Lecture 21

Gradient boosting

COURSE MATH 637: Mathematical Techniques in Data Science

WEBSITE https://jacobcalvert.com/teaching/math637fall2026/

INSTRUCTOR Jacob Calvert (calvert@udel.edu)

OFFICE HOURS MW 2:45pm–3:45pm, Ewing Hall 509

Plan for today

• Recap: bagging and random forests	2 min
• Boosting (regression)	12 min
• Gradient boosting	8 min
• Boosting (classification)	14 min
• Shrinkage, depth, and validation	8 min

Recall: Bagging and random forests

- deep trees have small bias and large variance
- bagging: grow trees on bootstrap samples, and average their estimates
- random forest: bagging, but each split chooses among m random features, to lower the correlation ρ between trees
- more trees do not overfit; the out-of-bag error estimates the error on new data

Recall: Averaging lowers variance, but not bias!

Proposition

Let $T_1, \ldots, T_B$ be identically distributed, with mean μ, variance σ^2, and correlation ρ between any two. Then

$$\mathbb{E}\left[\frac{1}{B}\sum_b T_b\right] = \mu, \qquad \mathrm{Var}\left(\frac{1}{B}\sum_b T_b\right) = \frac{1-\rho}{B}\sigma^2 + \rho\sigma^2.$$

Here, we're treating x as fixed:

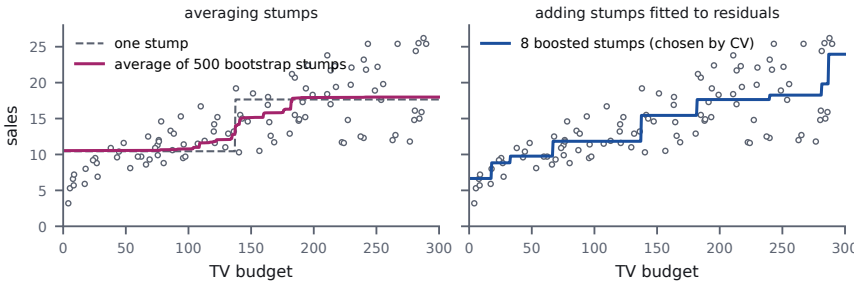
- $T_b = \hat{\eta}^{(b)}(x)$
- $\mathbb{E}[T_b] - \eta(x) = \mu - \eta(x)$ is the bias of tree T_b
- $\mathbb{E}[\frac{1}{B}\sum_b T_b] - \eta(x) = \mu - \eta(x)$ is also the bias of $\hat{\eta}_{\mathrm{bag}}(x) = \frac{1}{B}\sum_b T_b$!

Plan for today

• Recap: bagging and random forests	2 min
• Boosting (regression)	12 min
• Gradient boosting	8 min
• Boosting (classification)	14 min
• Shrinkage, depth, and validation	8 min

Example: Averaging can't reduce bias on sales data

Example: Regress sales against TV using 100 training markets.

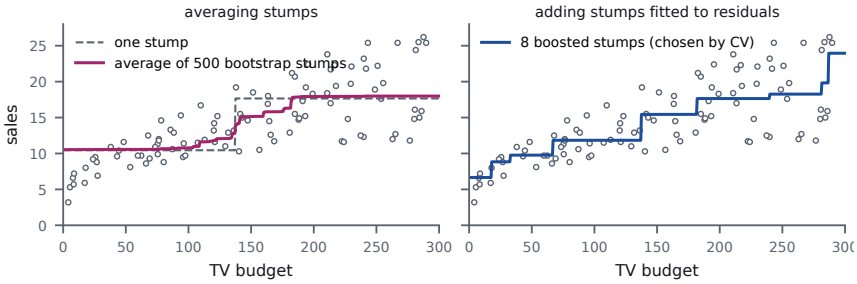


Stump $=$ a tree with one split.

Each stump has one step. Averaging blurs it, but the bias remains!

Example: Averaging can't reduce bias on sales data

Example: Regress sales against TV using 100 training markets.



Stump $=$ a tree with one split.

Each stump has one step. Averaging blurs it, but the bias remains!

> **Idea:** Grow trees sequentially, each fitted to the errors of earlier trees.

