Supervised learning for computer vision: Theory and algorithms - Part I

Francis Bach\textsuperscript{1} & Jean-Yves Audibert\textsuperscript{2,1}

1. INRIA - Ecole Normale Supérieure
2. ENPC

ECCV Tutorial - Marseille, 2008
Outline

• Probabilistic model

• Local averaging algorithms
  – Link between binary classification and regression
  – $k$-Nearest Neighbors
  – Kernel estimate
  – Partitioning estimate

• Empirical risk minimization and variants
  – Neural networks
  – Convexification in binary classification
  – Support Vector Machines
  – Boosting
Probabilistic model

• Training data = \( n \) input-output pairs:

\[(X_1, Y_1), \ldots, (X_n, Y_n)\] i.i.d.

from some unknown distribution \( P \)

• A new input \( X \) comes.

• Goal: predict the corresponding output \( Y \).

• probabilistic assumption:

\[(X, Y) = \text{another independent realization of } P.\]
Some typical examples

- **Computer Vision**
  - object recognition
    \[ X = \text{an image} \]
    \[ Y = +1 \text{ if the image contains the object, } Y = 0 \text{ otherwise} \]

- **Textual document**
  - \[ X = \text{a mail} \]
    \[ Y = \text{spam vs non spam} \]

- **Insurance**
  - \[ X = \text{data of a future policy holder} \]
    \[ Y = \text{premium} \]

- **Finance**
  - \[ X = \text{data of a loanee} \]
    \[ Y = \text{loan rate} \]
  - \[ X = \text{data of a company} \]
    \[ Y = \text{buy or sell} \]
Measuring the quality of prediction (1/2)

- $\ell(y, \hat{y}) = \text{measure the loss incurred by predicting } \hat{y} \text{ while the true output is } y$

- Typical losses are:
  - the $p$-power loss for real outputs
    \[ \ell(y, \hat{y}) = |y - \hat{y}|^p \]
  - the classification loss for discrete outputs (e.g. in $\{0, 1\}$)
    \[ \ell(y, \hat{y}) = \mathbb{1}_{y \neq \hat{y}} \]
Measuring the quality of prediction (2/2)

- A prediction function = mapping from input space to output space

\[ f : X \mapsto f(X) \]

- Quality of a prediction function

Risk of \( f \) = \( R(f) = \mathbb{E} \ell[Y, f(X)] \)

- The best prediction function (=Bayes predictor):

\[ f^* = \arg\min_f R(f) \]
Noise in classification:

\[ R(f) = \mathbb{E} 1_{Y \neq f(X)} = \mathbb{P}(Y \neq f(X)) \]

\[ \Rightarrow R(f^*) = 0 \]

\[ \Rightarrow R(f^*) > 0 \]
Bayes predictors for typical losses

- In classification (when $\ell(y, \hat{y}) = 1_{y \neq \hat{y}}$):

$$f^*_{\text{cla}}(x) = \arg\max_y P(Y = y|X = x)$$

- In least square regression (when $\ell(y, \hat{y}) = (y - \hat{y})^2$)

$$f^*_{\text{reg}}(x) = \mathbb{E}(Y|X = x)$$
What is formally a supervised learning algorithm?

• An estimator of the unobservable $f^*$

• An algorithm = a training sample is mapped to a prediction function

$$\hat{f} : \mathcal{T} = \{(X_1, Y_1), \ldots , (X_n, Y_n)\} \mapsto \hat{f}_\mathcal{T}$$

• Quality of an algorithm for training samples of size $n$

$$\mathbb{E}_\mathcal{T} R(\hat{f}_\mathcal{T})$$

Here the expectation is wrt the training sample distribution.
Uniformly universal consistency

- An algorithm is **uniformly universally consistent** if we have

  \[
  \lim_{n \to \infty} \sup_P \{ \mathbb{E}_T R(\hat{f}_T) - R(f^*) \} = 0
  \]

- Bad news: uniformly universally consistent algorithms do not exist [Devroye (1982); Audibert (2008)]

- **Practical meaning:** you will never know beforehand how much data is required to reach a predefined accuracy
Universal consistency [Stone (1977)]

- An algorithm is *universally consistent* if for any $P$ generating the data, we have
  \[ \mathbb{E}_T R(\hat{f}_T) \xrightarrow{n \to +\infty} R(f^*) , \]
  in other words:
  \[ \sup_P \lim_n \left\{ \mathbb{E}_T R(\hat{f}_T) - R(f^*) \right\} = 0 \]

  ( ≠ unif. univ. consistency: $\lim_n \sup_P \left\{ \mathbb{E}_T R(\hat{f}_T) - R(f^*) \right\} = 0$)

- Good news: universally consistent algorithms do exist

- Practical meaning: for a sufficiently large amount of data, you will reach any desired accuracy
What should we expect from a good supervised learning algorithm?

