TREE-BASED METHODS (ch 8)

12/4/2021

Non-linear methods

Main idea: Divide the feature space into (simple) regions (rectangles/boxes)

The predicted outcome ($\hat{y}$ or class) is a function of the region each observation belongs to

→ stratification or segmentation of the feature space

The splitting process has a tree structure.

REGRESSION TREES: MOTIVATING EXAMPLE

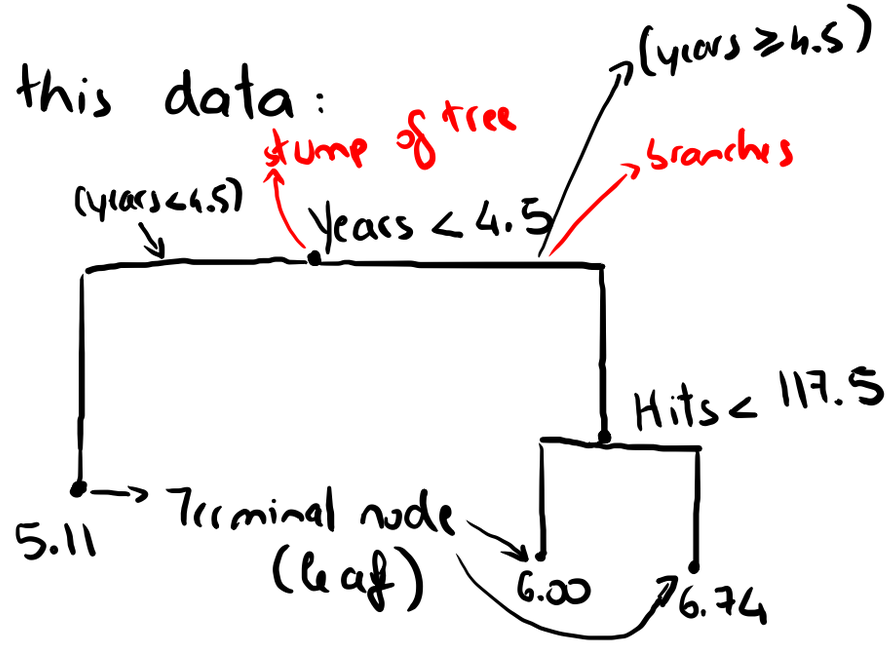
Predicting next year salary of baseball players using this year's statistics:

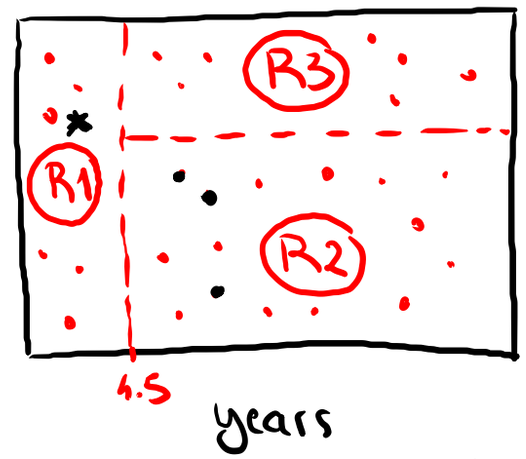
Predictor 1: # years played in major leagues

Predictor 2: # hits

Response: salary (next season)

Decision tree fitted to this data:

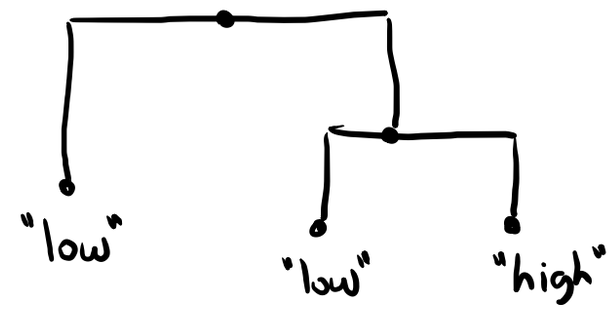




3 terminal leaves ⟷ 3 non-overlapping regions

CLASSIFICATION TREE : EXAMPLE

Recoding response variable "Salary" as { "low" / "high" } according to some threshold (eg mean 5.93)



Let's look at the details of how one fits a tree to data :

① Dividing the space into J non-overlapping regions

② Predicting a value of the response Y in each region

Step 2 : Assume we have a partition with J leaves / regions: $R_1, \ldots, R_J$.

