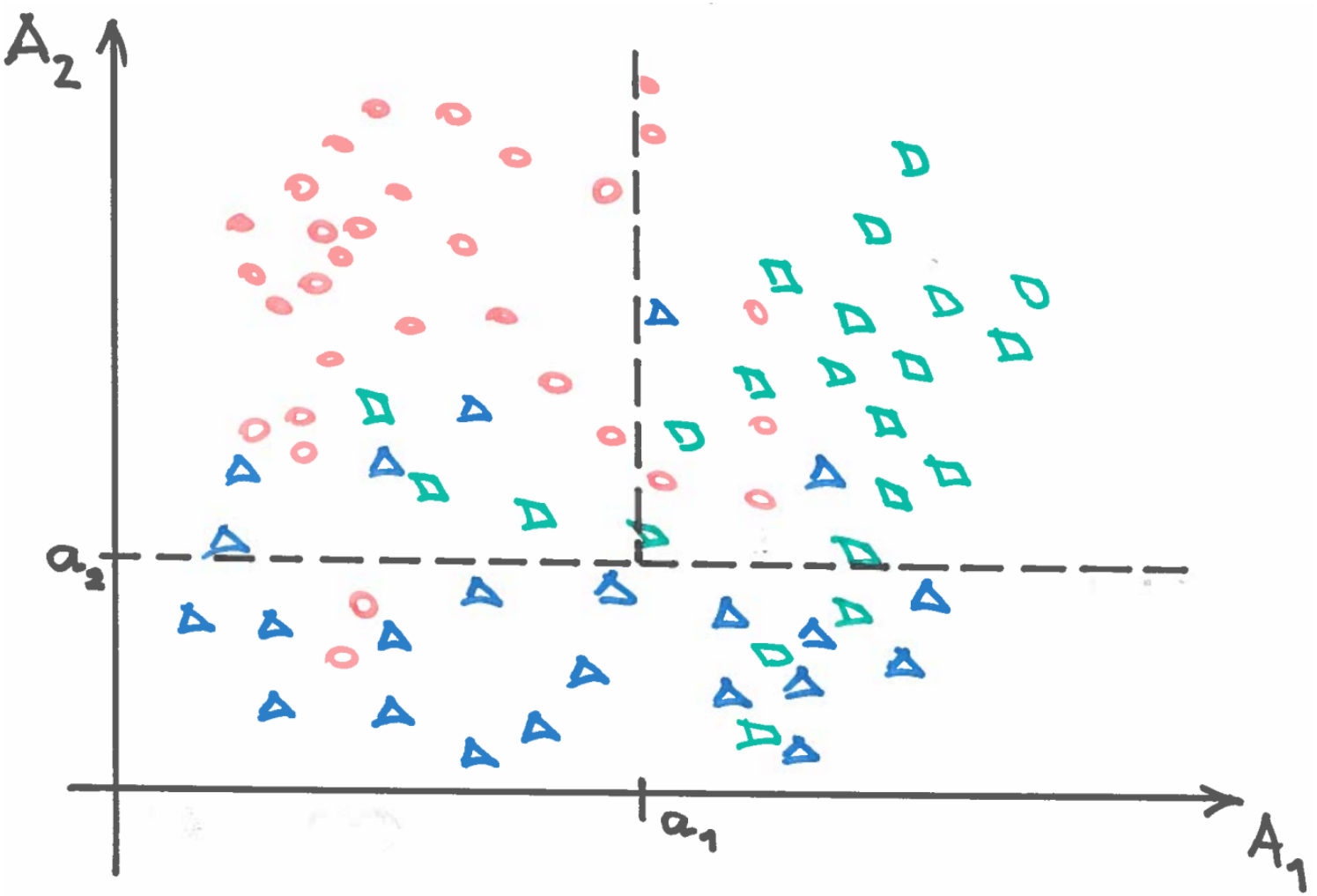
Learning decision tree – example first split

First split divides the training data set into two partitions by condition $A_2 \geq a_2$



