

Machine Learning

Theory and Applications: third lecture

Arnak Dalalyan

ENSAE PARISTECH

12 avril 2016

Framework and notation

- ▶ We observe

$$(X_i, Y_i) \in \mathcal{X} \times \mathcal{Y}, \quad i = 1, \dots, n$$

independent randomly drawn from P over $\mathcal{X} \times \mathcal{Y} = \mathcal{Z}$.

- ▶ For a random pair $(X, Y) \sim P$, the goal is to predict Y with the help of X .
- ▶ For a prediction $\hat{Y} = g(X)$ based on the prediction rule $g : \mathcal{X} \rightarrow \mathcal{Y}$, we measure the incurred error by

$$R_P(g) = \mathbf{E}_P[\ell(Y, \hat{Y})],$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ is a loss function.

k -nearest neighbors

- ▶ We observe

$$(X_i, Y_i) \in \mathcal{X} \times \mathcal{Y}, \quad i = 1, \dots, n$$

independent randomly drawn from P over $\mathcal{X} \times \mathcal{Y} = \mathcal{Z}$.

- ▶ For a new $x \in \mathcal{X}$, we look for k nearest neighbors of x among X_1, \dots, X_n .
- ▶ Let us denote these neighbors by X_{i_1}, \dots, X_{i_k} .
- ▶ We use the corresponding labels Y_{i_1}, \dots, Y_{i_k} for predicting the label of x .
- ▶ In regression, the prediction is the average of Y_{i_1}, \dots, Y_{i_k} .
- ▶ In binary classification, the prediction is the most frequent label among Y_{i_1}, \dots, Y_{i_k} .

Partition based methods

- ▶ **Principle** : Divide and conquer.
- ▶ Let $\mathcal{A} = \{A_1, \dots, A_M\}$ be a partition of \mathcal{X} :

$$\mathcal{X} = \bigsqcup_{m=1}^M A_m.$$

- ▶ Let

$$\mathcal{G}_{\mathcal{A}} = \left\{ g : \mathcal{X} \rightarrow \mathcal{Y} : \forall m \quad g \text{ is constant on } A_m \right\}.$$

- ▶ The minimizer of the empirical risk is :

$$\widehat{g}_{n, \mathcal{A}} \in \arg \min_{g \in \mathcal{G}_{\mathcal{A}}} \widehat{R}_n(g) = \arg \min_{g \in \mathcal{G}_{\mathcal{A}}} \frac{1}{n} \sum_{i=1}^n \ell(Y_i, g(X_i)),$$

Binary Classification and least squares regression

- ▶ Assume that $\mathcal{Y} \subset \mathbb{R}$ and define

$$N_m = \sum_{i=1}^n \mathbb{1}_{A_m}(X_i) \quad \text{and} \quad \bar{Y}_{A_m} = \frac{1}{N_m} \sum_{i=1}^n Y_i \mathbb{1}_{A_m}(X_i).$$

- ▶ For binary classification with $\mathcal{Y} = \{0, 1\}$ and the loss $\ell(y, y') = \mathbb{1}(y \neq y')$, any minimizer of the empirical risk satisfies

$$\forall m = 1, \dots, M; \forall x \in A_m \quad \widehat{g}_{n, \mathcal{A}}(x) = \begin{cases} 1, & \text{if } \bar{Y}_{A_m} > 1/2, \\ 0, & \text{if } \bar{Y}_{A_m} < 1/2. \end{cases}$$

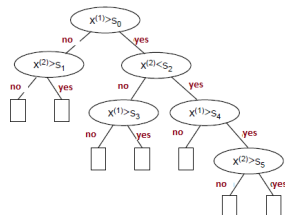
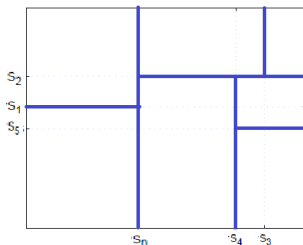
- ▶ For regression with $\mathcal{Y} = \mathbb{R}$ and squared loss $\ell(y, y') = (y - y')^2$,

$$\widehat{g}_{n, \mathcal{A}}(x) = \sum_{m=1}^M \bar{Y}_{A_m} \mathbb{1}_{A_m}(x) = \sum_{m=1}^M \left\{ \frac{1}{N_m} \sum_{i=1}^n Y_i \mathbb{1}_{A_m}(X_i) \right\} \mathbb{1}_{A_m}(x).$$

Decision trees : general principle

The goal : construct a partition \mathcal{A} de \mathcal{X} .

