Institut
Mines-Telecom

# Ensemble methods

Florence d'Alché-Buc,
florence.dalche@telecom-paristech.fr

# Outline

# Outline

# Ensemble methods and meta-learning

- Improve upon a single predictor by building an ensemble of predictors (with no hyperparameter)
- $\longrightarrow$ meta-learning : the parameter of the *meta-learning* algorithm is those of the base learner and the size of the ensemble

# Ensemble methods

Let $f_t, t = 1, \ldots, T$ be T different regressors.
Notations :

$$\begin{aligned}
\epsilon_t(x) &= y - f_t(x) \\
MSE(f_t) &= \mathbb{E}[\epsilon_t(x)^2] \\
f_{ens}(x) &= \frac{1}{T}\sum_t f_t(x) \\
&= y - \frac{1}{T}\sum_t \epsilon_t(x).
\end{aligned}$$

# Encourage the diversity of base predictors

$$MSE(f_{ens}) = \mathbb{E}[(y - f_{ens}(x))^2]$$

If $\epsilon_t$ are mutually independent with zero mean, then we have :

$$MSE(f_{ens}) = \frac{1}{T^2}\mathbb{E}[\sum_t \epsilon_t(x)^2]$$

The more diverse are the classifiers, the more we reduce the mean square error !

# Ensemble methods

- **Encourage the diversity of base predictors by :**
  - using bootstrap samples (Bagging and Random forests)
  - using randomized predictors (ex : Random forests)
  - using weighted version of the current sample (Boosting) with weights dependent on the previous predictor (adaptive sampling)

# Ensemble methods at a glance

- 1995 : Boosting, Freund and Schapire
- 1996 : Bagging, Breiman
- 2001 : Random forests, Breiman

# Outline

# Decomposition bias/variance in regression

Given $x$,

$$\mathbb{E}_S \mathbb{E}_{y|x}(y - f_S(x))^2 = noise(x) + bias^2(x) + \text{variance}(x) \quad (1)$$

noise(x): $E_{y|x}[(y - E_{y|x}(y))^2]$ :
quantifies the error made by the Bayes model ($E_{y|x}(y)$)
$bias^2(x) = (E_{y|x}(y) - E_S[f_S(x)])^2$
measures the difference between minimal error (Bayes error) and the average model
$variance(x) = E_S[(f_S(x) - E_S[f_S(x)])^2]$
measures how much $h_S(x)$ varies from one training set to another

# Introduction to bagging (regression) - 1

Assume we can generate several training samples $\mathcal{S}_1, \ldots, \mathcal{S}_T$ from P(x,y).

A first algorithm :

- draw T training samples $\{\mathcal{S}_1, \ldots, \mathcal{S}_T\}$
- learn a model $f_t \in \mathcal{F}$ from each training sample $\mathcal{S}_t; t = 1, \ldots, T$
- compute the average model : $f_{ens}(x) = \frac{1}{T} \sum_{t=1}^{T} f_t(x)$

# Introduction to bagging - 2

The bias remains the same :

$bias(x) = E_{\mathcal{S}_1,\dots,\mathcal{S}_T}[f_{ens}(x)] = \frac{1}{T}\sum_t E_{\mathcal{S}_t}[f_t(x)] = E_{\mathcal{S}}[f_t(x)]$

The variance is divided by T :

$E_{\mathcal{S}_1,\dots,\mathcal{S}_T}[(f_{ens}(x) - E_{\mathcal{S}_1,\dots,\mathcal{S}_T}[f_{ens}(x)])^2] = \frac{1}{T} E_{\mathcal{S}}[f_{\mathcal{S}}(x) - E_{\mathcal{S}}[f_{\mathcal{S}}(x))^2]$