Definition: Boosting for squared loss

Definition (Boosting regression trees)

Start from the constant $F_0(x) = \bar{y}$. For $m = 1, \dots, M$:

1. compute the residuals $r_i = y_i - F_{m-1}(x_i)$
2. fit a tree h_m to the pairs (x_i, r_i) using RSS to split
3. update $F_m(x) = F_{m-1}(x) + \nu h_m(x)$

- $M =$ the number of trees
- $\nu \in (0, 1] =$ the learning rate
- h_m predicts the part of y that F_{m-1} still misses

Example: Boosting six points

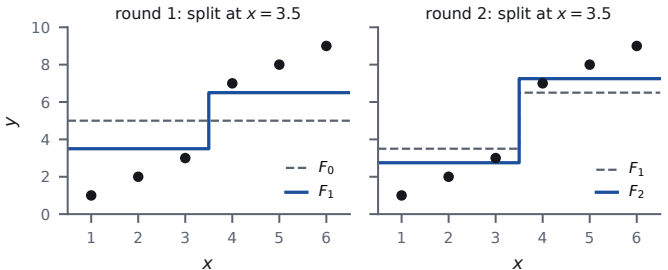
Problem

Six points: $x = 1, \dots, 6$ and $y = 1, 2, 3, 7, 8, 9$. Do boosting with stumps, learning rate $\nu = \frac{1}{2}$. Start with $F_0 = 5$. Find F_1, F_2 and F_3.

Example: Boosting six points

Problem

Six points: $x = 1, \ldots, 6$ and $y = 1, 2, 3, 7, 8, 9$. Do boosting with stumps, learning rate $\nu = \frac{1}{2}$. Start with $F_0 = 5$. Find F_1, F_2 and F_3.



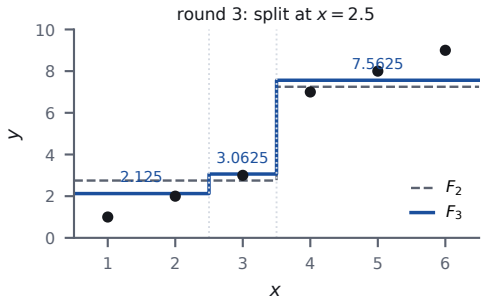
Rounds $1, 2$ both split at $x = 3.5$, as residuals jump between $x = 3$ and 4.

Round 3: the split changes

Residuals after round 2:

$$(-1.75,\ -0.75,\ 0.25,\ -0.25,\ 0.75,\ 1.75)$$

- no jump at 3.5 any more: what is left is the trend inside each group

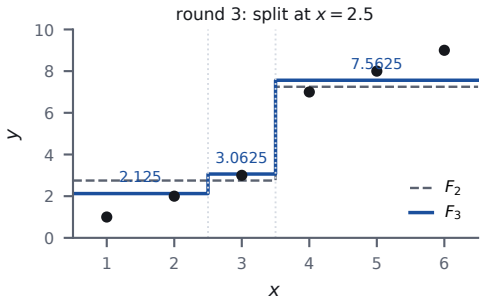


Round 3: the split changes

Residuals after round 2:

$$(-1.75,\ -0.75,\ 0.25,\ -0.25,\ 0.75,\ 1.75)$$

- no jump at 3.5 any more: what is left is the trend inside each group
- best stump now splits at $x = 2.5$ (tied with 4.5)

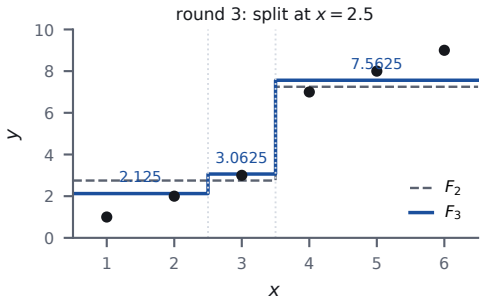


Round 3: the split changes

Residuals after round 2:

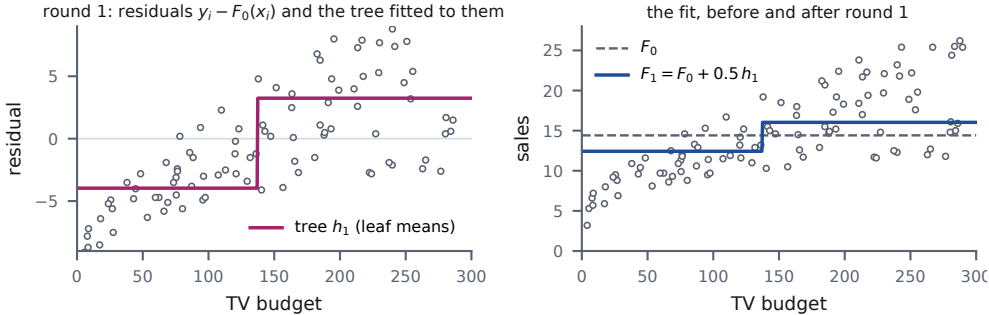
$$(-1.75,\ -0.75,\ 0.25,\ -0.25,\ 0.75,\ 1.75)$$

