University of Amsterdam
Amsterdam School of Economics

Machine Learning for Econometrics

Lecture 4: Boosting

Yi He

November 22, 2022

Plan for Today

1. Introduction
2. Least Squares Boosting
3. AdaBoost

Introduction

Example: Predicting Stock Direction

Let the target $Y_t = \texttt{sign}(r_t) \in \{-1, 1\}$ be the sign of the stock return at time t, and the feature $X_t = r_{t-1}$ to be the return at time $t-1$.

- Momentum Rule

$$g(X_t) = \begin{cases} 1 & X_t > 0 \\ -1 & X_t < 0 \end{cases}$$

- Reversion Rule

$$-g(X_t) = \begin{cases} -1 & X_t > 0 \\ 1 & X_t < 0 \end{cases}$$

- Error rates for daily returns on S&P 500 index between 2012 and 2021 ($n = 2516$): Momentum $= 52\%$, Reversion $= 48\%$.

Decision Stumps

- Target $Y \in \{-1, 1\}$ and a univariate feature $X \in \mathbb{R}$.
- Consider a simple threshold rule for decision:

$$g(X) = \begin{cases} 1 & X > \theta \\ -1 & X < \theta \end{cases} \text{ or } -g(X) = \begin{cases} -1 & X > \theta \\ 1 & X < \theta \end{cases}$$

- Decision stumps:

$$\mathcal{G} = \{g(x) = \texttt{sign}(\theta - x) \cdot b : \ \theta \in \mathbb{R}, b = \pm 1\},$$

- For multivariate feature $X \in \mathbb{R}^d$:

$$\mathcal{G} = \left\{ g(x) = \texttt{sign}(\theta - x_j) \cdot b : \ \underbrace{j \in \{1, \ldots, d\}}_{\text{pick one feature}}, \ \theta \in \mathbb{R}, \ b = \pm 1 \right\}.$$

Decision Stumps As Weak Classifiers

- Take any decision stump $g \in \mathcal{G}$, then $-g \in \mathcal{G}$
- $g \in \mathcal{G}$ or $-g \in \mathcal{G}$ is no worse than a random guess but shows only weak predictive performance in general:

$$\min \{\mathbb{P}(g(X) \neq Y), \mathbb{P}(-g(X) \neq Y)\} \leq \frac{1}{2}$$

because

$$\begin{aligned} &\mathbb{P}(g(X) \neq Y) + \mathbb{P}(-g(X) \neq Y) \\ =&\mathbb{P}(g(X) \neq Y) + \mathbb{P}(g(X) = Y) = 1. \end{aligned}$$

- In our stock example: reversion rule $= 48\% < 50\%$.

Does Ensembling Work?

Can we combine the performance of many weak classifiers, say, $g_1(x), \ldots, g_M(x)$ to produce a powerful committee?

Consider $M = 25$ weak classifiers and fix some value x:

- Each classifier $g_m \in \{-1, 1\}$ has equal base error rate $\varepsilon = 0.35$
- Assume errors made by classifiers are independent (hardly true in practice, but let us assume it here for simplicity)
- Ensemble classifier

