Gradient Boosting

Daeyoung Kim, Sreyas Mohan, Brett Bernstein

CDS at NYU

14 Apr 2021

Intro Question

Question

Suppose 10 different meteorologists have produced functions $f_1, \ldots, f_{10} : \mathbb{R}^d \to \mathbb{R}$ that forecast tomorrow's noon-time temperature using the same d features. Given a dataset containing 1000 data points $(x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$ of similar forecast situations, describe a method to forecast tomorrow's noon-time temperature. Would you use boosting, bagging or neither?

Intro Solution

Solution

Let $\hat{x}_i = (x_i, f_1(x_i), \ldots, f_{10}(x_i)) \in \mathbb{R}^{d+10}$. Then use any fitting method you like to produce an aggregate decision function $\underline{f} : \mathbb{R}^{d+10} \to \mathbb{R}$. This method is sometimes called <u>stacking</u>.

1. This isn't bagging - we didn't generate bootstrap samples and learn a decision function on each of them.
2. This isn't boosting - boosting learns decision functions on varying datasets to produce an aggregate classifier.

Additive Models

1. *Additive models* over a *base hypothesis space* $\mathcal{H}$ take the form

$$\mathcal{F} = \left\{ f(x) = \sum_{m=1}^{M} \nu_m \underline{h_m(x)} \mid h_m \in \mathcal{H}, \nu_m \in \mathbb{R} \right\}.$$

2. Since we are taking linear combinations, we assume the h_m functions take values in $\mathbb{R}$ or some other vector space.
3. Empirical risk minimization over $\mathcal{F}$ tries to find

$$\underset{f \in \mathcal{F}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(x_i)).$$

4. This in general is a difficult task, as the number of base hypotheses M is unknown, and each base hypothesis h_m ranges over all of $\mathcal{H}$.

Forward Stagewise Additive Modeling (FSAM)

The FSAM method fits additive models using the following (greedy) algorithmic structure:

1. Initialize $\underline{f_0} \equiv 0$.
2. For stage $m = 1, \ldots, M$:
    1. Choose $h_m \in \mathcal{H}$ and $\nu_m \in \mathbb{R}$ so that
    $$f_m = f_{m-1} + \nu_m h_m$$
    has the minimum empirical risk.
    2. The function f_m has the form
    $$f_m = \nu_1 h_1 + \cdots + \nu_m h_m.$$

Forward Stagewise Additive Modeling (FSAM)

The FSAM method fits additive models using the following (greedy) algorithmic structure:

1. Initialize $f_0 \equiv 0$.
2. For stage $m = 1, \ldots, M$:
    1. Choose $h_m \in \mathcal{H}$ and $\nu_m \in \mathbb{R}$ so that
    $$f_m = f_{m-1} + \nu_m h_m$$
    has the minimum empirical risk.
    2. The function f_m has the form
    $$f_m = \nu_1 h_1 + \cdots + \nu_m h_m.$$

- When choosing h_m, ν_m during stage m, we must solve the minimization

$$(\nu_m, h_m) = \underset{\nu \in \mathbb{R}, h \in \mathcal{H}}{\arg\min} \sum_{i=1}^{n} \ell(y_i, f_{m-1}(x_i) + \nu h(x_i)).$$

Gradient Boosting

1. Can we simplify the following minimization problem:

$$(\nu_m, h_m) = \underset{\nu\in\mathbb{R}, h\in\mathcal{H}}{\arg\min} \sum_{i=1}^{n} \ell(y_i, f_{m-1}(x_i) + \nu h(x_i)).$$

2. What if we linearize the problem and take a step along the steepest descent direction?
3. Good idea, but how do we handle the constraint that h is a function that lies in $\mathcal{H}$, the base hypothesis space?

Gradient Boosting

1. Can we simplify the following minimization problem:

$$(\nu_m, h_m) = \underset{\nu \in \mathbb{R}, h \in \mathcal{H}}{\arg\min} \sum_{i=1}^{n} \ell(y_i, f_{m-1}(x_i) + \nu h(x_i)).$$

2. What if we linearize the problem and take a step along the steepest descent direction?
3. Good idea, but how do we handle the constraint that h is a function that lies in $\mathcal{H}$, the base hypothesis space?
4. First idea: since we are doing empirical risk minimization, we only care about the values h takes on the training set. Thus we can think of h as a vector $(h(x_1), \ldots, h(x_n))$.
5. Second idea: first compute unconstrained steepest descent direction, and then constrain (project) onto possible choices from $\mathcal{H}$.

