



école —————
normale —————
supérieure —————
paris-saclay —————

université
PARIS-SACLAY

Introduction to Statistical Learning

Nicolas Vayatis

Lecture # 5 - Statistical analysis of mainstream ML algorithms

Part II - Analysis of Ensemble Methods and Boosting

Reminder on Part I

Kernel methods applied to classification
(*aka* Support Vector Machines)

Algorithmic Principles of SVM

- Search space : soft classifiers in an RKHS (kernel k)

$$\mathcal{H}(X) \doteq \left\{ h = \sum_{i=1}^n \alpha_i k(X_i, \cdot) : \alpha_1, \dots, \alpha_n \in \mathbb{R} \right\}$$

- Regularized optimization problem : set $\lambda > 0$

$$\hat{h}_\lambda = \arg \min_{\mathcal{H}_k} \left\{ \sum_{i=1}^n (1 - Y_i h(X_i))_+ + \lambda \|h\|_k \right\}$$

- Resolution of the dual formulation thanks to quadratic optimization solvers

Theoretical analysis of SVM

- First ingredient : Complexity control
When RKHS norm bounded by M and kernel bounded by R^2 , we have the bound on Rademacher complexity of kernel classes :

$$\hat{R}_n(\mathcal{F}_M) \leq \frac{MR}{\sqrt{n}}$$

- Second ingredient : key inequalities (concentration and contraction or Zhang's Lemma)
- Two theorems can be derived : margin bound or classification error bound based on CRM

Global methods (e.g. CRM)

- Based on empirical minimization of error functionals
- Example in the case of *soft* classifiers $h : \mathbb{R}^d \rightarrow \mathbb{R}$
- Convex risk minimization, with φ positive convex cost function :

$$\widehat{A}(h) = \frac{1}{n} \sum_{i=1}^n \varphi(-Y_i h(X_i))$$

- Note that if $h \in \text{span}(\mathcal{H})$ with \mathcal{H} some class of classifiers, then the minimization problem is convex.
- Main issue : complexity of the class \mathcal{H} of candidate decision rules

Rademacher complexity of SVM

- Let X_1, \dots, X_n be an n -sample in \mathbb{R}^d , and denote by K the Gram matrix with coefficients $k(X_i, X_j)$, $1 \leq i, j \leq n$.
- Introduce the subspace of functions with bounded RKHS norm :

$$\mathcal{F}_\tau = \{h \in \mathcal{H}_k : \|h\|_k \leq \tau\}$$

- We then have :

$$\hat{R}_n(\mathcal{F}_\tau) \leq \frac{\tau \sqrt{\text{trace}(K)}}{n}$$

- In addition, if we have : $k(X_i, X_i) \leq R^2$ for $1 \leq i \leq n$, then

$$\hat{R}_n(\mathcal{F}_\tau) \leq \frac{\tau R}{\sqrt{n}}$$

Margin loss

- Fix $\rho > 0$
- The *margin loss* is defined, for any $u, v \in \mathbb{R}$, as :
 $\ell(u, v) = m_\rho(uv)$ where

$$m_\rho(t) = \begin{cases} 0 & \text{if } \rho \leq t \\ 1 - \frac{t}{\rho} & \text{if } 0 \leq t \leq \rho \\ 1 & \text{if } t \leq 0 \end{cases}$$

- Empirical margin error on a sample D_n :

$$\widehat{L}_{n,\rho}(f) = \frac{1}{n} \sum_{i=1}^n m_\rho(Y_i f(X_i))$$

Margin bounds for SVM classification

Theorem. (Fixed margin)

Let \mathcal{H}_k the RKHS with bounded kernel $k \leq R^2$.