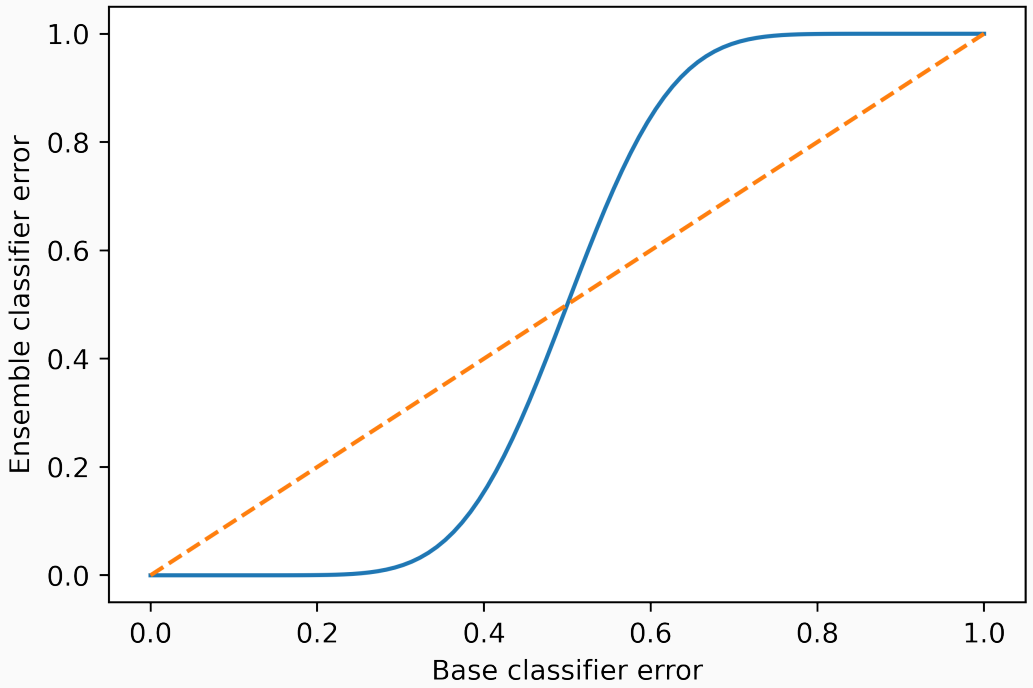
$$\texttt{sign}\left(g_1 + \ldots + g_{25}\right) = \begin{cases} 1 & g_1 + \ldots + g_{25} > 0 \\ -1 & g_1 + \ldots + g_{25} < 0, \end{cases}$$

  where `sign` denotes the sign function.

Probability that the committee makes a wrong prediction:

$$\mathbb{P}[\text{sgn}\,(g_1 + \ldots + g_{25}) \neq Y | X = x]$$

$$= \mathbb{P}\left(\sum_{m=1}^{25} \mathbb{1}[g_m \neq Y] \geq 13 \mid X = x\right) = \sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$



How can we use this idea to improve learning?

Least Squares Boosting

Additive Regression Models

Consider the candidate prediction rule in an additive form given by

$$f(x) = \sum_{m=1}^{M} f_m(x), \quad f_m \in \mathcal{G}, \tag{1}$$

where

- $\mathcal{G}$ is a set of base regression learners (stumps or T-terminal node trees) such that

$$g \in \mathcal{G} \;\Rightarrow w \cdot g \in \mathcal{G}, \; \forall w \in \mathbb{R}$$

- M is some positive integer (a hyperparameter)

Let span($\mathcal{G}$) denote the set of function generated by (1) over all possible choices of $M \in \{1, 2, 3, \ldots\}$.

Least-Squares Regression With Additive Models

Our goal is to estimate an ideal prediction rule $f^* \in \text{span}(\mathcal{G})$ that (nearly) minimizes the population risk

$$L_{\mathcal{D}}(f) = \mathbb{E}[\ell(f(X), Y)]$$

for the (half) squared loss function

$$\ell(f(X), Y) = \frac{1}{2}(f(X) - Y)^2$$

- If $\mu(x) = \mathbb{E}[Y|X = x] \in \text{span}(\mathcal{G})$, then it is the ideal rule.
- Unlike neural networks, there is no guarantee of universal approximation property with stumps. [For advanced reader, see Theorem 7 in Alon et al.(2022)]

Fitting Additive Regression Models: First Attempt

Let us take the bases as stumps such that

$$\begin{aligned} f_m(x) =& g(x; j_m, \theta_m, c_{m1}, c_{m2}) \\ =& c_{m1}\mathbf{1}[x_{j_m} < \theta_m] + c_{m2}\mathbf{1}[x_{j_m} \geq \theta_m]. \end{aligned}$$

Given a dataset $\{Y_i, X_i : i = 1, \ldots, n\}$, the empirical risk is

$$L_S(f) = \frac{1}{2n}\sum_{i=1}^{n}\left(\sum_{m=1}^{M} f_m(X_i) - Y_i\right)^2.$$

- Difficult to solve for large M: better optimization/statistical method?
- How to regularize the estimation?

