Boosting

Victor Kitov
v.v.kitov@yandex.ru

Yandex School of Data Analysis

Table of Contents

Linear ensembles

Linear ensemble:

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

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

- Notation: $h_1(x), ...h_M(x)$ are called *base learners, weak learners, base models*.
- Too expensive to optimize $f_0(x), h_1(x), ...h_M(x)$ and $c_1, ...c_M$ jointly for large M.
- Idea: optimize $f_0(x)$ and then each pair $(h_m(x),\ c_m)$ greedily.
- After ensemble is built we can fine-tune $c_1, ...c_M$ by fitting features $f_0(x), h_1(x), ...h_M(x)$ with linear regression/classifier.

Forward stagewise additive modeling (FSAM)

Input: training dataset (x_i, y_i), $i = 1, 2, ...N$; loss function $L(f, y)$, general form of "base learner" $h(x|\gamma)$ (dependent from parameter γ) and the number M of successive additive approximations.

1. Fit initial approximation $f_0(x) = \arg\min_f \sum_{i=1}^{N} L(f(x_i), y_i)$
2. For $m = 1, 2, ...M$:
    1. find next best classifier
    $$(c_m, h_m) = \arg\min_{h,c} \sum_{i=1}^{N} L(f_{m-1}(x_i) + ch(x_i),\ y_i)$$
    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 $f_0(x) = \arg\min_{\beta \in \mathbb{R}} \sum_{i=1}^{N} L(\beta, y_i)$ 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 $L(y, f(x)) = e^{-yf(x)}$ 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 $h(x) = h(x|\gamma)$ where γ is some fitted parametrization.
- $h(x) \in \{+1, -1\}$
- classification is performed with $\widehat{y} = sign\{f_0(x) + c_1 f_1(x) + ... + c_M f_M(x)\}$
- optimized loss is $L(y, f(x)) = e^{-yf(x)}$
- FSAM is applied

Adaboost (discrete version): algorithm

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

1. Initialize observation weights $w_i = 1/n$, $i = 1, 2, ...n$.
2. for $m = 1, 2, ...M$:
    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 $\alpha_m = \ln\left((1 - E_m)/E_m\right)$
    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

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

$$F(w) \to \min_{w}, \quad w \in \mathbb{R}^{N}$$

Gradient descend algorithm:

```
**INPUT**:
$\eta$-parameter, controlling the speed of convergence
$M$-number of iterations

**ALGORITHM**:
initialize $w$
for $m = 1, 2, ...M$:
    $\Delta w = \frac{\partial F(w)}{\partial w}$
    $w = w - \eta \Delta w$
```

Modified gradient descent algorithm

```
INPUT:
$M$-number of iterations

ALGORITHM:
initialize $w$
for $m = 1, 2, ...M$:
    $\Delta w = \frac{\partial F(w)}{\partial w}$
    $c^* = \arg\min_c F(w - c\Delta w)$
    $w = w - c^*\Delta w$
```

Gradient boosting

- Now consider $F\left(f(x_1), ...f(x_N)\right) = \sum_{n=1}^{N} L\left(f(x_n), y_n\right)$
- 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 $z_i = -\frac{\partial L(r,y)}{\partial r}\big|_{r=f^{m-1}(x)}$
    - fit base learner $h_m(x)$ to $\{(x_i, z_i)\}_{i=1}^{N}$

Gradient boosting

Input: training dataset (x_i, y_i), $i = 1, 2, ...N$; loss function $L(f, y)$ 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), $i = 1, 2, ...N$; loss function $L(f, y)$ 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 $m = 1, 2, ...M$:

Gradient boosting

Input: training dataset (x_i, y_i), $i = 1, 2, ...N$; loss function $L(f, y)$ 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 $m = 1, 2, ...M$:
    1. calculate derivatives $z_i = -\frac{\partial L(r,y)}{\partial r}\big|_{r=f^{m-1}(x_i)}$

Gradient boosting

Input: training dataset (x_i, y_i), $i = 1, 2, ...N$; loss function $L(f, y)$ 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 $m = 1, 2, ...M$:
    1. calculate derivatives $z_i = -\frac{\partial L(r,y)}{\partial r}\big|_{r=f^{m-1}(x_i)}$
    2. fit h_m to $\{(x_i, z_i)\}_{i=1}^{N}$, for example by solving

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

Gradient boosting

Input: training dataset (x_i, y_i), $i = 1, 2, ...N$; loss function $L(f, y)$ 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 $m = 1, 2, ...M$:
    1. calculate derivatives $z_i = -\frac{\partial L(r,y)}{\partial r}|_{r=f^{m-1}(x_i)}$
    2. fit h_m to $\{(x_i, z_i)\}_{i=1}^N$, for example by solving
    $$\sum_{n=1}^{N} (h_m(x_n) - z_n)^2 \to \min_{h_m}$$
    3. solve univariate optimization problem:
    $$\sum_{i=1}^{N} L\left(f_{m-1}(x_i) + c_m h_m(x_i), y_i\right) \to \min_{c_m \in \mathbb{R}_+}$$

Gradient boosting

Input: training dataset (x_i, y_i), $i = 1, 2, ...N$; loss function $L(f, y)$ 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 $m = 1, 2, ...M$:
    1. calculate derivatives $z_i = -\frac{\partial L(r,y)}{\partial r}|_{r=f^{m-1}(x_i)}$
    2. fit h_m to $\{(x_i, z_i)\}_{i=1}^N$, for example by solving
    $$\sum_{n=1}^{N} (h_m(x_n) - z_n)^2 \to \min_{h_m}$$
    3. solve univariate optimization problem:
    $$\sum_{i=1}^{N} L\left(f_{m-1}(x_i) + c_m h_m(x_i), y_i\right) \to \min_{c_m \in \mathbb{R}_+}$$
    4. set $f_m(x) = f_{m-1}(x) + c_m h_m(x)$

