Machine Learning

TSIA-SD210 - P3

Lecture 4 - Local Averaging - Decision and regression trees

Florence d'Alché-Buc

Contact: florence.dalche@telecom-paristech.fr,
Télécom ParisTech,Université of Paris-Saclay, France

Table of contents

Outline

Supervised learning

- Let's call X a random vector that takes its value in $\mathcal{X} = \mathbb{R}^p$
- Y a random variable that takes its value in $\mathcal{Y}$
- $\mathcal{D}$ is the joint probability distribution of (X, Y)
- $\mathcal{Y} = \mathbb{R}$ in case of regression
- $\mathcal{Y} = \{1, -1\}$ in case of supervised binary classification
- $\mathcal{Y} = \{1, \ldots, C\}$ in case of supervised multiclass classification

Minimizing the true risk

- Denote $\mathcal{H}$ the hypothesis class: e.g. the set of models we consider
- Define a local loss function $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+$,

One wishes to solve this problem:

$$\arg\min_{f \in \mathcal{H}} \mathbb{E}_{(X,Y)\sim\mathcal{D}}[\ell(Y, f(X))]$$

from the knowledge of $\{(x_i, y_i), i = 1, \dots n\}$.

Target functions

Classification
For $\ell(y, f(x))$: $(0/1)$ prediction loss, the best classifier is the **Bayes classifier**: $f_{Bayes}(x) = \arg\max_c \mathbb{P}(Y = c|x)$

Regression
for $\ell(y, f(x))) = (y - f(x))^2$: ℓ_2 loss, the best solution in regression is the regression function: $f_{reg}(x) = \mathbb{E}[Y|x]$

Minimizing the (regularized) empirical risk

$\mathcal{S} = \{(x_i, y_i), i = 1, \ldots, n\}$, i.i.d. sample; Ω: regularization term

$$\arg\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(x_i)) + \Omega(f)$$

Statistical Learning in a nutshell

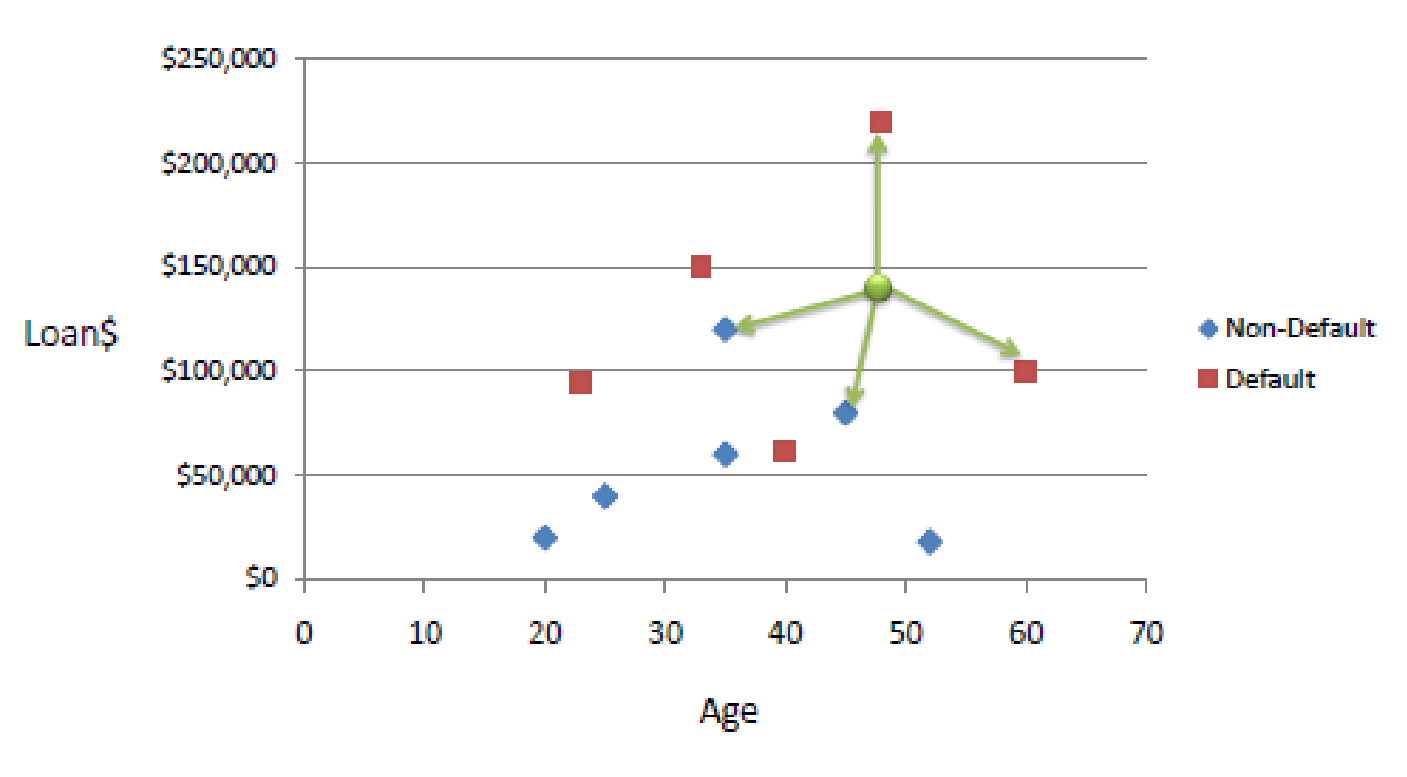
- **Learning** $f_n = \mathcal{A}(\mathcal{S}_n, \mathcal{H}, \ell, \Omega)$ with
  - $\mathcal{A}$: learning algorithm
  - $\mathcal{S}_n$: training data
  - $\mathcal{H}$: class of functions
  - Ω: some complexity measure
  - ℓ : Local loss function
- **Prediction**: give me a new x, and compute $f_n(x)$

Statistical Learning with local average models and trees

- Nonparametric approach to machine learning
- Training data are the main parameters !
- Local average / Majority vote: prediction is made locally, looking at the neighbours
- **Why is it important ?** Simple way to handle a learning problem, very general, can be extended to many kinds of outputs
- Limited number of hyparameters
- **Drawback:** heavily depends on the number of data
- **Advantage**: the model itself contains the training data on which it is based (transparency, pre-training)

Outline

Algorithme des K-plus-proches voisins



Algorithme des K-plus-proches voisins

K-PPV (en anglais K-Nearest neighbours: K-NN)