Greedy Stagewise Approach: One At A Time

- The least-squares estimation is easy with $M = 1$, however, and denote the fitted function to be $\widehat{f}_1$.
- For $M = 2$, the standard algorithm does not store $\widehat{f}_1$, but optimize f_1 and f_2 simultaneously again. Note that $\widehat{f}_1$ may change.
- To save computational efforts, the stagewise method stores $\widehat{f}_1$ but only fit $\widehat{f}_2$ by minimizing

$$\frac{1}{2n}\sum_{i=1}^{n}\left(\widehat{f}_1(X_i) + f_2(X_i) - Y_i\right)^2 = \frac{1}{2n}\sum_{i=1}^{n}\left(f_2(X_i) - \widetilde{Y}_i^{(1)}\right)^2,$$

  where $Y_i^{(1)} = Y_i - \widehat{f}_1(X_i)$ are the first-step residuals.
- Repeat this procedure to fit new bases one-at-a-time sequentially: fit $\widehat{f}_m$ to the residuals from the previous step $Y_i^{(m-1)} = Y_i - \widehat{f}_1(X_i) - \ldots - \widehat{f}_{m-1}(X_i)$.

Sequential Optimization

- Initialize $F^{(0)}(x)$ with zero or a constant $F^{(0)}(x) = \bar{Y}$ the sample average of the target values.
- For $m = 1$ to M do:
  1. Calculate the residuals $\widetilde{Y}_i = Y_i - F^{(m-1)}(X_i)$
  2. Fit a base to the residuals:

$$f_m = \underset{f \in \mathcal{G}}{\operatorname{argmin}} \frac{1}{2n} \sum_{i=1}^{n} \left(f(X_i) - \widetilde{Y}_i^{(m-1)} \right)^2 .$$

  3. Accumulate the base learners:

$$F^{(m)} = F^{(m-1)} + f_m$$

- Output $\widehat{f}(x) = F^{(M)}(x)$.

Regularization: Early Stopping

- Boosting *forever* can overfit (although slowly)!

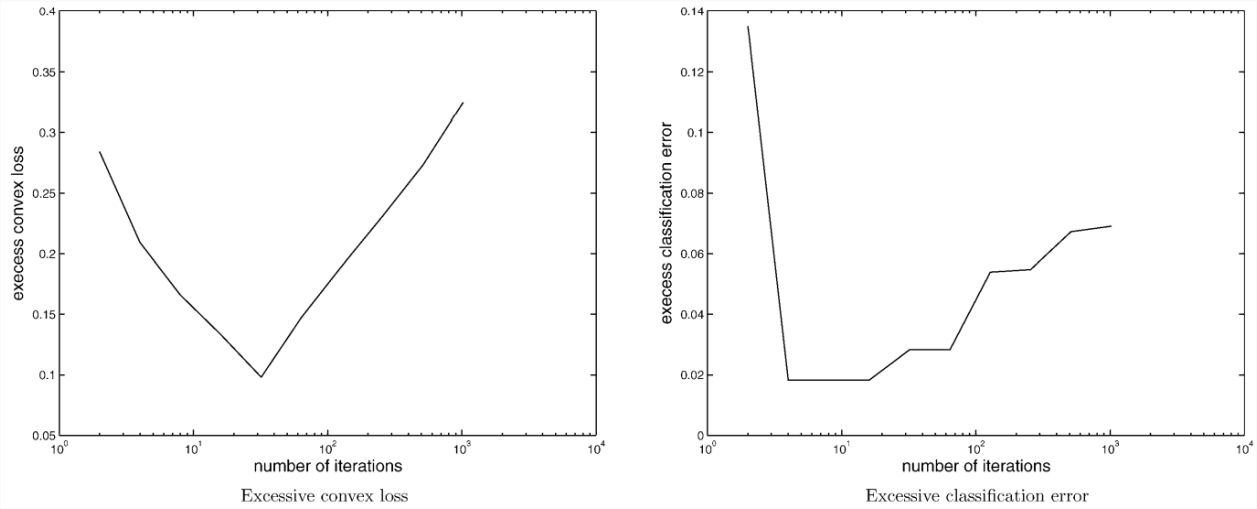


Figure 3 from Zhang and Yu (2005)

- A simple strategy is early-stopping: choose M based on a validation set

Regularization: Shrinkage

- Initialize $F^{(0)}(x)$ with zero or a constant $F^{(0)}(x) = \bar{Y}$ the sample average of the target values.
- For $m = 1$ to M do:
  1. Calculate the residuals $\widetilde{Y}_i = Y_i - F^{(m-1)}(X_i)$
  2. Fit a base to the residuals:

$$g_m = \underset{f \in \mathcal{G}}{\operatorname{argmin}} \frac{1}{2n} \sum_{i=1}^{n} \left(f(X_i) - \widetilde{Y}_i^{(m-1)} \right)^2 .$$

  3. Set $f_m = \eta \cdot g_m \in \mathcal{G}$. Accumulate the base learners:

$$F^{(m)} = F^{(m-1)} + f_m = F^{(m-1)} + \eta \cdot g_m$$

- Output $\widehat{f}(x) = F^{(M)}(x)$.

