Gradient Boosting

David Rosenberg

New York University

April 12, 2016

Review: AdaBoost and FSAM

Adaptive Basis Function Model

- AdaBoost produces a classification score function of the form

$$\sum_{m=1}^{M} \alpha_m G_m(x)$$

  - each G_m is a **weak classifier**
- The G_m's are like basis functions, but they are learned from the data.
- Let's move beyond classification models...

Adaptive Basis Function Model

- **Base hypothesis space $\mathcal{F}$**
  - the "weak classifiers" in boosting context
- An **adaptive basis function expansion** over $\mathcal{F}$ is

$$f(x)=\sum_{m=1}^{M}\nu_m h_m(x),$$

  - $h_m \in \mathcal{F}$ chosen in a learning process ("adaptive")
  - $\nu_m \in \mathbf{R}$ are **expansion coefficients.**
- **Note:** We are taking linear combination of outputs of $h_m(x)$.
  - Functions in $h_m \in \mathcal{F}$ must produce values in $\mathbf{R}$ (or a vector space)

How to fit an adaptive basis function model?

- **Loss function**: $\ell(y, \hat{y})$
- **Base hypothesis space**: $\mathcal{F}$ of **real-valued** functions
- Want to find

$$f(x) = \sum_{m=1}^{M} \nu_m h_m(x)$$

that **minimizes empirical risk**

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

- We'll proceed in stages, adding a new h_m in every stage.

Forward Stagewise Additive Modeling (FSAM)

- Start with $f_0 \equiv 0$.
- After $m-1$ stages, we have

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

where $h_1, \ldots, h_{m-1} \in \mathcal{F}$.

- Want to find
  - **step direction** $h_m \in \mathcal{F}$ and
  - **step size** $\nu_m > 0$
- So that

$$f_m = f_{m-1} + \nu_m h_m$$

minimizes empirical risk.

Forward Stagewise Additive Modeling

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

Example 1: Exponential Loss & Classifiers (AdaBoost)

- Loss function: $\ell(y, f(x)) = \exp(-yf(x))$.
- Base hypothesis space: $\mathcal{F} = \{h(x) : \mathcal{X} \to \{-1, 1\}\}$ (weak classifiers)
- Then Forward Stagewise Additive Modeling (FSAM) reduces to an instance of AdaBoost.
    - (See HTF Section 10.4 for proof.)

Example 2: Square Loss & Regression (L_2-Boosting)

- Loss function: $\ell(y, f(x)) = (y - f(x))^2$
- Base hypothesis space: $\mathcal{F} = \{h(x) : \mathcal{X} \to \mathbf{R}\}$ (real-valued functions)
- Key step is

$$\min_{\nu \in \mathbf{R}, h \in \mathcal{F}} \sum_{i=1}^{n} \left(y_i - \left[f_{m-1}(x_i) \underbrace{+\nu h(x_i)}_{\text{new piece}} \right] \right)^2$$

$$= \min_{\nu \in \mathbf{R}, h \in \mathcal{F}} \sum_{i=1}^{n} \left(\underbrace{y_i - f_{m-1}(x_i)}_{\text{residual}} - \nu h(x_i) \right)^2$$

Example 2: Square Loss & Regression (L_2-Boosting)

- Simplifying assumption: $\mathcal{F}$ is closed under scalar multiplication:
  - If $h \in \mathcal{F}$ then $ch \in \mathcal{F}$ for all $c \in \mathbf{R}$.
- Then step size is absorbed into $\mathcal{F}$ we can just compute

$$\min_{h \in \mathcal{F}} \sum_{i=1}^{n} \left(\underbrace{y_i - f_{m-1}(x_i)}_{\text{residual}} - h(x_i) \right)^2$$

- This is square-loss regression on $(x_1, r_1), \ldots, (x_n, r_n)$, where

$$r_i = y_i - f_{m-1}(x_i).$$

- [**Not linear regression** unless $\mathcal{F}$ comprises linear functions.]
- This is called L_2-**Boosting**.

FSAM: More Examples?

- The challenge with FSAM is solving

$$\min_{\nu \in \mathbf{R}, h \in \mathcal{F}} \sum_{i=1}^{n} \ell\left(y_i, f_{m-1}(x_i) \underbrace{+\nu h(x_i)}_{\text{new piece}}\right).$$

- Possibilities so far:
  - reduce it to weighted classification (e.g. AdaBoost)
  - reduce it to regression (e.g. L_2-Boosting).
- But finding minimizer is not always easy for arbitrary
  - loss function and
  - base hypothesis space

Coordinate Descent Method

Coordinate Descent Method

Goal: Minimize $L(w) = L(w_1, \ldots w_d)$ over $w = (w_1, \ldots, w_d) \in \mathbf{R}^d$.

