Trees, Bagging, Random Forest and Boosting

E. Le Pennec

Fall 2017

Outline

1. Trees
    - Questions, Trees and Partitions
    - Classification and Regression Trees
    - Tree construction: Branching and Pruning

2. Bagging and Forest
    - Bagging and Bootstrap
    - Construction rules

3. Boosting
    - AdaBoost as a Greedy Scheme
    - Generic Boosting
    - Gradient Boosting
    - Variations

Outline

1. Trees
    - Questions, Trees and Partitions
    - Classification and Regression Trees
    - Tree construction: Branching and Pruning

2. Bagging and Forest
    - Bagging and Bootstrap
    - Construction rules

3. Boosting
    - AdaBoost as a Greedy Scheme
    - Generic Boosting
    - Gradient Boosting
    - Variations

Outline

1. Trees
   - Questions, Trees and Partitions
   - Classification and Regression Trees
   - Tree construction: Branching and Pruning
2. Bagging and Forest
   - Bagging and Bootstrap
   - Construction rules
3. Boosting
   - AdaBoost as a Greedy Scheme
   - Generic Boosting
   - Gradient Boosting
   - Variations

Guess Who?

A game of questions

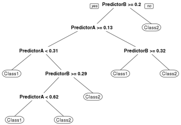
- Game invented in 1979 in the UK.
- **Goal:** discover the character chosen by your opponent before he discovers yours.
- **Optimal strategy:** choose at each step the question that splits the remaining characters in two group with the least possible difference in size.
- **Information Theory!**

- Adaptive construction of a tree of questions!
- Optimal tree of questions can be constructed without knowing the answers...but during a game only a path of the tree is used...

Outline

1. Trees
   - Questions, Trees and Partitions
   - Classification and Regression Trees
   - Tree construction: Branching and Pruning
2. Bagging and Forest
   - Bagging and Bootstrap
   - Construction rules
3. Boosting
   - AdaBoost as a Greedy Scheme
   - Generic Boosting
   - Gradient Boosting
   - Variations

Classification and Regression Trees



Tree principle

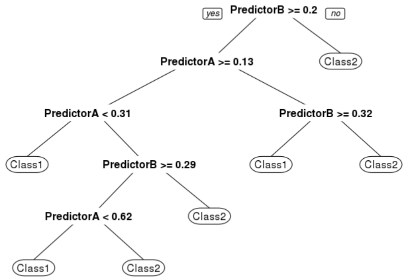
- Construction of a recursive partition through a tree structured set of questions (splits around a given value of a variable)
- For a given partition, statistical approach **and** optimization approach yields the same classifier!
- A simple majority vote/averaging in each leaf

- **CART/ID3:** specific choice for the partition construction proposed independently by Breiman and Quinlan in respectively 85 and 86.
- Amounts to a **local estimate** of proportions or $\mathbb{E}[Y|X]$!

Outline

1. Trees
   - Questions, Trees and Partitions
   - Classification and Regression Trees
   - Tree construction: Branching and Pruning
2. Bagging and Forest
   - Bagging and Bootstrap
   - Construction rules
3. Boosting
   - AdaBoost as a Greedy Scheme
   - Generic Boosting
   - Gradient Boosting
   - Variations

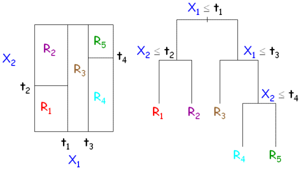
CART



Tree and partition

- Quality of the prediction depends on the tree (the partition).
- **Intuitively:**
  - small leaves lead to low bias but large variance
  - large leaves lead to large bias but low variance...
- **Issue:** Minim. of the (penalized) empirical error is NP hard!
- Practical tree construction are all based on **two steps**:
  - a top-down step in which branches are created (branching)
  - a bottom-up in which branches are removed (pruning)

Branching



Greedy top-bottom approach

- Start from a single region containing all the data
- Recursively split those regions along a certain variable and a certain value

- **No regret strategy** on the choice of the splits!
- **Heuristic:** choose a split so that the two new regions are as *homogeneous* possible...

