

Boosting

Victor Kitov

v.v.kitov@yandex.ru

Yandex School of Data Analysis



General idea

- Bagging, Random forest, extra random trees fit independent models.
 - can be done in parallel
 - sampling: objects - with replacement, features - without replacement.
 - base learners-complex
- Boosting fits models sequentially and they depend on each other.
 - each model is fitted to correct errors of the ensemble of previous models.
 - base learners-simple

Linear ensembles

Linear ensemble:

$$F(x) = f_0(x) + c_1 h_1(x) + \dots + c_M h_M(x)$$

Regression: $\hat{y}(x) = F(x)$

Binary classification: $\text{score}(y|x) = F(x)$, $\hat{y}(x) = \text{sign } F(x)$

- Notation: $h_1(x), \dots, h_M(x)$ are called *base learners*, *weak learners*, *base models*.
- Too expensive to optimize $f_0(x), h_1(x), \dots, h_M(x)$ and c_1, \dots, c_M jointly for large M .
- Idea: optimize $f_0(x)$ and then each pair $(h_m(x), c_m)$ greedily.

Forward stagewise additive modeling (FSAM)

Input: training dataset (x_i, y_i) , $i = 1, 2, \dots, N$; loss function $\mathcal{L}(f, y)$, parametrized base learner $h(x|\theta)$ and initial approximation $f(x|\theta')$, the number M of successive additive approximations.

Forward stagewise additive modeling (FSAM)

Input: training dataset (x_i, y_i) , $i = 1, 2, \dots, N$; loss function $\mathcal{L}(f, y)$, parametrized base learner $h(x|\theta)$ and initial approximation $f(x|\theta')$, the number M of successive additive approximations.

- 1 Fit initial approximation $f_0(x) = \arg \min_f \sum_{i=1}^N \mathcal{L}(f(x_i), y_i)$

Forward stagewise additive modeling (FSAM)

Input: training dataset (x_i, y_i) , $i = 1, 2, \dots, N$; loss function $\mathcal{L}(f, y)$, parametrized base learner $h(x|\theta)$ and initial approximation $f(x|\theta')$, the number M of successive additive approximations.

- ① Fit initial approximation $f_0(x) = \arg \min_f \sum_{i=1}^N \mathcal{L}(f(x_i), y_i)$
- ② For $m = 1, 2, \dots, M$:
 - ① find next best classifier

$$(c_m, h_m) = \arg \min_{h, c} \sum_{i=1}^N \mathcal{L}(f_{m-1}(x_i) + ch(x_i), y_i)$$

Forward stagewise additive modeling (FSAM)

Input: training dataset (x_i, y_i) , $i = 1, 2, \dots, N$; loss function $\mathcal{L}(f, y)$, parametrized base learner $h(x|\theta)$ and initial approximation $f(x|\theta')$, the number M of successive additive approximations.

- 1 Fit initial approximation $f_0(x) = \arg \min_f \sum_{i=1}^N \mathcal{L}(f(x_i), y_i)$
- 2 For $m = 1, 2, \dots, M$:
 - 1 find next best classifier

$$(c_m, h_m) = \arg \min_{h, c} \sum_{i=1}^N \mathcal{L}(f_{m-1}(x_i) + ch(x_i), y_i)$$

- 2 set

$$f_m(x) = f_{m-1}(x) + c_m h_m(x)$$

Output: approximation function

$$f_M(x) = f_0(x) + \sum_{m=1}^M c_m h_m(x)$$

Comments on FSAM

- Number of steps M should be determined by performance on validation set.
- Step 1 need not be solved accurately, since its mistakes are expected to be corrected by future base learners.
 - we can take $f_0(x) = \arg \min_{\beta \in \mathbb{R}} \sum_{i=1}^N \mathcal{L}(\beta, y_i)$ or simply $f_0(x) \equiv 0$.
- By similar reasoning there is no need to solve 2.1 accurately
 - typically very simple base learners are used such as trees of depth=1,2,3.
- For some loss functions, such as $\mathcal{L}(y, f(x)) = e^{-yf(x)}$ we can solve FSAM explicitly.
- For general loss functions gradient boosting scheme should be used.