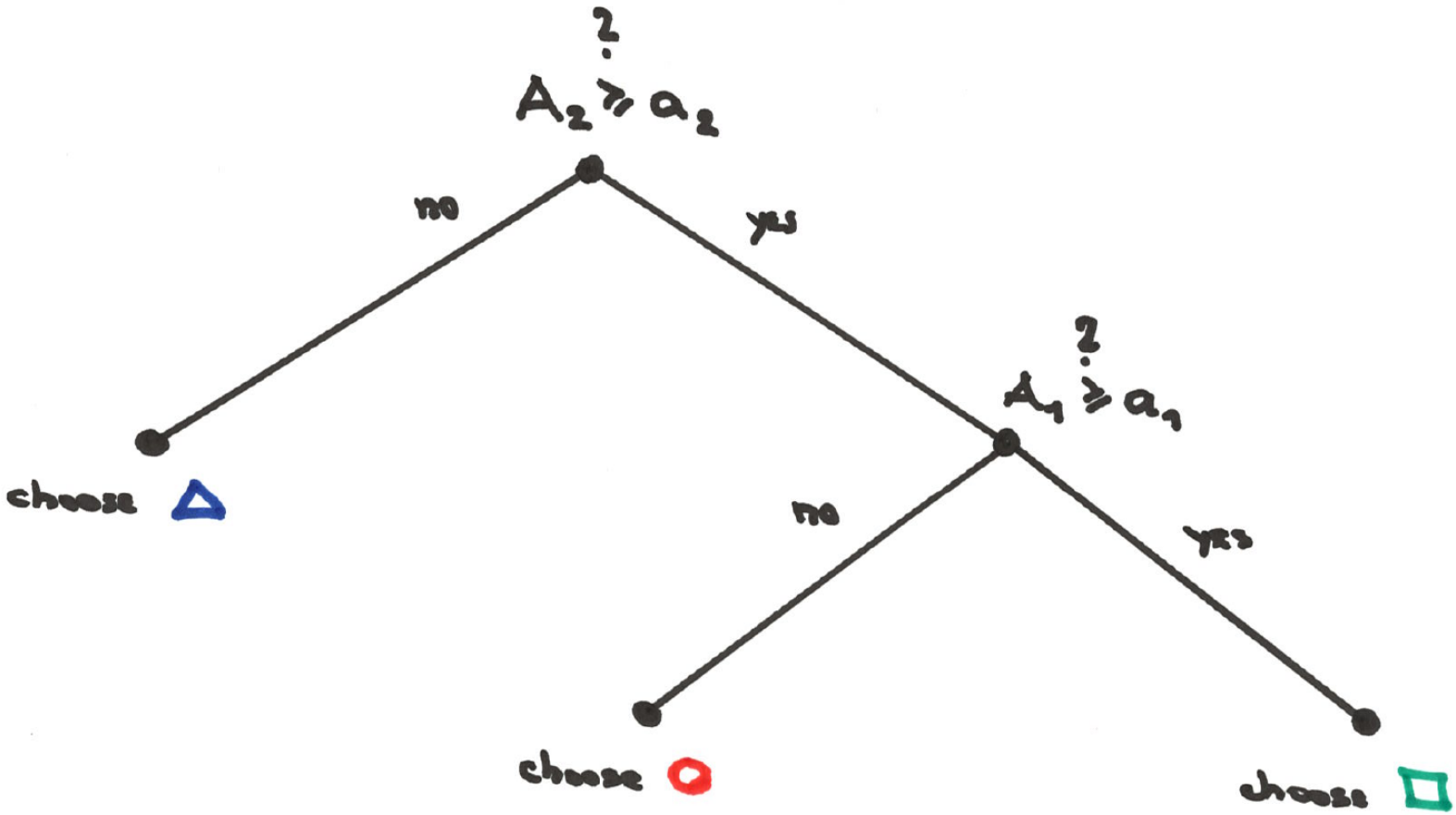
Learning decision tree – example second split

