Module 1: Introduction au Machine-Learning

Stéphan Clémençon,
stephan.clemencon@telecom-paristech.fr

Telecom Evolution, Paris, France

Basic algorithms for pattern recognition

Parametric logistic regression

- Explicit modelling of $\eta(x) = \mathbb{P}(Y = +1 \mid X = x) \in]0, 1[$
- **Logistic** transform: $f(x) = \operatorname{logit} \eta(x) = \log(\frac{\eta(x)}{1-\eta(x)})$
- Inverse transform: $\eta(x) = \frac{e^{f(x)}}{1+e^{f(x)}}$
- Assume $f \in \mathcal{F} = \{f_\theta(x);\ \theta \in \Theta\}$ with $\Theta \subset \mathbb{R}^d$

$$\eta_\theta(x) = \frac{e^{f_\theta(x)}}{1 + e^{f_\theta(x)}}$$

- Ex: **linear** logistic regression
  $f(x) = \alpha +^t \beta \cdot x, \quad \theta = (\alpha, \beta)$
- Maximize the **log-likelihood**

$$l_n(\theta) = \sum_{i=1}^{n} \left\{ \frac{1 + y_i}{2} \log(\eta_\theta(x_i)) + \frac{1 - y_i}{2} \log(1 - \eta_\theta(x_i)) \right\}$$

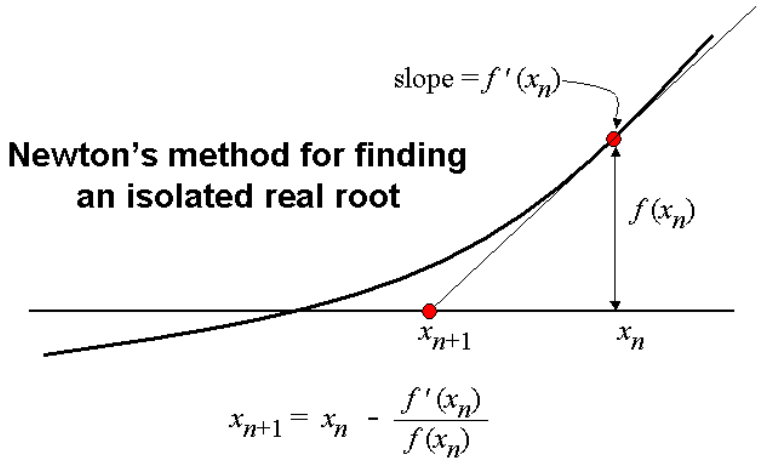
Parametric logistic regression

- Even in the additive model, the score equation

$$\nabla_\theta l_n(\theta) = 0$$

  cannot be solved explicitly!

- Implement Newton-Raphson method (gradient descent)

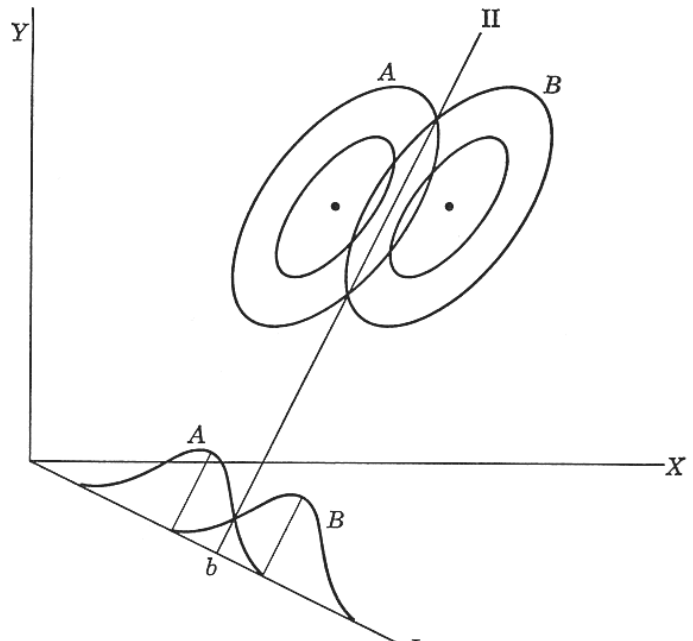


- Alternative to logit: probit model $\Phi^{-1}(\eta(X)) = \alpha +^t \beta X$

$$\text{with } \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp(-\frac{t^2}{2}) dt.$$