Adaboost (discrete version): assumptions

- binary classification task $y \in \{+1, -1\}$
- family of base classifiers $h(x) = h(x|\theta)$ where θ is some fitted parametrization.
- $h(x) \in \{+1, -1\}$
- classification is performed with
$$\hat{y} = \text{sign}\{f_0(x) + c_1 f_1(x) + \dots + c_M f_M(x)\}$$
- optimized loss is $L(y, f(x)) = e^{-yf(x)}$
- FSAM is applied

Adaboost (discrete version): algorithm

Input: training dataset (x_i, y_i) , $i = 1, 2, \dots, N$; number of additive weak classifiers M , a family of weak classifiers $h(x) \in \{+1, -1\}$, trainable on weighted datasets.

- ① Initialize observation weights $w_i = 1/n$, $i = 1, 2, \dots, n$.
- ② for $m = 1, 2, \dots, M$:
 - ① fit $h^m(x)$ to training data using weights w_i
 - ② compute weighted misclassification rate:

$$E_m = \frac{\sum_{i=1}^N w_i \mathbb{I}[h^m(x) \neq y_i]}{\sum_{i=1}^N w_i}$$

- ③ if $E_m > 0.5$ or $E_m = 0$: terminate procedure.
- ④ compute $\alpha_m = \ln((1 - E_m)/E_m)$
- ⑤ increase all weights, where misclassification with $h^m(x)$ was made:

$$w_i \leftarrow w_i e^{\alpha_m}, i \in \{i : h^m(x_i) \neq y_i\}$$

Output: composite classifier $f(x) = \text{sign} \left(\sum_{m=1}^M \alpha_m h^m(x) \right)$

Table of Contents

1 Gradient boosting

Motivation

- Problem: For general loss function L FSAM cannot be solved explicitly
- Analogy with function minimization: when we can't find optimum explicitly we use numerical methods
- Gradient boosting: numerical method for iterative loss minimization

Gradient descent algorithm

$$F(w) \rightarrow \min_w, \quad w \in \mathbb{R}^N$$

Gradient descend algorithm:

INPUT:

η -parameter, controlling the speed of convergence

M -number of iterations

ALGORITHM:

initialize w

for $m = 1, 2, \dots, M$:

$$\Delta w = \frac{\partial F(w)}{\partial w}$$

$$w = w - \eta \Delta w$$

Modified gradient descent algorithm

INPUT:

M -number of iterations

ALGORITHM:

initialize w

for $m = 1, 2, \dots, M$:

$$\Delta w = \frac{\partial F(w)}{\partial w}$$

$$c^* = \arg \min_c F(w - c\Delta w)$$

$$w = w - c^* \Delta w$$

Gradient boosting

- Now consider $F(f(x_1), \dots, f(x_N)) = \sum_{n=1}^N \mathcal{L}(f(x_n), y_n)$
- Gradient descent performs pointwise optimization, but we need generalization, so we optimize in space of functions.
- Gradient boosting implements modified gradient descent in function space:
 - find $z_i = -\frac{\partial \mathcal{L}(r, y)}{\partial r} \big|_{r=f^{m-1}(x)}$
 - fit base learner $h_m(x)$ to $\{(x_i, z_i)\}_{i=1}^N$

Gradient boosting

Input: training dataset (x_i, y_i) , $i = 1, 2, \dots, N$; loss function $\mathcal{L}(f, y)$ and the number M of successive additive approximations.

- 1 Fit initial approximation $f_0(x)$ (might be taken $f_0(x) \equiv 0$)

Gradient boosting

Input: training dataset (x_i, y_i) , $i = 1, 2, \dots, N$; loss function $\mathcal{L}(f, y)$ and the number M of successive additive approximations.

- 1 Fit initial approximation $f_0(x)$ (might be taken $f_0(x) \equiv 0$)
- 2 For each step $m = 1, 2, \dots, M$:

Gradient boosting

Input: training dataset (x_i, y_i) , $i = 1, 2, \dots, N$; loss function $\mathcal{L}(f, y)$ and the number M of successive additive approximations.