Second split is defined by $A_1 \geq a_1$ and applies only if $A_2 \geq a_2$



Learning decision tree – example resulting tree

Two splits in the example produce a tree with two inner nodes and three leaves



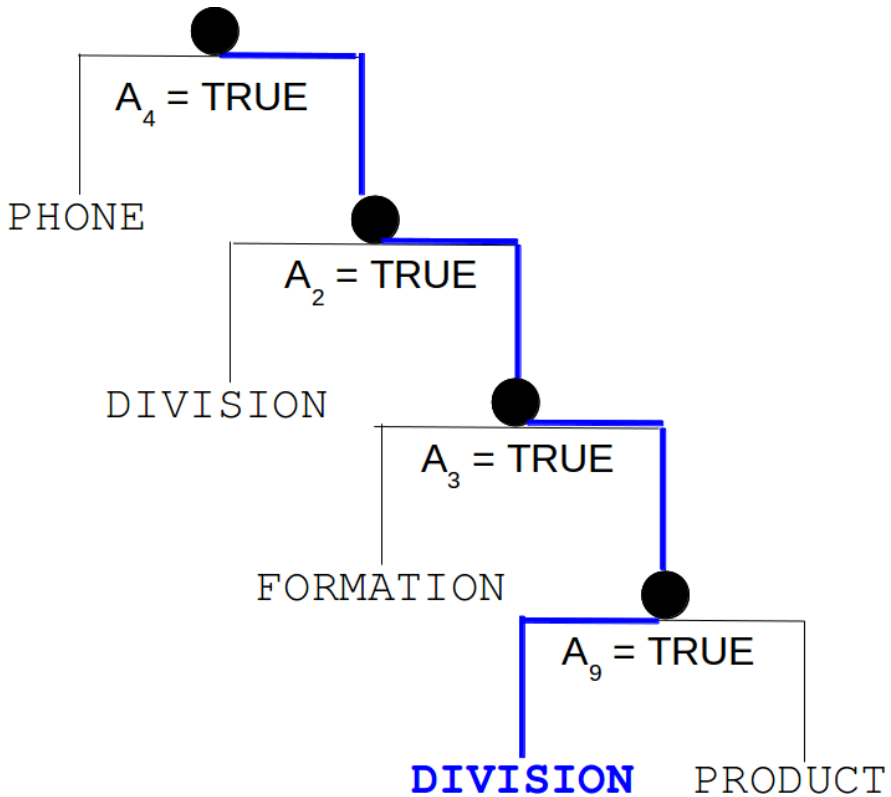
Prediction on test data

Once a decision tree predictor is built, an unseen instance is predicted by starting at the root node and moving down the tree branch corresponding to the feature values asked in decisions.

Prediction on test data – example

Decision tree predictor for the WSD-*line* task

According to existing feature values in a given test instance you can use the decision tree as a predictor to get the classification of the instance.