Cas 2 classes:

$$f_{KNN}(x) = \arg \max_{y \in \{-1,1\}} \frac{N_y^K(x)}{K},$$

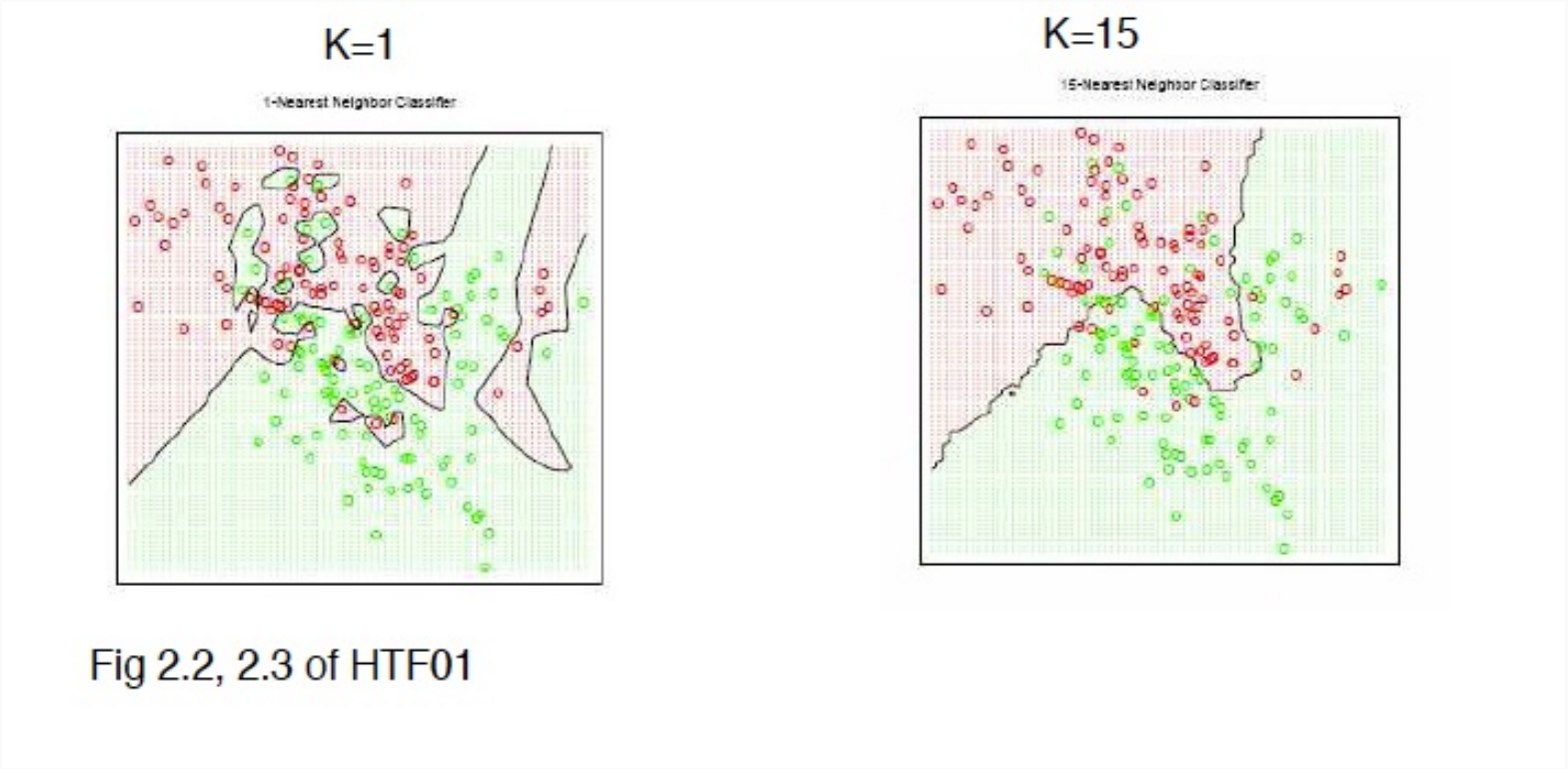
avec :

- Soit K un entier strictement positif.
- Soit d une métrique définie sur $\mathcal{X} \times \mathcal{X}$
- $S = \{(x_i, y_i), i = 1, \ldots, n\}$
- Pour une donnée x, on définit σ la permutation d'indices dans $\{1, \ldots, n\}$ telle que:
  - $d(x, x_{\sigma(1)}) \leq d(x, x_{\sigma(1)}) \ldots \leq d(x, x_{\sigma(n)})$
- $S_x^K = \{x_{\sigma(1)}, \ldots, x_{\sigma(K)}\}$: K premiers voisins de x
- $N_y^K(x) = |\{x_i \in S_x^K, y_i = y\}|$

Le paramètre de lissage K

K: trop petit : la fonction f est trop sensible aux données : large variance

K : trop large : la fonction f devient trop peu sensible aux données : biais important

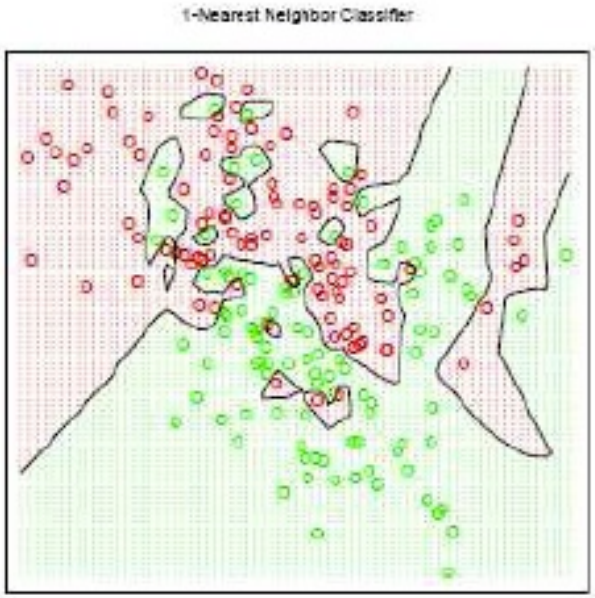


Book of Hastie, Tibshirani and Friedman (The elements of statistical learning, Springer)

Question: Tracer la frontière de décision lorsque K = 50

Le paramètre de lissage K

K=1



K=15

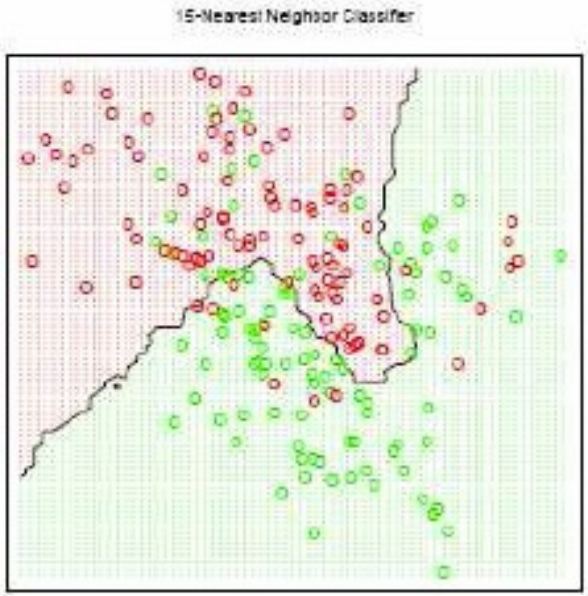


Fig 2.2, 2.3 of HTF01

Book of Hastie, Tibshirani and Friedman (The elements of statistical learning, Springer)

Décomposition biais-variance

Calcul du risque (de l'erreur en généralisation)
On suppose: $Y = f(X) + \epsilon$ avec ϵ centré et de variance σ_ϵ^2. Fixons x.
Soit $\hat{f}$ dépendant de l'échantillon $\mathcal{S}$.