- its universal consistency

\[ \lim_{n \to \infty} \sup_{P} \left\{ \mathbb{E}_T R(\hat{f}_T) - R(f^*) \right\} = 0 \]

- a locally uniform universal consistency

\[ \sup_{P \in \mathcal{P}} \left\{ \mathbb{E}_T R(\hat{f}_T) - R(f^*) \right\} \text{ goes to } 0 \text{ fast (typically in } 1/n^{\gamma}) \]

for \( \mathcal{P} \) a known class of distributions in which (we hope/know that) the unknown distribution \( P \) is.
Outline

- Probabilistic model

- Local averaging algorithms
  - Link between binary classification and regression
  - $k$-Nearest Neighbors
  - Kernel estimate
  - Partitioning estimate

- Empirical risk minimization and variants
  - Neural networks
  - Convexification in binary classification
  - Support Vector Machines
  - Boosting
Link between binary classification and regression

\[ Y \in \{0, 1\} \]

\[ f_{\text{reg}}^*(x) = P(Y = 1|X = x) \]

\[ \Rightarrow 1_{f_{\text{reg}}^*(x) \geq 1/2} = \arg\max_y P(Y = y|X = x) = f_{\text{cla}}^*(x) \]

**Theorem:**

\[ f_{\text{reg}}: \text{real-valued function defined on the input space} \]

\[ f_{\text{cla}} = 1_{f_{\text{reg}} \geq 1/2} \]

\[ R_{\text{cla}}(f_{\text{cla}}) - R_{\text{cla}}(f_{\text{cla}}^*) \leq 2 \sqrt{R_{\text{reg}}(f_{\text{reg}}) - R_{\text{reg}}(f_{\text{reg}}^*)} \]

**Corollary:**

\[ \hat{f}_{\text{reg}} \text{ universally consistent} \Rightarrow \hat{f}_{\text{cla}} = 1_{\hat{f}_{\text{reg}} \geq 1/2} \text{ universally consistent} \]
Local averaging methods [Györfi et al. (2004)]

Context: \( Y \in \mathbb{R} \) \( \ell(y, y') = (y - y')^2 \)

- **Recall**: \( f^*(x) = \mathbb{E}(Y | X = x) \) unknown but \((X_1, Y_1), \ldots, (X_n, Y_n)\) observed

- **Implementation**
  For an input \( x \), predict the average of the \( Y_i \) of the \( X_i \)'s close to \( x \)

\[
\hat{f} : x \mapsto \sum_{i=1}^{n} W_i(x) Y_i,
\]

with \( W_i(x) \) appropriate functions of \( x, n, X_1, \ldots, X_n \).
Stone’s Theorem [Stone (1977)]:
sufficient conditions for universal consistency

Assume that the weights $W_i$ satisfies for any distribution $P$

1. $\forall \varepsilon > 0 \ P\left\{ \left| \sum_{i=1}^{n} W_i(X) - 1 \right| > \varepsilon \right\} \xrightarrow{n \to +\infty} 0$

2. $\forall a > 0 \ \mathbb{E}\left\{ \sum_{i=1}^{n} |W_i(X)|1_{\|x_i-x\|>a} \right\} \xrightarrow{n \to +\infty} 0$

3. $\mathbb{E} \sum_{i=1}^{n} [W_i(X)]^2 \xrightarrow{n \to +\infty} 0$

4. + two technical assumptions

Then $\hat{f} : x \mapsto \sum_{i=1}^{n} W_i(x)Y_i$ is universally consistent
First example: the $k$-Nearest Neighbors

$W_i(x) = \begin{cases} \frac{1}{k} & \text{if } X_i \text{ belongs to the } k\text{-n.n. of } x \text{ among } X_1, \ldots, X_n \\ 0 & \text{otherwise} \end{cases}$

e.g. for $k = 1$: if $X_i$ N.N. of $x$, then $\hat{f}_1(x) = Y_i$. More generally:

$$\hat{f}_k(x) = \frac{1}{k} \sum_{j=1}^{k} Y_{ij}$$

Universal consistency [Stone (1977)]:

The $k_n$-N.N. is univ. consistent iff $k_n \rightarrow +\infty$ and $k_n/n \rightarrow 0$

• The nearest neighbor ($k = 1$) algorithm is not universally consistent [Cover and Hart (1967)].
Using k-N.N.