- ❶ Fit initial approximation $f_0(x)$ (might be taken $f_0(x) \equiv 0$)
- ❷ For each step $m = 1, 2, \dots, M$:
 - ❶ calculate derivatives $z_i = -\frac{\partial \mathcal{L}(r, y_i)}{\partial r} \Big|_{r=f^{m-1}(x_i)}$

Gradient boosting

Input: training dataset (x_i, y_i) , $i = 1, 2, \dots, N$; loss function $\mathcal{L}(f, y)$ and the number M of successive additive approximations.

- ❶ Fit initial approximation $f_0(x)$ (might be taken $f_0(x) \equiv 0$)
- ❷ For each step $m = 1, 2, \dots, M$:
 - ❶ calculate derivatives $z_i = -\frac{\partial \mathcal{L}(r, y_i)}{\partial r} \Big|_{r=f^{m-1}(x_i)}$
 - ❷ fit h_m to $\{(x_i, z_i)\}_{i=1}^N$, for example by solving

$$\sum_{n=1}^N (h_m(x_n) - z_n)^2 \rightarrow \min_{h_m}$$

Gradient boosting

Input: training dataset (x_i, y_i) , $i = 1, 2, \dots, N$; loss function $\mathcal{L}(f, y)$ and the number M of successive additive approximations.

- ① Fit initial approximation $f_0(x)$ (might be taken $f_0(x) \equiv 0$)
- ② For each step $m = 1, 2, \dots, M$:
 - ① calculate derivatives $z_i = -\frac{\partial \mathcal{L}(r, y_i)}{\partial r} \Big|_{r=f^{m-1}(x_i)}$
 - ② fit h_m to $\{(x_i, z_i)\}_{i=1}^N$, for example by solving

$$\sum_{n=1}^N (h_m(x_n) - z_n)^2 \rightarrow \min_{h_m}$$

-
-
- ③ solve univariate optimization problem:

$$\sum_{i=1}^N \mathcal{L}(f_{m-1}(x_i) + c_m h_m(x_i), y_i) \rightarrow \min_{c_m \in \mathbb{R}_+}$$

Gradient boosting

Input: training dataset (x_i, y_i) , $i = 1, 2, \dots, N$; loss function $\mathcal{L}(f, y)$ and the number M of successive additive approximations.

- ① Fit initial approximation $f_0(x)$ (might be taken $f_0(x) \equiv 0$)
- ② For each step $m = 1, 2, \dots, M$:
 - ① calculate derivatives $z_i = -\frac{\partial \mathcal{L}(r, y_i)}{\partial r} \Big|_{r=f^{m-1}(x_i)}$
 - ② fit h_m to $\{(x_i, z_i)\}_{i=1}^N$, for example by solving

$$\sum_{n=1}^N (h_m(x_n) - z_n)^2 \rightarrow \min_{h_m}$$

- ③ solve univariate optimization problem:

$$\sum_{i=1}^N \mathcal{L}(f_{m-1}(x_i) + c_m h_m(x_i), y_i) \rightarrow \min_{c_m \in \mathbb{R}_+}$$

- ④ set $f_m(x) = f_{m-1}(x) + c_m h_m(x)$

Gradient boosting

Input: training dataset (x_i, y_i) , $i = 1, 2, \dots, N$; loss function $\mathcal{L}(f, y)$ and the number M of successive additive approximations.

- ① Fit initial approximation $f_0(x)$ (might be taken $f_0(x) \equiv 0$)
- ② For each step $m = 1, 2, \dots, M$:
 - ① calculate derivatives $z_i = -\frac{\partial \mathcal{L}(r, y_i)}{\partial r} \Big|_{r=f^{m-1}(x_i)}$
 - ② fit h_m to $\{(x_i, z_i)\}_{i=1}^N$, for example by solving

$$\sum_{n=1}^N (h_m(x_n) - z_n)^2 \rightarrow \min_{h_m}$$

- ③ solve univariate optimization problem:

$$\sum_{i=1}^N \mathcal{L}(f_{m-1}(x_i) + c_m h_m(x_i), y_i) \rightarrow \min_{c_m \in \mathbb{R}_+}$$