Fix $\rho \in (0, 1)$, and $\delta > 0$. Then with probability at least $1 - \delta$, we have, for any SVM classifier g :

$$L(g) \leq \hat{L}_{n,\rho}(g) + 2 \left(\frac{\tau R}{\rho \sqrt{n}} \right) + \sqrt{\frac{\log(1/\delta)}{2n}}$$

and

$$L(g) \leq \hat{L}_{n,\rho}(g) + 2 \left(\frac{\tau \sqrt{\text{trace}(K)}}{\rho n} \right) + 3 \sqrt{\frac{\log(2/\delta)}{2n}}$$

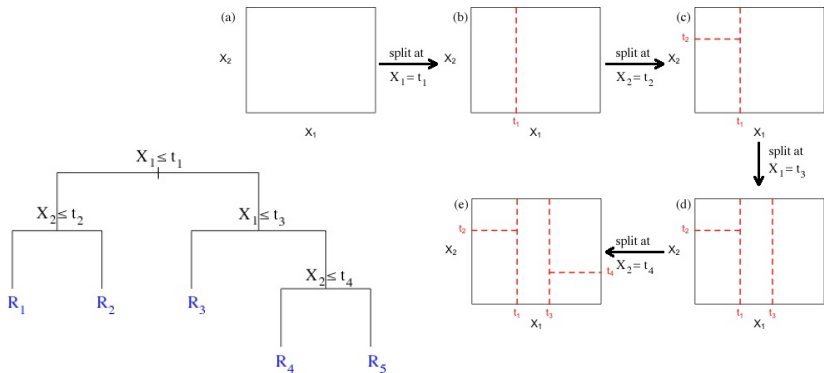
Interlude - The (al)most perfect algorithm

Decision trees

Algorithmic principles of Decision Trees

- Belongs to the family of *local* methods using (a) recursive partitioning, and (b) label averaging (or majority voting)
- Main inputs : (1) Geometry/number of splits (perpendicular, linear, binary or more, ...), (2) Local cost function (impurity) : entropy, Gini, classification error, (3) Stopping criterion : number of layers, minimal number of points per cell..., (4) with/out pruning
- Historical references : Hyafil, Rivest (1976), Breiman, Friedman, Olshen, Stone (1984), Quinlan (1986)

Construction through recursive partitioning



Stopping criteria and tree balancing

- Hyperparameters : type/number of splits, cost function, minimal number of points per partition cell, maximal depth of the tree (number of layers)
- Pruning the tree : amounts to exploring the class of all subpartitions (subtrees) and optimize a penalized criterion of the form

$$\arg \min_c \hat{L}_n(h_c) + \lambda |c|$$

where $c \subset \hat{c}$ is the collection of subpartitions obtained from the learned partition by pruning from bottom to top

Theory for partition-based classifiers

- Case of fixed and regular partitions with cells which are hypercubes of \mathbb{R}^d with edges of length δ_n :

$$\mathbb{E}L(\hat{h}(\cdot, \delta_n)) \rightarrow L^*$$

under the condition : $n\delta_n^d \rightarrow \infty$ and $\delta_n \rightarrow 0$ when $n \rightarrow \infty$
(need enough data points in every cell and cell diameter go to zero as sample size grows)

- Case of empirical data-driven partitions : Vapnik-Chervonenkis and Rademacher theory do apply, but also Minimum Description Length...

Complexity of decision trees

Binary splits, continuous features :

- Time complexity : $\mathcal{O}(\log n)$
- Runtime complexity : $\mathcal{O}(dn^2 \log n)$, can be optimized to $\mathcal{O}(dn \log n)$
- VC dimension : $\mathcal{O}(m \log(md))$ where m is the number of internal nodes (Leboeuf, Leblanc, Marchand, 2020)
- Codelength : $(n + 1) \log_2(d + 3)$

Take-home message on decision trees

Virtues of decision trees :

- Can handle missing/categorical data, scale change
- Can be expressed in terms of logical rule \rightarrow explainable machine learning

Major limitations :

- Prediction performance below state-of-the-art methods since the mid-90s (SVM, boosting, neural networks)
- Decision trees are extremely unstable :
 - Typical hack : Bumping (Tibshirani, 1997)
 - Recent improvements : see Bertsimas, Digalakis (2023)
- Numerical cost for the pruning step is high - as $\mathcal{O}(2^L)$ for binary trees where L is the number of layers of the master tree

Part II. Ensemble methods

What can be saved from decision trees?