- Requires full storage of training points

- Naive implementation: $O(n)$

- Refined implementation using trees: $O(\log n)$ at test time (but $O(n \log n)$ for building the tree)  
  (http://www.cs.umd.edu/~mount/ANN/)

- How to choose $k$? answer: by cross-validation i.e. take the $k$ by minimizing the risk estimate of $\hat{f}_k$ by

$$
\frac{1}{n} \sum_{j=1}^{p} \sum_{(x,y) \in B_j} \left[ y - \hat{f}_k(\cup_{l \neq j} B_l)(x) \right]^2,
$$

where $B_1, ..., B_p$ is a partition of the training sample: 
$\{(X_1, Y_1), \ldots, (X_n, Y_n)\} = B_1 \sqcup \ldots \sqcup B_p$. 

18
Second example: kernel estimate [Nadaraya (1964); Watson (1964)]

\[ X \in \mathbb{R}^d \quad h > 0 \quad K : \mathbb{R}^d \to \mathbb{R} \]

\[ \hat{f}(x) = \sum_{i=1}^{n} \left( \frac{K(\frac{x-X_i}{h})}{\sum_{l=1}^{n} K(\frac{x-X_l}{h})} \right) Y_i \]

Universal consistency [Devroye and Wagner (1980); Spiegelman and Sacks (1980)]: Let \( B(0, u) \) be the Euclidean ball in \( \mathbb{R}^d \) of radius \( u > 0 \). If there are \( 0 < a \leq A \) and \( b > 0 \) s.t.

\[ \forall u \in \mathbb{R}^d \quad b \mathbf{1}_{B(0,a)} \leq K(u) \leq \mathbf{1}_{B(0,A)} \]

and if \( h_n \xrightarrow{n \to +\infty} 0 \) and \( nh_n^{d} \xrightarrow{n \to +\infty} +\infty \), then \( \hat{f} \) is universally consistent.
Partitioning estimate [Tukey (1947)]

\[ X \in [0, 1]^d = \mathcal{X}_1 \sqcup \cdots \sqcup \mathcal{X}_p \]

\[ \hat{f}(x) = \sum_{i=1}^{n} \left( \frac{1_{x_i \in \mathcal{X}_j(x)}}{\sum_{l=1}^{n} 1_{x_l \in \mathcal{X}_j(x)}} \right) Y_i, \]

with \( j(x) \) such that \( x \in \mathcal{X}_j(x) \)

Universal consistency [Györfi (1991)]:

Let \( \text{Diam}(\mathcal{X}_j) = \sup_{x_1, x_2 \in \mathcal{X}_j} \| x_1 - x_2 \| \). If \( p/n \to 0 \) and \( \max_j \text{Diam}(\mathcal{X}_j) \to 0 \) then \( \hat{f} \) is universally consistent

- Meaning for a regular grid of width \( h_n \): \( nh_n^d \to +\infty \) and \( h_n \to 0 \)
Using the partitioning estimate

- Fast and simple but ...

- border effects: “Mind the gap!”

- nonobvious choice of the partition

- Variants of the partitioning estimate: decision trees with partitions built from the training data ...
Outline

- Probabilistic model

- Local averaging algorithms
  - Link between binary classification and regression
  - $k$-Nearest Neighbors
  - Kernel estimate
  - Partitioning estimate

- Empirical risk minimization and variants
  - Neural networks
  - Convexification in binary classification
  - Support Vector Machines
  - Boosting
Empirical risk minimization

\[ R(f) = \mathbb{E} \ell(Y, f(X)) \quad \text{unobservable} \]

- **Empirical risk:** \( r(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, f(X_i)) \quad \text{observable} \)