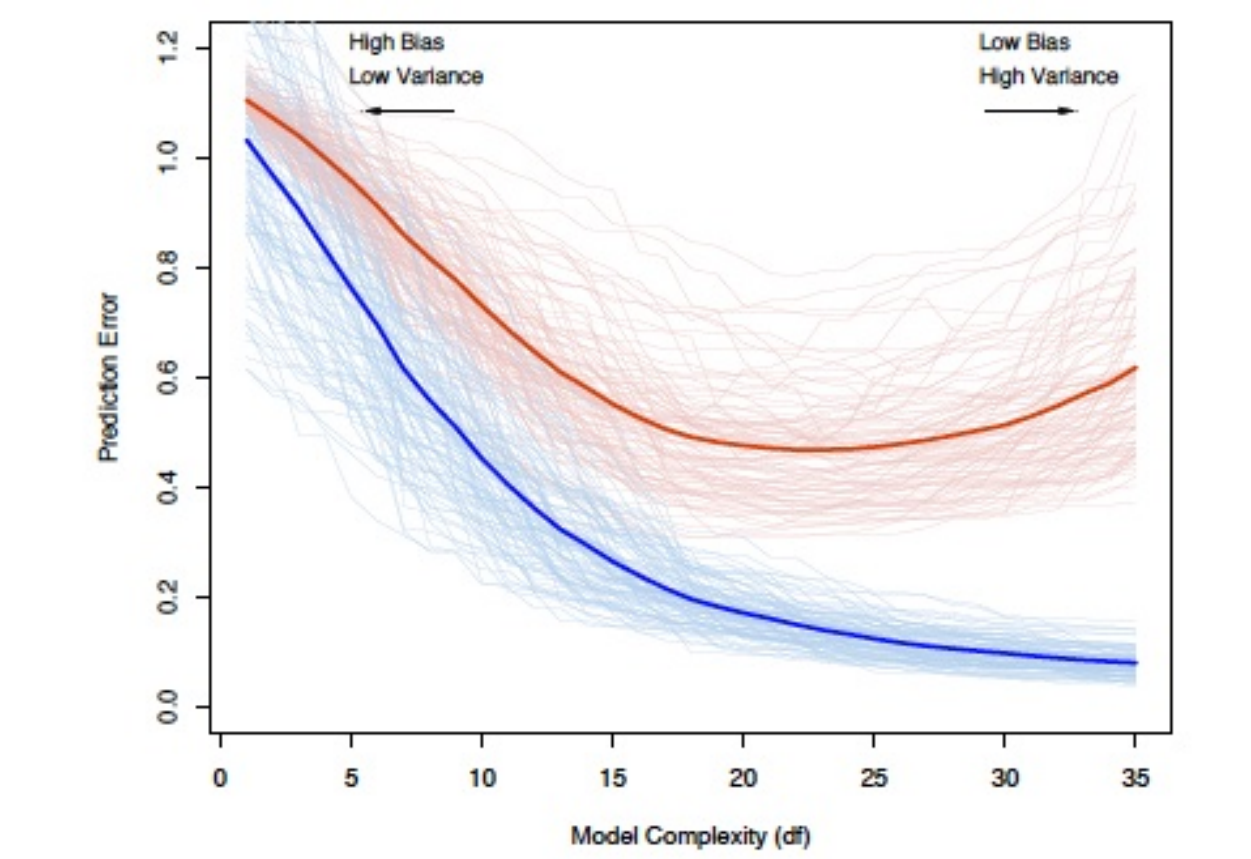
$$\begin{aligned} E_{\mathcal{S}} E_Y[(Y - \hat{f}(x))^2]] &= E_{\mathcal{S}}[E_Y[Y^2] + \hat{f}(x)^2 - 2E_Y[Y]\hat{f}(x)] \\ &= E_Y[Y^2] + E_{\mathcal{S}}[\hat{f}(x)^2] - 2E_{\mathcal{S}}[E_Y[Y\hat{f}(x)] \end{aligned}$$

On sait qu'on a: $E[Z^2] = E[(Z - E[Z])^2] + E[Z]^2$

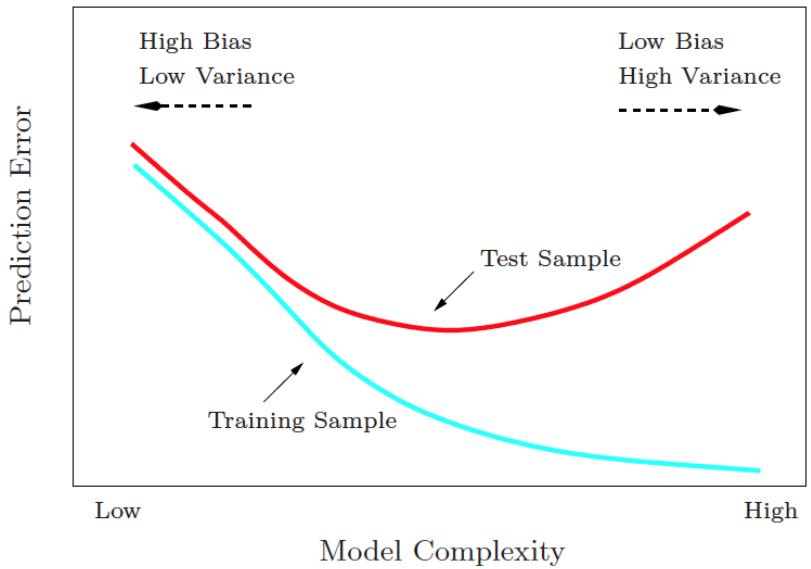
$$\begin{aligned} E_{\mathcal{S}}[E_Y[(Y - \hat{f}(x))^2]] &= VarY + E[Y]^2 + Var\hat{f}(x) + E[\hat{f}(x)]^2 - 2E[f(x) + \epsilon]E[\hat{f}(x)] \\ &= \sigma_\epsilon^2 + E[f(x) + \epsilon]^2 + E[\hat{f}(x)]^2 - 2E[f(x)]E[\hat{f}(x)] + Var\hat{f}(x) \\ &= \sigma_\epsilon^2 + E[\hat{f}(x) - f(x)]^2 + Var\hat{f}(x) \\ &= \sigma_\epsilon^2 + \text{Biais}^2 + \text{variance} \end{aligned}$$

terme incompressible : bruit des données
Biais au carré: mesure à quel point l'estimateur $\hat{f}$ est loin de la cible
Variance de $\hat{f}(x)$: mesure à quel point l'estimateur $\hat{f}(x)$ est sensible aux données

Biais variance



Biais variance tradeoff



Décomposition biais-variance des k-plus-proches voisins

Posons x. Supposons que l'aléa ne vient que des y. On peut montrer que:

$$E[(Y - \hat{f}(x))^2] = \sigma_\epsilon^2 + (f(x) - \frac{1}{K}\sum_{\ell=1}^{K} f(x_{(\ell)})^2 + \frac{\sigma_\epsilon}{K}$$

NB: $x_{(\ell)}$: ℓ-ième plus proche voisin
K contrôle le terme de variance : plus grande est la valeur de K, plus la variance décroit; mais K contrôle aussi le biais, plus petite est la valeur de K, plus petit est le biais : **dilemne biais-variance**.