The hyperparameter $\eta \in (0, 1)$ is called 'learning rate' or 'shrinkage parameter'.

AdaBoost

Log Odds

- Binary target $Y \in \{-1, 1\}$ and features $X \in \mathbb{R}^d$
- Recall the softmax representation

$$\begin{bmatrix} \mathbb{P}(Y = 1|X = x) \\ \mathbb{P}(Y = -1|X = x) \end{bmatrix} = \begin{bmatrix} \frac{\exp(a_1(x))}{\exp(a_1(x))+\exp(a_2(x))} \\ \frac{\exp(a_2(x))}{\exp(a_1(x))+\exp(a_2(x))} \end{bmatrix} = \begin{bmatrix} \sigma(a(x)) \\ \sigma(-a(x)) \end{bmatrix}$$

  where

$$a(x) = a_1(x) - a_2(x) = \log \frac{\mathbb{P}(Y = 1|X = x)}{\mathbb{P}(Y = -1|X = x)}$$

  is called the log odds, and $\sigma(a) = \frac{1}{1+\exp(-a)}$ is the sigmoid function (Lecture 2).
- The Bayes classifier $C^{\text{Bayes}}(x) = \texttt{sign}(a(x))$.

Additive Logistic Model

Suppose we want to estimate $a(x)$ with some additive models

$$f(x) = \sum_{m=1}^{M} f_m(x), \quad f_m \in \mathcal{B}.$$

such that the resulted Bayes decision rule

$$h(x) = \texttt{sign}\left(\sum_{m=1}^{M} f_m(x)\right) = \texttt{sign}\left(\sum_{m=1}^{M} \alpha_m g_m(x)\right)$$

is a weighted majority vote. Each $g_m(x) \in \{-1, 1\}$ is a 'voter', and $\alpha_m > 0$ is its voting weight.

$\Rightarrow$ Take $\mathcal{B}$ to be stumps such that

$$\mathcal{B} = \{f(x) = \alpha g(x) : \alpha > 0, g \in \mathcal{G}\},$$

where $\mathcal{G}$ denotes the set of base classifiers so that $g(x) \in \{-1, 1\}$.

Exponential Loss

(Tute) The true log odds minimize the expected exponential loss:

$$a(x) = \operatorname*{argmin}_{f:\mathbb{R}^d \to \mathbb{R}} \mathbb{E}[\ell_{\exp}(f(X), Y) \mid X = x],$$

with

$$\ell_{\exp}(f(x), y) = \phi(f(x) \cdot y), \quad \phi(x) = \exp\left(-\frac{1}{2}x\right).$$

- The exponential loss is similar to hinge loss:

$$\ell_{\text{Hinge}}(f(x), y) = \phi(f(x) \cdot y), \quad \phi(x) = \max\{0, 1 - x\}.$$

- The exponential loss is similar to the zero-one loss

$$\ell_{0\text{-}1}(f(x), y) = \phi(f(x) \cdot y), \quad \phi(x) = \mathbb{1}[x < 0].$$

- However, the function $\phi(x) = \exp\left(-\frac{1}{2}x\right)$ is strictly convex and identifies $a(x)$.

Factorizing the Empirical Exponential Risk

Again, it is hard to optimize the empirical (exponential) risk over all base functions $f_1, \ldots, f_M$ simultaneously for large M.