Parametric approach - Linear Discriminant Analysis

- Assume that the conditional distributions of X given $Y = +1$ and given $Y = -1$ are **Gaussian** with same covariance matrix Γ but different means μ_+ and μ_-. Let $p = \mathbb{P}\{Y = +1\}$.
- Estimate the moments of first and second orders, next the likelihood ratio and assign the likeliest labels



Parametric approach - Linear Discriminant Analysis

At point X, predict $Y = +1$ if

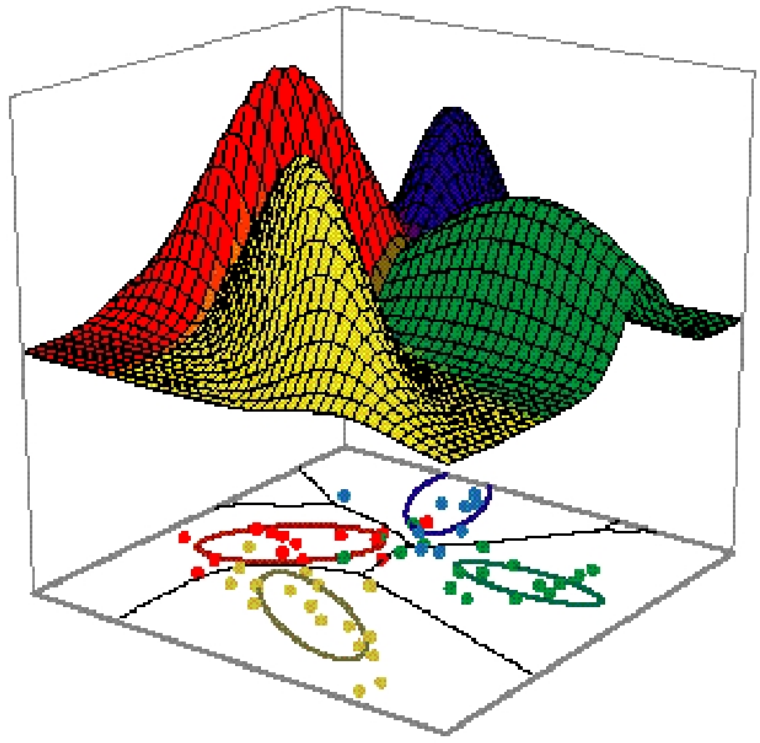
$$\log\left(\frac{\mathbb{P}\{Y = +1 \mid X\}}{\mathbb{P}\{Y = -1 \mid X\}}\right) > 0 \Leftrightarrow$$

$$\log(\frac{p}{1-p}) - \frac{1}{2}(\mu_+ - \mu_-)^t \Gamma^{-1}(\mu_+ - \mu_-) + x^t \Gamma^{-1}(\mu_+ - \mu_-) > 0$$

- Linear separator ($\neq$ linear logistic regression, except when $p = 1/2$)
- Replace μ_+, μ_- and Γ by empirical estimates

Linear Discriminant Analysis

- Naive Bayes: given Y, the input variables $X^{(1)}, \ldots, X^{(d)}$ are independent
- Nonlinear decision boundaries: quadratic discriminant analysis (QDA), Gaussian mixtures, kernels
- LDA can be easily extended to the multiclass framework



The (single-layer) perceptron algorithm

- The output Y is connected to the input X by

$$y = sign({}^t w \cdot X - \theta)$$

- The input space is separated into two regions by a **hyperplane**
- **Rosenblatt's algorithm (1962)** for minimizing