Branching

Various definition of *homogeneous*

- **CART:** empirical loss based criterion (least squares/prediction error)
$$C(R,\overline{R}) = \sum_{x_i \in R} \ell(Y_i, y(R)) + \sum_{x_i \in \overline{R}} \ell(Y_i, y(\overline{R}))$$
- **CART:** Gini index (Classification)
$$C(R,\overline{R}) = \sum_{x_i \in R} p(R)(1-p(R)) + \sum_{x_i \in \overline{R}} p(\overline{R})(1-p(\overline{R}))$$
- **C4.5:** entropy based criterion (Information Theory)
$$C(R,\overline{R}) = \sum_{x_i \in R} H(R) + \sum_{x_i \in \overline{R}} H(\overline{R})$$

- CART with Gini is probably the most used technique...
- Other criterion based on χ^2 homogeneity or based on different local predictors (generalized linear models...)

Branching

Choice of the split in a given region

- Compute the criterion for **all features and all possible splitting points** (necessarily among the data values in the region)
- Choose the one **minimizing** the criterion

- **Variations:** split at all categories of a categorical variables using a clever category ordering (ID3), split at a fixed position (median/mean)
- **Stopping rules:**
  - when a leaf/region contains less than a prescribed number of observations
  - when the region is sufficiently homogeneous...
- May lead to a quite complex tree: over-fitting possible!
- Additional pruning often use.

Pruning

- **Model selection** within the (rooted) subtrees of previous tree!
- Number of subtrees can be quite large but the tree structure allows to find the best model efficiently.

Key idea

- The predictor in a leaf depends only on the values in this leaf.
- **Efficient bottom-up (dynamic programming) algorithm** if the criterion used satisfies an additive property

$$C(\mathcal{T}) = \sum_{\mathcal{L} \in \mathcal{T}} c(\mathcal{L})$$

- Example: AIC / CV.
- Goal: limit overfitting!

Pruning

Examples of criterion satisfying this assumption

- AIC type criterion:

$$\sum_{i=1}^{n} \ell'(Y_i, f_{\mathcal{L}(\mathbf{X}_i)}(\mathbf{X}_i) + \lambda |\mathcal{T}| = \sum_{\mathcal{L} \in \mathcal{T}} \left(\sum_{\mathbf{X}_i \in \mathcal{L}} \ell'(Y_i, f_{\mathcal{L}}(\mathbf{X}_i) + \lambda \right)$$

- Simple cross-Validation (with $(\mathbf{X}'_i, \mathbf{X}'_i)$ a different dataset):

$$\sum_{i=1}^{n'} \ell'(Y'_i, f_{\mathcal{L}}(\mathbf{X}'_i) = \sum_{\mathcal{L} \in \mathcal{T}} \left(\sum_{\mathbf{X}'_i \in \mathcal{L}} \ell'(Y'_i, f_{\mathcal{L}}(\mathbf{X}'_i) \right)$$

Pruning

- **Key observation:** at a given node, the best subtree is either the current node or the union of the best subtrees of its child.

Dynamic programming algorithm

- Compute the individual cost $c(\mathcal{L})$ of each node (including the leaves)
- Scan all the nodes in reverse order of depth:
  - If the node $\mathcal{L}$ has no child, set its best subtree $\mathcal{T}(\mathcal{L})$ to $\{\mathcal{L}\}$ and its current best cost $c'(\mathcal{L})$ to $c(\mathcal{L})$
  - If the children $\mathcal{L}_1$ and $\mathcal{L}_2$ are such that $c'(\mathcal{L}_1) + c'(\mathcal{L}_2) \geq c(\mathcal{L})$, then prune the child by setting $\mathcal{T}(\mathcal{L}) = \{\mathcal{L}\}$ and $c'(\mathcal{L}) = c(\mathcal{L})$
  - Otherwise, set $\mathcal{T}(\mathcal{L}) = \mathcal{T}(\mathcal{L}_1) \cup \mathcal{T}(\mathcal{L}_2)$ and $c'(\mathcal{L}) = c'(\mathcal{L}_1) + c'(\mathcal{L}_2)$
- The best subtree is the best subtree $\mathcal{T}(\mathcal{R})$ of the root $\mathcal{R}$.