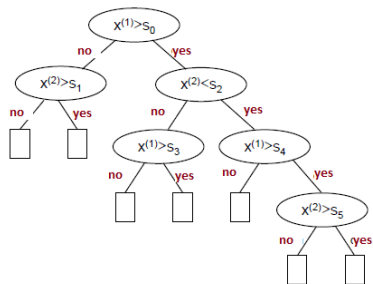
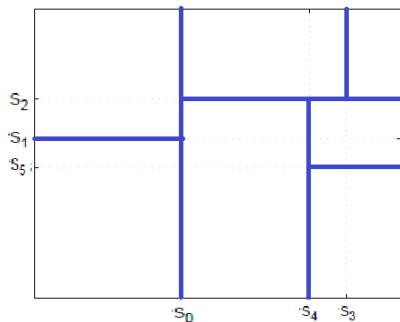
Example :



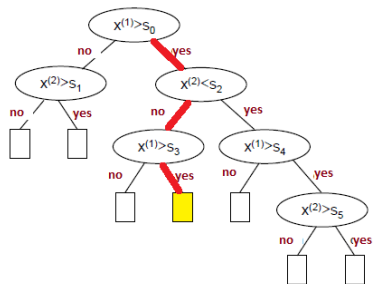
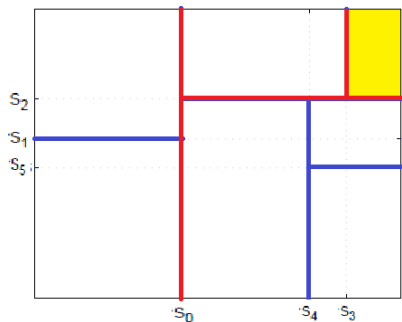
Principle : Each node corresponds to a subset of the feature space \mathcal{X} .

- ▶ the root corresponds to \mathcal{X} ,
- ▶ if A is the set corresponding to a node and if $A^{(1)}, \dots, A^{(k)}$ are the parts corresponding to the descendents of this node, then $A^{(1)}, \dots, A^{(k)}$ form a partition of A .

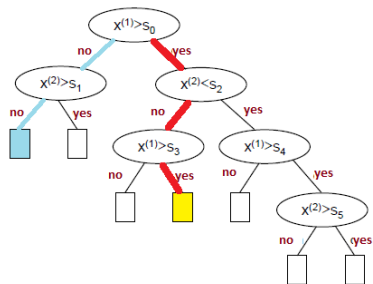
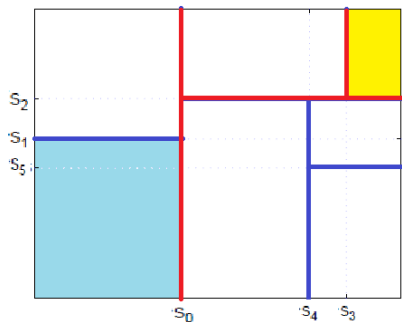
Decision trees : general principle



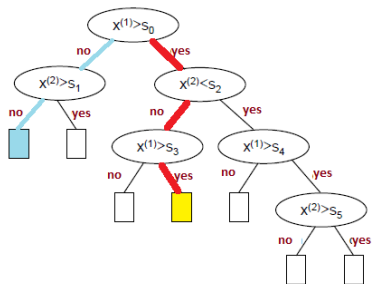
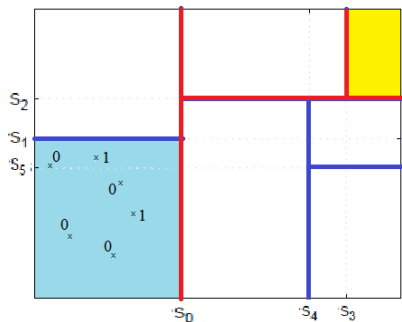
Decision trees : general principle