# Bagging (Breiman 1996)

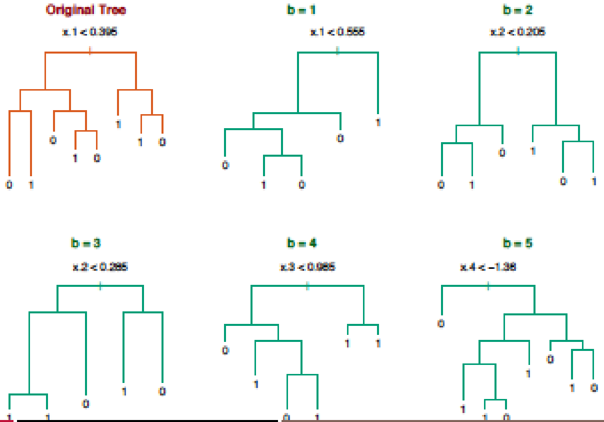
In practice, we do not know P(x,y) and we have only one training sample $\mathcal{S}$.

Bagging = Bootstrap Aggregating :

- draw $T$ bootstrap samples $\{\mathcal{B}_1 \ldots, \mathcal{B}_T\}$ from $\mathcal{S}$
- Learn a model $f_t$ for each $\mathcal{B}_t$
- Build the average model : $f_{bag}(x) = \frac{1}{T} \sum_t f_t(x)$

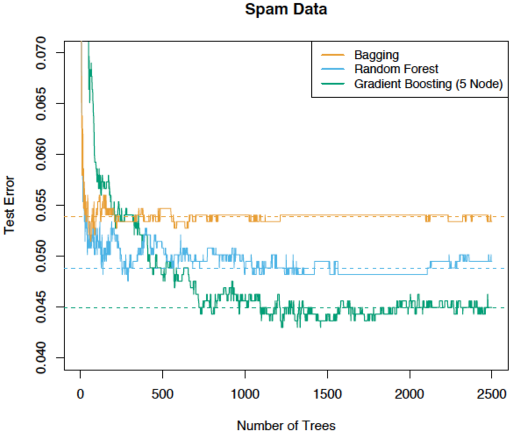
# Example of bagged trees

[Book : The elements of statistical learning, Hastie, Tibshirani, Friedman, 2001]



# Example of bagged trees

[Book : The elements of statistical learning, Hastie, Tibshirani, Friedman, 2001]



# Bagging in practise

- Variance is reduced but the bias can increase a bit (the effective size of a bootstrap sample is 30% smaller than the original training set $\mathcal{S}$
- The obtained model is however more complex than a single model
- Bagging works for unstable predictors (neural nets, trees)
- In supervised classification, bagging a good classifier usually makes it better but bagging a bad classifier can make it worse

# Outline

# Other ensemble methods

- Perturbe and combine algorithms
  - Perturbe the base predictor
  - Combine the perturbed predictors

REFS : Random forests : Breiman 2001
Geurts, Ernst, Wehenkel, Extra-trees, 2006

# Random forests : Breiman 2001

## Random forests algorithm

- INPUT : candidate feature splits F, $\mathcal{S}_{train}$
- for t=1 to T
  - $\mathcal{S}_{train}^{(t)}$ m instance randomly drawn with replacement from $\mathcal{S}_{train}$
  - $h_{tree}^{(t)} \leftarrow$ randomized decision tree learned from $\mathcal{S}_{train}^{(t)}$
- OUTPUT : $H^T = \frac{1}{T} h_{tree}^{(t)}$

# Random forests :

Learning a single randomized tree :

- To select a split at a node :
  - $R_f(F) \leftarrow$ randomly select (without replacement) $f$ feature splits from $F$ with $f << |F|$
  - Choose the best split in $R_f(F)$ (consider the different cut-points)
- Do not prune this tree

# Randomized tree :

Learning a single randomized tree :

- To select a split at a node :
  - $R_K(F) \leftarrow$ randomly select (without replacement) $f$ feature splits from $F$ with $f << |F|$
  - Choose the best split in $R_f(F)$ (consider the different cut-points)
- Do not prune this tree

# Extra-trees : Geurts et al. 2006

## Extra-trees

- INPUT : candidate feature splits F, $\mathcal{S}_{train}$
- for t=1 to T
  - Always use $\mathcal{S}_{train}$
  - $h_{tree}^{(t)} \rightarrow$ : randomized decision tree learned from $\mathcal{S}_{train}$
- OUTPUT : $H^T = \frac{1}{T} h_{tree}^{(t)}$