- Optimization cost proportional to the **number of nodes** and not the number of subtrees!

Extensions

- **Local estimation** of the proportions or of the conditional mean.
- **Recursive Partitioning methods:**
  - Recursive construction of a partition
  - Use of simple local model on each part of the partition
- **Examples:**
  - CART, ID3, C4.5, C5
  - MARS (local linear regression models)
  - Piecewise polynomial model with a dyadic partition...
- **Book:** *Recursive Partitioning and Applications* by Zhang and Singer

CART: Pros and Cons

Pros

- Leads to a easily interpretable model
- Fast computation of the prediction
- Easily deals with categorical features

Cons

- Greedy optimization
- Hard decision boundaries
- Lack of stability

Ensemble methods

- Lack of robustness for single trees.
- How to combine trees?

Parallel construction

- Construct several trees from bootstrapped samples and average the responses **(Bagging)**
- Add more randomness in the tree construction **(Eandom forests)**

Sequential construction

- Construct a sequence of trees by reweighting sequentially the samples according to their difficulties **(AdaBoost)**
- Reinterpretation as a stagewise additive model **(Boosting)**

Outline

1. Trees
   - Questions, Trees and Partitions
   - Classification and Regression Trees
   - Tree construction: Branching and Pruning
2. Bagging and Forest
   - Bagging and Bootstrap
   - Construction rules
3. Boosting
   - AdaBoost as a Greedy Scheme
   - Generic Boosting
   - Gradient Boosting
   - Variations

Outline

1. Trees
   - Questions, Trees and Partitions
   - Classification and Regression Trees
   - Tree construction: Branching and Pruning
2. Bagging and Forest
   - Bagging and Bootstrap
   - Construction rules
3. Boosting
   - AdaBoost as a Greedy Scheme
   - Generic Boosting
   - Gradient Boosting
   - Variations

Independent Average

Stability through averaging

- Very simple idea to obtain a more stable estimator.
- **Vote/average** of B predictors $f_1, \ldots, f_B$ obtained with **independent datasets** of size n!

$$f_{\text{agr}} = \operatorname{sign}\left(\frac{1}{B}\sum_{b=1}^{B} f_b\right) \quad \text{or} \quad f_{\text{agr}} = \frac{1}{B}\sum_{i=1}^{B} f_b$$

- **Regression:** $\mathbb{E}\left[f_{\text{agr}}(x)\right] = \mathbb{E}\left[f_b(x)\right]$ and $\mathbb{V}\left[f_{\text{agr}}(x)\right] = \frac{\mathbb{V}[f_b(x)]}{B}$
- **Prediction:** slightly more complex analysis
- Averaging leads to **variance reduction**, i.e. stability!

- **Issue:** cost of obtaining B independent datasets of size n!

Bagging and Bootstrap

- Strategy proposed by Breiman in 1994.

Stability through bootstrapping

- Instead of using B independent dataset of size n, draw B dataset from a single one using a **uniform with replacement** scheme (Bootstrap).
- **Rk:** On average, a fraction of $(1 - 1/e) \simeq .63$ examples are unique among each drawn dataset...
- The f_b are still identically distributed but **not independent** anymore.
- Price for the non independence: $\mathbb{E}\left[f_{\mathrm{agr}}(x)\right] = \mathbb{E}\left[f_b(x)\right]$ and
$$\mathbb{V}\left[f_{\mathrm{agr}}(x)\right] = \frac{\mathbb{V}\left[f_b(x)\right]}{B} + \left(1 - \frac{1}{B}\right)\rho(x)$$
with $\rho(x) = \mathbb{C}\mathrm{ov}\left[f_b(x), f_{b'}(x)\right] \leq \mathbb{V}\left[f_b(x)\right]$ with $b \neq b'$.
- **Bagging:** Bootstrap Aggregation

- Better aggregation scheme exists...