AdaBoost fits f_m sequentially:

- The empirical exponential risk function at m-th step

$$\frac{1}{n}\sum_{i=1}^{n}\exp\left(-\frac{1}{2}\sum_{j=1}^{m-1}f_j(X_i)Y_i - \frac{1}{2}f_m(X_i)Y_i\right)$$

$$= \sum_{i=1}^{n}\frac{1}{n}\exp\left(-\frac{1}{2}\sum_{j=1}^{m-1}f_j(X_i)Y_i\right)\cdot\exp\left(-\frac{1}{2}f_m(X_i)Y_i\right)$$

$$= \sum_{i=1}^{n}\widetilde{w}_i^{(m)}\cdot\ell_{\exp}(f_m(X_i), Y_i)$$

- We are minimizing *weighted* exponential error at each step.
- The weights $\widetilde{w}_i^{(m)}$ are known from the previous steps.

Normalizing the Weights

- Normalizing the weights $\widetilde{w}_i^{(m)}$ does not change the solution:

$$w_i^{(m)} = \frac{\widetilde{w}_i^{(m)}}{\sum_{i=1}^n \widetilde{w}_i^{(m)}}$$

- Normalized weights are easier to interpret as probabilities.
- Henceforth we minimize the error function at m-th step:

$$f_m = \operatorname*{argmin}_{f \in \mathcal{B}} \sum_{i=1}^n w_i^{(m)} \cdot \ell_{\exp}(f(X_i), Y_i)$$

with the normalized weights such that $\sum_{i=1}^n w_i^{(m)} = 1$.

AdaBoost: Solving the Base Learner

Now consider any candidate base $f = \alpha g$, $g \in \mathcal{G}$ and $\alpha \geq 0$.

Because $g(X_i), Y_i \in \{-1, 1\}$, $g(X_i)Y_i = \begin{cases} -1 & g(X_i) \neq Y_i \\ 1 & g(X_i) = Y_i, \end{cases}$

$$\ell_{\text{exp}}(f(X_i), Y_i)$$

$$= \exp\left(-\frac{1}{2}\alpha g(X_i)Y_i\right) = \begin{cases} \exp\left(\frac{1}{2}\alpha\right) & g(X_i) \neq Y_i \\ \exp\left(-\frac{1}{2}\alpha\right) & g(X_i) = Y_i \end{cases}$$

$$= \exp\left(\frac{1}{2}\alpha\right) \mathbb{1}[g(X_i) \neq Y_i] + \exp\left(-\frac{1}{2}\alpha\right) \underbrace{\mathbb{1}[g(X_i) = Y_i]}_{1-\mathbb{1}[g(X_i)\neq Y_i]}.$$

$$= \left(\underbrace{\exp\left(\frac{1}{2}\alpha\right) - \exp\left(-\frac{1}{2}\alpha\right)}_{>0,\ \Leftrightarrow\ \alpha>0} \right) \mathbb{1}[g(X_i) \neq Y_i] + \exp\left(-\frac{1}{2}\alpha\right)$$

AdaBoost: Key Steps

Using the normalization condition $\sum_{i=1}^{n} w_i^{(m)} = 1$:

$$\sum_{i=1}^{n} w_i^{(m)} \ell_{\exp}(f(X_i), Y_i)$$

$$= \left(\underbrace{\exp\left(\frac{1}{2}\alpha\right) - \exp\left(-\frac{1}{2}\alpha\right)}_{>0 \;\Leftrightarrow\; \alpha>0} \right) \underbrace{\sum_{i=1}^{n} w_i^{(m)} \mathbb{1}[g(X_i) \neq Y_i]}_{J_m(g)} + \exp\left(-\frac{1}{2}\alpha\right)$$

- Fit a classifier $g_m \in \mathcal{G}$ minimizing the weighed misclassification error $J_m(g)$
- Then solve the first-order condition for α gives that

$$\alpha_m = \log \frac{1-\epsilon_m}{\epsilon_m}, \text{ with weighted error } \epsilon_m = J_m(g_m)$$

- $J_m(g_m) < J_m(-g_m)$ implies that $\epsilon_m < 1/2$ and $\alpha_m > 0$

To reduce computational costs, save the history of $w_i^{(m)}$ and update the data weighting coefficients along the path.

