Gradient Boosting

David Rosenberg

New York University

November 1, 2015

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

- Hypothesis space $\mathcal{F}$
  - Can be classifiers or regression functions
  - These would be the "weak classifiers" or "base classifiers"
- An **adaptive basis function expansion** over $\mathcal{F}$ is

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

  - Each $h_m \in \mathcal{F}$ is chosen in a learning process, and
  - ν_m are **expansion coefficients.**
- For example, $\mathcal{F}$ could be all decision trees of depth at most 4.
- We now discuss one approach to fitting such a model.

Forward Stagewise Additive Modeling

1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to M:
    1. Compute:

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

    2. Set $f_m(x) = f_{m-1}(x) + \nu_m h(x)$.
3. Return: $f_M(x)$.

Exponential Loss and AdaBoost

- Take loss function to be

$$\ell(y, f(x)) = \exp\left(-yf(x)\right).$$

- Let $\mathcal{F} = \{h(x) : \mathcal{X} \to \{-1, 1\}\}$ be a hypothesis space of weak classifiers.
- Then Forward Stagewise Additive Modeling (FSAM) reduces to AdaBoost.
    - (See HTF Section 10.4 for proof.)

FSAM Looks Like Gradient Descent?

- Let's examine the key step of FSAM a bit more closely:

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

- This looks like one step of a numerical optimization method:
  - $h(x_i)$ is like a step direction
- This inspires a **new** approach to boosting.
  - We can choose h_m to be something like a gradient in function space.
  - Roughly speaking, it will be like the gradient projected onto $\mathcal{F}$.
  - Leads to a **functional gradient descent** method.
- Note: This will be a new method. AdaBoost will **not** be a special case.

Functional Gradient Descent: Main Idea

- We want to minimize
$$\sum_{i=1}^{n} \ell\{y_i, f(x_i)\}.$$
- 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: Unconstrained Objective

- Note that
$$\sum_{i=1}^{n}\ell\left(y_i, f\left(x_i\right)\right)$$
only depends on f at the training points.
- Define
$$\mathbf{f}=\left(f\left(x_1\right),\ldots,f\left(x_n\right)\right)^T$$
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:

$$\mathbf{g}_m = \left(\frac{\partial}{\partial f(x_i)} \left(\sum_{i=1}^{n} \ell\{y_i, f(x_i)\} \right) \Bigg|_{f(x_i)=f_{m-1}(x_i)} \right)_{i=1}^{n}$$

    2. Fit regression model to $-\mathbf{g}_m$:

$$h_m = \operatorname*{arg\,min}_{h \in \mathcal{F}} \sum_{i=1}^{n} \left((-\mathbf{g}_m)_i - h(x_i) \right)^2.$$

    3. Choose fixed step size $\nu_m = \nu \in (0,1]$, or take

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

    4. Take the step:

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

The Gradient Boosting Machine: Recap

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

Gradient Tree Boosting

- Common form of gradient boosting machine takes

$$\mathcal{F} = \{\text{regression trees of size } J\},$$

  where J is the number of terminal nodes.
- $J = 2$ gives decision stumps
- HTF recommends $4 \leqslant J \leqslant 8$.
- Software packages:
  - Gradient tree boosting is implemented by the **gbm package** for R
  - as `GradientBoostingClassifier` and `GradientBoostingRegressor` in **sklearn**
- For trees, there are other tweaks on the algorithm one can do
  - See HTF 10.9-10.12 and