

Forward Stagewise Additive Modeling

He He

CDS, NYU

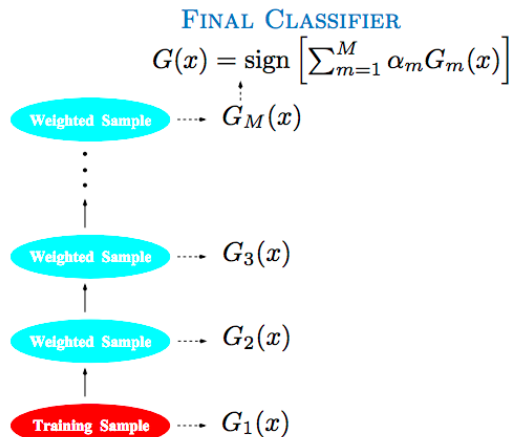
April 13, 2021

Today's lecture

- Another way to get non-linear models in a linear form—adaptive basis function models.
- A general algorithm for greedy function approximation—gradient boosting machine.

Motivation

Recap: Adaboost



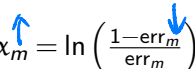
From ESL Figure 10.1

AdaBoost: Algorithm

Given training set $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$.

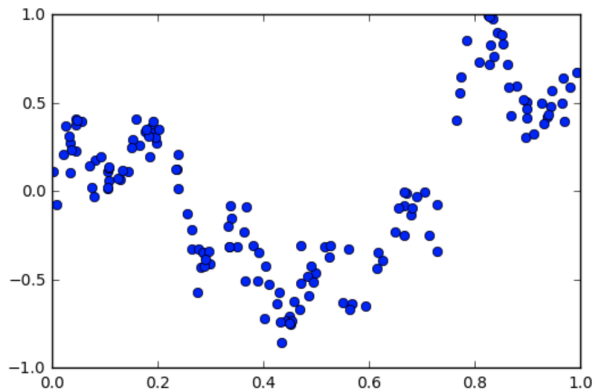
- ① Initialize observation weights $w_i = 1, i = 1, 2, \dots, n$.
- ② For $m = 1$ to M :
 - ① Base learner fits weighted training data and returns $G_m(x)$
 - ② Compute *weighted empirical 0-1 risk*:

$$\text{err}_m = \frac{1}{W} \sum_{i=1}^n w_i \mathbf{1}(y_i \neq G_m(x_i)) \quad \text{where } W = \sum_{i=1}^n w_i.$$

- ③ Compute *classifier weight*: $\alpha_m = \ln \left(\frac{1 - \text{err}_m}{\text{err}_m} \right)$.

 - ④ Update *example weight*: $w_i \leftarrow w_i \cdot \exp[\alpha_m \mathbf{1}(y_i \neq G_m(x_i))]$
- ③ Return *voted classifier*: $G(x) = \text{sign} \left[\sum_{m=1}^M \alpha_m G_m(x) \right]$. Why not learn $G(x)$ directly?

Nonlinear Regression

- How do we fit the following data?



Linear Model with Basis Functions

- Fit a linear combination of transformations of the input:

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

variable (pointing to v_m)

where h_m 's are called **basis functions** (or feature functions in ML):

$$h_1, \dots, h_M : \mathcal{X} \rightarrow \mathbb{R}$$

- Example: polynomial regression where $h_m(x) = x^m$.
- Can we use this model for classification?
- Can fit this using standard methods for linear models (e.g. least squares, lasso, ridge, etc.)
 - Note that h_m 's are fixed and known, i.e. chosen ahead of time.

Adaptive Basis Function Model

- What if we want to learn the basis functions? (hence *adaptive*)
- Base hypothesis space \mathcal{H} consisting of functions $h : \mathcal{X} \rightarrow \mathbb{R}$.
- An **adaptive basis function expansion** over \mathcal{H} is an ensemble model:

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

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

- Combined hypothesis space:

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

- What are the learnable?

Empirical Risk Minimization

- What's our learning objective?

$$\hat{f} = \arg \min_{f \in \mathcal{F}_M} \frac{1}{n} \sum_{i=1}^n \ell(y_i, f(x_i)), \quad \text{ERM}$$

for some loss function ℓ .

- Write ERM objective function as

$$J(v_1, \dots, v_M, h_1, \dots, h_M) = \frac{1}{n} \sum_{i=1}^n \ell \left(y_i, \underbrace{\sum_{m=1}^M v_m h_m(x)}_{f(x)} \right).$$

- How to optimize J ? i.e. how to learn?

Gradient-Based Methods

- Suppose our base hypothesis space is parameterized by $\Theta = \mathbb{R}^b$:

$$J(v_1, \dots, v_M, \theta_1, \dots, \theta_M) = \frac{1}{n} \sum_{i=1}^n \ell \left(y_i, \sum_{m=1}^M v_m h(x; \theta_m) \right).$$

- Can we optimize it with SGD?
 - Can we differentiate J w.r.t. v_m 's and θ_m 's?
- For some hypothesis spaces and typical loss functions, yes!
 - Neural networks fall into this category! (h_1, \dots, h_M are neurons of last hidden layer.)