The concept of weak classifier

- Intuition : a weak classifier performs at least slightly better than random guessing (probability $\frac{1}{2} + \gamma$ for some $\gamma > 0$ to predict the true label).
- Formalization : weakly PAC-learnable algorithm - see Kearns and Valiant (1989), Freund (1990), Schapire (1990)
- Typical example : decision trees with single-variable splits (aka decision stumps) and fixed number of layers (say 3 to 7)

Definition of ensemble methods

- Consider a weak learning algorithm over a base class \mathcal{H} of predictors
- An ensemble method has a search space which is either

$$\mathcal{F}_\alpha = \left\{ \sum_{t \geq 1} \alpha_t h_t : \forall t \geq 1, \alpha_t \in \mathbb{R}, h_t \in \mathcal{H} \right\}$$

or

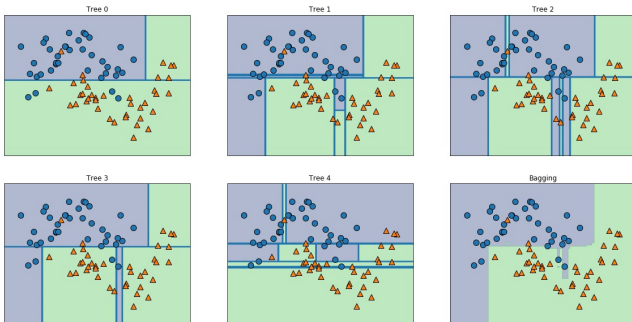
$$\mathcal{F}_1 = \left\{ \sum_{t \geq 1} h_t : \forall t \geq 1, h_t \in \mathcal{H} \right\}$$

- Popular ensemble methods : Bagging (Breiman (1996)), Random forests (Amit, Geman (1997), Breiman (2000)), Boosting (Freund, Schapire (1995))

Complexity of ensembles

- Vapnik-Chervonenkis dimension of \mathcal{F}_α is $+\infty$ even for base class with finite VC dimension V
- For truncated sums (with T terms), the VC dimension is upper bounded by $2(V + 1)(T + 1) \log_2(e(T + 1))$
- the Rademacher average of ensembles is in $\mathcal{O}\left(\sqrt{\frac{V}{n}}\right)$

Ensemble of decision trees with stumps



Ensemble methods

Bagging and Random Forests

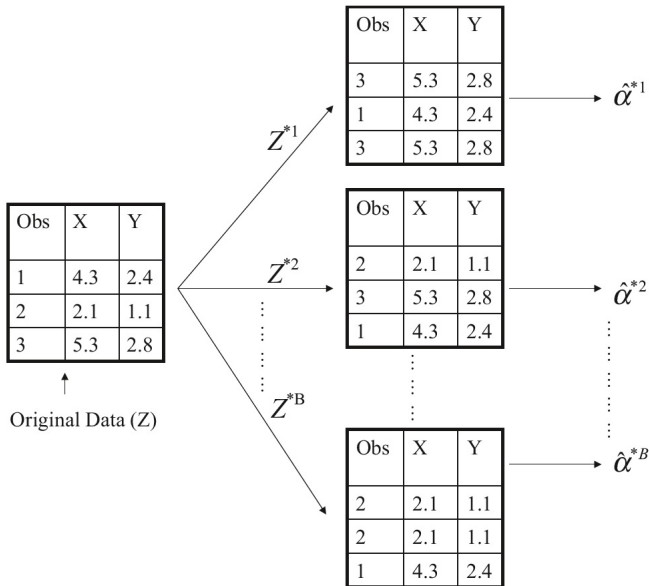
How to generate the ensemble? Bootstrap and aggregation

- Bagging and random forests operate on the search space

$$\mathcal{F}_1 = \left\{ \sum_{t \geq 1} h_t : \forall t \geq 1, h_t \in \mathcal{H} \right\}$$

- Bagging and random forests rely on bootstrap samples of the training data, meaning if we denote by D_n the training data, we assume that we can sample functions $\hat{h}_1, \dots, \hat{h}_t$ (the ensemble) from \mathcal{H} conditionally to D_n
- They differ by some different specifications of the recursive partitioning procedure to build each tree (no pruning involved)

What is bootstrap in general?