Then:

Regression: Given a new observation x, consider the region R_j it falls into and predict $\hat{y}$ by the average y for all training observations in region R_j

Classification Given a new observation x, consider the region R_j it falls into and use for classification

$\hat{p}(c \mid R_j)$ = proportion of training observations in region R_j that belong to class c (similar to k-nn)

Use these probabilities within a Bayes classifier, eg assign x to class with highest probability (threshold 0.5)

Step 1 HOW TO BUILD A TREE

Aim : Divide the feature space into boxes

Idea : Choose regions so as to minimize some optimality criterion

Two Key aspects :

① Choice of optimality criterion

② Find a computationally efficient approach

CHOICE OF OPTIMALITY CRITERION

In regression, the most natural choice is RSS :

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

$\hat{y}_{R_j}$: mean response in region R_j

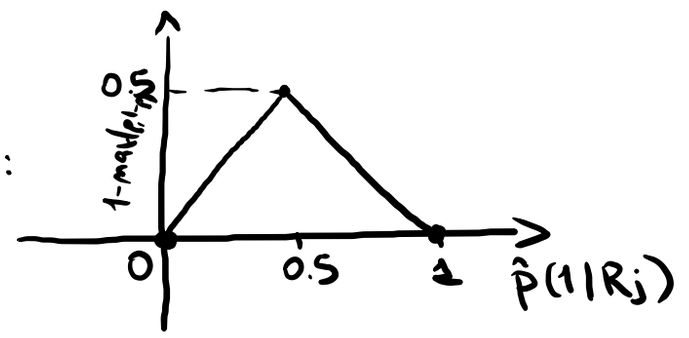
In classification, a natural choice is

$$E = \sum_{j=1}^{J} E_j \quad \text{where} \quad E_j = 1 - \max_c \hat{p}(c|R_j)$$

E_j: fraction of observations in region R_j that are misclassified

$\max_c \hat{p}(c|R_j)$: probability associated to the predicted class (0.5 threshold)

This function is not very "smooth" for optimisation purposes:

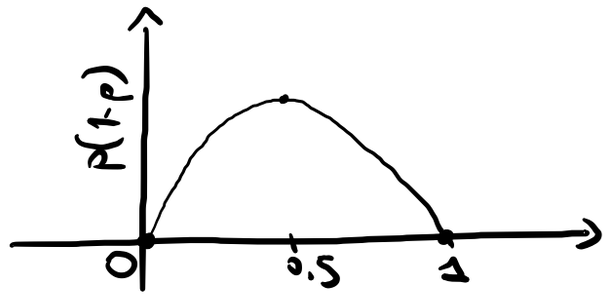


$$E_j = 1 - \max(\hat{p}(1|R_j), 1-\hat{p}(1|R_j))$$

There are two alternatives:

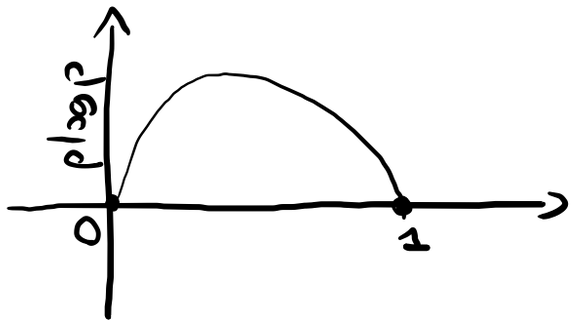
Gini index: $G_j = \sum_{c=1}^{K} \hat{p}(c|R_j)\,(1-\hat{p}(c|R_j))$

variance of Bernoulli



Entropy (or cross-entropy)

$$D_j = -\sum_{c=1}^{K} \hat{p}(c|R_j) \log \hat{p}(c|R_j)$$



RECURSIVE BINARY SPLITTING

- top-down approach : start with the whole space as a single region
- binary : at each step, one region is selected and split into two regions
- greedy : at each step, the best split is made without looking ahead (local rather than global optimum)

