Lecture 10 boosting

wbg231

December 2022

1 motivation

- recall in ada boosting we learn weak learners $g_m(x)$ and weights α_m and use those to get our final predictor $G(x) = sing(\sum_{i=1}^{m} \alpha_m G_m(x))$
- why not just learn $G(x)$ directly?
- well it is a linear sum of weak learners that are not nesscarrily linear so we learn it sequentially in ada boost

nonlinear regression

- we can find wide types of data by fitting a linear combination of transformations to the input
$$f(x) = \sum_{m=1}^{m} v_m h_m(x)$$
where h_m is called a basis function such that
$$h_1 \cdots h_m : X \to \mathbb{R}$$
- example polynomial regression $h_i(x) \in \{x^i : i \in \mathbb{Z}\}$
- we can fit this with standard linear models if our basis functions are fixed ahead of time all we are learning are weighting factors

adaptive bayes function model

- what if we want to learn the basis functions
- we define a base hypotheses space $\mathcal{H} : X \to \mathbb{R}$ so all scalar values function

- an adaptive basis function expansion over $\mathcal{H}$ is an ensample model

$$f(x) = \sum_{m}^{M} v_m h_m(x)$$

where $v_m \in \mathbb{R}, h_m \in \mathcal{H}$

- so then we can combine these to get a new hypotheses space

$$\mathcal{F}_M = \{\sum_m v_m h_m(x) | v_m \in \mathbb{R}, h_m \in \mathcal{H} \forall m \in [1...m]\}$$

- so we our objective is

$$j(v_1 \cdots v_m, h_1 \cdots h_m) = \frac{1}{n}\ell(y_i, f(x_i)) = \frac{1}{n}\ell(y_i, \sum_{m=1}^{M} v_m h_m(x))$$

- if we want to optimize this sometimes we can use gradient descent or find a closed form (but that is not always the case)
- in cases where we can not differentiate we can try using a greedy algorithm similar to ada boost

gradient boosting

- applies when ever our loss function is sub differentiable wrt our training predictions $f(x_i)$ we can do regressions with the hypothesis base space $\mathcal{H}$

forward Stagewise adaptive modeling

- to recap our goal is to find the model

$$f(x) = \sum_m v_m h_m(x)$$

that is a weighed sum of basis functions given some loss function

- we do this by greedily fitting one function at a time without adjusting previous functions "forward Stagewise"
- so after $m-1$ stages we will have

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

- and then at the mth round we are trying to find the basis function $h \in \mathcal{H}$ and $v_m > 0$ such that

$$f_m = f_{m-1} + v_m h_m = c + v_m h_m$$

improves our loss as much as possible

- so this is what our algorithm looks like

_et's plug in our objective function.

1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to M:
   1. Compute:
   $$(v_m, h_m) = \underset{v \in \mathrm{R}, h \in \mathcal{H}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y_i, f_{m-1}(x_i) \underbrace{+ v h(x_i)}_{\text{new piece}}\right).$$
   2. Set $f_m = f_{m-1} + v_m h_m$.
3. Return: f_M.

- we are going to set our loss function as exponential ie $\ell(y, f(x)) = e^{-yf(x)}$
- and assume our $\mathcal{H} = \{h : x \to \{-1, 1\}\}$ that is our base functions are binary classifiers
- $$\begin{aligned} J(v, h) &= \sum_{i=1}^{n} \exp\left[-y_i \left(f_{m-1}(x_i) + v h(x_i)\right)\right] \\ &= \sum_{i=1}^{n} w_i^m \exp\left[-y_i v h(x_i)\right] && w_i^m \overset{\text{def}}{=} \exp\left[-y_i f_{m-1}(x_i)\right] \\ &= \sum_{i=1}^{n} w_i^m \left[\mathbb{I}(y_i = h(x_i))\, e^{-v} + \mathbb{I}(y_i \neq h(x_i))\, e^{v}\right] && h(x_i) \in \{1, -1\} \\ &= \sum_{i=1}^{n} w_i^m \left[(e^{v} - e^{-v}) \mathbb{I}(y_i \neq h(x_i)) + e^{-v}\right] && \mathbb{I}(y_i = h(x_i)) = 1 - \mathbb{I}(y_i \neq h(x_i \end{aligned}$$
- so i mean the above is mainly algebra i am not sure if there is that much to it except for a helpful way to re-write the objective
- there is a lot of kind of messy algebra in this section that i am not sure is super useful
- the real take away is basically this a generalization of ada boost
- in practice this has a robustness issue since ada boost is not robust to outliers