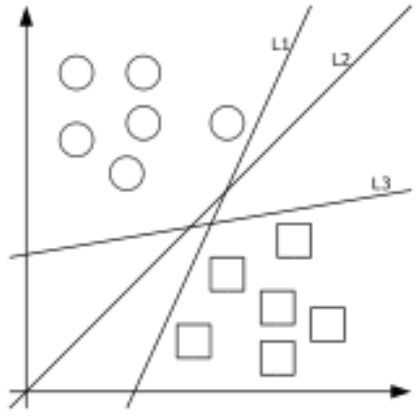
$$-\sum_i y_i({}^t w \cdot x_i + \theta)$$

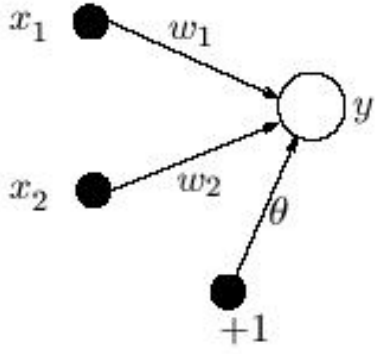
1. Choose at random (x_i, y_i) for "feeding" the perceptron
2. Gradient descent with rate ρ

$$\begin{pmatrix} w \\ \theta \end{pmatrix} \leftarrow \begin{pmatrix} w \\ \theta \end{pmatrix} + \rho \begin{pmatrix} y_i x_i \\ y_i \end{pmatrix}$$

3. Converges only when the data are **separable in a linear fashion**

The (single-layer) perceptron algorithm





A simplistic nonparametric method: K-nearest neighbours

- Let $K \geq 1$. On $\mathbb{R}^D$, consider a **metric** d (ex: euclidean distance)
- For any input value x, let $\sigma = \sigma_x$ be the permutation of $\{1, \ldots, n\}$ such that