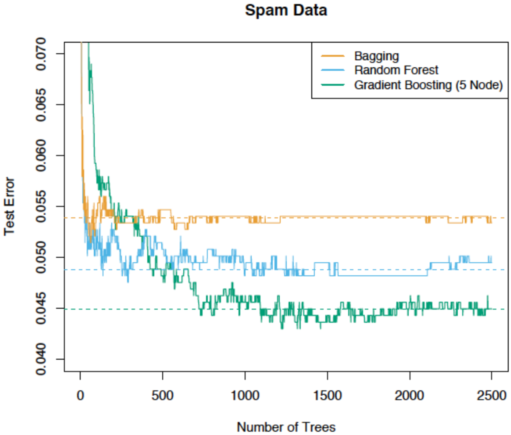
Learning a single randomized tree in extra-trees :

- To select a split at a node :
  - randomly select (without replacement) $K$ feature splits from $F$ with $K << |F|$
  - Draw $K$ splits using the procedure Pick-a-random-split($\mathcal{S}$,i):
    - let $a^i_{max}$ and $a^i_{min}$ denote the maximal and minimal value of $x_i$ in $\mathcal{S}$
    - Draw uniformly a cut-point $a_c$ in $[a^i_{max}, a^i_{min}]$
- Choose the best split among the $K$ previous splits

Do not prune this tree

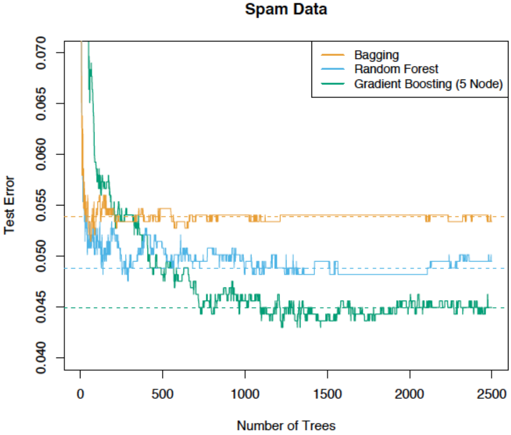
# Random forest

Example of decision frontier :



# Comparison (just an example)

[Book : The elements of statistical learning, Hastie, Tibshirani, Friedman, 2001]

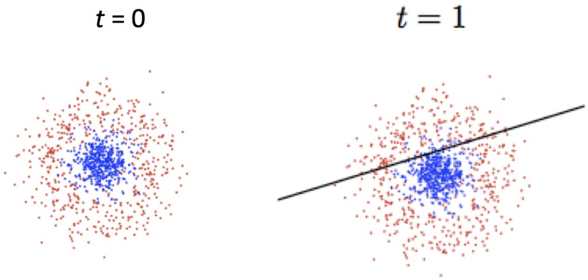


# Outline

# AdaBoost

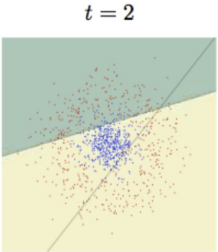
- Idea : learn a sequence of predictors trained on weighted dataset with weights depending on the loss so far.
- Iterative scheme proposed by Schapire and Freund :

