Forward Stagewise Additive Modeling

He He
Slides based on Lecture 11c from David Rosenberg's course materials
(https://github.com/davidrosenberg/mlcourse)

CDS, NYU

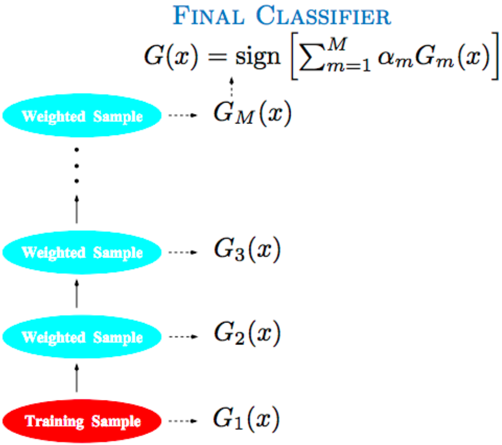
April 13, 2021

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 $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.

1. Initialize observation weights $w_i = 1,\ i = 1, 2, \ldots, n$.
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*:

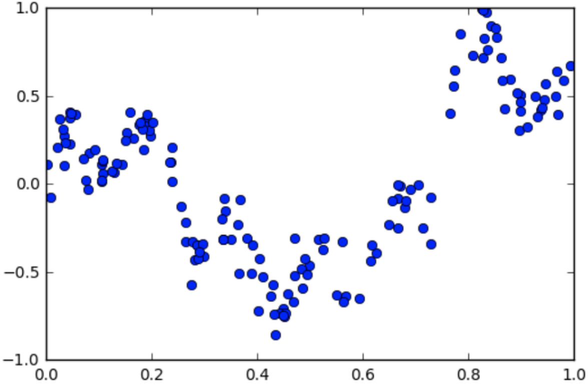
    $$\mathrm{err}_m = \frac{1}{W} \sum_{i=1}^{n} w_i 1(y_i \neq G_m(x_i)) \quad \text{where } W = \sum_{i=1}^{n} w_i.$$

    3. Compute *classifier weight*: $\alpha_m = \ln\left(\frac{1-\mathrm{err}_m}{\mathrm{err}_m}\right)$.
    4. Update *example weight*: $w_i \leftarrow w_i \cdot \exp\left[\alpha_m 1(y_i \neq G_m(x_i))\right]$
3. Return *voted classifier*: $G(x) = \operatorname{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

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:

variable

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

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

$$h_1, \ldots, h_M : \mathcal{X} \to \mathrm{R}$$

- 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 $h : \mathcal{X} \to \mathrm{R}$.
- An **adaptive basis function expansion** over $\mathcal{H}$ is an ensemble model:

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

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

- Combined hypothesis space:

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

- What are the learnable?

Empirical Risk Minimization

- What's our learning objective?

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

  for some loss function ℓ.

- Write ERM objective function as

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

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

Gradient-Based Methods

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

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

- 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! ($h_1, \ldots, h_M$ 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 $\Theta = \mathrm{R}^b$?
- 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

History

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.

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

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

   2. Set $f_m = f_{m-1} + v_m h_m$.
3. Return: f_M.

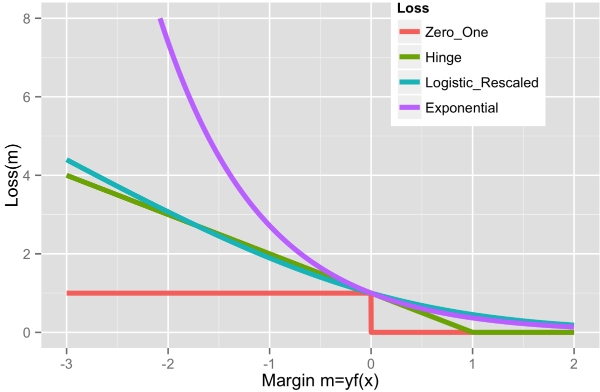
Recap: margin-based classifier

Binary classification

- Outcome space $\mathcal{Y} = \{-1, 1\}$
- Action space $\mathcal{A} = \mathrm{R}$ (model outoput)
- Score function $f : \mathcal{X} \to \mathcal{A}$.
- 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.

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

FSAM with Exponential Loss: objective function

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

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

$$= \sum_{i=1}^{n} w_i^m \exp\left[-y_i vh(x_i)\right] \qquad \boxed{w_i^m \stackrel{\text{def}}{=} \exp\left[-y_i f_{m-1}(x_i)\right]} \tag{3}$$

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

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

$$\tag{5}$$

FSAM with Exponential Loss: basis function

- Objective function in the m'th round:

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

- If $v > 0$, then

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

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

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

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:

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

- Exercise: show that the optimal v is:

$$v_m = \frac{1}{2}\log\frac{1-\text{err}_m}{\text{err}_m} \tag{11}$$

  - 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:

$$
\begin{aligned}
w_i^{m+1} &\stackrel{\text{def}}{=} \exp\left[-y_i f_m(x_i)\right] && (12)\\
&= w_i^m \exp\left[-y_i v_m h_m(x_i)\right] \qquad f_m(x_i) = f_{m-1}(x_i) + v_m h_m(x_i) && (13)\\
&= w_i^m \exp\left[-v_m \mathbb{I}\left(y_i = h_m(x_i)\right) + v_m \mathbb{I}\left(y_i \neq h_m(x_i)\right)\right] && (14)\\
&= w_i^m \exp\left[2 v_m \mathbb{I}\left(y_i \neq h_m(x_i)\right)\right] \underbrace{\exp^{-v_m}}_{\text{scaler}} && (15)
\end{aligned}
$$

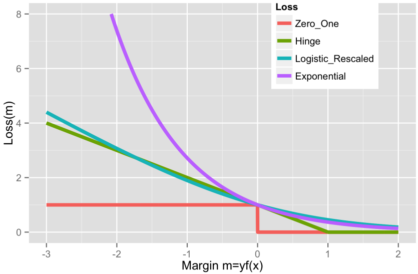
- The constant scaler will cancel out during normalization.
- $2v_m = \alpha_m$ in Adaboost.

Why Exponential Loss

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

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

- 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.

Review

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.