Gradient Boosting Machine

1. Initialize $f_0 \equiv 0$.
2. For stage $m = 1, \ldots, M$:
    1. Compute the steepest descent direction (also called *pseudoresiduals*):
    $$r_m = -\left(\frac{\partial}{\partial f_{m-1}(x_1)}\ell(y_1, f_{m-1}(x_1)), \ldots, \frac{\partial}{\partial f_{m-1}(x_n)}\ell(y_n, f_{m-1}(x_n))\right).$$
    2. Find the closest base hypothesis (using Euclidean distance):
    $$h_m = \underset{h \in \mathcal{H}}{\arg\min} \sum_{i=1}^{n} ((r_m)_i - h(x_i))^2.$$
    3. Choose fixed step size $\nu_m \in (0, 1]$ or line search:
    $$\nu_m = \underset{\nu \geq 0}{\arg\min} \sum_{i=1}^{n} \ell(y_i, f_{m-1}(x_i) + \nu h_m(x_i)).$$
    4. Take the step:
    $$f_m(x) = f_{m-1}(x) + \nu_m h_m(x).$$

Gradient Boosting Machine

1. Each stage we need to solve the following step:

$$h_m = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{n} ((r_m)_i - h(x_i))^2.$$

How do we do this?

Gradient Boosting Machine

1. Each stage we need to solve the following step:

$$h_m = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{n} ((r_m)_i - h(x_i))^2.$$

   How do we do this?

2. This is a standard least squares regression task on the “mock” dataset

$$\mathcal{D}^{(m)} = \{(x_1, (r_m)_1), \ldots, (x_n, (r_m)_n)\}.$$

3. We assume that we have a learner that (approximately) solves least squares regression over $\mathcal{H}$.

Gradient Boosting Comments

1. The algorithm above is sometimes called AnyBoost or Functional Gradient Descent.
2. The most commonly used base hypothesis space is small regression trees (HTF recommends between 4 and 8 leaves).

Practice With Different Loss Functions

Question

Explain how to perform gradient boosting with the following loss functions:

1. Square loss: $\ell(y,a) = (y-a)^2/2$.
2. Absolute loss: $\ell(y,a) = |y-a|$.
3. Exponential margin loss: $\ell(y,a) = e^{-ya}$.

Solution: Square loss

Using $\ell(y,a) = (y-a)^2/2$
To compute an arbitrary pseudoresidual we first note that

$$\partial_a (y-a)^2/2 = -(y-a)$$

giving

$$-\partial_{\bigcirc{2}}\ell(y_i, f_{m-1}(x_i)) = (y_i - f_{m-1}(x_i)).$$

In words, for the square loss, the pseudoresiduals are simply the residuals from the previous stage's fit. Thus, in stage m our step direction h_m is given by solving

$$h_m := \arg\min_{h\in\mathcal{H}} \sum_{i=1}^{n} ((y_i - f_{m-1}(x_i)) - h(x_i))^2.$$

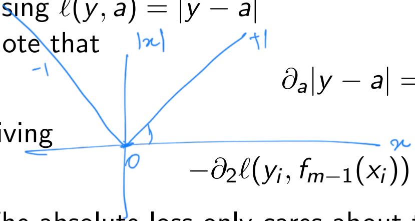
Solution: Absolute Loss

$$\frac{\partial |x|}{\partial x} = \frac{\partial \sqrt{x^2}}{\partial x} = \frac{1}{\cancel{2}} (x^2)^{-1/2} \cancel{2}x$$

$$= \frac{x}{\sqrt{x^2}} = \frac{x}{|x|} \begin{cases} +1 \\ -1 \end{cases}$$

Using $\ell(y, a) = |y - a|$

Note that



$$\partial_a |y - a| = -\underbrace{\boxed{\mathrm{sgn}(y - a)}}_{x}$$

giving

$$-\partial_2 \ell(y_i, f_{m-1}(x_i)) = \underline{\mathrm{sgn}(y_i - f_{m-1}(x_i))}.$$