$$d(x, x_{\sigma(1)}) \leq \ldots \leq d(x, x_{\sigma(n)})$$

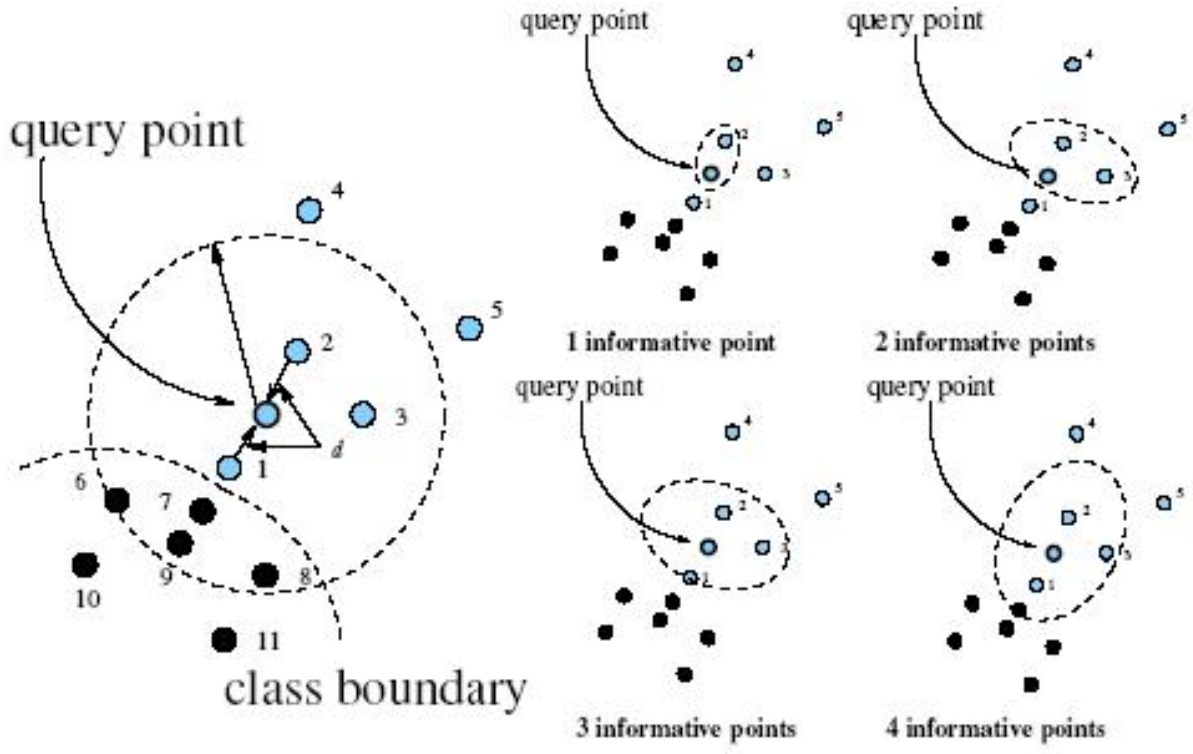
- Consider the K-**nearest neighbours**

$$\{x_{\sigma(1)}, \ldots, x_{\sigma(K)}\}$$

- **Majority vote:** $N_y = \text{Card}\{k \in \{1, ..., K\};\ y_{\sigma(k)} = y\}$, $y \in \{-1, 1\}$

$$C(x) = \arg \max_{y \in \{-1, +1\}} N_y,$$

A simplistic nonparametric method: K-nearest neighbours



K-nearest neighbours

Consistency (Stone '77)

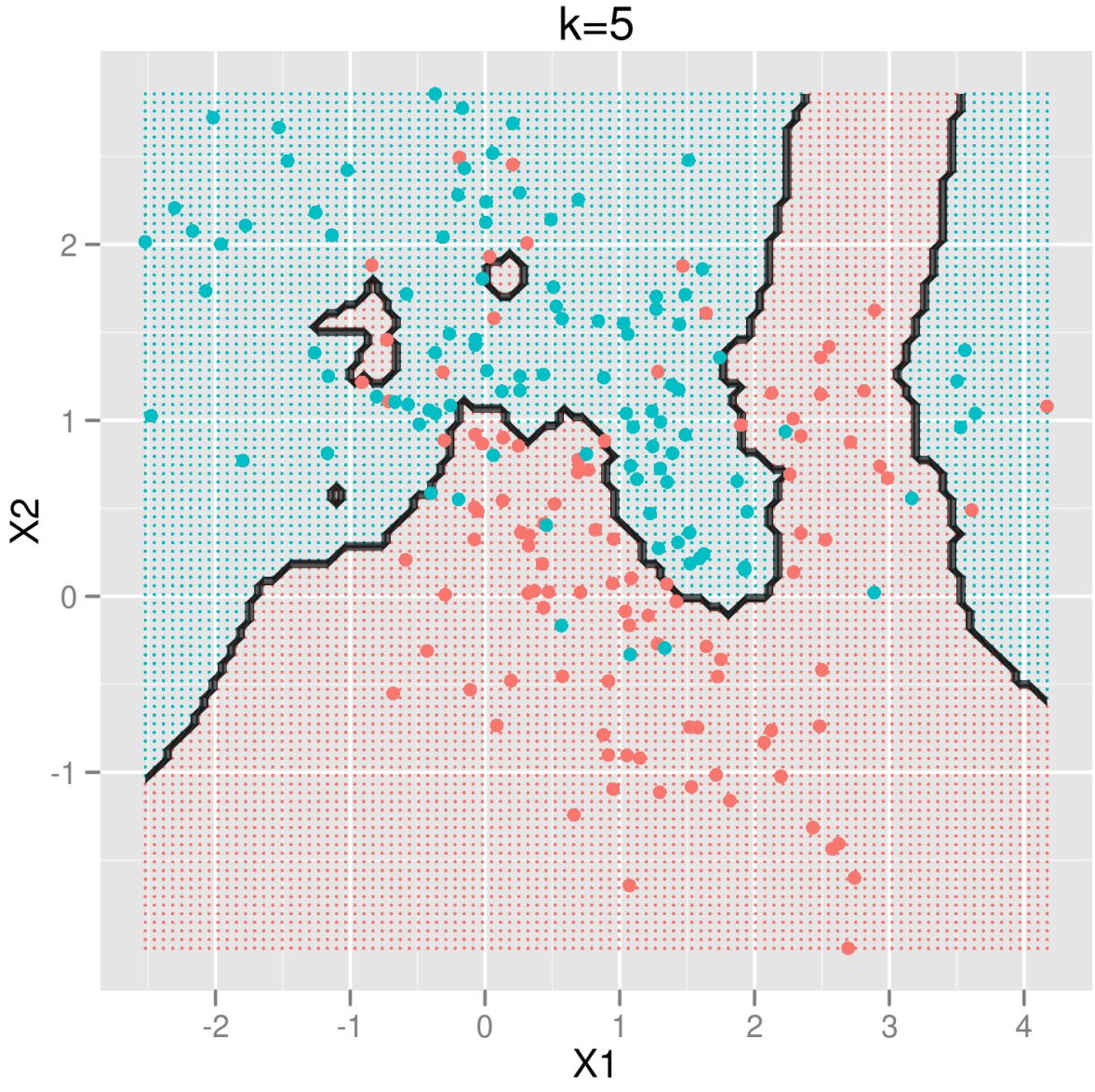
If $k = k_n \to \infty$ such that $k_n = o(n)$, then the K-NN rule is consistent