Randomized rules (1/2)

- For a given sample $D_n = \{(X_i, Y_i) : i = 1, \dots, n\}$
- Introduce \mathcal{Z} a measurable space and Z a random variable over \mathcal{Z}
- Conditionally on the sample D_n and on (X, Y) , draw independent sequences Z_1, \dots, Z_B of B copies of Z
- Design a pool of decision rules $\hat{g}_{n,b}(x) = \hat{g}_{n,b}(x, Z_b, D_n)$ for $b = 1, \dots, B$

Randomized rules (2/2)

- *Voting classifier* :

$$\hat{g}_n^B(x) = \mathbb{I} \left\{ \sum_{b=1}^B \hat{g}_{n,b}(x) > B/2 \right\} ,$$

- *Averaging classifier* (which is not a randomized classifier) :

$$\bar{g}_n^B(x) = \mathbb{I} \{ \mathbb{E}_Z \hat{g}_n(x, Z) > 1/2 \} .$$

Bagging - Breiman, 1996

- Randomization through bootstrap replicates of D_n
- Randomized rule through bagging :

$$g_n(x, Z, D_n) = g_n(x, D_n(Z))$$

- ... and $D_n(Z) = \{(X_i^*, Y_i^*) : i = 1, \dots, n\}$ where the points are drawn through random sampling from D_n
- Typical sampling is sampling with replacement and $|D_n(Z)| = n$

Bagging - a consistency result

- Special case with subsampling and without replicates in the bootstrap sample
- $|D_n(Z)| = N \leq n$ and ...
- ... we assume $N \sim \text{Bin}(n, q_n)$ and $q_n = \mathbb{P}((X_i, Y_i) \in D_n(Z))$
- Consistency of both voting classifier and averaging classifier under assumptions :
 - $\{g_n\}$ sequence of classifiers that is consistent for P
 - $nq_n \rightarrow \infty$ when $n \rightarrow \infty$

Bagging can render consistent rules that are inconsistent

- Biau, Devroye and Lugosi (2008) have considered bagging 1-NN
- 1-NN is consistent if and only if $L^* \in \{0, 1/2\}$
- Bagging averaged 1-NN classifier is consistent for any P if and only if $q_n \rightarrow 0$ and $nq_n \rightarrow \infty$ when $n \rightarrow \infty$
- Proof follows the lines of Stone theorem (cf. Devroye, Györfi, Lugosi (1996))

Random Forest consistency

- Simplified model : the *purely random forest* in Biau, Devroye, Lugosi (2008) (further work in Scornet, Biau, Vert (2015))
- Start with a tree classifier with rectangular cells with orthogonal splits over $[0, 1]^d$ and build randomized versions as follows :
 - Draw a leaf according to a uniform distribution over the set of leaves of the tree
 - Draw one split variable among the d dimensions of input space with a uniform
 - Position the split at random (uniform distribution again)
 - Repeat k times splitting of terminal cells of the tree
- Result : The averaged classifier is consistent if $k_n \rightarrow \infty$ and $k_n/n \rightarrow 0$ when $n \rightarrow \infty$

Ensemble methods

Boosting

Historical perspective on Boosting

- Original paper : Freund and Schapire (ECML, 1995).
- Interpretation of the optimization problem solved as stochastic gradient descent : Friedman (CSDA, 2002).
- Wald Memorial lecture (IMS, 2000) : Leo Breiman declares that *"understanding Boosting is the most important problem in Machine Learning"*
- Proofs of boosting consistency : Jiang (2004), Lugosi, G. and Vayatis, N. (2004), Zhang (2004), Bartlett and Traskin (2007)
- Xgboost, a scalable implementation : Chen, T. and Guestrin, C. (ACM SIGKDD, 2016).