Decision trees for classification and for regression

Decision trees can be used both for classification and regression tasks

Classification trees

- Categorical output value

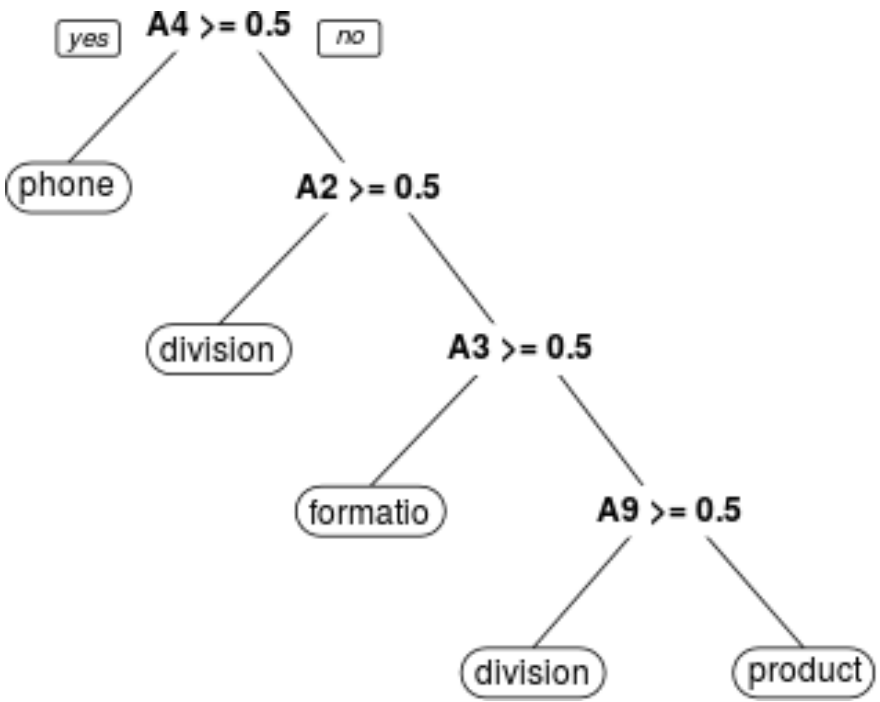


Figure: Tree for predicting the sense of *line* based on binary features.

Regression trees

- Numerical output value

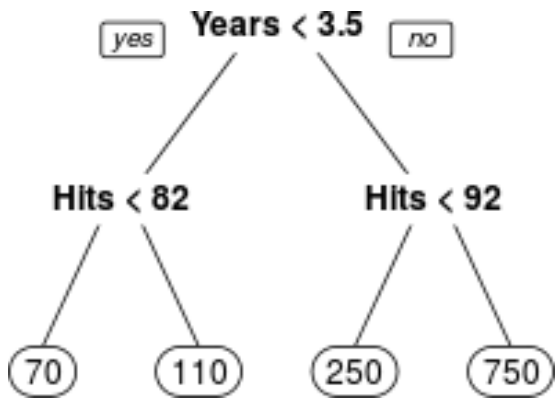


Figure: Tree for predicting the salary of a baseball player based on the number of years that he has played in the major leagues (`Year`) and the number of hits that he made in the previous year (`Hits`). See the `ISLR Hitters` data set.

Classification and regression trees

Each terminal node in the decision tree is associated with one of the regions in the feature space. Then

Classification trees

- **output value**: the most common class in the data associated with the terminal node