Outline

1. Trees
   - Questions, Trees and Partitions
   - Classification and Regression Trees
   - Tree construction: Branching and Pruning

2. Bagging and Forest
   - Bagging and Bootstrap
   - Construction rules

3. Boosting
   - AdaBoost as a Greedy Scheme
   - Generic Boosting
   - Gradient Boosting
   - Variations

Randomized Predictors

- Correlation leads to less variance reduction:
$$\mathbb{V}\left[f_{\text{agr}}(x)\right] = \frac{\mathbb{V}\left[f_b(x)\right]}{B} + \left(1 - \frac{1}{B}\right)\rho(x)$$
with $\rho(x) = \mathbb{C}\text{ov}\left[f_b(x), f_{b'}(x)\right]$ with $b \neq b'$.
- **Idea:** Reduce the correlation by adding more randomness in the predictor.

Randomized Predictors

- Construct predictors that depends from a **randomness source** R that may be chosen independently for all bootstrap samples.
- This **reduces** the correlation between the estimates and thus the **variance**...
- But may **modify heavily the estimates** themselves!

- **Performance gain** not obvious from theory...

Random Forest

- Example of randomized predictors based on trees proposed by Breiman in 2001...

Random Forest

- Draw B resampled datasets from a single one using a uniform with replacement scheme (**Bootstrap**)
- For each resampled datasets, construct a tree using a different **randomly drawn subset of variables** at each split.

- Most important parameter is the **subset size**:
  - if it is too large then we are back to bagging
  - if it is too small the mean of the predictors is probably not a good predictor...
- **Recommendation:**
  - Classification: use a proportion of $1/\sqrt{p}$
  - Regression: use a proportion of $1/3$
- **Sloppier stopping rules** and pruning than in CART...

Random Forest

Out Of the Box Estimate

- For each sample x_i, a prediction can be made using only the resampled datasets not containing x_i...
- The corresponding empirical prediction error is **not prone to overfitting** but does not corresponds to the final estimate...
- Good proxy nevertheless.

Random Forest and Variable Ranking

- **Importance:** Number of use or criterion gain at each split can be used to rank the variables.
- **Permutation tests:** Difference between OOB estimate using the true value of the jth feature and a value drawn a random from the list of possible values.

- Theoretical performance analysis very challenging!

Outline

1. Trees
   - Questions, Trees and Partitions
   - Classification and Regression Trees
   - Tree construction: Branching and Pruning
2. Bagging and Forest
   - Bagging and Bootstrap
   - Construction rules
3. Boosting
   - AdaBoost as a Greedy Scheme
   - Generic Boosting
   - Gradient Boosting
   - Variations

Outline

1. Trees
    - Questions, Trees and Partitions
    - Classification and Regression Trees
    - Tree construction: Branching and Pruning
2. Bagging and Forest
    - Bagging and Bootstrap
    - Construction rules
3. Boosting
    - AdaBoost as a Greedy Scheme
    - Generic Boosting
    - Gradient Boosting
    - Variations

AdaBoost

- **Idea:** learn a sequence of predictor trained on weighted dataset with weights depending on the loss so far.

Iterative scheme proposed by Schapire and Freud

- Set $w_1(i) = 1/n$; $t = 0$ and $f = 0$
- For $t = 1$ to $= T$
  - $t = t + 1$
  - $h_t = \operatorname{argmin}_{h \in \mathcal{H}} \sum_{i=1}^n w_t(i) \ell^{0/1}(y_i, h(x_i))$
  - Set $\epsilon_t = \sum_{i=1}^n w_t(i) \ell^{0/1}(y_i, g(x_i))$ and $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
  - 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 $\sum_{i=1}^n w_i(t+1) = 1$
  - $f = f + \alpha_t h_t$
- Use $f = \sum_{i=1}^T \alpha_t h_t$

- **Intuition:** $w_i(t)$ measures the difficulty of learning the sample i at step t...
- No simple explanation of such a scheme!