- ④ set $f_m(x) = f_{m-1}(x) + c_m h_m(x)$

Output: approximation function $f_M(x) = f_0(x) + \sum_{m=1}^M c_m h_m(x)$

Gradient boosting: examples

In gradient boosting

$$\sum_{n=1}^N \left(h_m(x_n) - \left(-\frac{\partial \mathcal{L}(r, y_n)}{\partial r} \Big|_{r=f^{m-1}(x_n)} \right) \right)^2 \rightarrow \min_{h_m}$$

Commonly used loss-functions¹:

- $\mathcal{L} = \frac{1}{2} (r - y)^2$
- $\mathcal{L} = e^{-ry}$
- $\mathcal{L} = [-ry]_+$
- $\mathcal{L} = \ln(1 + e^{-ry})$

¹Estimate targets for them.

Gradient boosting of trees

Input: training dataset (x_i, y_i) , $i = 1, 2, \dots, N$; loss function $\mathcal{L}(f, y)$ and the number M of successive additive approximations.

- 1 Fit constant initial approximation $f_0(x)$:

$$f_0(x) = \arg \min_{\gamma} \sum_{i=1}^N \mathcal{L}(\gamma, y_i)$$

Gradient boosting of trees

Input: training dataset (x_i, y_i) , $i = 1, 2, \dots, N$; loss function $\mathcal{L}(f, y)$ and the number M of successive additive approximations.

- 1 Fit constant initial approximation $f_0(x)$:

$$f_0(x) = \arg \min_{\gamma} \sum_{i=1}^N \mathcal{L}(\gamma, y_i)$$

- 2 For each step $m = 1, 2, \dots, M$:

Gradient boosting of trees

Input: training dataset (x_i, y_i) , $i = 1, 2, \dots, N$; loss function $\mathcal{L}(f, y)$ and the number M of successive additive approximations.

- ❶ Fit constant initial approximation $f_0(x)$:
$$f_0(x) = \arg \min_{\gamma} \sum_{i=1}^N \mathcal{L}(\gamma, y_i)$$
- ❷ For each step $m = 1, 2, \dots, M$:
 - ❶ calculate derivatives $z_i = -\frac{\partial \mathcal{L}(r, y_i)}{\partial r} \Big|_{r=f^{m-1}(x_i)}$

Gradient boosting of trees

Input: training dataset (x_i, y_i) , $i = 1, 2, \dots, N$; loss function $\mathcal{L}(f, y)$ and the number M of successive additive approximations.

- ① Fit constant initial approximation $f_0(x)$:
$$f_0(x) = \arg \min_{\gamma} \sum_{i=1}^N \mathcal{L}(\gamma, y_i)$$
- ② For each step $m = 1, 2, \dots, M$:
 - ① calculate derivatives $z_i = -\frac{\partial \mathcal{L}(r, y_i)}{\partial r} \Big|_{r=f^{m-1}(x_i)}$
 - ② fit regression tree h^m on $\{(x_i, z_i)\}_{i=1}^N$ with some loss function, get leaf regions $\{R_{jm}\}_{j=1}^{J_m}$.

Gradient boosting of trees

Input: training dataset (x_i, y_i) , $i = 1, 2, \dots, N$; loss function $\mathcal{L}(f, y)$ and the number M of successive additive approximations.

- ① Fit constant initial approximation $f_0(x)$:

$$f_0(x) = \arg \min_{\gamma} \sum_{i=1}^N \mathcal{L}(\gamma, y_i)$$
- ② For each step $m = 1, 2, \dots, M$:
 - ① calculate derivatives $z_i = -\frac{\partial \mathcal{L}(r, y_i)}{\partial r} \Big|_{r=f^{m-1}(x_i)}$
 - ② fit regression tree h^m on $\{(x_i, z_i)\}_{i=1}^N$ with some loss function, get leaf regions $\{R_{jm}\}_{j=1}^{J_m}$.
 - ③ for each terminal region R_{jm} , $j = 1, 2, \dots, J_m$ solve univariate optimization problem:

$$\gamma_{jm} = \arg \min_{\gamma} \sum_{x_i \in R_{jm}} \mathcal{L}(f_{m-1}(x_i) + \gamma, y_i)$$

Gradient boosting of trees

Input: training dataset (x_i, y_i) , $i = 1, 2, \dots, N$; loss function $\mathcal{L}(f, y)$ and the number M of successive additive approximations.