$$L(C_{K-NN}) - L^* \to 0, \quad as \quad n \to \infty$$

But...

- The rate can be arbitrarily slow
- Curse of dimensionality: sorting data is computationally expensive
- Instability: choice of K? metric D?
- **Metric learning** (*e.g.* Mahalanobis distance)
- Variants with *weights*

K-nearest neighbours - A too flexible method?



Histogram rules - Local averaging

- 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 rules - Local averaging

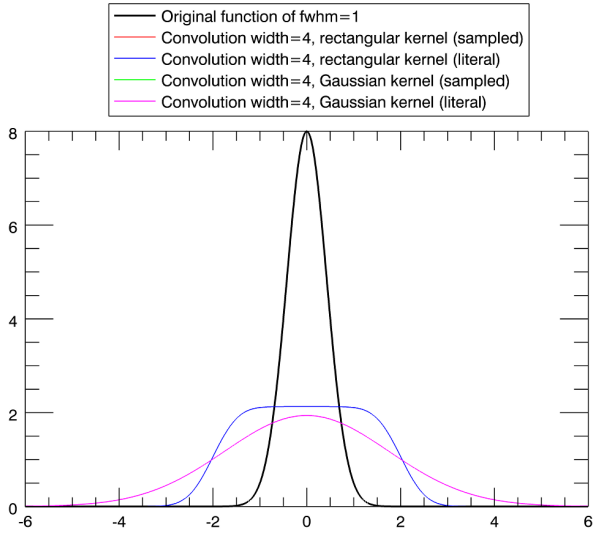
- Smooth the estimator/boundary decision!
- Replace the indicator function by a **convolution kernel**:

