

11.1 Decision Trees

Machine Learning 1: Foundations

Marius Kloft (TUK)

Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?

Manuel Fernández-Delgado

Eva Cernadas

Senén Barro

CITIUS: Centro de Investigación en Tecnologías de la Información de USC

University of Santiago de Compostela

Campus Vida, 15872, Santiago de Compostela, Spain

MANUEL.FERNANDEZ.DELGADO@USC.ES

EVA.CERNADAS@USC.ES

SENEN.BARRO@USC.ES

Dinani Amorim

Departamento de Tecnologia e Ciências Sociais- DTCS

Universidade do Estado da Bahia

Av. Edgar Chastinet S/N - São Geraldo - Juazeiro-BA, CEP: 48.305-680, Brasil

DINANIAMORIM@GMAIL.COM

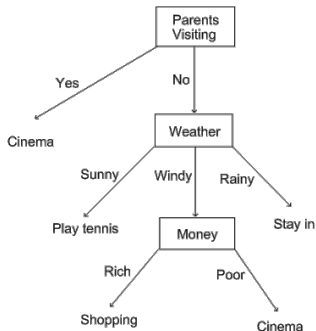
Editor: Russ Greiner

Abstract

We evaluate 179 classifiers arising from 17 families (discriminant analysis, Bayesian, neural networks, support vector machines, decision trees, rule-based classifiers, boosting, bagging, stacking, random forests and other ensembles, generalized linear models, nearest-neighbors, partial least squares and principal component regression, logistic and multinomial regression, multiple adaptive regression splines and other methods), implemented in Weka, R (with and without the caret package), C and Matlab, including all the relevant classifiers available today. We use 121 data sets, which represent the whole UCI data base (excluding the large-scale problems) and other own real problems, in order to achieve significant conclusions about the classifier behavior, not dependent on the data set collection. The classifiers most likely to be the bests are the random forest (RF) versions, the best of which (implemented in R and accessed via caret) achieves 94.1% of the maximum accuracy overcoming 90% in the 84.3% of the data sets. However, the difference is not statistically significant with the second best, the SVM with Gaussian kernel implemented in C using LibSVM, which achieves 92.3% of the maximum accuracy. A few models are clearly better than the remaining ones: random forest, SVM with Gaussian and polynomial kernels, extreme learning machine with Gaussian kernel, C5.0 and avNNet (a committee of multi-layer perceptrons implemented in R with the caret package). The random forest is clearly the best family of classifiers (3 out of 5 bests classifiers are RF), followed by SVM (4 classifiers in the top-10), neural networks and boosting ensembles (5 and 3 members in the top-20, respectively).

Decision tree learning [1]

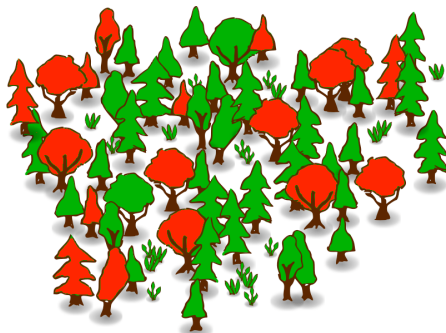
- ▶ historically one of the first non-linear ML methods



Today (2)

Bagging and random forests

- ▶ methods for growing an ensemble of decision trees



Contents of this Class

Random Forests

1 Decision Trees

2 Random Forests

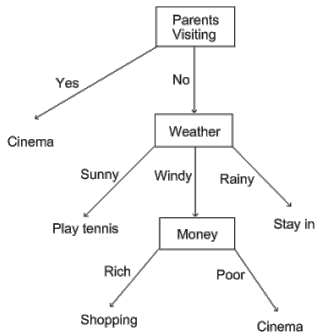
1 Decision Trees

2 Random Forests

Decision Tree

Definition

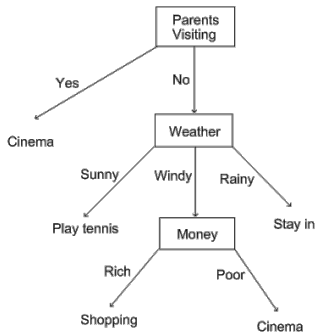
A **decision tree** is a decision support tool from operations research that uses a tree-like graph or model of decisions and their possible consequences.