ALGORITHM

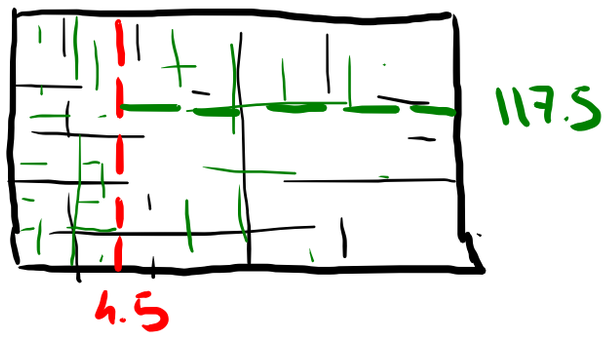
① For each predictor X_k (Years) and all its cut points s (h.s), define two regions :

$$R_-(k,s) = \{X : X_k < s\} \quad , \quad R_+(k,s) = \{X : X_k \geq s\}$$

Compute the total optimality by summing up the optimality of R_- and R_+.

Remarks : . For categorical variables, the splitting divides the categories into 2 groups
. For continuous variables, a selection of cutpoints is made (efficiently)

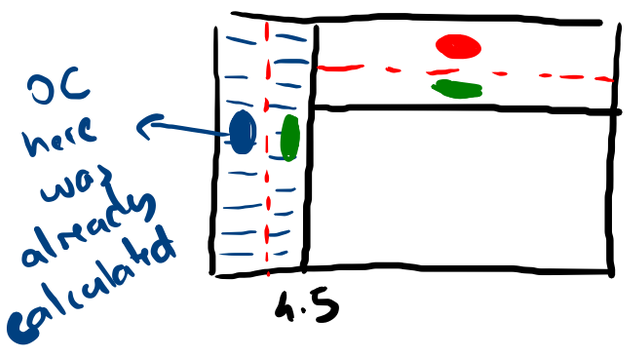
② Choose (K, s) that minimizes the optimality criterion
(RSS, Gini / Entropy)



③ Repeat the procedure inside each of the two regions, getting a new region

④ Continue until some stopping criterion is reached, eg maximal number of observations in each region...

Computational tricks



Big issue with trees : Very high variance, predict very well training data but not as well the test data (overfitting)

Some solutions :

① Fit a smaller size tree (lower variance but typically too high bias)

② Stop dividing when reduction in RSS / Gini / Entropy is below some (high) threshold (problems because of the greediness of the algorithm → an "unimportant" split early or could result in good splits later on)

③ Build a big tree and then prune it back, ie remove branches

HOW TO PRUNE A TREE

OPTIMAL WAY : . consider all subtrees
. calculate optimality criterion on a validation set or via CV
for each subtree
. choose the best one

Computationally too expensive.

Solution : COST-COMPLEXITY PRUNITY

Consider a sequence of subtrees indexed by a tuning parameter $\alpha (\geq 0)$

ALGORITHM

① Use recursive binary spliting to build a big tree (eg 5 observation maximum in each region)
Denote this with T_0

② For each value of α (in a pre-selected sequence) choose subtree $T_\alpha \subset T_0$ such that

$$T_\alpha = \underset{T \subset T_0}{\operatorname{argmin}} \underbrace{\sum_{j=1}^{|T|} QC(R_j) + \alpha |T|}_{\text{objective function}}$$

$|T|$ → number of terminal nodes in T

$\alpha |T|$ → penalise too complex subtrees

Choice of α :
- $\alpha = 0 \rightarrow T_0$
- As α increases, the tree is pruned in a nested way
- Choice of α corresponds to finding optimal bias/variance trade-off

CV for choice of α (k-fold)

Assume we have a sequence of α values $= (\alpha_1, \ldots, \alpha_n)$

① Split the data in K folds : choose $K-1$ folds as training data and the remaining one as test data

② For each fold and for each α : use recursive binary splitting on training data to build T_0 and then select subtree T_α. Measure error on test data.

③ Compute average error across folds for each α :

$$CV(\alpha_1),\ CV(\alpha_2),\ \ldots,\ CV(\alpha_n)$$

④ Choose α_{OPT} as the one that minimises CV error.

⑤ Typically, one would refit the optimal model on full data.