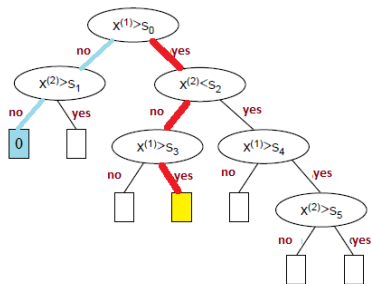
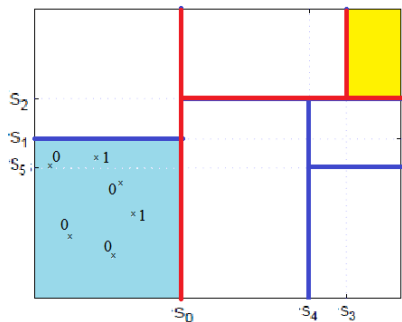
Decision trees : general principle



Decision trees : general principle



Decision trees : general principle



Decision trees : construction

1. For every node, we look for the variable and the test (of type $x = a, x > a, x < a, \dots$) that offers the best split of the sample into homogeneous parts,
2. We create the new nodes and assign to each node a part of the sample.
3. We repeat this procedure until the stopping rule is attained.
4. We perform a pruning in order to eliminate "useless" branches.

The most famous algorithms : CART, C4.5, C5.0,...

CART : classification and regression trees

- ▶ 1984, L. Breiman, J.H. Friedman, R.A. Olshen et C.J. Stone,
- ▶ among the most frequently used and the most accurate,
- ▶ can be found in : SAS, R, S-Plus, ...
- ▶ binary : each node is split into two new nodes,
- ▶ treats missing values,
- ▶ handle any type of variable,
- ▶ separation criteria :
 - ▶ Gini index for the classification,
 - ▶ least squares for the regression.

Binary Classification : more details

- ▶ Gini index of the node corresponding to $A \subset \mathcal{X}$:

$$G(A) = 1 - \bar{Y}_A^2 - (1 - \bar{Y}_A)^2 \quad [0 \leq G(A) \leq 0.5, \forall A]$$

Idea : for a good node $G(A) \approx 0$, while for a poor node $G(A) \approx 0.5$.

- ▶ We assess the quality of a test of splitting of A into A_1 and A_2 by the homogeneity gain :

$$I_G(A_1, A_2) = G(A) - qG(A_1) - (1 - q)G(A_2)$$

where $q = N_{A_1}/N_A$, the proportion of $X_i \in A$ that go to A_1 .

- ▶ Among all the candidate partitions $\{A_1, A_2\}$ of A , we choose the one minimizing the gain of homogeneity.

Cross Validation

Input :

- training sample $\mathcal{D}_n = \{(X_1, Y_1), \dots, (X_n, Y_n)\}$
- K prediction algorithms : $g^{(1)}, \dots, g^{(k)}$
- an integer $V \in \{2, \dots, n\}$

Output : index k^* of the algorithm with the smallest estimated error.


Algorithm :

- partition the set $\{1, \dots, n\}$ into V well balanced parts B_1, \dots, B_V ,
- $\forall (k, v) \in \llbracket 1, K \rrbracket \times \llbracket 1, V \rrbracket$, compute the predictor $g^{(k,v)}$ by applying the rule $g^{(k)}$ to the sub-sample $\mathcal{D}_n \setminus \{(X_i, Y_i) : i \in B_v\}$.
- define the estimated error of $g^{(k)}$ by

$$e_k = \frac{1}{V} \sum_{v=1}^V \frac{1}{|B_v|} \sum_{i \in B_v} \ell(Y_i, g^{(k,v)}(X_i)).$$

- set $k^* = \arg \min_k e_k$.

Starting with R

- ▶ Start R by clicking on  R 3.96 3.1.0.
- ▶ Go to File menu (on the top left corner of the R window), then click on New script. You will see a new window. Type all your code in this window and save it in a file, in order to be able to use it in the future.
- ▶ Type Ctrl - S for saving the file. Give the file the extension .R and be sure to save it in a location that will be easy to find.
- ▶ Just to check that it works, type the following lines :

```
n = 10^4
```

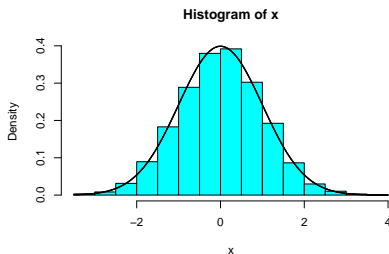
```
x = rnorm(n)
```

```
hist(x, freq = FALSE, col = "cyan") # color cyan
```

```
curve(dnorm(x), add=TRUE, lwd=2) # line width 2
```


Starting with R

- ▶ Type `Ctrl A` then `Ctrl R` in order to execute the code. If everything goes well, you will see the following plot



- ▶ Note that you can save the plot by going to the `File` menu and clicking on `save as`, then choosing the file format (`pdf`, `pnd`, `jpeg`, ...)