Decision Tree

Definition

A **decision tree** is a decision support tool from operations research that uses a tree-like graph or model of decisions and their possible consequences.

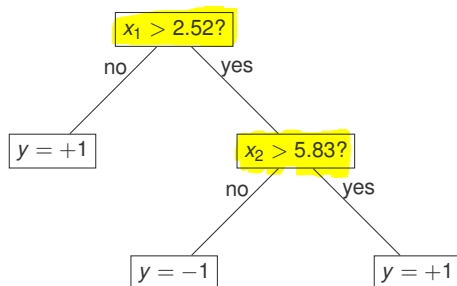


What does that have to do with ML?

Decision Trees in ML

Can use decision trees for prediction:

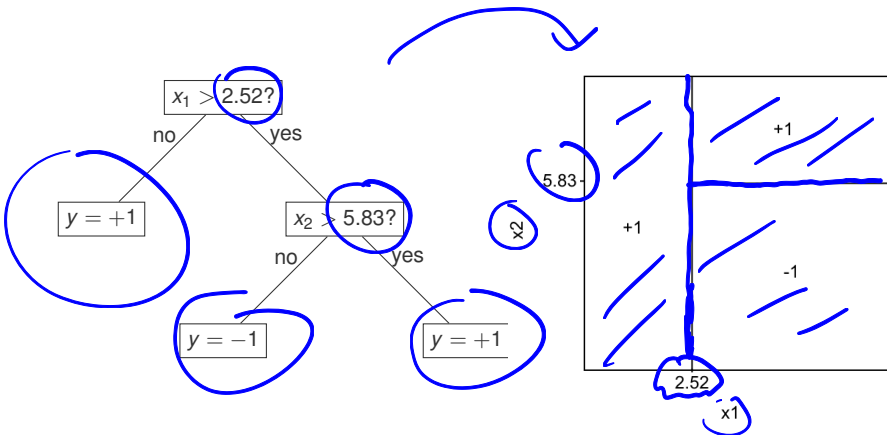
- ▶ given $\mathbf{x} = (x_1, x_2)^\top \in \mathbb{R}^2$
- ▶ follow the tree down to a terminal node
- ▶ take its label as predicted label



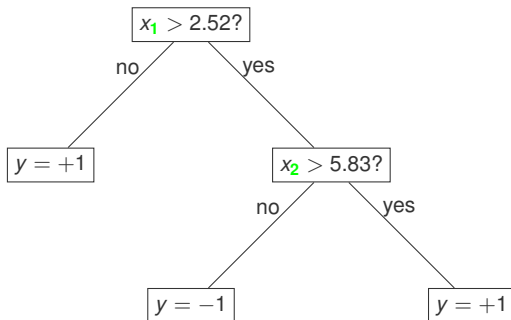
Decision Trees in ML

Can use decision trees for prediction:

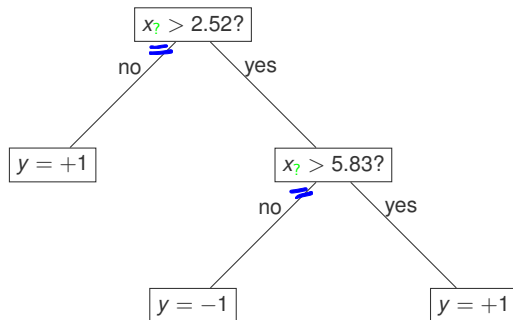
- ▶ given $\mathbf{x} = (x_1, x_2)^\top \in \mathbb{R}^2$
- ▶ follow the tree down to a terminal node
- ▶ take its label as predicted label