- ① Fit constant initial approximation $f_0(x)$:

$$f_0(x) = \arg \min_{\gamma} \sum_{i=1}^N \mathcal{L}(\gamma, y_i)$$
- ② For each step $m = 1, 2, \dots, M$:
 - ① calculate derivatives $z_i = -\frac{\partial \mathcal{L}(r, y_i)}{\partial r} \Big|_{r=f_{m-1}(x_i)}$
 - ② fit regression tree h^m on $\{(x_i, z_i)\}_{i=1}^N$ with some loss function, get leaf regions $\{R_{jm}\}_{j=1}^{J_m}$.
 - ③ for each terminal region R_{jm} , $j = 1, 2, \dots, J_m$ solve univariate optimization problem:

$$\gamma_{jm} = \arg \min_{\gamma} \sum_{x_i \in R_{jm}} \mathcal{L}(f_{m-1}(x_i) + \gamma, y_i)$$

- ④ update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} \mathbb{I}[x \in R_{jm}]$

Gradient boosting of trees

Input: training dataset (x_i, y_i) , $i = 1, 2, \dots, N$; loss function $\mathcal{L}(f, y)$ and the number M of successive additive approximations.

- ① Fit constant initial approximation $f_0(x)$:

$$f_0(x) = \arg \min_{\gamma} \sum_{i=1}^N \mathcal{L}(\gamma, y_i)$$
- ② For each step $m = 1, 2, \dots, M$:
 - ① calculate derivatives $z_i = -\frac{\partial \mathcal{L}(r, y_i)}{\partial r} \Big|_{r=f_{m-1}(x_i)}$
 - ② fit regression tree h^m on $\{(x_i, z_i)\}_{i=1}^N$ with some loss function, get leaf regions $\{R_{jm}\}_{j=1}^{J_m}$.
 - ③ for each terminal region R_{jm} , $j = 1, 2, \dots, J_m$ solve univariate optimization problem:

$$\gamma_{jm} = \arg \min_{\gamma} \sum_{x_i \in R_{jm}} \mathcal{L}(f_{m-1}(x_i) + \gamma, y_i)$$

- ④ update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} \mathbb{I}[x \in R_{jm}]$

Output: approximation function $f_M(x)$

Modification of boosting for trees

- Compared to first method of gradient boosting, boosting of regression trees finds additive coefficients individually for each terminal region R_{jm} , not globally for the whole classifier $h^m(x)$.
- This is done to increase accuracy: forward stagewise algorithm cannot be applied to find R_{jm} , but it can be applied to find γ_{jm} , because second task is solvable for arbitrary L .
- Max leaves J
 - interaction between no more than $J - 1$ terms
 - usually $J \leq 8$
 - M controls underfitting-overfitting tradeoff and selected using validation set

Shrinkage & subsampling

- Shrinkage of general GB, step (d):

$$f_m(x) = f_{m-1}(x) + \nu c_m h_m(x)$$

- Shrinkage of trees GB, step (d):

$$f_m(x) = f_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} \mathbb{I}[x \in R_{jm}]$$

- Comments:
 - $\nu \in (0, 1]$
 - $\nu \downarrow \implies M \uparrow$
- Subsampling
 - increases speed of fitting
 - may increase accuracy

Types of boosting

- Loss function F :
 - $\mathcal{L}(|f(x) - y|)$ - regression
 - $\mathcal{L}(y \cdot \text{score}(y = +1|x))$ - binary classification
 - multiclass classification - may extend with one vs. all scheme
- Optimization
 - analytical (AdaBoost)
 - gradient based
- Base learners: continuous, discrete
- Extensions: shrinkage, subsampling