- **Law of large numbers and central limit theorem:**
  \[
  r(f) \xrightarrow{\text{p.s.}}_{n \to +\infty} R(f) \\
  \sqrt{n} \left[ r(f) - R(f) \right] \xrightarrow{\mathcal{L}}_{n \to +\infty} \mathcal{N}(0, \text{Var} \ell[Y, f(X)])
  \]

- **Goal of learning:** predict as well as \( f^* = \arg\min_{f} R(f) \)

- a “natural” algorithm is therefore: \( \hat{f}_{\text{ERM}} \in \arg\min_{f} r(f) \)
Natural choice does not work

\[ \hat{f}_{\text{ERM}} \in \arg\min_{f} r(f) \]

- There is an infinity of minimizers. Most of them will not perform well on test data. \(\rightarrow\) Overfitting
Why is it not working?

\[ R(\hat{f}_{\text{ERM}}) - R(f^*) = R(\hat{f}_{\text{ERM}}) - r(\hat{f}_{\text{ERM}}) + r(\hat{f}_{\text{ERM}}) - r(f^*) \]

\[ + r(f^*) - R(f^*) \]

\[ \leq \sup_{f} \{ R(f) - r(f) \} + 0 + O(1/\sqrt{n}) \]

\[ \forall f, R(f) - r(f) = O(1/\sqrt{n}) \neq \sup_{f} \{ R(f) - r(f) \} \xrightarrow{n \to +\infty} 0 \]
\[
\hat{f}_{\text{ERM}} \in \arg\min_{f \in \mathcal{F}} r(f) \text{ with } \mathcal{F} \text{ appropriately chosen}
\]

**Choice of \( \mathcal{F} \)?** Introduce \( \tilde{f} \in \arg\min_{f \in \mathcal{F}} R(f) \),

\[
R(\hat{f}_{\text{ERM}}) - R(f^*) = R(\hat{f}_{\text{ERM}}) - R(\tilde{f}) + R(\tilde{f}) - R(f^*)
\]

\(\underline{\text{Estimation error}}\) \(\underline{\text{Approximation error}}\)

\[
R(\hat{f}_{\text{ERM}}) - R(\tilde{f}) = R(\hat{f}_{\text{ERM}}) - r(\hat{f}_{\text{ERM}}) + r(\hat{f}_{\text{ERM}}) - r(\tilde{f})
\]

\[
+ r(\tilde{f}) - R(\tilde{f})
\]

\[
\leq \sup_{f \in \mathcal{F}} \{ R(f) - r(f) \} + 0 + O(1/\sqrt{n})
\]

**\( \mathcal{F} \) should be small enough to ensure** \( \sup_{f \in \mathcal{F}} \{ R(f) - r(f) \} \xrightarrow{n \to +\infty} 0 \)

**\( \mathcal{F} \) should be large enough to ensure** \( R(\tilde{f}) - R(f^*) \xrightarrow{n \to +\infty} 0 \)
First example: “neural networks” [Rosenblatt (1958, 1962)]

- **Squashing function** $\sigma$: a nondecreasing function with

\[
\sigma(x) \xrightarrow{x \to -\infty} 0 \quad \sigma(x) \xrightarrow{x \to +\infty} 1
\]

- e.g. $\sigma(x) = \mathbb{1}_{x \geq 0}$ or $\sigma(x) = 1/(1 + e^{-x})$

- **(Artificial) neuron**: function defined on $\mathbb{R}^d$ by

\[
g(x) = \sigma \left( \sum_{j=1}^{d} a_j x^{(j)} + a_0 \right) = \sigma(a \cdot \tilde{x})
\]

where $a = (a_0, \ldots, a_d)^T$ and $\tilde{x} = (1, x^{(1)}, \ldots, x^{(d)})^T$
Neural network with one hidden layer: function defined on $\mathbb{R}^d$ by

$$f(x) = \sum_{j=1}^{k} c_j \sigma(a_j \cdot \tilde{x}) + c_0$$

where $\tilde{x} = \begin{pmatrix} 1 \\ x \end{pmatrix} = (1, x^{(1)}, \ldots, x^{(d)})^T$. 