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

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

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

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



Kernel rules - Local averaging

- 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

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

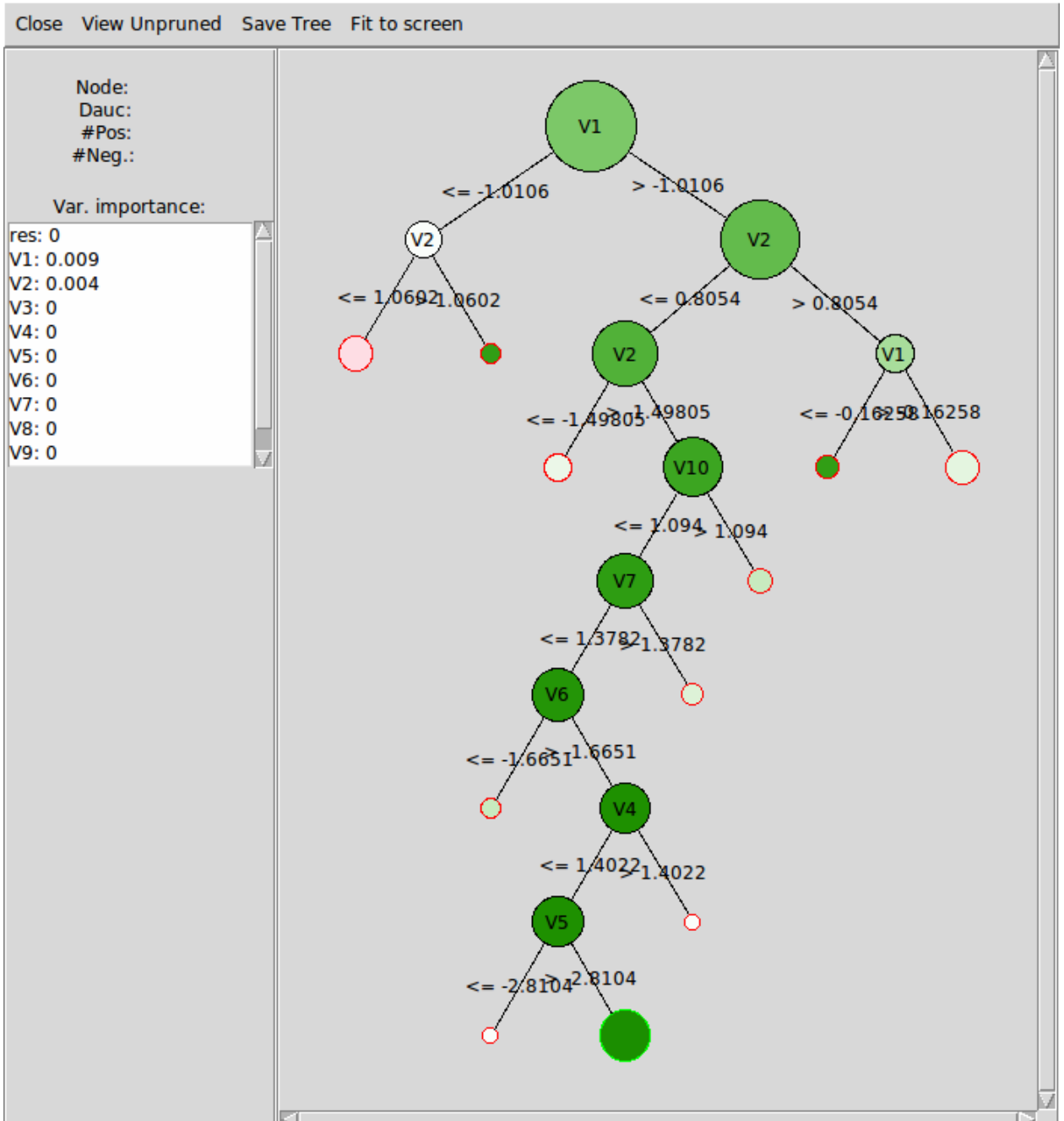
  is the **Nadaraya-Watson estimator** of the posterior probability.

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

Decision Trees: the CART Algorithm

- If the partition is picked in advance (before observing the data), many cells may be empty!
- Choose the partition **depending on the traning data!**
- The CART Book - Breiman, Friedman, Olshen & Stone (1986)
- **Greedy** Recursive Dyadic Partitioning:
  $X = (X^{(1)}, \ldots, X^{(d)}) \in \mathbb{R}^d$
- The algorithm will be explained in the next Machine-Learning Session