Outline

Local Averaging and Histogram rules

For binary classification

- K-NN limitations: a nearest neighbor may be very far from X!
- Consider a **partition** of the feature space:

$$C_1 \bigcup \cdots \bigcup C_K = \mathcal{X}$$

- Apply the **majority rule**: suppose that X lies in C_k,
  1. Count the number of training examples with positive label lying in C_k
  2. If $\sum_{i:\, X_i \in C_k} \mathbb{I}\{Y_i = +1\} > \sum_{i:\, X_i \in C_k} \mathbb{I}\{Y_i = -1\}$, predict $Y = +1$. Otherwise predict $Y = -1$.
- This corresponds to the "plug-in" classifier $2\mathbb{I}\{\widehat{\eta}(x)\} - 1$, where

$$\widehat{\eta}(x) = \sum_{k=1}^{K} \mathbb{I}\{x \in C_k\} \frac{\sum_{i=1}^{n} \mathbb{I}\{Y_i = +1,\ X_i \in C_k\}}{\sum_{i=1}^{n} \mathbb{I}\{X_i \in C_k\}}$$

is the **Nadaraya-Watson estimator** of the posterior probability.

Kernel smoothing for classification 1

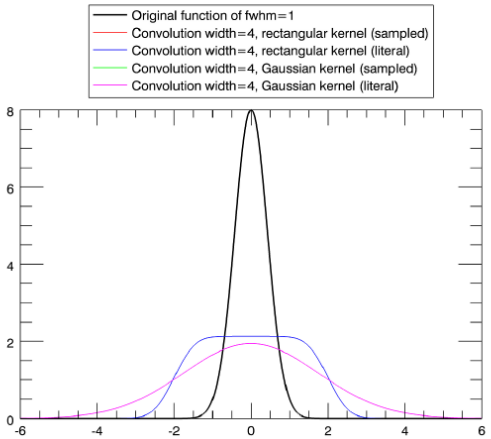
- Smooth the estimator/boundary decision!
- Replace the indicator function by a **convolution kernel**:

$$K : \mathbb{R}^d \to \mathbb{R}_+, \quad K \geq 0, \text{ symmetric and } \int K(x)dx = 1$$

- Bandwidth $h > 0$ and **rescaling**

$$K_h(x) = \frac{1}{h} K(x/h)$$

- Examples: Gaussian kernel, Novikov, Haar, *etc.*



Kernel smoothing for classification 2

- If $\sum_{i=1}^n \mathbb{I}\{Y_i = +1\} K_h(x - X_i) > \sum_{i=1}^n \mathbb{I}\{Y_i = -1\} K_h(x - X_i)$, predict $Y = +1$. Otherwise predict $Y = -1$.

- This corresponds to the "plug-in" classifier $2\mathbb{I}\{\widetilde{\eta}(x)\} - 1$, where

$$\widetilde{\eta}(x) = \frac{\sum_{i=1}^n \mathbb{I}\{Y_i = +1\} K_h(x - X_i)}{\sum_{i=1}^n K_h(x - X_i)}$$

  is the **Nadaraya-Watson estimator** of the posterior probability.

- **Statistical argument:** if η is a "smooth" function, $\widetilde{\eta}$ may be a better estimate than $\widehat{\eta}$ (smaller variance but... biased)

Reminder: k-nearest neighbour regression

$$\hat{h}(x) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(x)} y_i,$$

where $\mathcal{N}_k$ indicates the k closest points of $x_1, \ldots, x_n$ to x.
Limitations: the fitted function looks jagged, see why :

$$\hat{h}(x) = \sum_{i=1}^{n} w_i(x) y_i$$

where the weights are defined as:

$$w_i(x) = 1/k$$

if x_i is one fo the nearest neighbour of x ,0, otherwise.
Note that $w_i(x)$ as a function of x is discontinuous and so is $\hat{h}(x)$

Kernel Smoothing

As in kernel density estimation, kernel smoothing begins with a kernel function K, such that $\int K(x)dx = 1$, $\int xK(x)dx = 0$, $0 < \int x^2 K(x)dx < \infty$. Similarly to the classification case, we have (input dimensiond=1):

$$\hat{h}(x) = \frac{\sum_{i=1}^{n} K(\frac{x-x_i}{h}) y_i}{\sum_{i=1}^{n} K(\frac{x-x_i}{h})}$$

The kernel smoothing (Nadaraya-Watson estimator) is a smooth moving average of outputs.

A shortcoming: the kernel smoothing suffers from poor bias at the boundaries of the domain of the x_i's. One can replace y_i by polynomials.

Outline

Decision trees

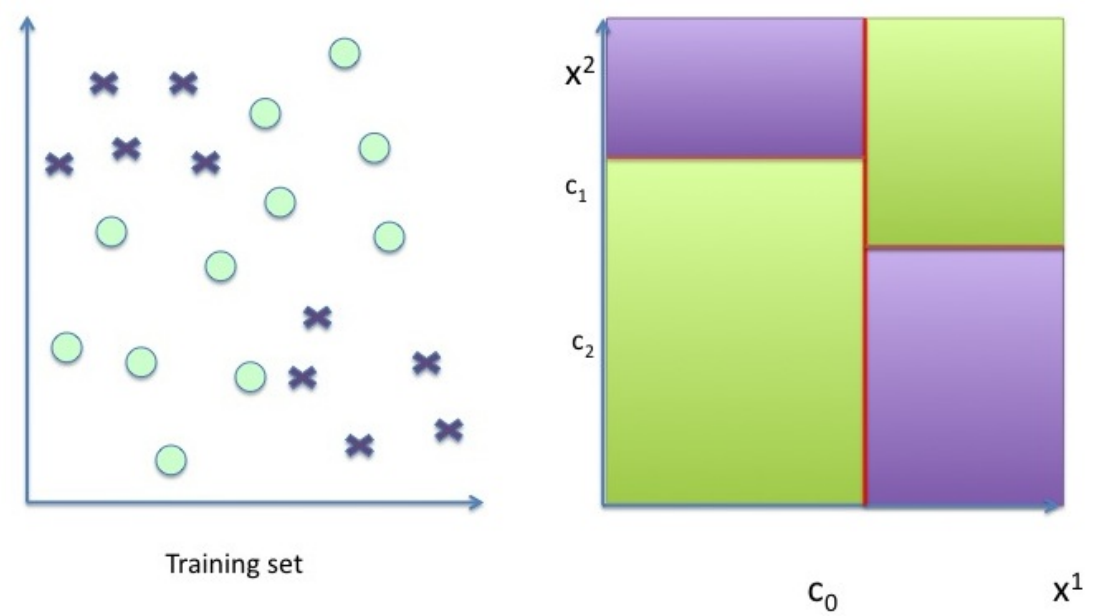
How to build automatically the partition $\mathcal{C}_1 \cup \ldots \mathcal{C}_m$ from the training set?

$$f(x) = \sum_{\ell=1}^{m} \mathbb{I}(x \in \mathcal{C}_\ell)(\arg\max_c [\sum_{i, x_i \in \mathcal{C}_\ell} \mathbb{I}(y_i = c)])$$