- no jump at 3.5 any more: what is left is the trend inside each group
- best stump now splits at $x = 2.5$ (tied with 4.5)
- F_3 steps where stumps split, giving three levels



Example: Boosting on advertising data

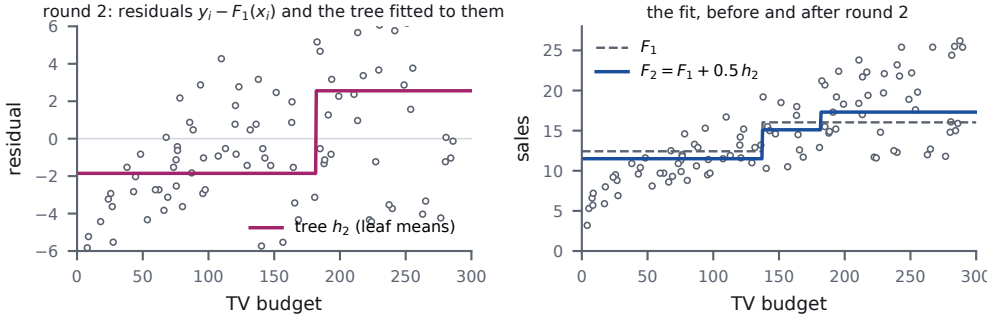
Example: Regress sales against TV using 100 training markets.



each round adds one step where the residuals are most out of balance

Example: Boosting on advertising data

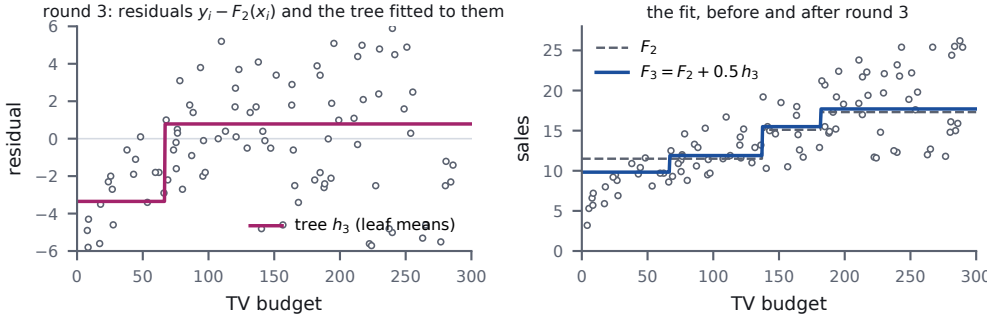
Example: Regress sales against TV using 100 training markets.



each round adds one step where the residuals are most out of balance

Example: Boosting on advertising data

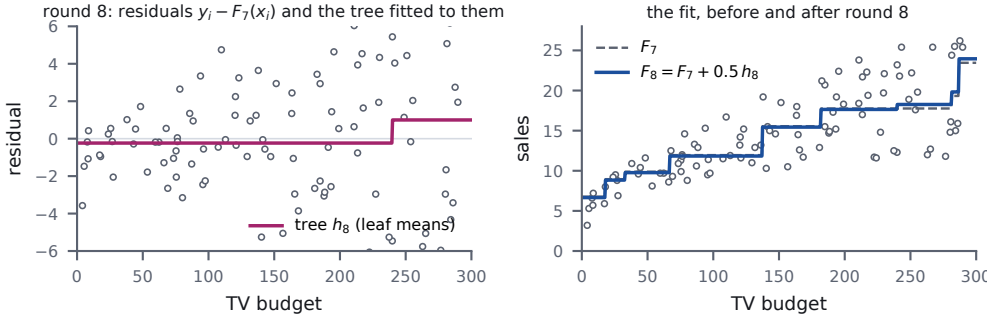
Example: Regress sales against TV using 100 training markets.