Decision Trees: the CART Algorithm

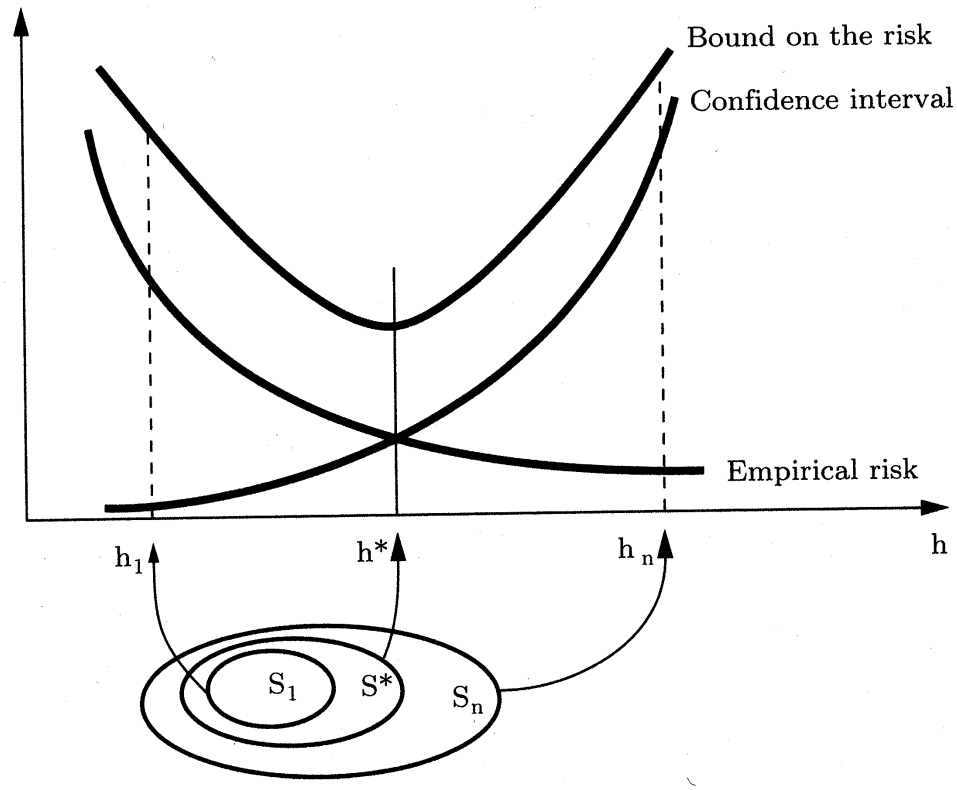


Model Assessment
Model Selection

Agenda

- Generalization ability
- Bias, variance and model complexity
- The "data-rich situation": Train-Validation-Test
- The training error: a too optimistic estimate
- Structural risk minimization (VC theory)
- Cross-validation: a popular method for prediction error estimation
- Bootstrap techniques

Looking for the right amount of complexity



Errors, training errors, generalization errors

- Learning is based on a training sample

$$\mathcal{D}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$$

- The classifier $\hat{C}_n \in \mathcal{G}$ selected through an "ERM like" method is **random**, depending on $\mathcal{D}_n$, as well as its **error**:

$$L(\hat{C}_n) = \mathbb{E}\left[\mathbb{I}\{Y \neq \hat{C}_n(X)\} \mid \mathcal{D}_n\right]$$

  Expectation is taken over a pair (X, Y) independent from training data $\mathcal{D}_n$

- The **generalization error**: take next expectation over $\mathcal{D}_n$

$$Err = \mathbb{E}\left[L(\hat{C}_n)\right]$$

Methods for performance assessment, for model selection

- Training error is not a good estimate!

$$\hat{L}_n(\hat{C}_n) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\{Y_i \neq C(X_i)\}$$

  It vanishes as soon as the class $\mathcal{G}$ is complex enough
  $\Rightarrow$ Overfitting and poor generalization

- The objective is twofold
  - Model selection: choose the best model among a collection of models
  - Model assessment: for a given model, estimate its generalization error

When data are not expensive

- Divide the data into three parts:

  Training - Validation - Test

- Typical choice: 50% - 25% - 25%

- $K \geq 1$ model candidates: $\mathcal{G}_1, \ldots, \mathcal{G}_K$

  - For each $k \in \{1, \ldots, K\}$, apply ERM to training data $\Rightarrow \hat{C}^{(k)}$
  - Use validation data to find the "best" $\hat{k} \in \{1, \ldots, K\}$
  - Estimate the error using the test data (independent from $\hat{k}$)

- How to proceed in a data-poor situation?