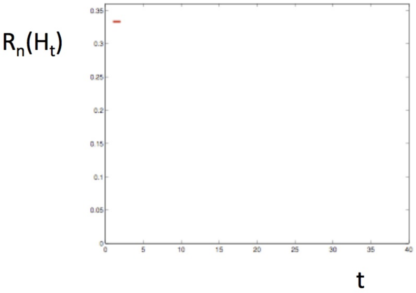
# Boosting a linear classifier




*Source Jiri Matas (Oxford U.)*

# Boosting a linear classifier

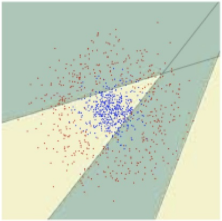


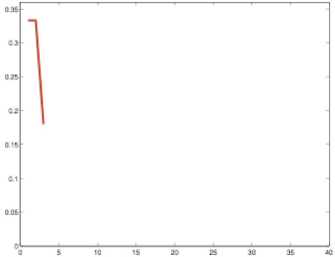




*Source Jiri Matas (Oxford U.)*

# Boosting a linear classifier

$t = 3$

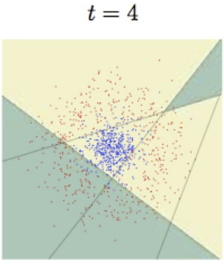


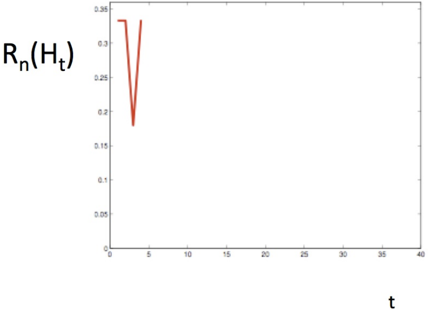



*Source*

*Jiri Matas (Oxford U.)*

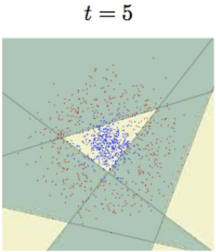
# Boosting a linear classifier

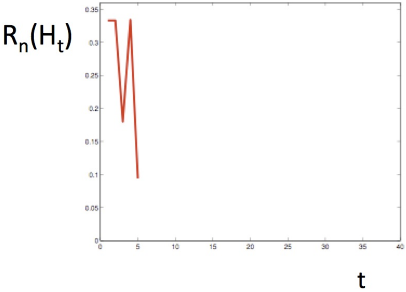



*Source Jiri Matas (Oxford U.)*

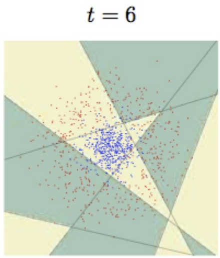
# Boosting a linear classifier

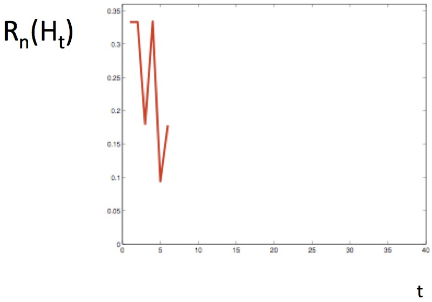



*Source Jiri Matas (Oxford U.)*

# Boosting a linear classifier

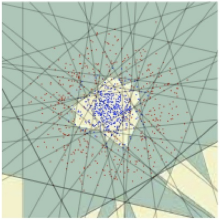


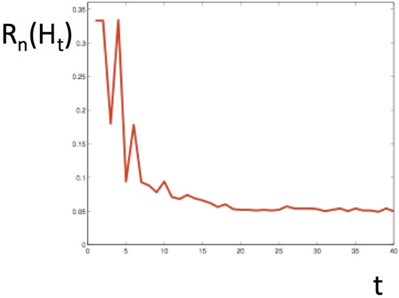




*Source Jiri Matas (Oxford U.)*

# Boosting a linear classifier

$t = 40$






*Source Jiri Matas (Oxford U.)*

# AdaBoost (Freund and Schapire 1996)

$\mathcal{H}$ : a chosen class of "weak" binary classifiers

- Set $w_1(i) = 1/n$ ; $t = 0$ and $f_0 = 0$
- For $t = 1$ to $T$
  - $t = t + 1$
  - $h_t = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{n} \epsilon_t(h)$
  - With $\epsilon_t = \sum_{i=1}^{n} w_t(i) \ell^{0/1}(y_i, h(x_i))$
  - Set $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
  - let $w_i(t+1) = \frac{w_t(i) e^{-\alpha_t y_i h_t(x_i)}}{Z_{t+1}}$ where $Z_{t+1}$ is a renormalization constant such that $\sum_{i=1}^{n} w_i(t+1) = 1$
  - $f_t = f_{t-1} + \alpha_t h_t$
- Use $f_T = \sum_{i=1}^{T} \alpha_t h_t$

Intuition : $w_i(t)$ measures the difficulty of learning the sample $i$ at step $t$...