each round adds one step where the residuals are most out of balance

Example: Boosting on advertising data

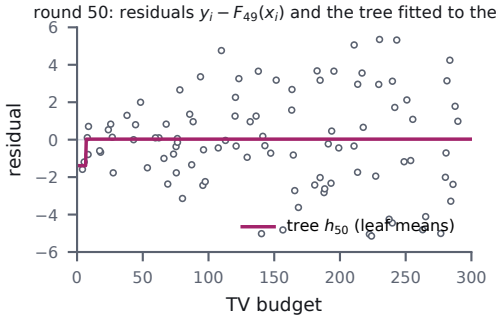
Example: Regress sales against TV using 100 training markets.

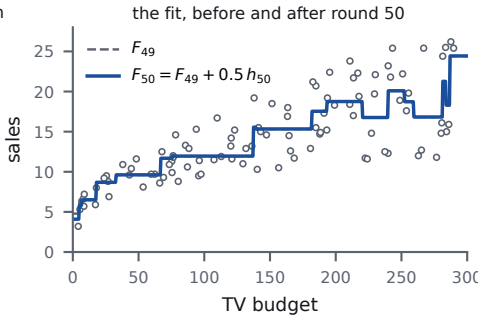


$M = 8$: the number of trees chosen by 5-fold cross-validation

Example: Boosting on advertising data

Example: Regress sales against TV using 100 training markets.





$M = 50$: the later stumps fit individual markets

Bagging averages, boosting adds

	bagging, random forests	boosting
each tree fits	resampled data	residuals (x_i, r_i)
trees grown	separately	sequentially
starts from	deep trees: large variance	small trees: large bias
combined by	averaging: $\frac{1}{B}\sum_b \hat{\eta}^{(b)}$	adding: $F_0 + \nu \sum_m h_m$
reduces	variance	bias
more trees	average converges: no overfitting	more flexible: can overfit
how many	as many as you can afford	choose by CV
error est. for tuning	out-of-bag error	CV error

Plan for today

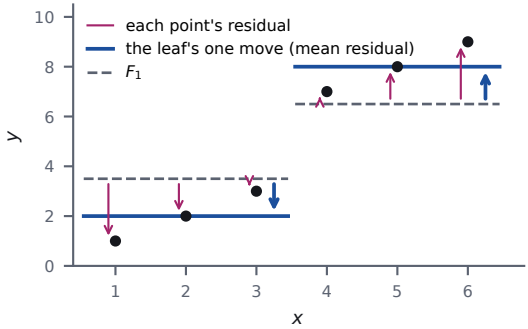
• Recap: bagging and random forests	2 min
• Boosting (regression)	12 min
• Gradient boosting	8 min
• Boosting (classification)	14 min
• Shrinkage, depth, and validation	8 min

Boosting is a kind of gradient descent

Key idea: With fitted values $F(x_i)$ and loss $L(y, F) = \frac{1}{2}(y - F)^2$, the residuals are *negative gradients*:

$$-\frac{\partial L}{\partial F}(y_i, F(x_i)) = y_i - F(x_i).$$

- gradient descent would move each $F(x_i)$ by its own residual (arrows)

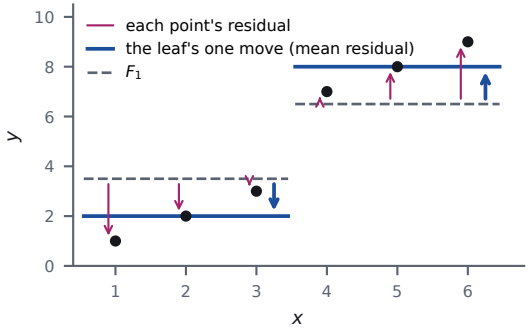


Boosting is a kind of gradient descent

Key idea: With fitted values $F(x_i)$ and loss $L(y, F) = \frac{1}{2}(y - F)^2$, the residuals are *negative gradients*:

$$-\frac{\partial L}{\partial F}(y_i, F(x_i)) = y_i - F(x_i).$$

- gradient descent would move each $F(x_i)$ by its own residual (arrows)
- a tree can't do this because each leaf has one value!

