Boosting

Victor Kitov
v.v.kitov@yandex.ru

Yandex School of Data Analysis

Linear ensembles

Linear ensemble:

$$F_M(x) = f_0(x) + c_1 f_1(x) + ... + c_M f_M(x)$$

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

- Notation: $f_1(x), ...f_M(x)$ are called *base learners, weak learners, base models*.
- Too expensive to optimize $f_0(x), f_1(x), ...f_M(x)$ and $c_1, ...c_M$ jointly for large M.
- Idea: optimize $f_0(x)$ and then each pair $(f_m(x)$, $c_m)$ greedily.

Forward stagewise additive modeling (FSAM)

Input:

- training dataset (x_n, y_n), $n = 1, 2, ...N$
- loss function $\mathcal{L}(f, y)$
- parametric form of base learner $f(x|\gamma)$ (parametrized by γ)
- the number of base learners M.

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

Forward stagewise additive modeling (FSAM)

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

$$(c_m, f_m) = \arg\min_{f,c} \sum_{n=1}^{N} \mathcal{L}(F_{m-1}(x_n) + cf(x_n),\ y_n)$$

   - reevaluate ensemble

$$F_m(x) = F_{m-1}(x) + c_m f_m(x)$$

Comments

- M should be determined by performance on validation set.
    - may overfit!
- Each step should be coarse to leave room for future base learners improvement:
    - initial approximation may be zero or constant
    - optimization can be coarse (just few steps)
    - base learner should be simple
        - such as trees of depth=1,2,3.
- For some loss functions (see Adaboost) we can solve minimization explicitly.
- For general loss functions gradient boosting should be used.

Table of Contents

Adaboost (discrete version)

Assumptions:

- binary classification task $y \in \{+1, -1\}$
- $f_m(x) \in \{+1, -1\}$
- classification is performed with $\widehat{y} = sign\{f_0(x) + c_1 f_1(x) + ... + c_M f_M(x)\}$
- optimized loss is $\mathcal{L}(F(x), y) = e^{-yF(x)}$

Optimization in FSAM can be solved explicitly!

Adaboost (discrete version): algorithm

Input:

- training dataset (x_n, y_n), $n = 1, 2, ...N$
- number of additive weak classifiers M
- a family of weak classifiers $h(x) \in \{+1, -1\}$
  - should be trainable on weighted datasets.

Output: composite classifier $F_M(x) = \text{sign}\left(\sum_{m=1}^{M} c_m f_m(x)\right)$

Adaboost (discrete version): algorithm

1. Initialize observation weights $w_i = 1/N$, $i = 1, 2, ...N$.
2. for $m = 1, 2, ...M$:
    1. fit $f_m(x)$ to training data using weights w_i
    2. compute weighted misclassification rate:
    $$E_m = \frac{\sum_{i=1}^{N} w_i \mathbb{I}[f_m(x) \neq y_i]}{\sum_{i=1}^{N} w_i}$$
    3. if $E_M > 0.5$ or $E_M = 0$: terminate procedure.
    4. compute $c_m = \frac{1}{2} \ln\left((1 - E_m)/E_m\right)$
    5. increase all weights, where misclassification with $f_m(x)$ was made:
    $$w_i \leftarrow w_i e^{2c_m},\ i \in \{i:\ f_m(x_i) \neq y_i\}$$

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

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

Gradient descend algorithm

- based on approximation $L(w + \Delta w) \approx L(w) + g(w)\Delta w$ where $g(w) = \nabla_w L(w)$
- fit $\Delta w := -g(w)$

INPUT:
step size ε
number of iterations M

ALGORITHM:
initialize w
for $m = 1, 2, ...M$:
  $\Delta w = -g(w)$
  $w = w - \varepsilon \Delta w$

Modified gradient descent algorithm

INPUT:
number of iterations M

ALGORITHM:
initialize $w = (f_1, ...f_M)$
for $m = 1, 2, ...M$:
    $\Delta w = -g(w)$
    $c^* = \arg\min_{c>0} F(w - c\Delta w)$
    $w = w - c^* \Delta w$

Gradient boosting

- Now consider $L\left(f(x_1),...f(x_N)\right)=\sum_{n=1}^{N}\mathcal{L}\left(f(x_n),y_n\right)$
- Gradient descent performs pointwise optimization, but we need generalization, so we optimize in space of functions.
- Gradient boosting = modified gradient descent in function space:
  - find $z_n=-g(x_n)$, where $g(x_n)=\frac{\partial\mathcal{L}(r,y_n)}{\partial r}|_{r=f^{m-1}(x_n)}$
  - fit base learner $f_m(x)$ to $\{(x_n,z_n)\}_{n=1}^{N}$

Gradient boosting

Input: training dataset (x_n, y_n), $n = 1, 2, ...N$; 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_n, y_n), $n = 1, 2, ...N$; 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 $m = 1, 2, ...M$:

Gradient boosting

Input: training dataset (x_n, y_n), $n = 1, 2, ...N$; 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 $m = 1, 2, ...M$:
    1. calculate targets $z_n := -g_n$ $\quad \left(g_n = \frac{\partial \mathcal{L}(r, y_n)}{\partial r} \big|_{r=F_{m-1}(x_n)} \right)$

Gradient boosting

Input: training dataset (x_n, y_n), $n = 1, 2, ...N$; 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 $m = 1, 2, ...M$:
    1. calculate targets $z_n := -g_n$ $\quad \left(g_n = \frac{\partial \mathcal{L}(r, y_n)}{\partial r} \big|_{r = F_{m-1}(x_n)} \right)$
    2. fit $f_m(\cdot)$ to $\{(x_n, z_n)\}_{n=1}^N$, for example by solving

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

Gradient boosting

Input: training dataset (x_n, y_n), $n = 1, 2, ...N$; 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 $m = 1, 2, ...M$:
    1. calculate targets $z_n := -g_n$ $\quad \left(g_n = \frac{\partial \mathcal{L}(r, y_n)}{\partial r} \Big|_{r = F_{m-1}(x_n)} \right)$
    2. fit $f_m(\cdot)$ to $\{(x_n, z_n)\}_{n=1}^{N}$, for example by solving
    $$\sum_{n=1}^{N} (f_m(x_n) - z_n)^2 \to \min_{f_m}$$
    3. solve univariate optimization problem:
    $$\sum_{n=1}^{N} \mathcal{L}\left(F_{m-1}(x_n) + c_m f_m(x_n), y_n\right) \to \min_{c_m \in \mathbb{R}_+}$$

Gradient boosting

Input: training dataset (x_n, y_n), $n = 1, 2, ...N$; 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 $m = 1, 2, ...M$:
    1. calculate targets $z_n := -g_n$ $\quad \left(g_n = \frac{\partial \mathcal{L}(r, y_n)}{\partial r} \Big|_{r=F_{m-1}(x_n)} \right)$
    2. fit $f_m(\cdot)$ to $\{(x_n, z_n)\}_{n=1}^N$, for example by solving
    $$\sum_{n=1}^{N} (f_m(x_n) - z_n)^2 \to \min_{f_m}$$
    3. solve univariate optimization problem:
    $$\sum_{n=1}^{N} \mathcal{L}\left(F_{m-1}(x_n) + c_m f_m(x_n), y_n\right) \to \min_{c_m \in \mathbb{R}_+}$$
    4. set $F_m(x) = F_{m-1}(x) + c_m f_m(x)$

Gradient boosting

Input: training dataset (x_n, y_n), $n = 1, 2, ...N$; 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 $m = 1, 2, ...M$:
   1. calculate targets $z_n := -g_n$ $\left(g_n = \frac{\partial \mathcal{L}(r, y_n)}{\partial r} \Big|_{r=F_{m-1}(x_n)} \right)$
   2. fit $f_m(\cdot)$ to $\{(x_n, z_n)\}_{n=1}^N$, for example by solving
   $$\sum_{n=1}^{N} (f_m(x_n) - z_n)^2 \to \min_{f_m}$$
   3. solve univariate optimization problem:
   $$\sum_{n=1}^{N} \mathcal{L}\left(F_{m-1}(x_n) + c_m f_m(x_n), y_n\right) \to \min_{c_m \in \mathbb{R}_+}$$
   4. set $F_m(x) = F_{m-1}(x) + c_m f_m(x)$

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

Gradient boosting: examples

In gradient boosting

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

Sample cases:

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

Table of Contents

Gradient boosting of trees

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

1. Fit initial approximation *with constant*:
   $f_0(x) = \arg\min_\gamma \sum_{n=1}^{N} \mathcal{L}(\gamma, y_n)$

Gradient boosting of trees

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

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

Gradient boosting of trees

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

1. Fit initial approximation *with constant*:
   $f_0(x) = \arg\min_\gamma \sum_{n=1}^{N} \mathcal{L}(\gamma, y_n)$
2. For each step $m = 1, 2, ...M$:
   1. calculate targets $z_n := -g_n$ $\quad \left(g_n = \frac{\partial \mathcal{L}(r, y_n)}{\partial r}\Big|_{r=F_{m-1}(x_n)}\right)$

Gradient boosting of trees

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

1. Fit initial approximation *with constant*:
   $f_0(x) = \arg\min_\gamma \sum_{n=1}^{N} \mathcal{L}(\gamma,\ y_n)$
2. For each step $m = 1, 2, ...M$:
   1. calculate targets $z_n := -g_n \qquad \left(g_n = \frac{\partial \mathcal{L}(r, y_n)}{\partial r}\Big|_{r=F_{m-1}(x_n)}\right)$
   2. fit regression tree $f_m(\cdot)$ on $\{(x_n, z_n)\}_{n=1}^{N}$ with some loss function, get leaf regions $\{R_j^m\}_{j=1}^{J_m}$.