# Exponential Stagewise Additive Modeling (Friedman, Hastie, Tibshirani 1999)

- Greedy optimization of a classifier as a linear combination of $T$ classifier for the exponential loss.
  - Set $t = 0$ and $f_t = 0$.
  - For $t = 1$ to $T$,
    - $(h_t, \alpha_t) = \arg\min_{h,\alpha} \sum_{i=1}^{n} e^{-y_i(f_{t-1}(x_i)+\alpha h(x_i))}$
    - $f_t = f_{t-1} + \alpha_t h_t$
  - Use $f_T = \sum_{t=1}^{T} \alpha_t h_t$
- Adaboost and this algorithm are equivalent



TELECOM ParisTech

# Boosting

- Iterative scheme with only two parameters : the class $\mathcal{H}$ of *weak* classifier and the number of step $T$.
- In the literature, one can read that Adaboost does not overfit ! This not true (see work of Vayatis et al.) and $T$ should be chosen with care...

# Boosting

- General greedy optimization strategy to obtain a linear combination of *weak* predictor
  - Set $t = 0$ and $f = 0$.
  - For $t = 1$ to $T$,
    - $(h_t, \alpha_t) = \arg\min_{h,\alpha} \sum_{i=1}^{n} \ell'(y_i, f(x_i) + \alpha h(x_i))$
    - $f = f + \alpha_t h_t$
  - Use $f = \sum_{t=1}^{T} \alpha_t h_t$
- Forward Stagewise Additive Modeling :
  - AdaBoost with $\ell'(y, h) = e^{-yh}$
  - LogitBoost with $\ell'(y, h) = \log(1 + e^{-yh})$
  - $L_2$Boost with $\ell'(y, h) = (y - h)^2$ (Matching pursuit)
  - $L_1$Boost with $\ell'(y, h) = |y - h|$
  - HuberBoost with
    $\ell'(y, h) = |y - h|^2 \mathbf{1}_{|y-h|<\epsilon} + (2\epsilon|y - h| - \epsilon^2)\mathbf{1}_{|y-h|\geq\epsilon}$
- Simple principle but no easy numerical scheme except for AdaBoost and $L_2$Boost...

# Gradient Boosting I

- At each boosting step, one need to solve

$$(h_t, \alpha_t) =_{h,\alpha} \sum_{i=1}^{n} \ell'(y_i, f(x_i) + \alpha h) = L(y, f + \alpha h)$$

- Gradient approximation $L(y, f + \alpha h) \sim L(y, f) + \alpha \langle \nabla f, h \rangle$.
- Gradient boosting : replace the minimization step by a *gradient descent* type step :
  - Choose $h_t$ as the best possible descent direction in $\mathcal{H}$
  - Choose $\alpha_t$ that minimizes $L(y, f + \alpha h_t)$ (line search)
- Easy if finding the best descent direction is easy !

# Stochastic Gradient Boosting

- Variation of the Boosting scheme
- Idea : change the learning set at each step.
- Two possible reasons :
  - Optimization over all examples too costly
  - Add variability to use a averaged solution
- Two different samplings :
  - Use sub-sampling, if you need to reduce the complexity
  - Use re-sampling, if you add variability...
- Stochastic Gradient name mainly used for the first case...

# Outline

# References


- Perrone, Cooper, When classifiers disagree, 1992
- Tumer and Gosh, 1996
- Breiman, Bagging predictors, 1996
- Buhlman and Yu, Analyzing bagging, Annals of stats., 2002
- Breiman, Random Forests, Machine Learning, 2001.
- Geurts, Ernst, Wehenekl, Extra-trees, JMLR, 2006
- Boosting :
  - Freund and Schapire, 1996
  - Greedy function approximation, Friedman, 1999.
  - MarginBoost and AnyBoost : Mason et al. 1999.