1. For each internal node, which **feature** to take?

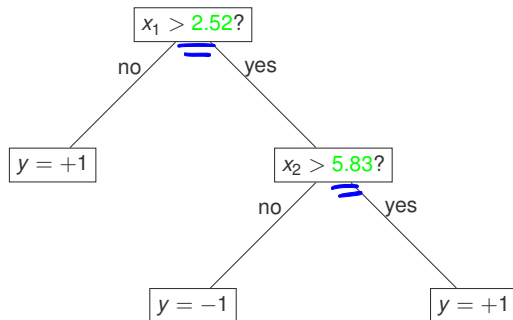


1. For each internal node, which **feature** to take?



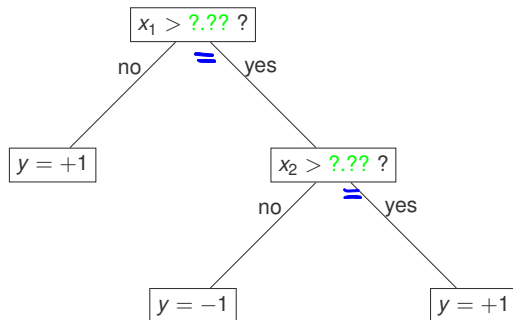
Training (2)

2. For each internal node, which **threshold** to take?



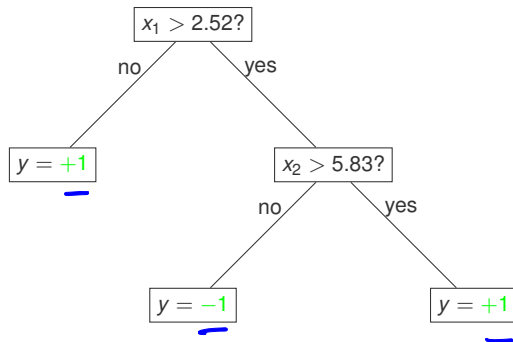
Training (2)

2. For each internal node, which **threshold** to take?



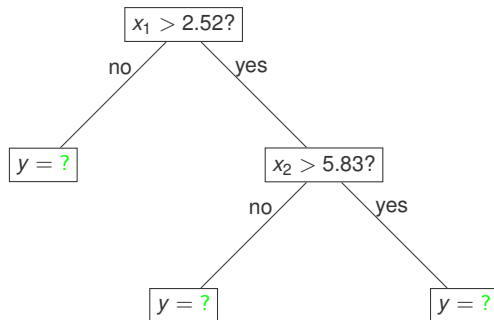
Training (3)

3. Which **labels** to use at the terminal nodes?



Training (3)

3. Which **labels** to use at the terminal nodes?

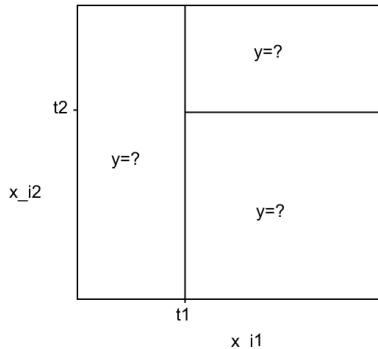
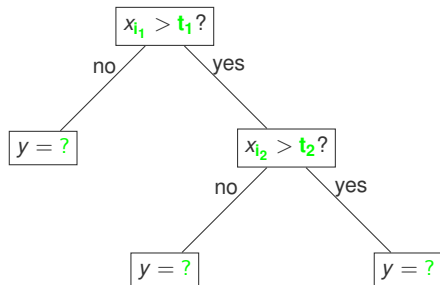


Bottom Line: Questions in Decision Tree Training

► Q1: How to split at the internal nodes?

1.1 Feature
1.2 Threshold

► Q2: How to assign labels at the terminal nodes?

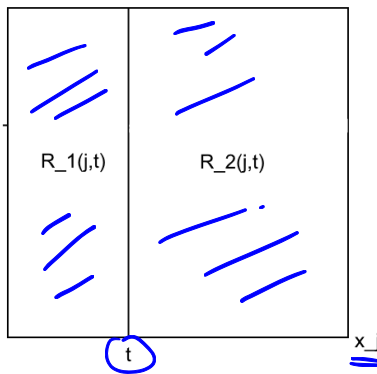


Q1: How to Split at an Internal Node?

Q1: How to Split at an Internal Node?