Iris dataset

- ▶ We start by calling the `iris` dataset

```
data(iris)
```

```
head(iris) #shows first six lines
```

```
summary(iris) #summary statistics
```

- ▶ You can type `?iris` in order to read the description of the dataset.
- ▶ If you need to check or to modify the data, you can type `edit(iris)`.

Iris dataset

k-NN prediction

- ▶ Prior to trying to predict the species of flowers, we split the dataset into two parts : training set and testing set.

```
train = iris[c(1:30, 51:80, 101:130), 1:5]
```

```
test = iris[c(31:50, 81:100, 131:150), 1:5]
```

- ▶ Thus, our goal is to use the first 90 data-points for predicting the species of the last 60 data-points.
- ▶ To perform prediction kNN prediction, we can type

```
library(class)
```

```
pred = knn(train[,1:4], test[,1:4], train[,5], k = 3)
```

```
# display the confusion matrix
```

```
table(pred, test[,5])
```

Iris dataset

Decision tree

- ▶ We can also use the decision trees for prediction :

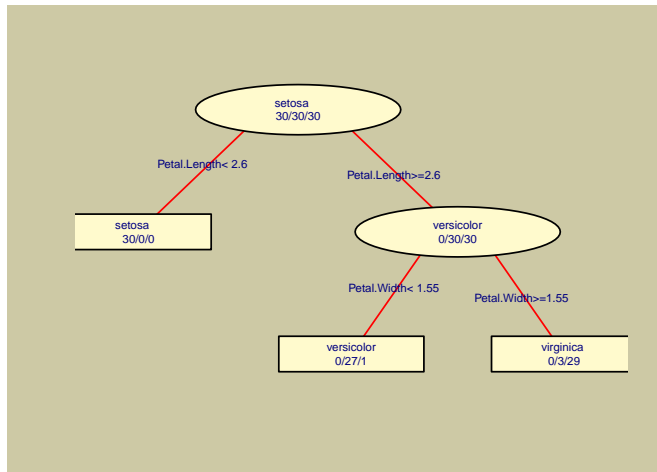
```
library(rpart)
rt = rpart(Species ~ ., data = train)
```
- ▶ We can see the result :

```
par(lwd=2, col="red")
plot(rt, compress=TRUE)
```
- ▶ A better result can be obtained by

```
install.packages("DMwR")
library(DMwR)
par(lwd=2, bg="lemonchiffon3")
prettyTree(rt, col="navy", bg="lemonchiffon")
```

Iris dataset

Decision tree



Iris dataset

Decision tree

- ▶ Predicted values on the test dataset can be obtained by

```
pred.rt = predict(rt, test, type="class")
```
- ▶ One can display the confusion matrix :

```
table(pred.rt, test[,5])
```
- ▶ Note that one can also get the predictions as probabilities of each class :

```
pred.rt1 = predict(rt, test, type="prob")  
head(pred.rt1)
```

Iris dataset

k-NN prediction and cross validation

- ▶ One can choose the value of k in k-NN by cross validation.
- ▶ The code of the next page performs 5-fold cross-validation to select k from the set $\{1, \dots, 10\}$.
- ▶ The output is something like this

```
> apply(cvpred, 2, function(x) sum(class!=x))  
[1] 3 3 2 2 4 3 5 5 8 4
```

Iris dataset

k-NN prediction and cross validation

```
fold = sample(rep(1:5, each=18))
cvpred = matrix(NA, nrow=90, ncol=10)
fold = sample(rep(1:5, each=18))
  for (k in 1:10)
    for (v in 1:5)
      {
        sample1 = train[which(fold!=v), 1:4]
        sample2 = train[which(fold==v), 1:4]
        class1 = train[which(fold!=v), 5]
        cvpred[which(fold==v), k] = knn(sample1, sample2, class1, k=k)
      }
    class = as.numeric(train[, 5])
  apply(cvpred, 2, function(x) sum(class!=x))
```