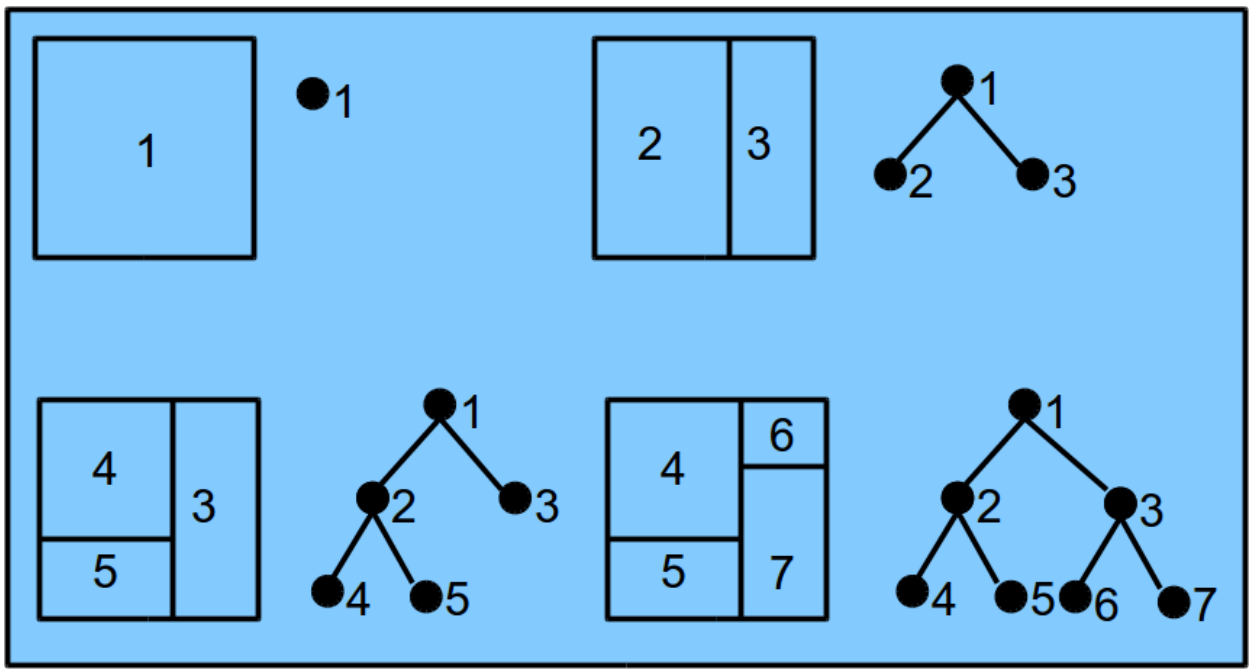
Regression trees

- **output value**: the mean output value of the training instances associated with the terminal node

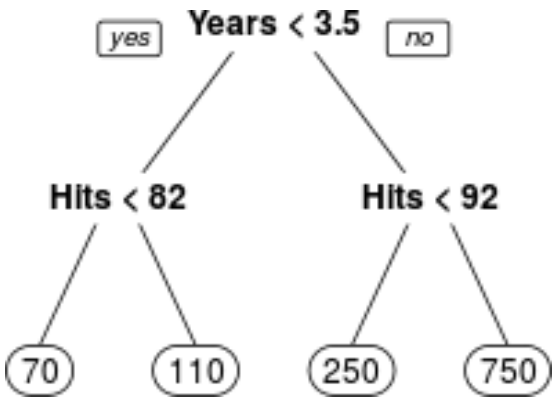
Building a tree = recursive data partitioning