![Diagram of a neural network with one hidden layer](image)
Universal consistency in least square setting [Lugosi and Zeger (1995); Faragó and Lugosi (1993)]:

\((k_n)\) integer sequence

\((\beta_n)\) real sequence

\(\mathcal{F}_n = \{ \text{n. n. with one hidden layer, } k \leq k_n \text{ and } \sum_{j=0}^{k} |c_j| \leq \beta_n \} \)

ERM on \(\mathcal{F}_n\) is universally consistent if \(k_n \to +\infty, \beta_n \to +\infty\) and

\[
\frac{k_n \beta_n^4 \log(k_n \beta_n^2)}{n} \xrightarrow{n \to +\infty} 0.
\]
Using neural networks

- In practice: use of multilayer neural nets

- Squashing function $\Rightarrow$ ERM = nonconvex optimization pb
  $\Rightarrow$ any algorithm will end in a local minimum

- With good intuitions on how to build the neural nets and good heuristics to perform the minimization [LeCun et al. (1998); LeCun (2005); Simard et al. (2003)], neural nets are great...
Convexification of empirical risk minimization in binary classification

\[ Y \in \{-1; +1\} \quad R(g) = \mathbb{P}[Y \neq g(X)] \]

- ERM: \( \hat{g} \in \arg\min_{g \in G} \sum_{i=1}^{n} \mathbb{1}_{Y_i \neq g(X_i)} \rightarrow \text{highly nonconvex} \)

- \( f \) real-valued function and \( g : x \mapsto \text{sign}[f(x)] \)

\[ \rightarrow \quad \hat{f} \in \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \mathbb{1}_{Y_i f(X_i) \leq 0} \quad \mathcal{F} \text{ convex} \]

\[ \rightarrow \quad \hat{f} \in \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \phi[Y_i f(X_i)] \quad \phi \text{ convex} \]
Criterion to choose the convex function $\phi$

- $\phi$-risk of $f$: $A(f) = \mathbb{E}\phi[Yf(X)]$.

- $\phi$ should satisfy:
  
  $\hat{f}$ univ. consistent for the $\phi$-risk 
  
  $\Rightarrow$ sign$(\hat{f})$ univ. consistent for the classification risk

- Necessary and sufficient cond. [Bartlett et al. (2006)]:
  
  $\phi$ is differentiable at 0 and $\phi'(0) < 0$
Some convex functions useful for classification and their remarkable property

- $f^*$ best function for the $\phi$-risk
- $f$ a real-valued function

- $\phi(u) = (1 - u)_+ = \max(1 - u, 0)$: S.V.M. loss
  $$R[\text{sign}(f)] - R(g^*) \leq A(f) - A(f^*)$$

- $\phi(u) = e^{-u}$: AdaBoost loss
  $$R[\text{sign}(f)] - R(g^*) \leq \sqrt{2} \sqrt{A(f) - A(f^*)}$$

- $\phi(u) = \log(1 + e^{-u})$: Logistic regression loss
  $$R[\text{sign}(f)] - R(g^*) \leq \sqrt{2} \sqrt{A(f) - A(f^*)}$$

- $\phi(u) = (1 - u)^2$: Least square regression loss
  $$R[\text{sign}(f)] - R(g^*) \leq \sqrt{A(f) - A(f^*)}$$
Support Vector Machines [Boser et al. (1992); Vapnik (1995)]

\[ C > 0 \quad \phi(u) = (1 - u)_+ \quad \mathcal{H} \text{ a Reproducing Kernel Hilbert Space} \]

\[
\inf_{b \in \mathbb{R}, h \in \mathcal{H}} C \sum_{i=1}^{n} \phi(Y_i[h(X_i) + b]) + \frac{1}{2}\|h\|_{\mathcal{H}}^2 \quad (P_C)
\]

- **Empirical \( \phi \)-risk minim. on \( \mathcal{F} = \{x \mapsto h(x) + b; \|h\|_{\mathcal{H}} \leq \lambda, b \in \mathbb{R}\} \)

\[
\inf_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \phi(Y_if(X_i)) \quad (Q_\lambda)
\]

- \((\hat{h}_C, \hat{b}_C)\) solution of \((P_C) \Rightarrow \hat{h}_C + \hat{b}_C \text{ sol. of } (Q_\lambda) \text{ for } \lambda = \|\hat{h}_C\|_{\mathcal{H}}

\[ \rightarrow \text{ SVM: } x \mapsto \text{sign}(\hat{h}_C(x) + \hat{b}_C) \approx \text{empirical } \phi \text{-risk minim. on } \mathcal{F} \]
Reproducing Kernel pre-Hilbert Space