Let us look on one particular internal node and denote:

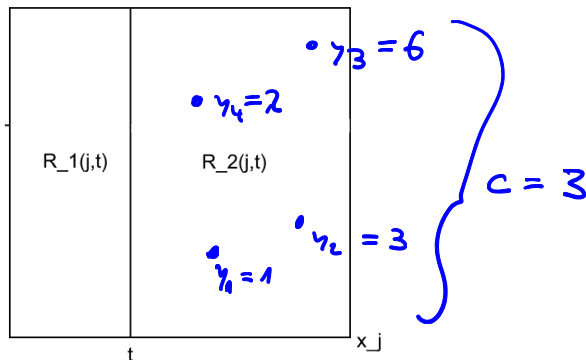
- ▶ j := feature that we split upon
- ▶ t := split threshold
- ▶ $R_1(j, t), R_2(j, t) \subset \mathbb{R}^d$:= the two regions under the internal node given by j and t



Q1: How to Split at an Internal Node?

Let us look on one particular internal node and denote:

- ▶ $j :=$ feature that we split upon
- ▶ $t :=$ split threshold
- ▶ $R_1(j, t), R_2(j, t) \subset \mathbb{R}^d :=$ the two regions under the internal node given by j and t



But how find a good feature j and threshold t ?

Q1—Regression Case

Choose feature j and threshold t by:

$$\arg \min_{j,t} \sum_{k=1}^2 \min_{c \in \mathbb{R}} \sum_{i: \mathbf{x}_i \in R_k(j,t)} (y_i - c)^2$$

Handwritten blue annotations: an equals sign under $\arg \min$, an equals sign under $\sum_{k=1}^2$, an upward arrow under $\min_{c \in \mathbb{R}}$, an equals sign under $\sum_{i: \mathbf{x}_i \in R_k(j,t)}$, and an upward arrow under $(y_i - c)^2$.

Q1—Regression Case

Choose feature j and threshold t by:

$$\arg \min_{\underline{j, t}} \sum_{k=1}^2 \min_{c \in \mathbb{R}} \sum_{i: \mathbf{x}_i \in R_k(j, t)} (y_i - c)^2$$

Note that the optimal c in the above is:

$$c = \text{mean}(\{y_i : \mathbf{x}_i \in R(j, t)\}) =: \underline{\hat{y}_{j, t, k}}$$

Interpretation:

- $\hat{y}_{j, t, k}$ is the mean label of the training instances falling into the region $R_k(j, t)$.

Q1—Classification Case

Denote:

- ▶ $p_{j,t,k} :=$ the fraction of instances in region $R_k(j, t)$ that are labeled +1

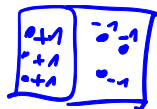
Q1—Classification Case

Denote:

- ▶ $p_{j,t,k} :=$ the fraction of instances in region $R_k(j, t)$ that are labeled +1

Choose feature j and threshold t by:

$$\arg \min_{j,t} \sum_{k=1}^2 \underline{g(p_{j,t,k})}$$



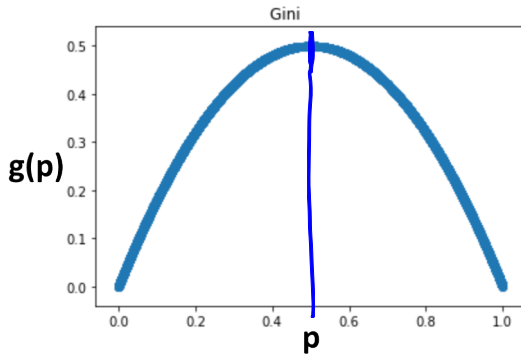
where g is the **Gini impurity measure**.

Gini Impurity Measure

Definition

The **Gini impurity measure** is defined as:

$$g(p) := 2p(1 - p)$$



Q2: How to Assign a Label at a Terminal Node?

Denote:

- ▶ $S \subset \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ is set of the instances falling into the region R under a terminal node

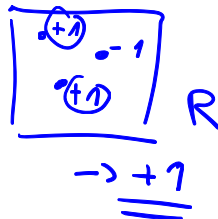
Q2: How to Assign a Label at a Terminal Node?

Denote:

- ▶ $S \subset \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ is set of the instances falling into the region R under a terminal node

Classification:

- ▶ majority vote over the labels of the instances in S



Q2: How to Assign a Label at a Terminal Node?