AdaBoost

Exponential Stagewise Additive Modeling

- Set $t = 0$ and $f = 0$.
- For $t = 1$ to T,
  - $(h_t, \alpha_t) = \operatorname{argmin}_{h,\alpha} \sum_{i=1}^{n} e^{-y_i(f(x_i)+\alpha h(x_i))}$
  - $f = f + \alpha_t h_t$
- Use $f = \sum_{t=1}^{T} \alpha_t h_t$

- **Greedy optimization** of a classifier as a linear combination of T classifier for the **exponential loss**.
- Those two algorithms are **exactly the same**!

Proof

- Denoting $f_t = \sum_{t'=1}^{t} \alpha_{t'} h_{t'}$,

$$\begin{aligned}\sum_{i=1}^{n} e^{-y_i(f_{t-1}(x_i)+\alpha h)} &= \sum_{i=1}^{n} e^{-y_i f_{t-1}(x_i)} e^{-\alpha y_i h(x_i)} \\ &= \sum_{i=1}^{n} w_i'(t) e^{-\alpha y_i h(x_i)} \\ &= (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{n} w_i'(t) \ell^{0/1}(y_i, h(x_i)) \\ &\quad + e^{-\alpha} \sum_{i=1}^{n} w_i'(t)\end{aligned}$$

- The minimizer h_t in h is independent of α and is also the minimizer of

$$\sum_{i=1}^{n} w_i'(t) \ell^{0/1}(y_i, h(x_i))$$

Proof

- The optimal α_t is then given by
$$\alpha_t = \frac{1}{2} \log \frac{1-\epsilon'_t}{\epsilon'_t}$$
with $\epsilon'_t = (\sum_{i=1}^n w'_i(t) \ell^{0/1}(y_i, h_t(x_i)))/(\sum_{i=1}^n w'_i(t))$
- One verify then by recursion that
$$w_i(t) = w'_i(t)/(\sum_{i=1}^n w'_i(t))$$
and thus the two procedures are equivalent!

Revisited AdaBoost

AdaBoost

- Set $t = 0$ and $f = 0$.
- For $t = 1$ to T,
  - $(h_t, \alpha_t) = \operatorname{argmin}_{h,\alpha} \sum_{i=1}^{n} e^{-y_i(f(x_i)+\alpha h(x_i))}$
  - $f = f + \alpha_t h_t$
- Use $f = \sum_{t=1}^{T} \alpha_t h_t$

- **Greedy 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 and T should be chosen with care...

Outline

1. Trees
   - Questions, Trees and Partitions
   - Classification and Regression Trees
   - Tree construction: Branching and Pruning
2. Bagging and Forest
   - Bagging and Bootstrap
   - Construction rules
3. Boosting
   - AdaBoost as a Greedy Scheme
   - Generic Boosting
   - Gradient Boosting
   - Variations

Generic Boosting

- **Greedy optim.** yielding a linear combination of *weak* learners.

Generic Boosting

- Algorithm:
  - Set $t = 0$ and $f = 0$.
  - For $t = 1$ to T,
    - $(h_t, \alpha_t) = \operatorname{argmin}_{h,\alpha} \sum_{i=1}^{n} \ell'(y_i, f(x_i) + \alpha h(x_i))$
    - $f = f + \alpha_t h_t$
  - Use $f = \sum_{t=1}^{T} \alpha_t h_t$
- AKA as **Forward Stagewise Additive Modeling**
  - AdaBoost with $\ell'(y,h) = e^{-yh}$
  - LogitBoost with $\ell'(y,h) = \log(1 + e^{-yh})$
  - L_2Boost with $\ell'(y,h) = (y-h)^2$ (Matching pursuit)
  - L_1Boost 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_2Boost...

Outline

1. Trees
    - Questions, Trees and Partitions
    - Classification and Regression Trees
    - Tree construction: Branching and Pruning
2. Bagging and Forest
    - Bagging and Bootstrap
    - Construction rules
3. Boosting
    - AdaBoost as a Greedy Scheme
    - Generic Boosting
    - Gradient Boosting
    - Variations

Gradient Boosting

- **Issue:** At each boosting step, one need to solve
$$(h_t, \alpha_t) = \operatorname*{argmin}_{h,\alpha} \sum_{i=1}^{n} \ell'(y_i, f(x_i) + \alpha h) = L(y, f + \alpha h)$$
- **Idea:** Replace the function by a **first order approximation**
$$L(y, f + \alpha h) \sim L(y, f) + \alpha \langle \nabla L(y, f), h \rangle$$

Gradient Boosting

- Replace the minimization step by a **gradient descent** step:
  - Choose h_t as the best possible descent direction in $\mathcal{H}$ according to the approximation
  - Choose α_t that minimizes $L(y, f + \alpha h_t)$ (line search)

- **Rk:** Exact gradient direction often not possible!
- Need to find efficiently this best possible direction...