What if Gradient Based Methods Don't Apply?

What if base hypothesis space \mathcal{H} consists of decision trees?

- Can we even parameterize trees with $\Theta = \mathbb{R}^b$?
- Even if we could, predictions would not change continuously w.r.t. $\theta \in \Theta$, so certainly not differentiable.

What about a greedy algorithm similar to Adaboost?

- Applies to non-parametric or non-differentiable basis functions.
- But is it optimizing our objective using some loss function?

Today we'll discuss **gradient boosting**.

- Gradient descent in the *function space*.
- It applies whenever
 - our loss function is [sub]differentiable w.r.t. training predictions $f(x_i)$, and

Kearns, Valiant (1989): Can weak learners (e.g., 51% accuracy) be transformed to strong learners (e.g., 99.9% accuracy)?

Schapire (1990) & Freund (1995): Yes, weak learners can be iteratively improved to a strong learner.

Freund, Schapire (1996): And here is a practical algorithm—Adaboost.

Breiman (1996 & 1998): Yes, it works! Boosting is the best off-the-shelf classifier in the world.

(Attempts to explain why Adaboost works and improvements)

Friedman, Hastie, Tibshirani (2000): Actually, boosting fits an additive model.

Friedman (2001): Furthermore, it can be considered as gradient descent in the function space.

Forward Stagewise Additive Modeling

Forward Stagewise Additive Modeling (FSAM)

Goal fit model $f(x) = \sum_{m=1}^M v_m h_m(x)$ given some loss function.

Approach Greedily fit one function at a time without adjusting previous functions, hence “forward stagewise”.

- After $m-1$ stages, we have

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

- In m 'th round, we want to find $h_m \in \mathcal{H}$ (i.e. a basis function) and $v_m > 0$ such that

$$f_m = \underbrace{f_{m-1}}_{\text{fixed}} + v_m h_m$$

improves objective function value by as much as possible.

Forward Stagewise Additive Modeling for ERM

Let's plug in our objective function.

❶ Initialize $f_0(x) = 0$.

❷ For $m = 1$ to M :

❶ Compute:

$$(v_m, h_m) = \arg \min_{v \in \mathbb{R}, h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \ell \left(y_i, \underbrace{f_{m-1}(x_i) + v h(x_i)}_{\text{new piece}} \right).$$

Handwritten blue arrow pointing to $f_{m-1}(x_i)$ with the word "fixed" above it.

❷ Set $f_m = f_{m-1} + v_m h_m$.

❸ Return: f_M .

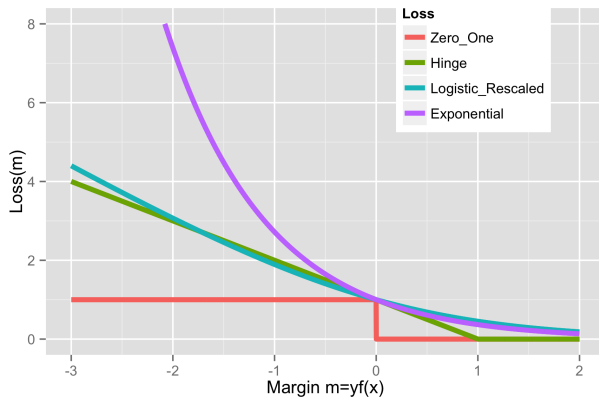
Recap: margin-based classifier

Binary classification

- Outcome space $\mathcal{Y} = \{-1, 1\}$
- Action space $\mathcal{A} = \mathbb{R}$ (model output)
- Score function $f : \mathcal{X} \rightarrow \mathcal{A}$.
- Margin for example (x, y) is $m = yf(x)$.
 - $m > 0 \iff$ classification correct
 - Larger m is better.
- **Concept check:** What are margin-based loss functions we've seen?

Exponential Loss

- Introduce the **exponential loss**: $\ell(y, f(x)) = \exp\left(-\underbrace{yf(x)}_{\text{margin}}\right)$.



Forward Stagewise Additive Modeling with exponential loss

Recall that we want to do FSAM with exponential loss.

❶ Initialize $f_0(x) = 0$.

❷ For $m = 1$ to M :

❶ Compute:

$$(v_m, h_m) = \arg \min_{v \in \mathbb{R}, h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \ell_{\text{exp}} \left(y_i, f_{m-1}(x_i) + \underbrace{vh(x_i)}_{\text{new piece}} \right).$$

❷ Set $f_m = f_{m-1} + v_m h_m$.

❸ Return: f_M .