Gradient boosting

Input: training dataset (x_i, y_i), $i = 1, 2, ...N$; loss function $L(f, y)$ 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 $m = 1, 2, ...M$:
    1. calculate derivatives $z_i = -\frac{\partial L(r,y)}{\partial r}|_{r=f^{m-1}(x_i)}$
    2. fit h_m to $\{(x_i, z_i)\}_{i=1}^N$, for example by solving
    $$\sum_{n=1}^{N} (h_m(x_n) - z_n)^2 \to \min_{h_m}$$
    3. solve univariate optimization problem:
    $$\sum_{i=1}^{N} L\left(f_{m-1}(x_i) + c_m h_m(x_i), y_i\right) \to \min_{c_m \in \mathbb{R}_+}$$
    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

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

Specific cases:

- $L=\frac{1}{2}(r-y)^2$ => $-\frac{\partial L}{\partial r}=-(r-y)=(y-r)$
    - $h_m(x)$ is fitted to compensate regression errors $(y-f_{m-1}(x))$
- $L=[-ry]_+$ => $-\frac{\partial L}{\partial r}=\begin{cases}0, & ry>0\\ y, & ry<0\end{cases}$
    - $h_m(x)$ is fitted to $y\mathbb{I}[f(x)y<0]$
- $L=\ln\left(1+e^{-ry}\right)$ => $-\frac{\partial L}{\partial r}=-\frac{-y}{1+e^{-ry}}e^{-ry}=\frac{y}{1+e^{ry}}$
    - $h_m(x)$ is fitted to $yp\left(-y|x\right)$ because for log-loss $p(y|x)=\frac{1}{1+e^{-f(x)y}}$
    - $p(-y|x)$ is probability of error on (x,y) pair

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 $J \leq 8$

Gradient boosting of trees

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

1. Fit constant initial approximation $f_0(x)$:
   $f_0(x) = \arg\min_\gamma \sum_{i=1}^N L(\gamma, y_i)$

Gradient boosting of trees

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

1. Fit constant initial approximation $f_0(x)$:
   $f_0(x) = \arg\min_\gamma \sum_{i=1}^N L(\gamma, y_i)$
2. For each step $m = 1, 2, ...M$:

Gradient boosting of trees

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

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

Gradient boosting of trees

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

1. Fit constant initial approximation $f_0(x)$:
   $f_0(x) = \arg\min_\gamma \sum_{i=1}^N L(\gamma, y_i)$
2. For each step $m = 1, 2, ...M$:
   1. calculate derivatives $z_i = -\frac{\partial L(r,y)}{\partial r}|_{r=f^{m-1}(x)}$
   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), $i = 1, 2, ...N$; loss function $L(f, y)$ and the number M of successive additive approximations.

1. Fit constant initial approximation $f_0(x)$:
   $f_0(x) = \arg\min_\gamma \sum_{i=1}^N L(\gamma, y_i)$
2. For each step $m = 1, 2, ...M$:
   1. calculate derivatives $z_i = -\frac{\partial L(r,y)}{\partial r}\big|_{r=f^{m-1}(x)}$
   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}, $j = 1, 2, ...J_m$ solve univariate optimization problem:

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

Gradient boosting of trees

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

1. Fit constant initial approximation $f_0(x)$:
   $f_0(x) = \arg\min_\gamma \sum_{i=1}^N L(\gamma, y_i)$
2. For each step $m = 1, 2, ...M$:
   1. calculate derivatives $z_i = -\frac{\partial L(r,y)}{\partial r}\Big|_{r=f^{m-1}(x)}$
   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}, $j = 1, 2, ...J_m$ solve univariate optimization problem:
   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L(f_{m-1}(x_i) + \gamma, y_i)$$
   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), $i = 1, 2, ...N$; loss function $L(f, y)$ and the number M of successive additive approximations.

1. Fit constant initial approximation $f_0(x)$:
   $f_0(x) = \arg\min_\gamma \sum_{i=1}^N L(\gamma, y_i)$
2. For each step $m = 1, 2, ...M$:
   1. calculate derivatives $z_i = -\frac{\partial L(r,y)}{\partial r}\big|_{r=f^{m-1}(x)}$
   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}, $j = 1, 2, ...J_m$ solve univariate optimization problem:
   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L(f_{m-1}(x_i) + \gamma, y_i)$$
   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 $f_M(x)$

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

Case of $C \geq 3$ classes

- Can fit C independent boostings (one vs. all scheme)
    - $\widehat{y} = \arg\max_y f_{my}(x)$
- Alternatively can optimize multivariate $L(f(x), y) = -\ln p(y|x)$
    - using linear or quadratic approximation
    - for quadratic approximation need to invert $\frac{\partial^2}{\partial r^2} F(r, y)\Big|_{r=f(x)}$.

      Can use diagonal approximation.

Types of boosting

- Loss function F:
  - $F(|f(x)-y|)$ - regression
  - $-\ln p(y|x)$ or $F(y\cdot score(y=+1|x))$ - binary classification
- Optimization
  - analytical (AdaBoost)
  - gradient based
- Base learners
  - continious
  - discrete
- Classification
  - binary
  - multiclass
- Extensions: shrinkage, subsampling