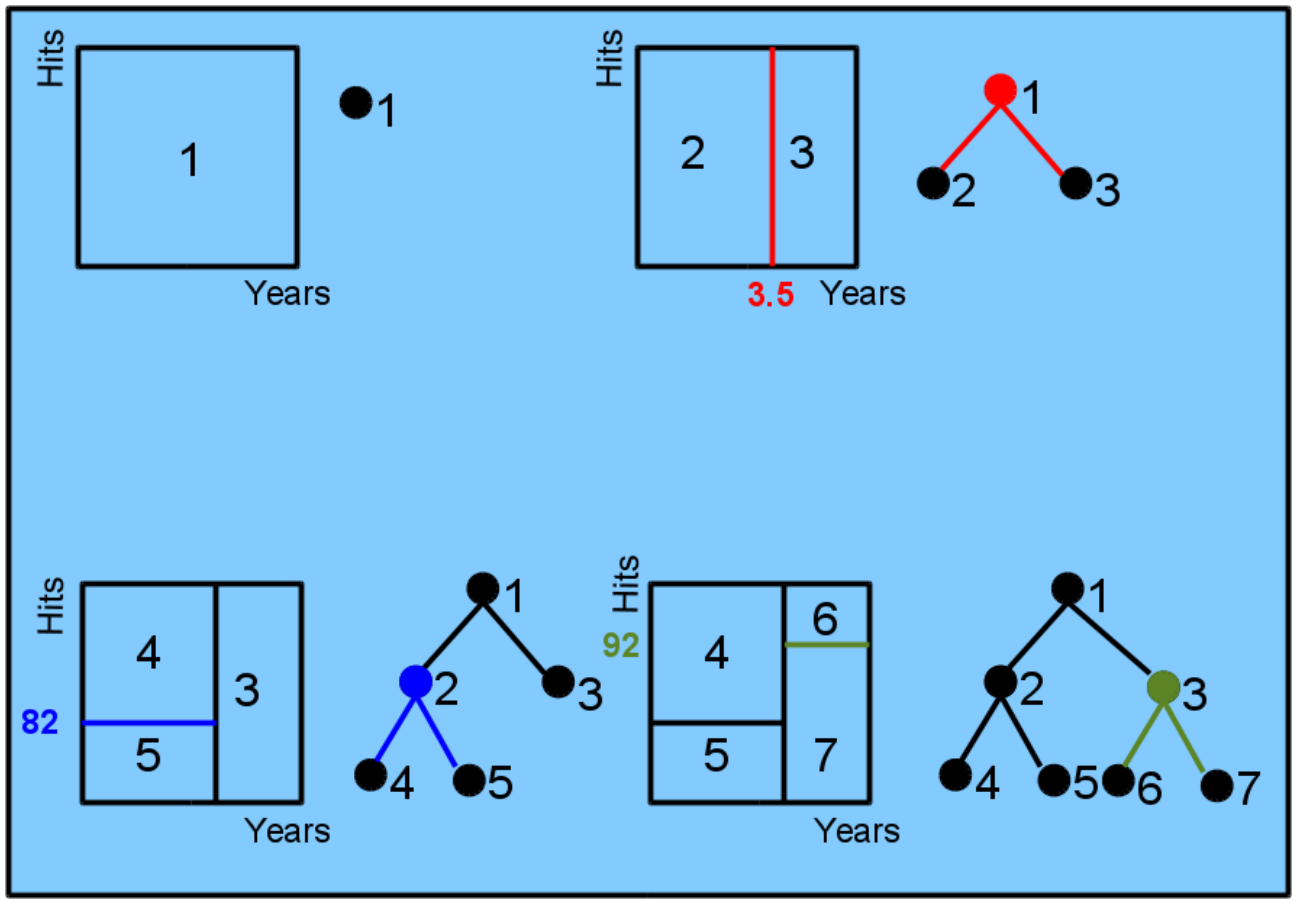
Building a decision tree is in fact a recursive partitioning process

Tree growing
The growing process is based on subdividing the feature space recursively into non-overlapping regions.

