Recitation 11

Gradient Boosting

DS-GA 1003 Machine Learning

Spring 2023

April 12, 2023

Motivation

- We started with linear models (regression, classification)
- We introduced feature transformation to incorporate non-linearity
- We covered decision trees: a non-linear, non-parametric model
- Another idea: Ensemble (Combining small models to tackle complex problems)

Additive Models

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

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

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 $f_0 \equiv 0$.
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. How about just take a step along the steepest descent direction?
3. Issue 1: h is a function instead of a vector
4. Solution 1: Treat h as a vector of the size of the training set $(h(x_1), \dots, h(x_n))$ rather than a function.
5. Issue 2: h must lies in $\mathcal{H}$, the base hypothesis space,
6. Solution 2: Compute unconstrained steepest descent direction, and then find the closest choices in $\mathcal{H}$.

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?

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 (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

$$\frac{\partial \ell}{\partial a} = -(y - a)$$

giving

$$-\frac{\partial \ell}{\partial f_{m-1}(x_i)} = (y_i - f_{m-1}(x_i)).$$

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

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

$$\frac{\partial \ell}{\partial a} = -\operatorname{sgn}(y - a)$$

giving

$$-\frac{\partial \ell}{\partial f_{m-1}(x_i)} = \operatorname{sgn}(y_i - f_{m-1}(x_i)).$$

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

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

Solution: Exponential Loss

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

$$\frac{\partial \ell}{\partial a} = -ye^{-ya}$$

giving

$$-\frac{\partial \ell}{\partial f_{m-1}(x_i)} = y_i e^{-y_i f_{m-1}(x_i)}.$$

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

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

Exponential Loss and Adaboost

If we have learners which 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

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

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

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

Exponential Loss and Adaboost continued

Note that

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

so

$$\begin{aligned} h_m &= \arg\max_{h \in \mathcal{H}} \sum_{i=1}^{n} e^{-y_i f_{m-1}(x_i)} - 2 \sum_{i=1}^{n} e^{-y_i f_{m-1}(x_i)} \mathbf{1}(h(x_i) \neq y_i) \\ &= \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{n} e^{-y_i f_{m-1}(x_i)} \mathbf{1}(h(x_i) \neq y_i). \end{aligned}$$

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

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

Python Demo

- 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.

Review

- Boosting is a sequential ensemble method (combine weak learners to produce a strong learner).
- Boosting greedily fits a (simple) additive model.
- Intuitively, we can think of gradient boosting as "gradient descent in the function space".