FSAM with Exponential Loss: objective function

- Base hypothesis: $\mathcal{H} = \{h: \mathcal{X} \rightarrow \{-1, 1\}\}$.
- Objective function in the m 'th round:

$$J(v, h) = \sum_{i=1}^n \exp[-y_i (f_{m-1}(x_i) + v h(x_i))] \quad (2)$$

$$= \sum_{i=1}^n w_i^m \exp[-y_i v h(x_i)] \quad w_i^m \stackrel{\text{def}}{=} \exp[-y_i f_{m-1}(x_i)] \quad (3)$$

$$= \sum_{i=1}^n w_i^m [\mathbb{I}(y_i = h(x_i)) e^{-v} + \mathbb{I}(y_i \neq h(x_i)) e^v] \quad h(x_i) \in \{1, -1\} \quad (4)$$

$$= \sum_{i=1}^n w_i^m [(e^v - e^{-v}) \mathbb{I}(y_i \neq h(x_i)) + e^{-v}] \quad \mathbb{I}(y_i = h(x_i)) = 1 - \mathbb{I}(y_i \neq h(x_i)) \quad (5)$$

FSAM with Exponential Loss: basis function

- Objective function in the m 'th round:

$$J(v, h) = \sum_{i=1}^n w_i^m [(e^v - e^{-v}) \mathbb{I}(y_i \neq h(x_i)) + e^{-v}]. \quad (6)$$

- If $v > 0$, then

$$\arg \min_{h \in \mathcal{H}} J(v, h) = \arg \min_{h \in \mathcal{H}} \sum_{i=1}^n w_i^m \mathbb{I}(y_i \neq h(x_i)) \quad (7)$$

$$h_m = \arg \min_{h \in \mathcal{H}} \sum_{i=1}^n w_i^m \mathbb{I}(y_i \neq h(x_i)) \quad (8)$$

$$= \arg \min_{h \in \mathcal{H}} \frac{1}{\sum_{i=1}^n w_i^m} \sum_{i=1}^n w_i^m \mathbb{I}(y_i \neq h(x_i)) \quad \text{multiply by a positive constant} \quad (9)$$

i.e. h_m is the minimizer of the weighted zero-one loss.

FSAM with Exponential Loss: classifier weights

- Define the weighted zero-one error:

$$\text{err}_m = \frac{\sum_{i=1}^n w_i^m \mathbb{I}(y_i \neq h(x_i))}{\sum_{i=1}^n w_i^m}. \quad (10)$$

- Exercise:** show that the optimal v is:

$$v_m = \frac{1}{2} \log \frac{1 - \text{err}_m}{\text{err}_m} \quad (11)$$

- Same as the classifier weights in Adaboost (differ by a constant).
- If $\text{err}_m < 0.5$ (better than chance), then $v_m > 0$.

FSAM with Exponential Loss: example weights

- Weights in the next round:

$$w_i^{m+1} \stackrel{\text{def}}{=} \exp[-y_i f_m(x_i)] \quad (12)$$

$$= w_i^m \exp[-y_i v_m h_m(x_i)] \quad f_m(x_i) = f_{m-1}(x_i) + v_m h_m(x_i) \quad (13)$$

$$= w_i^m \exp[-v_m \mathbb{I}(y_i = h_m(x_i)) + v_m \mathbb{I}(y_i \neq h_m(x_i))] \quad (14)$$

$$= w_i^m \exp[2v_m \mathbb{I}(y_i \neq h_m(x_i))] \underbrace{\exp^{-v_m}}_{\text{scaler}} \quad (15)$$

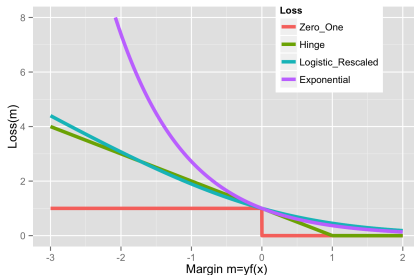
- The constant scaler will cancel out during normalization.
- $2v_m = \alpha_m$ in Adaboost.

Why Exponential Loss

- $\ell_{\text{exp}}(y, f(x)) = \exp(-yf(x))$.
- **Exercise:** show that the optimal estimate is

$$f^*(x) = \frac{1}{2} \log \frac{p(y = 1 | x)}{p(y = 0 | x)}. \quad (16)$$

- How is it different from other losses?



AdaBoost / Exponential Loss: Robustness Issues

- Exponential loss puts a high penalty on misclassified examples.
 - \implies not robust to outliers / noise.
- Empirically, AdaBoost has degraded performance in situations with
 - high Bayes error rate (intrinsic randomness in the label)
- Logistic/Log loss performs better in settings with high Bayes error.
- Exponential loss has some computational advantages over log loss though.

We've seen

- Use basis function to obtain *nonlinear* models: $f(x) = \sum_{i=1}^M v_m h_m(x)$ with known h_m 's.
- *Adaptive* basis function models: $f(x) = \sum_{i=1}^M v_m h_m(x)$ with unknown h_m 's.
- Forward stagewise additive modeling: greedily fit h_m 's to minimize the average loss.

But,

- We only know how to do FSAM for certain loss functions.
- Need to derive new algorithms for different loss functions.

Next, how to do FSAM in general.