The absolute loss only cares about the sign of the residual from the previous stage's fit. Thus, in stage m our step direction h_m is given by solving

$$h_m := \operatorname*{arg\,min}_{h \in \mathcal{H}} \sum_{i=1}^{n} \underline{(\mathrm{sgn}(y_i - f_{m-1}(x_i)) - h(x_i))^2}.$$

Solution: Exponential Loss

Using $\ell(y,a) = e^{-ya}$

Note that

$$\partial_a e^{-\underline{ya}} = \underline{-y\boxed{e^{-ya}}}$$

giving

$$(\gamma_m)_i \qquad -\partial_2 \ell(y_i, f_{m-1}(x_i)) = y_i e^{-y_i f_{m-1}(x_i)}.$$

Thus, in stage m our step direction h_m is given by solving

$$\underline{h_m} := \operatorname*{arg\,min}_{h \in \mathcal{H}} \sum_{i=1}^{n} \left(\underline{y_i e^{-y_i f_{m-1}(x_i)}} - h(x_i) \right)^2.$$

Exponential Loss and Adaboost

As an aside, we will now sketch an argument that shows that if we have learners in the sense of AdaBoost (i.e., they produce classification functions that minimize a weighted $0-1$ loss), we can use them with GBM and the exponential loss to recover the AdaBoost algorithm. Let

$$\vec{r} = \left(y_i e^{-y_i f_{m-1}(x_i)}\right)_{i=1}^{n} \quad \text{and} \quad \vec{h} = (h(x_i))_{i=1}^{n}.$$

Then we have

$$\Rightarrow h_m = \arg\min_{h\in\mathcal{H}} \|\vec{r}-\vec{h}\|_2^2 = \|\vec{r}\|_2^2 + \|\vec{h}\|_2^2 - 2\langle \vec{r}, \vec{h}\rangle.$$

Note that $\vec{h} \in \{-1,1\}^n$ so $\|\vec{h}\|_2^2 = n$, i.e., a constant. Thus this minimization is equivalent to

$$\boxed{\arg\max_{h\in\mathcal{H}} \langle \vec{r}, \vec{h}\rangle.}$$

Exponential Loss and Adaboost continued

Plugging in, we have

$$\Rightarrow \quad h_m = \underset{h\in\mathcal{H}}{\arg\max} \sum_{i=1}^{n} \boxed{h(x_i)y_i}\, e^{-y_i f_{m-1}(x_i)}.$$

Note that

$$h(x_i)y_i = \underbrace{1 - 2\cdot 1(h(x_i)\neq y_i)}_{=-1}$$

so

$$\begin{aligned} h_m &= \boxed{\underset{h\in\mathcal{H}}{\arg\max} \sum_{i=1}^{n} e^{-y_i f_{m-1}(x_i)}} - 2\sum_{i=1}^{n} e^{-y_i f_{m-1}(x_i)}\, 1(h(x_i)\neq y_i) \\ &= \underset{h\in\mathcal{H}}{\arg\min} \sum_{i=1}^{n} \underbrace{e^{-y_i f_{m-1}(x_i)}}_{w}\, 1(h(x_i)\neq y_i). \end{aligned}$$

Thus we see that h_m minimizes a weighted $0-1$ loss. The weights are

$$e^{-y_i f_{m-1}(x_i)} = e^{-y_i\left(\sum_{i=1}^{m-1} \nu_i h_i(x_i)\right)} = \prod_{i=1}^{m-1} e^{-y_i \nu_i h_i(x_i)} = \prod_{i=1}^{m-1} e^{-\nu_i \boxed{(1-2\,1(h_i(x_i)\neq y_i))}}.$$

Coding Part

- Next we apply GBM to square loss and absolute loss on a simple 1-d data set.
- We use decision stumps as our base hypothesis space.
- Run gbm.py to see the output.