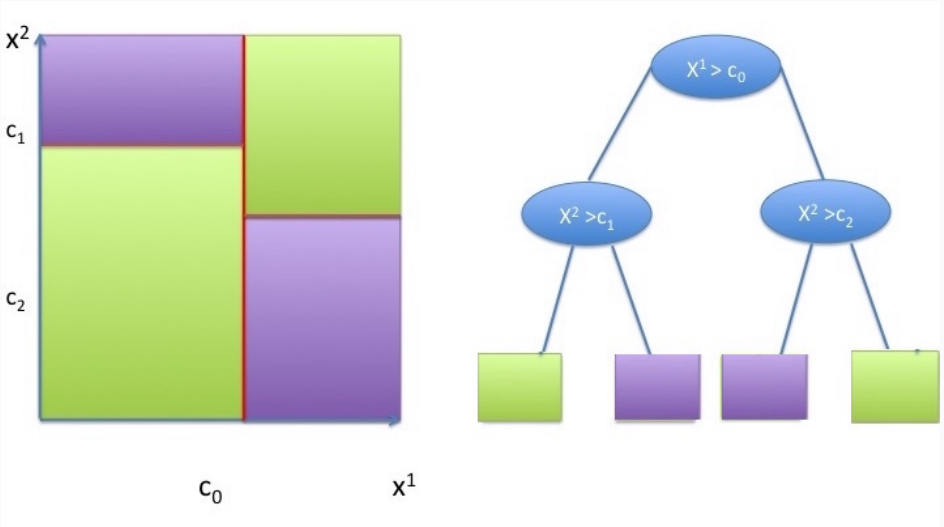
- Build a binary tree by choosing at each node, a splitting rule that splits the current dataset in two, minimizing a local criterion
- Greedy algorithm
- Each leaf of the tree corresponds to a subset of the partition: in the partition, take the **majority label**
- Works for binary classification as well as multi-class classification
- Works for Regression

Arbres de décision

Inventés quasi simultanément entre 1979 et 1983 par L. Breiman et col. (CART,Berkeley, USA) et R. Quinlan (ID3, Sydney, Australie) dans deux communautés complètement différentes: Breiman et col. en statistique, Quinlan dans une discipline nouvelle (Machine learning)

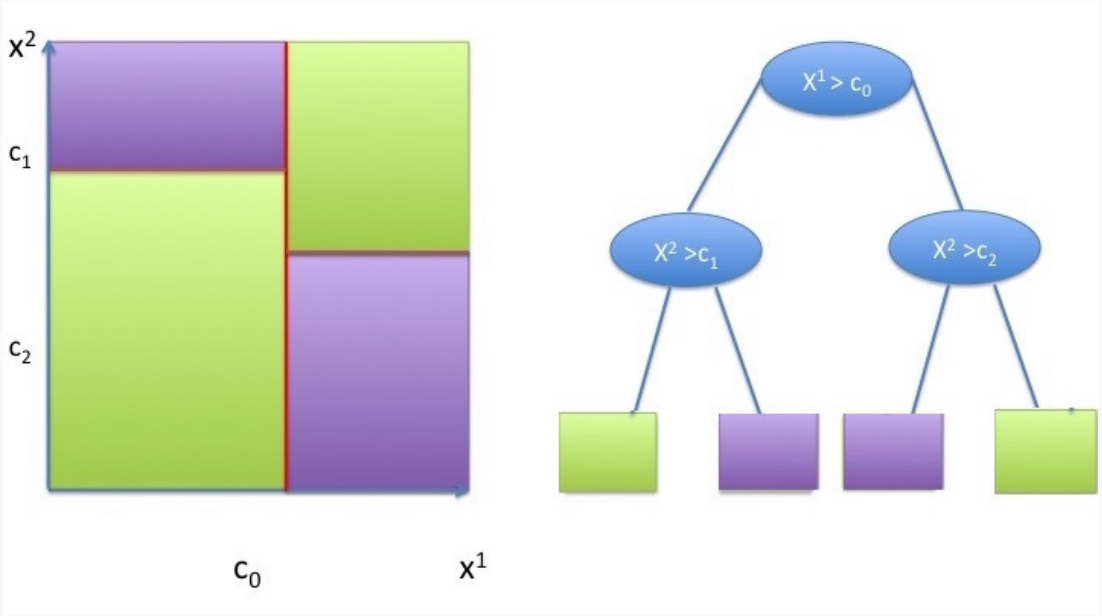


Arbres de décision 1



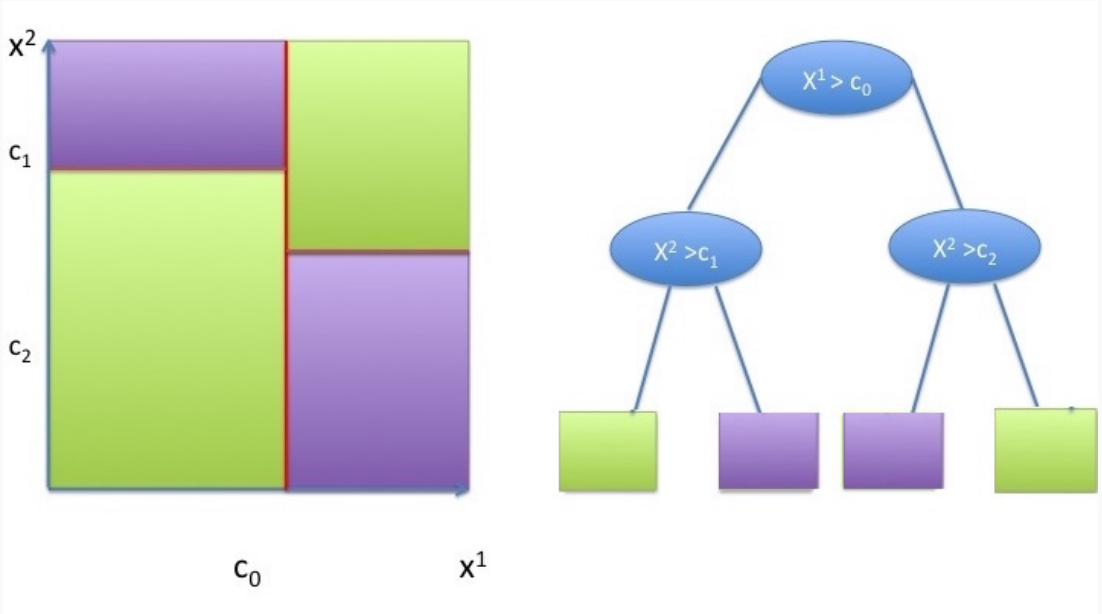
- Utiliser non pas 1 mais plusieurs séparateurs linéaires pour construire des frontières de décision non linéaires
- Utiliser des séparateurs linéaires orthogonaux à chaque vecteur de base, i.e. des hyperplans de la forme $x^j = \theta$ pour garder une interprétabilité de la fonction construite
- à l'issue de la phase d'apprentissage, on connaît les variables explicatives qui interviennent dans la fonction de décision construite

Arbres de décision 2



La fonction de décision peut être représentée par une structure d'arbre dont chaque noeud intermédiaire est associé à un hyperplan séparateur de la forme $x^j = c$ et chaque feuille est associée à une moyenne locale: une fonction constante, i.e. une classe.