Best Direction

- Gradient direction:

$$\nabla L(y, f) \quad \text{with} \quad \nabla_i L(y, f) = \frac{\partial}{dh(x_i)} \left(\sum_{i'=1}^{n} \ell'(y_{i'}, f(x_{i'}) + h(x_{i'})) \right)$$

$$= \frac{\partial}{dh} (\ell')(y_i, f(x_i))$$

Best Direction within $\mathcal{H}$

- Direct formulation:

$$h_t \in \underset{h \in \mathcal{H}}{\operatorname{argmin}} \frac{\sum_{i=1}^{n} \nabla_i L(y, f) h(x_i)}{\sqrt{\sum_{i=1}^{n} |h(x_i)|^2}} \left(= \frac{\langle \nabla L(y, f), h \rangle}{\|h\|} \right)$$

- Equivalent (least-squares) formulation: $h_t = -\beta_t h'_t$ with

$$(\beta_t, h'_t) \in \underset{(\beta, h) \in \mathbb{R} \times \mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{n} |\nabla_i L(y, f) - \beta h(x_i)|^2 \left(= \|\nabla - \beta h\|^2 \right)$$

- Choice of the formulation will depends on $\mathcal{H}$...

Gradient Boosting of Classifiers

- **Assumptions:**
  - h is a binary classifier, $h(x) = \pm 1$ and thus $\|h\| = n$.
  - $\ell'(y,f) = l(-yf)$ so that $\nabla_i L(y,f) = -y_i(\frac{\partial}{dx} l)(-y_i f(x_i))$.
- Best direction h_t in $\mathcal{H}$ using the first formulation

$$h_t = \operatorname*{argmin}_{h \in \mathcal{H}} \sum_i \nabla_i L(y,f) h(x_i)$$

AdaBoost Type Minimization

- Best direction rewriting

$$\begin{aligned} h_t &= \operatorname*{argmin}_{h \in \mathcal{H}} - \sum_i (\frac{\partial}{dx} l)(-y_i f(x_i)) y_i h(x_i) \\ &= \operatorname*{argmin}_{h \in \mathcal{H}} \sum_i (\frac{\partial}{dx} l)(-y_i f(x_i))(2\ell^{0/1}(y_i, h(x_i)) - 1) \end{aligned}$$

- **AdaBoost type weighted loss minimization** as soon as $(\frac{\partial}{dx} l)(-y_i f(x_i)) \geq 0$:

$$h_t = \operatorname{argmin} \sum_i (\frac{\partial}{dx} l)(-y_i f(x_i)) \ell^{0/1}(y_i, h(x_i))$$

Gradient Boosting of Classifiers

Gradient Boosting

- **(Gradient) AdaBoost:** $\ell'(y,f) = \exp(-yf)$
    - $l(x) = \exp(x)$ and thus $(\frac{\partial}{dx} l)(-y_i f(x_i)) = e^{-y_i f(x_i)} \geq 0$
    - h_t is the same than in AdaBoost
    - α_t also... (explicit computation)
- **LogitBoost:** $\ell'(y,f) = \log(1+e^{-yf})$
    - $l(x) = \log(1+e^x)$ and thus $(\frac{\partial}{dx} l)(-y_i f(x_i)) = \frac{e^{-y_i f(x_i)}}{1+e^{-y_i f(x_i)}} \geq 0$
    - Less weights on missclassified samples than in AdaBoost...
    - No explicit formula for α_t (line search)
    - Different path than with the (non computable) classical boosting!
- **SoftBoost:** $\ell'(y,f) = \max(1-yf,0)$
    - $l(x) = \max(1+x,0)$ and $(\frac{\partial}{dx} l)(-y_i f(x_i)) = \mathbf{1} y_i f(x_i) \leq 1$
    - Do not use the samples that are sufficiently well classified!