Denote:

- ▶ $S \subset \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ is set of the instances falling into the region R under a terminal node

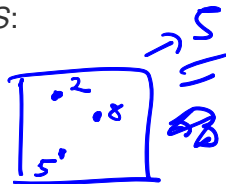
Classification:

- ▶ majority vote over the labels of the instances in S

Regression:

- ▶ average the labels of the training instances in S :

$$\hat{y} := \text{mean}(\{y_i : \mathbf{x}_i \in R\})$$



Pruning

What about overfitting?

Pruning

What about overfitting?

A large, fully grown tree is prone to overfit!
Remedy?

Pruning


What about overfitting?

A large, fully grown tree is prone to overfit!
Remedy?

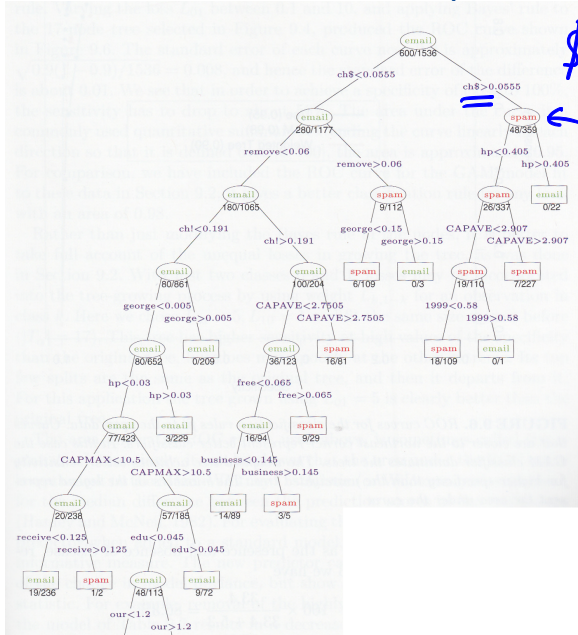
Pruning

Remove internal nodes that produce no substantial decrease in prediction accuracy as measured on a hold-out data set.

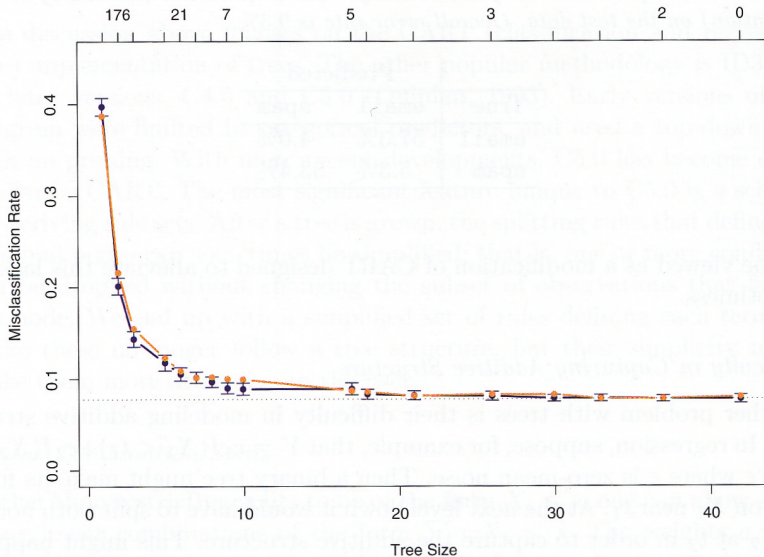
Further reading: sophisticated pruning strategies exist, e.g., **cost-complexity pruning** (e.g., Hastie, Tibshirani, and Friedman [2], Chapter 9).



Example: Decision Tree Grown on Spam Data [2]



Example: SPAM (continued)



Acknowledgment

This lecture is based on [2], Chapter 9 and 15. The figures on Slides 21 and 22 are taken from [2].

Refs I



L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, Classification and regression trees. CRC press, 1984.



T. Hastie, R. Tibshirani, and J. Friedman, The elements of statistical learning, 2nd edition. Springer series, 2009.

Chap 9