Algorithmic principle for Boosting

- **Input**

- Data sample $D_n = \{(X_i, Y_i) : i = 1, \dots, n\}$ with classification data $\{-1, +1\}$
- Base hypothesis class \mathcal{H} of *weak* classifiers such as decision trees (assumed to be symmetric, i.e. $h \in \mathcal{H}$ iff $-h \in \mathcal{H}$)

- **Iterations** $t = 1, \dots, T$.

- Compute weights $\hat{\alpha}_t > 0$ and weak classifiers $\hat{h}_t \in \mathcal{H}$

- **Output.**

- The Boosting classifier takes the sign of the following linear

combination of weak classifiers :
$$\hat{f}_n(x) = \sum_{t=1}^T \hat{\alpha}_t \hat{h}_t(x)$$

Weighting the data

- Boosting distributions on the data : sequence of discrete probability distributions over $\{1, \dots, n\}$ denoted by Π_t , $t \geq 1$
- Weighted training error : for any weak classifier $h \in \mathcal{H}$ and for $t \geq 1$

$$\hat{\varepsilon}_t(h) = \sum_{i=1}^n \Pi_t(i) \mathbb{I}\{h(X_i) \neq Y_i\}$$

The AdaBoost algorithm

- 1 **Initialization.** Π_1 is the uniform distribution on $\{1, \dots, n\}$
- 2 **Boosting iterations.** For $t = 1, \dots, T$, find the weak classifier such that :

$$\hat{h}_t = \arg \min_{h \in \mathcal{H}} \hat{\varepsilon}_t(h)$$

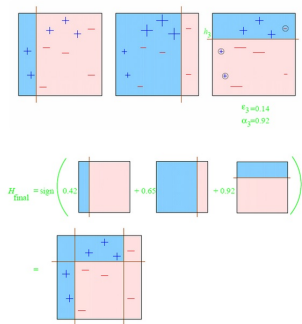
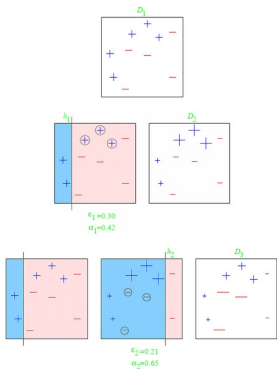
then set $e_t = \hat{\varepsilon}_t(\hat{h}_t)$ and take the weight to be

$$\hat{\alpha}_t = \frac{1}{2} \log \left(\frac{1 - e_t}{e_t} \right)$$

- 3 **Boosting distribution update.** For any $i = 1, \dots, n$,

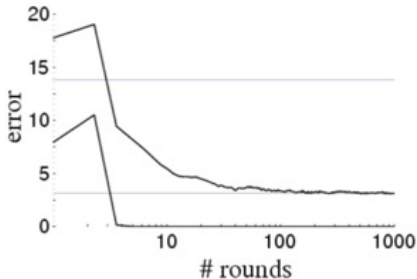
$$\Pi_{t+1}(i) \propto \Pi_t(i) \exp \left(-\hat{\alpha}_t Y_i \cdot \hat{h}_t(X_i) \right)$$

Illustration of Boosting on a toy example



Mysterious behavior of Boosting

The test error continues to drop along the iterations even though the training error is zero \rightarrow Regularization effect thanks to averaging??



Boosting as a CRM principle

- Boosting can be interpreted as a functional gradient descent on the following functional :

$$\hat{A}_n(f) = \frac{1}{n} \sum_{i=1}^n \exp(-Y_i f(X_i))$$

where f is taken in a hypothesis space which is the linear span of 'simple' set \mathcal{H} of classifiers.

- Exercise : why ?

Refer to : J. Friedman, "Greedy Function Approximation : A Gradient Boosting Machine", The Annals of Statistics, Vol. 29, No. 5, 2001.