Gradient Boosting and Least Squares

- Least squares formulation is often prefered when $|h| \neq 1$.

Least Squares Gradient Boosting

- Find $h_t = -\beta_t h'_t$ with

$$(\beta_t, h'_t) \in \operatorname*{argmin}_{(\beta,h)\in\mathbb{R}\times\mathcal{H}} \sum_{i=1}^{n} |\nabla_i L(y, f) - \beta h(x_i)|^2$$

- Classical least squares if $\mathcal{H}$ is a finite dimensional vector space!
- Not a usual least squares in general but a classical regression problem!

- Numerical scheme depends on the loss...

Gradient Boosting and Least Squares

Examples

- **Gradient L_2Boost:**
  - $\ell(y, f) = |y - f|^2$ and $\nabla_i L(y_i, f(x_i)) = -2(y_i - f(x_i))$:
    $$(\beta_t, h'_t) \in \operatorname*{argmin}_{(\beta,h)\in\mathbb{R}\times\mathcal{H}} \sum_{i=1}^{n} |2y_i - 2(f(x_i) - \beta/2h(x_i))|^2$$
  - $\alpha_t = -\beta_t/2$
  - Equivalent to classical L_2-Boosting
- **Gradient L_1Boost:**
  - $\ell(y, f) = |y - f|$ and $\nabla_i L(y_i, f(x_i)) = -\mathrm{sign}(y_i - f(x_i))$:
    $$(\beta_t, h'_t) \in \operatorname*{argmin}_{(\beta,h)\in\mathbb{R}\times\mathcal{H}} \sum_{i=1}^{n} |-\mathrm{sign}(y_i - f(x_i)) - \beta h(x_i)|^2$$
  - Robust to outliers...

- Classical choice for $\mathcal{H}$: Generalized Additive Model in which each h depends on a small subset of variables.

Gradient Boosting and Least Squares

- Least squares formulation can also be used in classification!
- Assumption:
  - $\ell(x,y) = l(-xy)$ so that $\nabla_i L(y_i, f(x_i)) = -y_i(\frac{\partial}{dx} l)(-y_i f(x_i))$

Least Squares Gradient Boosting for Classifiers

- Least Squares formulation:

$$(\beta_t, h'_t) \in \operatorname*{argmin}_{(\beta,h)\in\mathbb{R}\times\mathcal{H}} \sum_{i=1}^{n} |-y_i(\frac{\partial}{dx} l)(-y_i f(x_i)) - \beta h(x_i)|^2$$

- Equivalent formulation:

$$(\beta_t, h'_t) \in \operatorname*{argmin}_{(\beta,h)\in\mathbb{R}\times\mathcal{H}} \sum_{i=1}^{n} |-(\frac{\partial}{dx} l)(y_i f(x_i)) + \beta y_i h(x_i))|^2$$

- **Intuition:** Modify mis-classified examples without modifying too much the well-classified ones...

Outline

1. Trees
   - Questions, Trees and Partitions
   - Classification and Regression Trees
   - Tree construction: Branching and Pruning
2. Bagging and Forest
   - Bagging and Bootstrap
   - Construction rules
3. Boosting
   - AdaBoost as a Greedy Scheme
   - Generic Boosting
   - Gradient Boosting
   - Variations

Boosting Variations

Stochastic Boosting

- **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...

Second Order Boosting

- Replace the first order approximation by a second order one and avoid the line search...

XGBoost

- Very efficient boosting algorithm proposed by Chen and Guestrin in 2014.

eXtreme Gradient Boosting

- Gradient boosting for a (penalized) smooth loss using a second order approximation and the least squares approximation.
- Reduced stepsize with a shrinkage of the *optimal* parameter.
- Feature subsampling.
- Weak learners:
  - Trees: limited depth, penalized size and parameters, fast approximate best split.
  - Linear model: elastic-net penalization.

- Excellent baseline!