Conditional Probability Models: Poisson Distribution [1]

Suppose we are trying to predict a distribution of counts from some input covariates. The simplest distribution in this situation is the Poisson distribution

$$p(k; \lambda) = \frac{e^{-\lambda}\lambda^k}{k!}$$

on $k = 0, 1, 2, 3, \cdots$ and $\lambda \in (0, \infty)$. Per usual, the set up for this problem involves:

- Input: $x \in \mathbb{R}^d$
- Output: $y \in \{0, 1, 2, \cdots\}$
- Data: $\mathcal{D} = ((x_1, y_1), \cdots, (x_n, y_2)) \in \left(\mathbb{R}^d \times \{0, 1, 2, \cdots\}\right)^n$, assumed to be sampled i.i.d. from some distribution $P_{\mathcal{X}\times\mathcal{Y}}$.

[1] From David's note

Setup Continued

Recall our setup for learning conditional probability distributions involved two functions:

$$x \mapsto \underbrace{s = f(x)}_{\text{score } s \in \mathbb{R}} \mapsto \underbrace{\psi(x)}_{\psi(x) \in \text{ Action space}}$$

where we called $\psi(s)$ our "transfer function".

1. Give the action space for this problem.
2. Give a transfer function for the Poisson distribution. We would like it to be differentiable.

Setup Continued Solutions

1. $\lambda \in (0, \infty)$, where λ is the parameter of a Poisson distribution.
2. $\psi(s) = \exp(s)$

Linear Conditional Probability Model

1. Give the form of $f(x)$ for a linear conditional probability model – call the weight vector w (as we will use this convention for the rest of the problem)
2. Give the likelihood $p(y = y_i | x_i; w)$ for a particular example (x_i, y_i)
3. Give the log likelihood for a particular example.
4. Give the full data log likelihood.
5. To fit this model, we need to find w^* by maximizing the log-likelihood, or equivalently minimizing the negative log-likelihood. Show how you could solve this optimization problem using a gradient based method by:
   - Stating your objective function $J(w)$.
   - Finding the gradient $\nabla_w J$
   - Giving the update rule $w_{t+1} \leftarrow w_t$, including specifying a reasonable approach to step size selection.

Solution: LCPM

1. $f(x) = w^T x$
2. $$\begin{aligned} p(y = y_i | x_i; w) &= \frac{e^{-\psi(f(x_i))} \psi(f(x_i))^{y_i}}{y_i!} \\ &= \frac{e^{-\exp(w^T x_i)} \exp(w^T x_i)^{y_i}}{y_i!} \end{aligned}$$
3. $$\begin{aligned} \log(p(y = y_i | x_i; w)) &= \log\left(\frac{e^{-\exp(w^T x_i)} \exp(w^T x_i)^{y_i}}{y_i!}\right) \\ &= -\exp(w^T x_i) + y_i w^T x_i - \log(y_i!) \end{aligned}$$

Solution LCPM continued

4.

$$
\begin{aligned}
\log L_{\mathcal{D}}(w) &= \log p(\mathcal{D}; w) \\
&= log \prod_{i=1}^{n} p(y = y_i | x_i; w) \\
&= \sum_{i=1}^{n} \log(p(y = y_i | x_i; w)) \\
&= \sum_{i=1}^{n} \left[-\exp(w^T x_i) + y_i w^T x_i - \log(y_i!) \right]
\end{aligned}
$$

Solution LCPM continued

5. We'll use full batch gradient descent on the NLL, with a fixed step size η found via grid search. The objective after dropping the constant $\log(y_i!)$ is:

$$J(w) = -\log L_{\mathcal{D}}(w) = \sum_{i=1}^{n} \left[\exp(w^T x_i) - y_i w^T x_i\right]$$

The gradient is

$$\nabla_w J = \sum_{i=1}^{n} \left[x_i \exp(w^T x_i) - y_i x_i\right]$$

The update rule is

$$w_{t+1} = w_t - \eta \nabla_w J$$

Nonlinear Conditional Probability Model

In this section, we'll replace the linear score function $s = f_w(x)$ with a nonlinear function $s = f(x)$. We will use the same transfer function as in the linear case.