- Using the fact that $Y_i, g_m(X_i) \in \{1, -1\}$,

$$\begin{aligned}
\widetilde{w}_i^{(m+1)} &= \widetilde{w}_i^{(m)} \exp\left(-\frac{1}{2}\alpha_m g_m(X_i) Y_i\right) \\
&\propto w_i^{(m)} \exp\left(-\frac{1}{2}\alpha_m (1 - 2\mathbb{1}(g_m(X_i) \neq Y_i))\right) \\
&\propto w_i^{(m)} \exp(\alpha_m \mathbb{1}(g_m(X_i) \neq Y_i)) \cdot \cancel{\exp\left(-\frac{1}{2}\alpha_m\right)}
\end{aligned}$$

- Update and normalize:

$$w_i^{(m+1)} \leftarrow w_i^{(m)} \exp(\alpha_m \mathbb{1}(g_m(X_i) \neq Y_i)),$$

$$w_i^{(m+1)} \leftarrow \frac{w_i^{(m+1)}}{\sum_{i=1}^n w_i^{(m+1)}}$$

Points that are misclassified are given greater weight when used to train the next classifier in the sequence.

Remark: LogitBoost

- You may consider the logistic loss

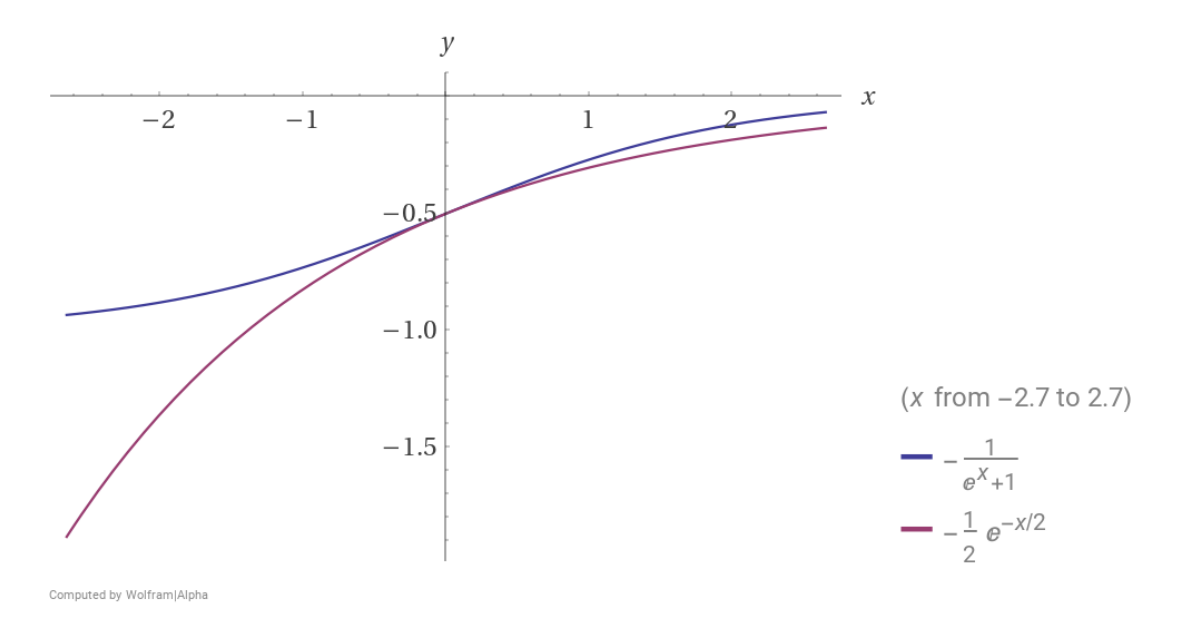
$$\begin{aligned}\ell_{\text{Logit}}(f(x), y) =& \ell_{\text{CE}}\left(\underbrace{\begin{pmatrix}\sigma(f(x)) \\ \sigma(-f(x))\end{pmatrix}}_{\text{Bivariate Regression Function}}, \begin{pmatrix}\mathbb{1}[y=1] \\ \mathbb{1}[y=-1]\end{pmatrix}\right) \\ =& \phi(f(x) \cdot y)\end{aligned}$$

  with

$$\phi(x) = \log(1 + \exp(-x)).$$

- ... which gives the 'LogitBoost' algorithm in Friedman et al. (2000).

Comparing the derivative $\phi'(x)$ for AdaBoost (purple) and LogitBoost (blue):



LogitBoost could be less sensitive to outliers

Friedman et al. (2000) conclude in their simulation studies:

> *It is likely that when the shrinkage parameter [η here] is carefully tuned ..., there would be little performance differential between them.*