Arbres de décision 2



A l'issue de la phase d'apprentissage, on connaît les variables explicatives qui interviennent dans la fonction de décision construite

L'arbre code pour un ensemble de règles logiques du type:

si $(x^{j_1} > c_{j_1})$ et $(x^{j_2} \leq c_{j_2})$ et ... alors x est de la classe k

Séparateur linéaire orthogonal à un vecteur de base

Variable x^j continue:

$$t_{j,c}(\mathbf{x}) = \text{signe}(x^j - c) \tag{1}$$

Remarque: on peut aussi traiter une variable x^j catégorielle à K valeurs $\{v_1^j, \ldots, v_K^j\}$:

$$t_{j,\mathbf{v},k}(\mathbf{x}) = 1(x^j = v_k^j) \tag{2}$$

Algorithme récursif de construction: arbre binaire

1. Soit $\mathcal{S}$ l'ensemble d'apprentissage
2. Construire un noeud racine
3. Chercher la meilleure séparation $t : \mathcal{X} \to \{0, 1\}$ à appliquer sur $\mathcal{S}$ telle que le coût local $L(t, \mathcal{S})$ soit minimal
4. Associer le séparateur choisi au noeud courant et séparer l'ensemble d'apprentissage courant $\mathcal{S}$ en $\mathcal{S}_d$ et $\mathcal{S}_g$ à l'aide de ce séparateur.
5. Construire un noeud fils à droite et un noeud à gauche.
6. Mesurer le critère d'arrêt à droite, s'il est vérifié, le noeud droit devient une feuille sinon aller en 3 avec $\mathcal{S}_d$ comme ensemble courant
7. Mesurer le critère d'arrêt à gauche, s'il est vérifié, le noeud gauche devient une feuille sinon aller en 3 avec $\mathcal{S}_g$ comme ensemble courant.

Fonction de coût locale

Soit un ensemble d'exemples d'apprentissage $\mathcal{S}$ et une fonction de séparation binaire $t_{j,\tau}$. Notons $\mathcal{D}(\mathcal{S}, j, \tau) = \{(\mathbf{x}, y) \in \mathcal{S}, t_{j,\tau}(\mathbf{x}) > 0\}$ et $\mathcal{G}(\mathcal{S}, j, \tau) = \{(\mathbf{x}, y) \in \mathcal{S}, t_{j,\tau}(\mathbf{x}) \leq 0\}$.

Parmi tous les paramètres $(j, \tau) \in \{1, \ldots, p\} \times \{\tau_1, \ldots, \tau_m\}$, on cherche $\hat{j}$ et $\hat{\tau}$ qui minimisent :

$$L(t_{j,\tau}, \mathcal{S}) = \frac{n_d}{n} H(\mathcal{D}(\mathcal{S}, j, \tau)) + \frac{n_g}{n} H(\mathcal{G}(\mathcal{S}, j, \tau)) \quad (3)$$

$$n_d = |\mathcal{D}(\mathcal{S}, j, \tau)| \quad (4)$$

$$n_g = |\mathcal{G}(\mathcal{S}, j, \tau)| \quad (5)$$

Fonction de coût locale pour la classification supervisée

On définit pour un ensemble $\mathcal{S}$ de n exemples étiquetés

$$p_c(\mathcal{S}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(y_i = c)$$

Voici les principaux critères H qui peuvent être utilisés:

Entropie croisée:

$$H(\mathcal{S}) = -\sum_{\ell=1}^{C} p_\ell(\mathcal{S}) \log p_\ell(\mathcal{S})$$

Critères de coût

Entropie croisée:

$$H(\mathcal{S}) = -\sum_{\ell=1}^{C} p_\ell(\mathcal{S}) \log p_\ell(\mathcal{S})$$

Index de Gini

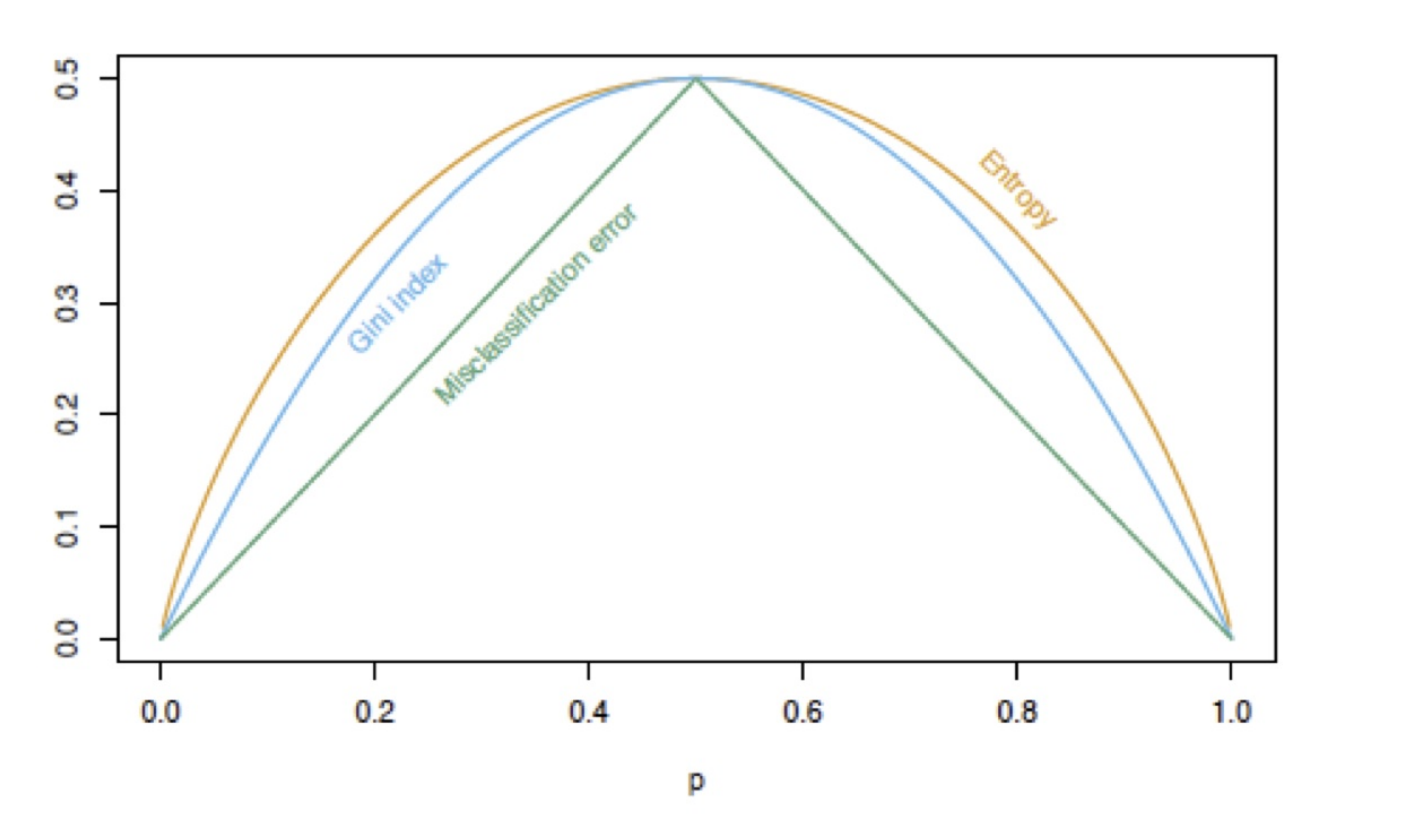
$$H(\mathcal{S}) = \sum_{\ell=1}^{C} p_\ell(\mathcal{S})(1 - p_\ell(\mathcal{S}))$$

Erreur de classification

$$H(\mathcal{S}) = 1 - p_{C(\mathcal{S})},$$

avec $C(\mathcal{S})$: classe majoritaire dans $\mathcal{S}$.