- **Initialize** $w^{(0)} = 0$
- **while** not converged:
  - Choose a coordinate $j \in \{1, \ldots, d\}$
  - $w_j^{\text{new}} \leftarrow \arg\min_{w_j} L(w_1^{(t)}, \ldots, w_{j-1}^{(t)}, \mathbf{w_j}, w_{j+1}^{(t)}, \ldots, w_d^{(t)})$
  - $w^{(t+1)} \leftarrow w^{(t)}$
  - $w_j^{(t+1)} \leftarrow w_j^{\text{new}}$
  - $t \leftarrow t+1$

FSAM as Coordinate Descent

- Suppose $\mathcal{F} = \{\hbar_1, \ldots, \hbar_N\}$.
- Then $f_m = w_1 \hbar_1 + \cdots w_N \hbar_N$.
- Represent f_m by parameter vector $w^{(m)} = (w_1, \ldots, w_N)$.
- Start with $w^{(0)} = 0$.
- After $m-1$ stages, we have $w^{(m-1)} = (w_1, \ldots, w_N)$.
- Suppose mth step chooses
  - $h_m = \hbar_j \in \mathcal{F}$ and $v_m \in \mathbf{R}$.
- Then $w^{(m)} = (w_1, \ldots, w_{j-1}, w_j + \nu_m, w_{j+1}, \ldots, w_N)$

Gradient Boosting / "Anyboost"

FSAM Looks Like Iterative Optimization

- The FSAM step

$$(\nu_m, h_m) = \underset{\nu \in \mathbf{R}, h \in \mathcal{F}}{\arg\min} \sum_{i=1}^{n} \ell\left(y_i, f_{m-1}(x_i) \underbrace{+\nu h(x_i)}_{\text{new piece}}\right).$$

- Hard part: finding the **best step direction** h.
- What if we looked for the **locally best** step direction?
  - like in gradient descent
- Approach:
  - Choose h_m to be something like a gradient in function space.
  - Roughly speaking, it will be the functional gradient projected onto $\mathcal{F}$.

Functional Gradient Descent: Main Idea

- We want to minimize
$$\sum_{i=1}^{n} \ell\left(y_i, f\left(x_i\right)\right).$$
- Take functional gradient w.r.t. f.
- Find function $h \in \mathcal{F}$ closest to gradient.
- Take a step in this "projected gradient" direction h.

"Functional" Gradient Descent

- We want to minimize
$$\sum_{i=1}^{n} \ell\left(y_i, f(x_i)\right).$$
- Only depends on f at the n training points.
- Define
$$\mathbf{f} = \left(f(x_1), \ldots, f(x_n)\right)^T$$
and write the objective function as
$$J(\mathbf{f}) = \sum_{i=1}^{n} \ell\left(y_i, \mathbf{f}_i\right).$$

Functional Gradient Descent: Unconstrained Step Direction

- Consider gradient descent on

$$J(\mathbf{f}) = \sum_{i=1}^{n} \ell\left(y_i, \mathbf{f}_i\right).$$

- The **negative gradient step direction** at $\mathbf{f}$ is

$$-\mathbf{g} = -\nabla_{\boldsymbol{f}} J(\mathbf{f}),$$

which we can easily calculate.

Functional Gradient Descent: Projection Step

- Unconstrained step direction is

$$-\mathbf{g} = -\nabla_{\boldsymbol{f}} J(\mathbf{f}).$$

- Suppose $\mathcal{F}$ is our weak hypothesis space.
- Find $h \in \mathcal{F}$ that is closest to $-\mathbf{g}$ at the training points, in the ℓ^2 sense:

$$\min_{h \in \mathcal{F}} \sum_{i=1}^{n} \left(-\mathbf{g}_i - h(x_i)\right)^2.$$

- This is a least squares regression problem.
- $\mathcal{F}$ should have **real-valued** functions.
- So the h that best approximates $-\mathbf{g}$ is our step direction.

Functional Gradient Descent: Step Size

- Finally, we choose a stepsize.
- Option 1 (Line search):

$$\nu_m = \operatorname*{arg\,min}_{\nu > 0} \sum_{i=1}^{n} \ell\{y_i, f_{m-1}(x_i) + \nu h_m(x_i)\}.$$

- Option 2: (Shrinkage parameter)
  - We consider $\nu = 1$ to be the full gradient step.
  - Choose a fixed $\nu \in (0,1)$ – called a **shrinkage parameter.**
  - A value of $\nu = 0.1$ is typical – optimize as a hyperparameter .

The Gradient Boosting Machine

1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to M:
    1. Compute the **"pseudo-residuals"**:
$$\mathbf{g}_m = \left(\frac{\partial}{\partial f(x_i)} \left(\sum_{i=1}^{n} \ell\{y_i, f(x_i)\} \right) \Bigg|_{f(x_i)=f_{m-1}(x_i)} \right)_{i=1}^{n}$$
    2. Fit regression model to $-\mathbf{g}_m$:
$$h_m = \underset{h \in \mathcal{F}}{\arg\min} \sum_{i=1}^{n} \left((-\mathbf{g}_m)_i - h(x_i) \right)^2.$$
    3. Choose fixed step size $\nu_m = \nu \in (0, 1]$, or take
$$\nu_m = \underset{\nu > 0}{\arg\min} \sum_{i=1}^{n} \ell\{y_i, f_{m-1}(x_i) + \nu h_m(x_i)\}.$$
    4. Take the step:
$$f_m(x) = f_{m-1}(x) + \nu_m h_m(x)$$

The Gradient Boosting Machine: Recap

- Take any [sub]differentiable loss function.
- Choose a base hypothesis space for regression.
- Choose number of steps (or a stopping criterion).
- Choose step size methodology.
- Then you're good to go!