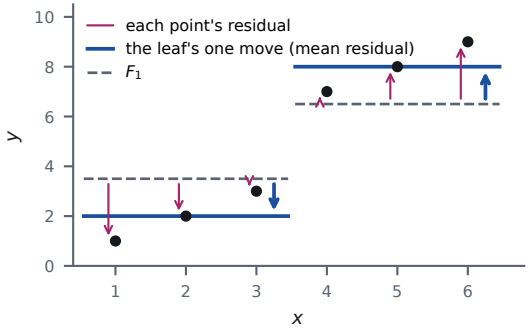


Boosting is a kind of gradient descent

Key idea: With fitted values $F(x_i)$ and loss $L(y, F) = \frac{1}{2}(y - F)^2$, the residuals are *negative gradients*:

$$-\frac{\partial L}{\partial F}(y_i, F(x_i)) = y_i - F(x_i).$$

- gradient descent would move each $F(x_i)$ by its own residual (arrows)
- a tree can't do this because each leaf has one value!
- it moves by *mean* residual instead

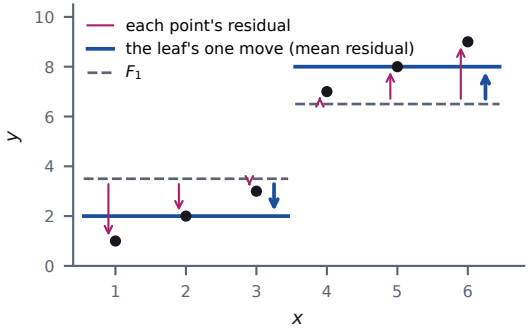


Boosting is a kind of gradient descent

Key idea: With fitted values $F(x_i)$ and loss $L(y, F) = \frac{1}{2}(y - F)^2$, the residuals are *negative gradients*:

$$-\frac{\partial L}{\partial F}(y_i, F(x_i)) = y_i - F(x_i).$$

- gradient descent would move each $F(x_i)$ by its own residual (arrows)
- a tree can't do this because each leaf has one value!
- it moves by *mean* residual instead

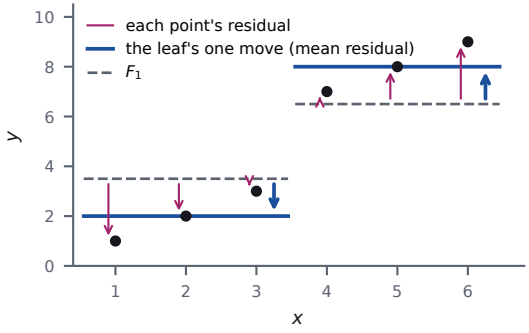


Boosting is a kind of gradient descent

Key idea: With fitted values $F(x_i)$ and loss $L(y, F) = \frac{1}{2}(y - F)^2$, the residuals are *negative gradients*:

$$-\frac{\partial L}{\partial F}(y_i, F(x_i)) = y_i - F(x_i).$$

- gradient descent would move each $F(x_i)$ by its own residual (arrows)
- a tree can't do this because each leaf has one value!
- it moves by *mean* residual instead



Note: For any other loss L, we can use $-\partial L/\partial F$ in similar way

Definition: Gradient boosting

Definition (Gradient boosting (Friedman, 2001))

Start from the best constant function, $F_0(x) = \arg\min_c \sum_i L(y_i, c)$. For $m = 1, \dots, M$:

1. compute the targets $r_{im} = -\frac{\partial L}{\partial F}(y_i, F_{m-1}(x_i))$
2. fit least-squares regression tree to the (x_i, r_{im}), keep leaves R_{jm}
3. for each leaf R_{jm}, choose best correction γ_{jm} to F_{m-1}:

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma),$$

and update $F_m = F_{m-1} + \nu h_m$, where $h_m(x) = \gamma_{jm}$ for $x \in R_{jm}$.

Choosing a loss

The loss decides what each tree is fitted to (step 1) and how far each leaf moves (step 3):

loss	targets r_{im}	leaf values γ_{jm}	use for
squared	$y_i - F(x_i)$	mean residual	regression
absolute	$\text{sign}(y_i - F(x_i))$	median residual	regression, y_i outliers
log loss	$y_i - p_i$	in general, no formula	classification

- squared loss predicts the mean, and a residual of 100 costs $10{,}000$ times one of 1; absolute loss predicts the median, and costs 100 times