review

- so far we have seen that using a basis function to obtain nonlinear model $f(x) = \sum_{m=1} v_m h_m(x)$ if you know the basis functions
- could use adaptive basis function models if you do not know the basis.
- and forward stage wise additive modeling greedily fits h_m to minimize average los
- but fsam only works for some loss functions
- we need a more general model

gradient boosting / any boost

FSAM with squared loss

- our objective function is

$$j(v, h) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (f_{m-1}(x_i) + vh(x_i)))^2$$

  if $\mathcal{H}$ is closed under scaling then we can just set $v = 1$ adn maximize and the model will adjust it's self

- doing v=1 yields

$$j(h) = \frac{1}{n}\sum_{i}([y_i - f_{m-1}(x_i) - h(x)])^2$$

  this si equivalent to fitting function residuals with least squares regression

- so we can think of this as sequentially building models such that each one minimizes least squared residuals with the ones before set

interpret the residual

- so our objective $J(f) = \frac{1}{n}\sum_{i=1}(y_i - f(x_i))^2 = \frac{1}{n}\sum_{i=1}(y_i - f_{m-1}(x) - f_m(x_i))^2$
- so we can see that $\frac{\partial J(j)}{\partial f(x_i)} = -2(y_i - f(x_i))$ this gradient with respect to f is saying how should we try to change the output of f to minimize square loss
- so in other words our residual is the gradient
- so what we are doing at each step is learning $h \in \mathcal{H}$ to fit the residual

$$f \leftarrow f + vh$$

functional gradient descent

- we want to minimize our objective

$$J(f) = \sum_{i=1}^{n} \ell(y_i, f(x_i))$$

- not that $j(f)$ only depends at f evaluated at n training points that is $f = (f(x_1) \cdots f(x_n))^t$ so treating these as parameters we can write

$$j(f) = \sum \ell(y_i, f_i)$$

- the negative gradient $-g$ is the vector of partial darivtives of the y with respect to $f_i = f(x_i)$
- with gradient descent the final predictor will be

$$f_0 + \sum v_t(-g_t)$$

  that is at every step we are updating our parameters in the direction of the gradient
- the unconstrained step direction $-g$ is called the "pseudo residual"
- so we only have h points which to use to estimate $h \in \mathcal{H}$ so we do projected least squares regression where

$$min_{h\in\mathcal{H}} \sum (-g_i - h(x_i))^2$$

recap

- so we have the following objects
  1. our objective function

  $$j(f) = \sum \ell(y_i, f(x_i))$$

  2. unconstrained gradient $g \in \mathbb{R}^n$ wrt $f = (f(x_1) \cdots f(x_n))^t$ (so this is a true gradient of the loss function with respect to what we have learned so far)
  3. then we have the projected negative gradient$h \in \mathcal{H}$ so that is

  $$h = argmin_{h\in\mathcal{H}} \sum_{i=1}^{n} (-g_i - h(x_i))^2$$

  4. and we update our function at each step acording to this projected gradient $f \leftarrow f + vh$

- Gradient boosting algorithm

  1. Initialize f to a constant: $f_0(x) = \arg\min_\gamma \sum_{i=1}^n \ell(y_i, \gamma)$.
  2. For m from 1 to M:
     1. Compute the pseudo-residuals (negative gradient):
     $$r_{im} = -\left[\frac{\partial}{\partial f(x_i)} \ell(y_i, f(x_i))\right]_{f(x_i) = f_{m-1}(x_i)} \tag{26}$$
     2. Fit a base learner h_m with squared loss using the dataset $\{(x_i, r_{im})\}_{i=1}^n$.
     3. [Optional] Find the best step size $v_m = \arg\min_v \sum_{i=1}^n \ell(yi, f_{m-1}(x_i) + vh_m(x_i))$.
     4. Update $f_m = f_{m-1} + \lambda v_m h_m$
  3. Return $f_M(x)$.

binomial boost with logistic loss

- recall that logistic loss with $y \in [-1, 1]$

$$\ell(y, f(x)) = log(1 + e^{-yf(x)})$$