- Let $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ symmetric (i.e. $K(u, v) = K(v, u)$) and positive semi-definite (i.e. $\forall J \in \mathbb{N}, \forall \alpha \in \mathbb{R}^J$ and $\forall x_1, \ldots, x_J$
  $\sum_{1 \leq j, k \leq J} \alpha_j \alpha_k K(x_j, x_k) \geq 0$)

  - $K$ is called a (Mercer) kernel

  - Examples: $\mathcal{X} = \mathbb{R}^d$
    * linear kernel $K(x, x') = \langle x, x' \rangle_{\mathbb{R}^d}$
    * polynomial kernel $K(x, x') = (1 + \langle x, x' \rangle_{\mathbb{R}^d})^p$ for $p \in \mathbb{N}^*$
    * gaussian kernel $K(x, x') = e^{-\|x - x'\|^2/(2\sigma^2)}$ for $\sigma > 0$.

- Let $\mathcal{H}'$ be the linear span of $K(x, \cdot) : x' \mapsto K(x, x')$, equipped with

  $$\left\langle \sum_{1 \leq i \leq I} \alpha_i K(x_i, \cdot), \sum_{1 \leq j \leq J} \alpha'_j K(x'_j, \cdot) \right\rangle_{\mathcal{H}'} = \sum_{i, j} \alpha_i \alpha'_j K(x_i, x'_j)$$
Reproducing Kernel Hilbert Space

- the closure $\mathcal{H}$ of $\mathcal{H}'$ is an Hilbert space
- $\mathcal{H}$ (as $\mathcal{H}'$) has the reproducing property:

$$\langle f, k(x, \cdot) \rangle_\mathcal{H} = f(x) \quad \text{for any } f \in \mathcal{H}$$

Back to S.V.M.: Training set: $(X_1, Y_1), \ldots, (X_n, Y_n)$

Let $\mathcal{H}_n = \{ \sum_{i=1}^{n} \alpha_i K(X_i, \cdot); \forall i, \alpha_i \in \mathbb{R} \} \subset \mathcal{H}$

S.V.M. pb = \(\min_{b \in \mathbb{R}, h \in \mathcal{H}_n} C \sum_{i=1}^{n} \phi(Y_i[h(X_i) + b]) + \frac{1}{2}\|h\|_{\mathcal{H}}^2\)

$\Rightarrow$ tractable $(n + 1)$-dimensional optimization task
Universal consistency and using S.V.M.

- **Universal consistency [Steinwart (2002)]:**
  
  \[ X \in [0; 1]^d \quad \sigma > 0 \]

  The S.V.M. with gaussian kernel \( K : (x, x') \mapsto e^{-\|x-x'\|^2/(2\sigma^2)} \) and parameter \( C = n^{\beta-1} \) with \( 0 < \beta < 1/d \) is universally consistent.

- **Practical choices:**
  - kernel: linear, polynomial, gaussian, ...
  - \( C \) (and parameters of the kernel) cross-validated

- **Choice of the kernel \( \leftrightarrow \) functions approximated by linear combinations of the functions \( K(x, \cdot) : x' \mapsto K(x, x') \)**

- **gaussian kernel with \( \sigma \to +\infty = \) linear kernel !**
Boosting methods

- Let $G$ be a set of functions from $\mathcal{X}$ to $\{-1, +1\}$

1. $G = \{ x \mapsto \text{sign}(x^{(j)} - \tau); j \in \{1, \ldots, d\}, \tau \in \mathbb{R} \}$
   \quad $\cup \{ x \mapsto \text{sign}(-x^{(j)} + \tau); j \in \{1, \ldots, d\}, \tau \in \mathbb{R} \}$

2. $G = \{ x \mapsto 1_{x \in A} - 1_{x \in A^c}; A \text{ hyper-rectangle of } \mathbb{R}^d \}$
   \quad $\cup \{ x \mapsto 1_{x \in A^c} - 1_{x \in A}; A \text{ hyper-rectangle of } \mathbb{R}^d \}$
3. \( \mathcal{G} = \{ x \mapsto 1_{x \in H} - 1_{x \in H^c}; H \text{ halfspace of } \mathbb{R}^d \} \)