Choosing a loss

The loss decides what each tree is fitted to (step 1) and how far each leaf moves (step 3):

loss	targets r_{im}	leaf values γ_{jm}	use for
squared	$y_i - F(x_i)$	mean residual	regression
absolute	$\text{sign}(y_i - F(x_i))$	median residual	regression, y_i outliers
log loss	$y_i - p_i$	in general, no formula	classification

- squared loss predicts the mean, and a residual of 100 costs $10{,}000$ times one of 1; absolute loss predicts the median, and costs 100 times
- absolute loss: the target is ± 1 (at $y_i = F(x_i)$, use 0)

Choosing a loss

The loss decides what each tree is fitted to (step 1) and how far each leaf moves (step 3):

loss	targets r_{im}	leaf values γ_{jm}	use for
squared	$y_i - F(x_i)$	mean residual	regression
absolute	$\text{sign}(y_i - F(x_i))$	median residual	regression, y_i outliers
log loss	$y_i - p_i$	in general, no formula	classification

- squared loss predicts the mean, and a residual of 100 costs $10{,}000$ times one of 1; absolute loss predicts the median, and costs 100 times
- absolute loss: the target is ± 1 (at $y_i = F(x_i)$, use 0)
- next: log loss for $y_i \in \{0, 1\}$, where F estimates log-odds

Plan for today

• Recap: bagging and random forests	2 min
• Boosting (regression)	12 min
• Gradient boosting	8 min
• Boosting (classification)	14 min
• Shrinkage, depth, and validation	8 min

Classification: F models the log-odds

Consider $y_i \in \{0, 1\}$. The tree value or "score" $F_m(x)$ is estimated log-odds:

$$F_m(x) \approx \log \frac{\eta(x)}{1 - \eta(x)}, \qquad \eta(x) = \mathbb{P}(Y = 1 \mid X = x).$$

Note: $F_0 = \log(\bar{y}/(1 - \bar{y}))$.

Idea: Compare to logistic regression, which models the log-odds by $\theta^\mathsf{T} x$. Here, it's a sum of trees!

Corresponding probability estimate is

$$\eta(x) \approx p_m(x) = \sigma(F_m(x)) = \frac{1}{1 + e^{-F_m(x)}}.$$

Scores and probabilities

$h_m(x)$ is a correction to log-odds, not a probability or class label:

$$F_m(x) = F_{m-1}(x) + \nu\, h_m(x), \qquad p_m(x) = \frac{1}{1 + e^{-F_m(x)}}.$$

Scores and probabilities

$h_m(x)$ is a correction to log-odds, not a probability or class label:

$$F_m(x) = F_{m-1}(x) + \nu\, h_m(x), \qquad p_m(x) = \frac{1}{1 + e^{-F_m(x)}}.$$

	one tree outputs	combined by	final estimate
tree (L19)	a probability, the class-1 fraction in the leaf: $\frac{1}{n_m} \sum_{x_i \in R_m} y_i$		that fraction, in $[0, 1]$
random forest (L20)	a probability $\hat{\eta}^{(b)}(x)$	averaging, on the probability scale	$\frac{1}{B} \sum_b \hat{\eta}^{(b)}(x)$, in $[0, 1]$
boosting	a correction $h_m(x)$ to the log-odds	adding, on the log-odds scale	$\sigma(F_0 + \nu \sum_m h_m(x))$, in $(0, 1)$

Where the targets $y - p$ come from

Recall: For one example with label $y \in \{0, 1\}$ and estimated probability $p = \sigma(F)$, the log loss is

$$L(y, F) = -\big[\, y \log p + (1 - y) \log(1 - p) \big], \qquad \frac{\partial L}{\partial F} = p - y.$$

L17's logistic regression gradient $\frac{1}{n} X^\mathsf{T}(p - y)$ sums these losses over the examples.

For gradient boosting with log loss, the targets are

$$r_{im} = -\frac{\partial L}{\partial F} = y_i - p_{m-1}(x_i).$$

Example: Two accounts

Credit default data. Suppose F_{m-1} has been fitted. For each training account, compute $p_{m-1}(x_i) = \sigma(F_{m-1}(x_i))$ and the target $y_i - p_{m-1}(x_i)$.

Problem

Find each target, and say which way that account's score should move.

(a) defaulted ($y = 1$), current estimate $p = 0.1$
(b) did not default ($y = 0$), current estimate $p = 0.9$

- (a) $1 - 0.1 = 0.9$: raise the log-odds; (b) $0 - 0.9 = -0.9$: lower it

Example: Two accounts

Credit default data. Suppose F_{m-1} has been fitted. For each training account, compute $p_{m-1}(x_i) = \sigma(F_{m-1}(x_i))$ and the target $y_i - p_{m-1}(x_i)$.