Gradient boosting of trees

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

1. Fit initial approximation *with constant*:
   $f_0(x) = \arg\min_\gamma \sum_{n=1}^{N} \mathcal{L}(\gamma,\, y_n)$
2. For each step $m = 1, 2, ...M$:
   1. calculate targets $z_n := -g_n$ $\quad\left(g_n = \frac{\partial \mathcal{L}(r, y_n)}{\partial r}\Big|_{r=F_{m-1}(x_n)}\right)$
   2. fit regression tree $f_m(\cdot)$ on $\{(x_n, z_n)\}_{n=1}^{N}$ with some loss function, get leaf regions $\{R_j^m\}_{j=1}^{J_m}$.
   3. for each terminal region R_j^m, $j = 1, 2, ...J_m$ solve univariate optimization problem:

$$\gamma_j^m = \arg\min_\gamma \sum_{x_i \in R_j^m} \mathcal{L}(F_{m-1}(x_i) + \gamma,\, y_i)$$

Gradient boosting of trees

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

1. Fit initial approximation *with constant*:
   $f_0(x) = \arg\min_\gamma \sum_{n=1}^{N} \mathcal{L}(\gamma,\ y_n)$
2. For each step $m = 1, 2, ...M$:
   1. calculate targets $z_n := -g_n \qquad \left(g_n = \frac{\partial \mathcal{L}(r, y_n)}{\partial r}\Big|_{r=F_{m-1}(x_n)}\right)$
   2. fit regression tree $f_m(\cdot)$ on $\{(x_n, z_n)\}_{n=1}^{N}$ with some loss function, get leaf regions $\{R_j^m\}_{j=1}^{J_m}$.
   3. for each terminal region R_j^m, $j = 1, 2, ...J_m$ solve univariate optimization problem:
   $$\gamma_j^m = \arg\min_\gamma \sum_{x_i \in R_j^m} \mathcal{L}(F_{m-1}(x_i) + \gamma,\ y_i)$$
   4. update $F_m(x) = F_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_j^m \mathbb{I}[x \in R_j^m]$

Gradient boosting of trees

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

1. Fit initial approximation *with constant*:
   $f_0(x) = \arg\min_\gamma \sum_{n=1}^N \mathcal{L}(\gamma,\, y_n)$
2. For each step $m = 1, 2, ...M$:
   1. calculate targets $z_n := -g_n$ $\quad \left(g_n = \frac{\partial \mathcal{L}(r, y_n)}{\partial r}\big|_{r=F_{m-1}(x_n)}\right)$
   2. fit regression tree $f_m(\cdot)$ on $\{(x_n, z_n)\}_{n=1}^N$ with some loss function, get leaf regions $\{R_j^m\}_{j=1}^{J_m}$.
   3. for each terminal region R_j^m, $j = 1, 2, ...J_m$ solve univariate optimization problem:
   $$\gamma_j^m = \arg\min_\gamma \sum_{x_i \in R_j^m} \mathcal{L}(F_{m-1}(x_i) + \gamma,\, y_i)$$
   4. update $F_m(x) = F_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_j^m \mathbb{I}[x \in R_j^m]$

Output: approximation function $F_M(x)$

Modification of boosting for trees

- Max leaves J
  - interaction between no more than $J-1$ terms
  - usually $4 \leq J \leq 8$
- M controls underfitting-overfitting tradeoff and selected using validation set

Shrinkage

- Shrinkage of general GB, step (d):

$$F_m(x) = F_{m-1}(x) + \alpha c_m f_m(x)$$

- Comments:
  - $\alpha \in (0, 1]$
  - $\alpha \downarrow \implies M \uparrow (\alpha M \approx const)$

xgBoost

- One of the most popular algorithms on kaggle.
- Uses decision trees as base learners:
  - $f_m \in \{f(x) = w_{q(x)}\}$,
  - T total number of leaves.
  - $q(x)$ maps $x \in \mathbb{R}^D$ to leaf number
  - $w \in \mathbb{R}^T$ predictions for leaves.

xgBoost

- Loss - 2nd order approximation with with regularization:

$$\begin{aligned}\mathcal{L}(f_m) &= \sum_{n=1}^{N}\mathcal{L}(F^{(m-1)}(x_n),\, y_n)\\ &\approx \sum_{n=1}^{N}\left[\mathcal{L}(F^{(m-1)}(x_n),\, y_n)+g_n f_m(x_n)+\frac{1}{2}h_n f_m^2(x_n)\right]\\ &\quad +\gamma T+\frac{1}{2}\lambda\sum_{t=1}^{T}w_t^2\end{aligned}$$

- Tree impurity function matches original loss $\mathcal{L}(\cdot,\cdot)$.
- Efficiency optimization:
    - feature values may be discretized for speed
    - parallelization over multiple CPU cores and with GPU

Types of boosting

- Loss function $\mathcal{L}$:
  - $\mathcal{L}(|f(x)-y|)$ - regression
  - $F(y \cdot score(y=+1|x))$ - binary classification
- Optimization
  - analytical (Adaboost)
  - gradient based
  - based on quadratic approximation
- Base learners
  - continious
  - discrete
- Shrinkage improves accuracy.