4. \( \mathcal{G} = \{ \text{ univariate decision trees with number of leaves } = d + 1 \} \)

- Boosting looks for classification function of the form

\[
x \mapsto \text{sign}\left( \sum_{j=1}^{m} \lambda_j g_j(x) \right)
\]

- Question: choice of \( \lambda_j \) and \( g_j \)?
Boosting by $L_1$-regularization

- Let $\mathcal{F}_\lambda = \left\{ \sum_{j=1}^{m} \lambda_j g_j ; \ m \in \mathbb{N}, \lambda_j \geq 0, g_j \in \mathcal{G}, \sum_{j=1}^{m} \lambda_j = \lambda \right\}$

- $\phi(u) = e^u \quad A_n(f) = \frac{1}{n} \sum_{i=1}^{n} \phi[Y_i f(X_i)]$

- Boosting by $L_1$-regularization:
  $$\hat{f}_\lambda = \operatorname{argmin}_{f \in \mathcal{F}_\lambda} A_n(f)$$

- **Universal consistency [Lugosi and Vayatis (2004)]:**
  If $\lambda = (\log n)/4$ and $\mathcal{G}$ is one of the previous choice (except choice 1), then $\operatorname{sign}(\hat{f}_\lambda)$ is universally consistent
Usual description of AdaBoost

- **Initialisation:** $w_i = 1/n$ for $i = 1, \ldots, n$

- **Iterate:** For $j = 1$ to $J$:
  - Take
    $$g_j \in \arg \min_{g \in G} \frac{1}{n} \sum_{i=1}^{n} w_i 1_{g(X_i) \neq Y_i}$$
    and $e_j$ the minimum value
  - $\lambda_j = \frac{1}{2} \log \left( \frac{1-e_j}{e_j} \right)$.
  - Update weights: for all $i$ s.t. $g_j(X_i) \neq Y_i$, $w_i \leftarrow w_i \frac{1-e_j}{e_j}$.
  - Normalize the weights: $w_i \leftarrow w_i / \sum_{i'=1}^{n} w_{i'}$ for $i = 1, \ldots, n$

- **Output:**
  $$x \mapsto \text{sign} \left( \sum_{j=1}^{J} \lambda_j g_j(x) \right)$$
AdaBoost = greedy empirical \( \phi \)-risk minimization

- \( \phi(u) = e^u \)
  
  \[ A_n(f) = \frac{1}{n} \sum_{i=1}^{n} \phi[Y_i f(X_i)] \]

- \( f_0 = 0 \)

- **For** \( j = 1 \) **to** \( J \)
  
  \[ (\lambda_j, g_j) \in \arg\min_{\lambda \in \mathbb{R}, g \in G} A_n(f_{j-1} + \lambda g) \]
  
  \[ f_j = f_{j-1} + \lambda_j g_j \]

- **Universal consistency** [Bartlett and Traskin (2007)]:
  
  If \( J = n^\nu \) with \( 0 < \nu < 1 \) and \( G \) is one of the previous choice (except choice 1), then AdaBoost is universally consistent
Link between boosting methods and S.V.M.

- AdaBoost output: \( x \mapsto \text{sign}\left(\sum_{j=1}^{J} \lambda_j g_j(x)\right) \)

- S.V.M. output: \( x \mapsto \text{sign}\left(\sum_{i=1}^{n} \alpha_i K(X_i, x) + b\right) \)

- Consider \( K(x, x') = \sum_{j=1}^{J} g_j(x)g_j(x') \). Then

\[
\sum_{i=1}^{n} \alpha_i K(X_i, x) = \sum_{i=1}^{n} \alpha_i \sum_{j=1}^{J} g_j(X_i)g_j(x) = \sum_{j=1}^{J} \lambda_j g_j(x)
\]

with

\[
\lambda_j = \sum_{i=1}^{n} \alpha_i g_j(X_i)
\]
Boosting vs S.V.M. vs Neural networks

- **Boosting advantages:**
  - Variable selection
  - Ability to handle very large amount of features
  - Simple tricks to reduce computational complexity
  - S.V.M. can be run at the end on the selected features

- **S.V.M. advantages:**
  - Easy to use off-the-shelf
  - Consistently good results

- **Neural networks advantages:**
  - Works well in practice