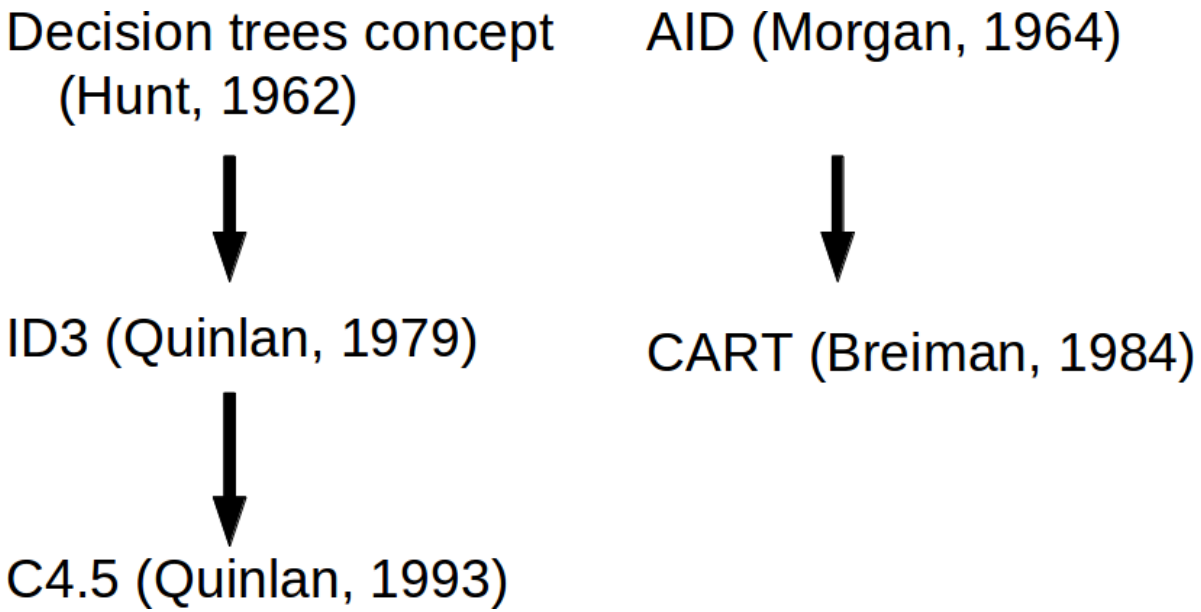


Recursive data partitioning – regression case





Historical excursion



- ID3 $\sim$ Iterative Dichotomiser
- AID $\sim$ Automatic Interaction Detection
- CART $\sim$ Classification and Regression Trees

Probably most well-known is the "C 5.0" algorithm developed by Quinlan for commercial use, which has also become the industry standard. C 5.0 is an improved extension of C 4.5. Single-threaded version is distributed under the terms of the GNU General Public License.

Learning a decision tree – key problems

Building a decision tree means to make a hierarchical sequence of splits. Each practical algorithm must be able to efficiently decide the following key questions:

(1) **How to choose a suitable splitting condition?**

(2) **When to stop the splitting process?**

Learning a decision tree – key problems

Building a decision tree means to make a hierarchical sequence of splits. Each practical algorithm must be able to efficiently decide the following key questions:

(1) How to choose a suitable splitting condition?

(2) When to stop the splitting process?

A practical answer to problem (1) is to employ entropy or another similar measure. Each node is defined by an associated subset of examples with a specific distribution of target values. After a split, the entropy in child nodes should decrease in comparison with entropy in the parent node.

Learning a decision tree – key problems

Building a decision tree means to make a hierarchical sequence of splits. Each practical algorithm must be able to efficiently decide the following key questions:

(1) How to choose a suitable splitting condition?

(2) When to stop the splitting process?

A practical answer to problem (1) is to employ entropy or another similar measure. Each node is defined by an associated subset of examples with a specific distribution of target values. After a split, the entropy in child nodes should decrease in comparison with entropy in the parent node.

The splitting process should be duly stopped just to not produce model that overfits the training data. To avoid overfitting, practical implementations usually use pruning after building a relatively deep tree.

Tree growing and tree pruning

Practical implementations of decision tree learning usually work in two main phases:

1. **Tree growing**
2. **Tree pruning**

Basic underlying idea

- First, grow a large tree that *fits the training data* quite well.
- Second, prune this tree to *avoid overfitting*.

Building a decision tree — how to avoid overfitting

Generally, overfitting can be avoided by

- applying a stopping criterion that prevents some sets of training instances from being subdivided,
- removing some of the structure of the decision tree after it has been produced.

Building a decision tree — how to avoid overfitting

Generally, overfitting can be avoided by

- applying a stopping criterion that prevents some sets of training instances from being subdivided,
- removing some of the structure of the decision tree after it has been produced.

Practically preferred strategy

- Grow a large tree T_0, stop the splitting process when only some minimum node size (say 5) is reached.
- Then prune T_0 using some pruning criteria.

Decision Trees – weak spots

- **data splitting**
  — deeper nodes can learn only from small data portions

- **sensitivity to training data set (unstable algorithm)**
  — learning algorithm is called unstable if small changes in the training set cause large differences in generated models

Decision trees — implementation in R

There are two widely used packages in R

- `rpart`
- `tree`

The algorithms used are very similar.

References

- An Introduction to Recursive Partitioning Using the RPART Routines
  by Terry M. Therneau, Elizabeth J. Atkinson, and Mayo Foundation
  (available online)
- *An Introduction to Statistical Learning with Application in R*
  Chapters 8.1, 8.3.1, and 8.3.2
  by Gareth James, Daniela Witten, Trevor Hastie and Rob Tibshirani
  (available online)
- R packages documentation — `rpart`, `tree`
  (available online)

Random Forests — an extension of Decision Trees
Resampling approach

Resampling can be used as a way to produce diversity among base learners

- Distribute the training data into K portions
- Run the learning process to get K different models
- Collect the output of the K models use a combining function to get a final output value

Bootstrapping principle

- New data sets $Data_1, \ldots, Data_K$ are drawn from $Data$ with replacement, each of the same size as the original $Data$, i.e. n.