Visualisation des critères de coût



Génération des fonctions de séparation candidates

Pour chaque variable explicative (feature), x_j:

- $x_j \in [a_j, b_j]$: définir des seuils réguliers
- Calculer l'histogramme sur les données, prendre les seuils entre les modes

Critères d'arrêt

On s'arrête localement dès qu'on atteint un de ces critères (on parle de "early stopping")

- La profondeur maximale
- Le nombre maximale de feuilles
- Le nombre minimal d'exemples dans un noeud (pas assez d'exemples)

Sinon, l'ensemble d'apprentissage est appris jusqu'au bout : **sur-apprentissage** !

Model selection 1

A single hyperparameter among the following list has to be controlled

- Maximal depth
- Maximal number of nodes
- **Minimal number of examples per leaf**

How ? selection by $\rightarrow$ **cross-validation**.

Model selection 2

Use a validation set to prune the tree

Re-visit a decision tree learned from a training set without any depth limitation (full size):

- Only keep the branches that bring an improvement in performance on the validation set.

Advantages and drawbacks of Decision trees

Advantages

- Produces an interpretable nonlinear decision function,
- Consistency of decision trees (Scott, Nowak, 2004 - a review)
- Works for multiple classes without any pre-processing
- Efficient prediction stage : $O(\log L)$, L: number of leaves
- Works for continuous and categorial features

Advantages and drawbacks of Decision trees

Drawbacks

- Large variance estimator, instability : a small variation in the training set produces a very different tree $\rightarrow$ so, ensemble of trees are therefore very attractive
- No global optimization

Outline

Regression trees

Build a constant piece-wise function

Same construction principle except that the local criterion changes:

$$L(t_{j,\tau}, \mathcal{S}) = VAR_{emp}(\mathcal{S}) - \frac{n_d}{n} VAR_{emp}(\mathcal{D}(j, \tau, \mathcal{S})) - \frac{n_g}{n} VAR_{emp}(\mathcal{G}(j, \tau, \mathcal{S}))$$

Soit $\mathcal{S}$.

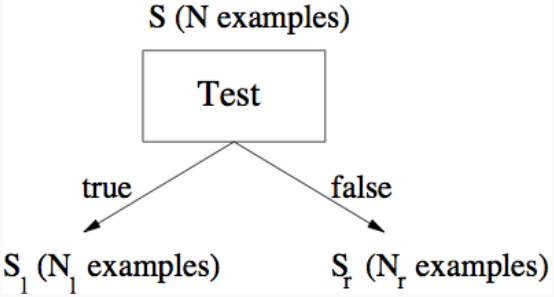
$$VAR_{emp}(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{(x_i, y_i) \in \mathcal{S}} (y_i - \bar{y})^2$$

On cherche à maximiser l'homogénéité des sorties.

Standard regression trees

- A learning algorithm that solves the regression problem ($\mathcal{Y} =$ and $\ell(y_1, y_2) = (y_1 - y_2)^2$) with tree structured models and provides a constant piece-wise approximation
- Basic idea of the learning procedure:
  - Recursively split the learning sample with tests based on the inputs trying to reduce as much as possible the variance of the output
  - Stop when the output is constant in the leaf (or some stopping criterion is met)

Standard regression trees



- The best split is the one that maximises the variance reduction:

$$\text{Score}_R(\mathit{Test}, S) = \text{var}\{y|S\} - \frac{N_l}{N}\text{var}\{y|S_l\} - \frac{N_r}{N}\text{var}\{y|S_r\},$$

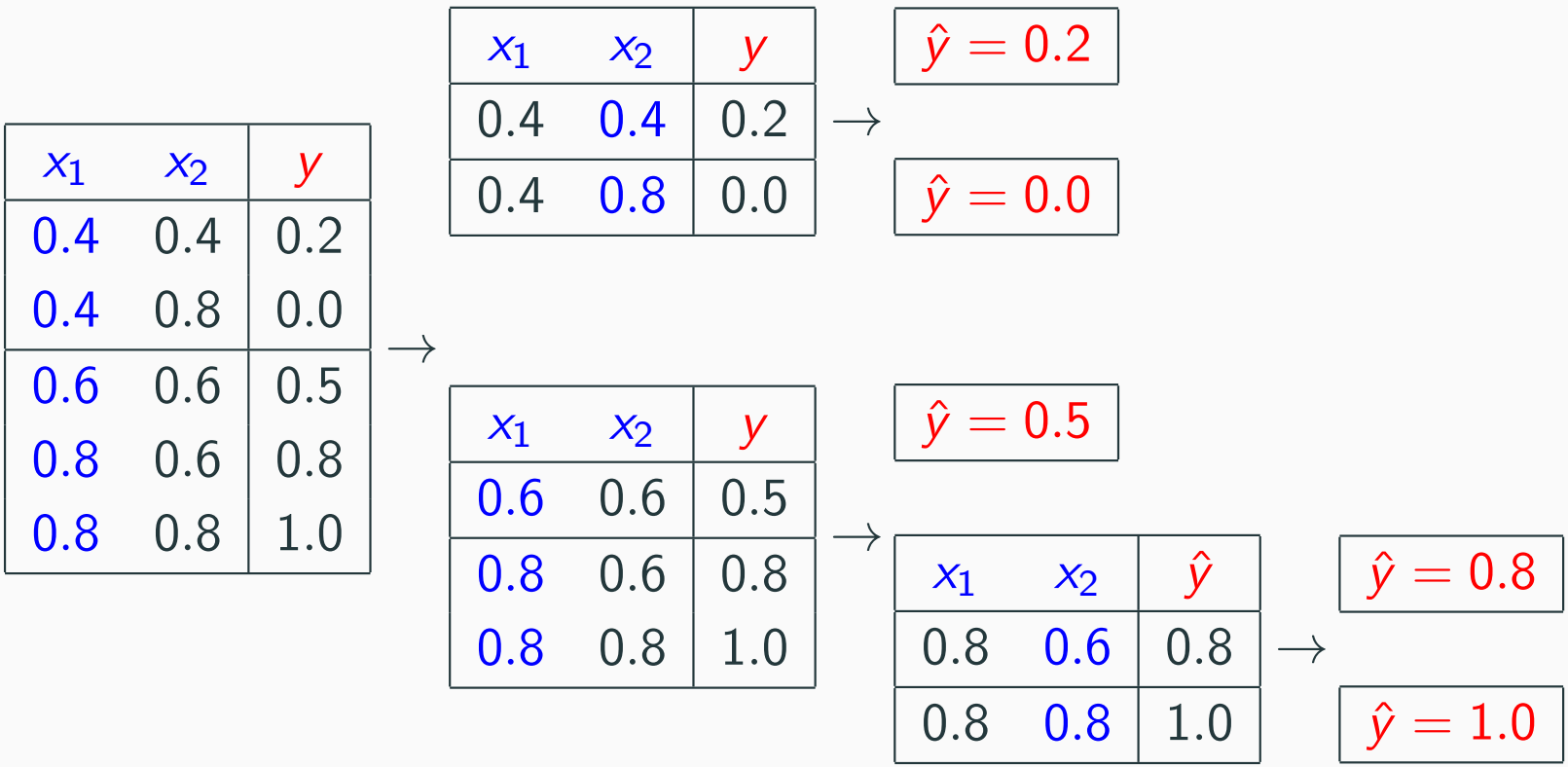
  where N is the size of S, N_l (resp. N_r) the size of S_l (resp. S_r), and $\text{var}\{Y|S\}$ denotes the variance of the output Y in the subset S:

$$\text{var}\{y|S\} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \frac{1}{N}\sum_{i=1} y_i)^2.$$

- Predictions at leaf nodes are computed as $\frac{1}{N_L}\sum_{i=1}^{N_L} y_i$ (N_L, the number of examples in the leaf).