  Complexity regularization (structural risk minimiation),
  resampling methods, *etc.*

Model selection by penalization

- Consider a sequence of model classes $\mathcal{G}_1, \mathcal{G}_2, \ldots$
  As $k \nearrow +\infty$, $\mathcal{G}_k$ gets richer
- Let $\hat{C}^{(k)}$ be the empirical risk minimizer over $\mathcal{G}_k$
- Our goal: select $\hat{k}$ so that $\mathbb{E}[L(\hat{C}^{(\hat{k})})] - L^*$ is close to

$$\min_k \mathbb{E}[L(\hat{C}^{(k)})] - L^* =$$

$$\min_k \left\{ \left(\mathbb{E}[L(\hat{C}^{(k)})] - \inf_{C \in G_k} L(C) \right) + \left(\inf_{C \in G_k} L(C) - L^* \right) \right\}$$

- Idea: add a complexity penalty to the training error to compensate the overfitting effect

$$\hat{L}_n(\hat{C}^{(k)}) + pen(n, k)$$

- The penalty may depend on the data or not
- The penalty is related to a distribution-free upper bound for

$$\sup |\hat{L}_n(C) - L(C)|$$

Complexity regularization

- Suppose that an estimate $R_{n,k}$ of $L(\hat{C}_k)$ is available, s.t. for all $\epsilon > 0$

$$\mathbb{P}\left\{L(\hat{C}_k) - R_{n,k} > \epsilon\right\} \leq ce^{-2m\epsilon^2}$$

  for fixed constants c, m

- The ideal optimization would be

$$L(\hat{C}_k) - \hat{L}_n(\hat{C}_k)$$

  that can be estimated by

$$R_{n,k} - \hat{L}_n(\hat{C}_k)$$

- This yields $pen(n,k) = R_{n,k} - \hat{L}_n(\hat{C}_k) + \sqrt{\log(k)/m}$

Complexity regularization

- Select the prediction rule

$$C_n^* = \arg\min_k \tilde{L}_n(\hat{g}_k)$$

based on the complexity penalized training error

$$\tilde{L}_n(\hat{g}_k) = \hat{L}_n(\hat{g}_k) + pen(n,k) = R_{n,k} + \sqrt{\log(k)/m}$$

- Penalization by the VC dimension

$$R_{n,k} = \hat{L}_n(\hat{g}_k) + 2\sqrt{\frac{V_{\mathcal{G}_k} \log(n+1) + \log 2}{n}}$$

Cross-Validation

- Goal: estimate the generalization error
- Let $K \geq 1$ (typical choices are 5 or 10), "K-fold cross-validation"
  (K=n "leave-one-out" estimation)
- Split the data into K parts (of same size)
- For all $k \in \{1, \ldots, K\}$,
  - learn $\hat{C}^{(-k)}$ based on all data except the k-th part
  - calculate the error of $\hat{C}^{(-k)}$ over the k-th part
- Average the K quantities

"Pulling yourself up by your own bootstrap" (Baron de Münchausen)

- Bootstrap (the plug-in principle): estimate the distribution of

$$\mathbb{E}^*[\mathbb{I}\{\hat{C}(X) \neq Y\}]$$

  where $\mathbb{E}^*[.]$ is the expectation w.r.t. the empirical df of the $(X_i, Y_i)'s$

- Heuristics: replace the unknown df by an estimate
- Monte-Carlo approximation
- Higher-order validity