1. Rewrite the full data log likelihood $\log L_{\mathcal{D}}(f)$ using f instead of the linear score function used above.
2. We'll use gradient boosting to learn f. To use gradient boosting, we'll need the negative gradient $-\mathrm{g}$ of $\log L_{\mathcal{D}}(f)$ with respect to $(f(x_1), \cdots, f(x_n))$. Find this negative gradient
3. Now fix some base hypothesis space $\mathcal{H}$ of function $h : \mathbb{R}^d \to \mathbb{R}$. Give the objective we solve to find $h \in \mathcal{H}$ that best fits $-\mathrm{g}$
4. Now let's put it all together. Assume we are using M rounds of boosting. Let $f^{(t)}$ be he score function after t rounds of boosting, $h^{(t)}$ be the function from the base hypothesis space learned in the t'th round of boosting, and $\mathrm{g}^{(t)}$ be the functional gradient in the t'th round of boosting. Give psuedocode for gradient boosting given this setup. Be sure to address step size selection.

Solution: NCPM

1. $\log L_{\mathcal{D}}(f) = \sum_{i=1}^{n} [-\exp(f(x_i)) + y_i f(x_i) - \log(y_i!)]$
2. Note for all i

$$\begin{aligned}\frac{\partial}{\partial f(x_i)}[\log L_{\mathcal{D}}(f)] &= \frac{\partial}{\partial f(x_i)}[-\exp(f(x_i)) + y_i f(x_i) - \log(y_i!)] \\ &= -\exp(f(x_i)) + y_i\end{aligned}$$

   Thus the negative gradient
   $-\mathrm{g} = (-y_1 + \exp(f(x_1)), \cdots, -y_n + \exp(f(x_n)))$.

3. $$\operatorname*{arg\,min}_{h\in\mathcal{H}} \sum_{i=1}^{n} (-\mathrm{g}_i - h(x_i))^2 = \operatorname*{arg\,min}_{h\in\mathcal{H}} \sum_{i=1}^{n} ([-y_i + \exp(f(x_i))] - h(x_i))^2$$

Solution NCPM continued

4.

1. Initialize $f^{(0)} = 0$.
2. For t from 1 to M:
    1. Find $-\mathrm{g}^{(t)}$
    2. Find $h^{(t)}$ (finding an (approximate) solution to the objective given in the previous problem).
    3. Add $\eta h^{(t)}$ to $f^{(t-1)}$ for some (grid searched and probably small η), yielding $f^{(t)} = \eta h^{(t)} + f^{(t-1)}$

More Questions [2]

Just in case we have time left. :-)

1. Show the exponential margin loss is a convex upper bound for the $0-1$ loss.
2. Show how to perform gradient boosting with the hinge loss.

[2]From Concept Checks

Q1 Solution

Recall that the exponential margin loss is given by $\ell(y, a) = e^{-ya}$ where $y \in \{-1, 1\}$ and $a \in \mathbb{R}$, and the $0-1$ loss is $1(y \neq \operatorname{sgn}(a))$. If $\operatorname{sgn}(y) \neq a$ then $ya \leq 0$ and

$$e^{-ya} \geq 1 - ya \geq 1 = 1(y \neq \operatorname{sgn}(a)).$$

In general $e^{-ya} \geq 0$ so the we obtain the upper bound. To prove convexity, we compute the second derivative and note that it is positive:

$$\frac{\partial^2}{\partial a^2} e^{-ya} = y^2 e^{-ya} > 0.$$

Q2 Solution

Recall that the hinge loss is given by $\ell(y, a) = \max(0, 1 - ya)$. Define g by

$$g(y, a) = \begin{cases} -y & \text{if } 1 - ya > 0, \\ 0 & \text{else.} \end{cases}$$

Then $g(y, a)$ is a subgradient of $\ell(y, a)$ with respect to a. At stage m of gradient boosting, we alredy have formed

$$f_{m-1} = \sum_{i=1}^{m-1} \nu_i h_i.$$

We then compute the pseudoresiduals r_m given by

$$r_m = -\left(g(y_1, f_{m-1}(x_1)), \ldots, g(y_n, f_{m-1}(x_n))\right).$$

After building the mock dataset $D^m = \{(x_1, (r_m)_1), \ldots, (x_n, (r_m)_n)\}$ we perform a least squares fit to obtain $h_m \in \mathcal{H}$. Then we can determine ν_m (usually a small fixed value). Finally we let $f_m = f_{m-1} + \nu_m h_m$.