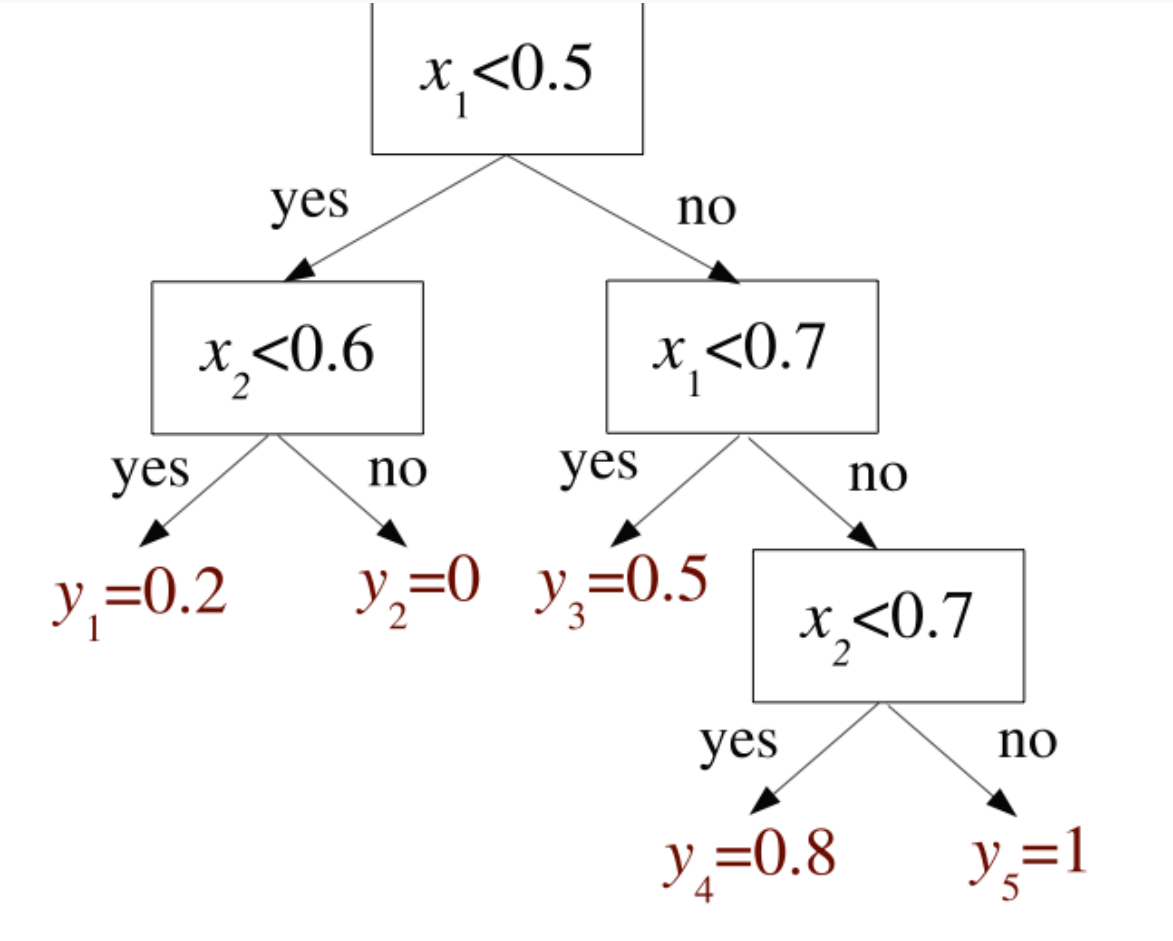
Illustration: tree growing=output variance reduction

(We show in blue the input variable that is used to split at each node)



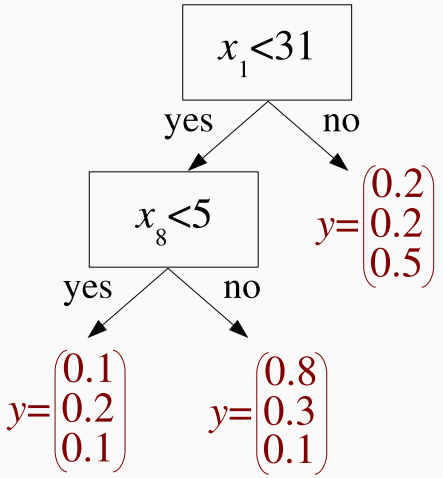
$\text{var}_{ls}\{y\}$=0.136 $\text{var}_{ls}\{y\}$=0.029 $\text{var}_{ls}\{y\}$=0.004 $\text{var}_{ls}\{y\}$=0

Illustration: resulting regression tree



Regression trees on multiple outputs

$\mathcal{Y} =^n$ and $\ell(y_1, y_2) = ||y_1 - y_2||^2$



- The variance becomes:

$$\text{var}\{y|S\} = \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{y}_i - \bar{\mathbf{y}}||^2 \text{ with } \bar{\mathbf{y}} = \frac{1}{N}\sum_{i=1} \mathbf{y}_i$$

  which is the average distance to the center of mass (or the average variance over all outputs).

- Predictions at leaf nodes become the centers of mass

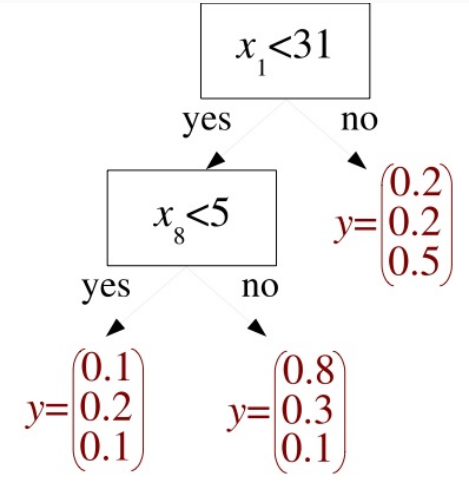
$$\frac{1}{N_L}\sum_{i=1}^{N_L} \mathbf{y}_i$$

Regression trees with multiple outputs

Model

$$h_{tree}(x) = \frac{1}{N_L} \sum_{i=1}^{N_L} \hat{y}_i . 1_i(t(x))$$

t is the tree indicator function that gives the number of the leaf where a data x falls. N_L the number of leaves



Regression trees: strengths and weaknesses

- Builds a constant piece-wise function
- Robustness to irrelevant attributes (to some extent)
- Good interpretability
- Very good computational efficiency and scalability
- Very high variance
  - The trees depend a lot on the random nature of the *ls*
  - *As a result, accuracy of tree based models is typically low*
  - *It is interesting to combine tree-based models in an ensemble*

References

- CART : Classification and Regression Trees, Breiman, Brieman, Olshen, Friedman and Stone, Wadsworth Statistics, 1984.
- Induction and Decision trees, Quinlan, Machine Learning Journal 1, 1986.
- Chapter Trees : The elements of statistical learning, Hastie, Tibshirani, Friedman, Springer.