- In the i-th step of the iteration, $Data_i$ is used as a training set, while the examples $\{\mathbf{x} \,|\, \mathbf{x} \in Data \wedge \mathbf{x} \notin Data_i\}$ form the test set.

Bootstrapping principle

- New data sets $Data_1, \ldots, Data_K$ are drawn from $Data$ with replacement, each of the same size as the original $Data$, i.e. n.
- In the i-th step of the iteration, $Data_i$ is used as a training set, while the examples $\{\mathbf{x} \,|\, \mathbf{x} \in Data \wedge \mathbf{x} \notin Data_i\}$ form the test set.
- The probability that we pick an instance is $1/n$, and the probability that we do not pick an instance is $1 - 1/n$. The probability that we do not pick it after n draws is $(1 - 1/n)^n \approx e^{-1} \doteq 0.368$.
- It means that for training the system will not use 36.8 % of the data, and the error estimate will be pessimistic. So the solution is to repeat the process many times.

Random Forests

- an ensemble method based on decision trees and bagging
- builds a number of random decision trees and then uses voting
- introduced by L. Breiman (2001), then developed by L. Breiman and A. Cutler
- very good (state-of-the-art) prediction performance
- a nice page with description
  `www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm`
- important: Random Forests helps to
  - avoid overfitting (by random sampling the training data set)
  - select important/useful features (by random sampling the feature set)

Building Random Forests

The algorithm for building a tree in the ensemble

1. Having a training set of the size n, sample n cases at random – with replacement, and use the sample to build a decision tree.

2. If there are M input features, choose a less number $m \ll M$. When building the tree, at each node a random sample of m features is selected as split candidates from the full set of M available features. Then the best split on these m features is used to split the node. A fresh sample of m features is taken at each split.
   – m is fixed for the whole procedure

3. Each tree is grown to the largest extent possible. There is no pruning.

The more trees in the ensemble, the better.
There is no risk of overfitting!

R packages for Random Forests

- **randomForest**: Breiman and Cutler's random forests for classification and regression
  – Classification and regression based on a forest of trees using random inputs.

- **RRF**: Regularized Random Forest
  – Feature Selection with Regularized Random Forest. This package is based on the 'randomForest' package by Andy Liaw. The key difference is the RRF function that builds a regularized random forest.
  – `http://cran.r-project.org/web/packages/RRF/index.html`

- **party**: A Laboratory for Recursive Partytioning
  – a computational toolbox for recursive partitioning
  – `cforest()` provides an implementation of Breiman's random forests
  – extensible functionality for visualizing tree-structured regression models is available

Examination requirements

- You should understand the basic ideas of building and using Decision Trees for classification task. You should also understand Random Forests, which is an important and effective extension of simple Decision Trees.

- You should be able to practically use `rpart()` and `randomForest()` packages in R.

- Later, we will revisit Decision Trees and go into more details. Also, later we will discuss Random Forests again, in connection with general ensemble methods.

References

- Breiman Leo, Friedman Jerome H., Olshen Richard A., Stone Charles J. *Classification and Regression Trees*. Chapman & Hall/CRC, 1984.
- Hunt, E. B. Concept Learning: An Information Processing Problem, Wiley. 1962.
- Morgan, J. N., Sonquist, J. A. Problems in the analysis of survey data, and a proposal. Journal of the American Statistical Association 58, pp. 415–434. 1963.
- Quinlan, J. R. Discovering rules from large collections of examples: A case study, in D. Michie, ed., Expert Systems in the Micro Electronic Age. Edinburgh University Press. 1979.
- Quinlan, J. R. C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, California. 1993.