Problem

Find each target, and say which way that account's score should move.

(a) defaulted ($y = 1$), current estimate $p = 0.1$
(b) did not default ($y = 0$), current estimate $p = 0.9$

- (a) $1 - 0.1 = 0.9$: raise the log-odds; (b) $0 - 0.9 = -0.9$: lower it
- in one leaf together: mean target 0, so no single correction helps both

Gradient boosting for classification

Definition (Gradient boosting with log loss)

Start from $F_0 = \log(\bar{y}/(1-\bar{y}))$. For $m = 1, \dots, M$:

1. compute $p_i = \sigma(F_{m-1}(x_i))$ and the targets $r_i = y_i - p_i$
2. fit least-squares regression tree to the (x_i, r_i), keep leaves R_{jm}
3. for each leaf, set

$$\gamma_{jm} = \frac{\sum_{x_i \in R_{jm}} (y_i - p_i)}{\sum_{x_i \in R_{jm}} p_i(1-p_i)}$$

and update $F_m(x) = F_{m-1}(x) + \nu\gamma_{jm}$ for $x \in R_{jm}$.

Estimate $p_M(x) = \sigma(F_M(x))$.

Why divide by $p(1-p)$?

- the minimizer γ_{jm} (step 3 of the algorithm) has no closed form when L is the log loss:

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma),$$

Why divide by $p(1-p)$?

- the minimizer γ_{jm} (step 3 of the algorithm) has no closed form when L is the log loss:

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma),$$

- instead, Taylor-expand $L(y_i, F_{m-1}(x_i) + \gamma)$ around $\gamma = 0$ to 2nd order

Why divide by $p(1-p)$?

- the minimizer γ_{jm} (step 3 of the algorithm) has no closed form when L is the log loss:

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma),$$

- instead, Taylor-expand $L(y_i, F_{m-1}(x_i) + \gamma)$ around $\gamma = 0$ to 2nd order
- doing so gives $\gamma_{jm} \approx -\frac{\text{sum of } \partial L/\partial F}{\text{sum of } \partial^2 L/\partial F^2} = \frac{\sum(y_i - p_i)}{\sum p_i(1-p_i)}$

Why divide by $p(1-p)$?

- the minimizer γ_{jm} (step 3 of the algorithm) has no closed form when L is the log loss:

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma),$$

- instead, Taylor-expand $L(y_i, F_{m-1}(x_i) + \gamma)$ around $\gamma = 0$ to 2nd order
- doing so gives $\gamma_{jm} \approx -\frac{\text{sum of } \partial L/\partial F}{\text{sum of } \partial^2 L/\partial F^2} = \frac{\sum(y_i - p_i)}{\sum p_i(1-p_i)}$

Why divide by $p(1-p)$?

- the minimizer γ_{jm} (step 3 of the algorithm) has no closed form when L is the log loss:

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma),$$

- instead, Taylor-expand $L(y_i, F_{m-1}(x_i) + \gamma)$ around $\gamma = 0$ to 2nd order
- doing so gives $\gamma_{jm} \approx -\dfrac{\text{sum of } \partial L/\partial F}{\text{sum of } \partial^2 L/\partial F^2} = \dfrac{\sum(y_i - p_i)}{\sum p_i(1-p_i)}$

> **Note:** Optimal γ_{jm} can be far from 0, making the approximation poor. But small step size can save it!

Why divide by $p(1-p)$?

- the minimizer γ_{jm} (step 3 of the algorithm) has no closed form when L is the log loss:

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma),$$

- instead, Taylor-expand $L(y_i, F_{m-1}(x_i) + \gamma)$ around $\gamma = 0$ to 2nd order
- doing so gives $\gamma_{jm} \approx -\dfrac{\text{sum of } \partial L/\partial F}{\text{sum of } \partial^2 L/\partial F^2} = \dfrac{\sum(y_i - p_i)}{\sum p_i(1-p_i)}$

Note: Optimal γ_{jm} can be far from 0, making the approximation poor. But small step size can save it!

Key idea: XGBoost and LightGBM use this quadratic approximation for every loss—to set the leaf values, and also to choose the splits!

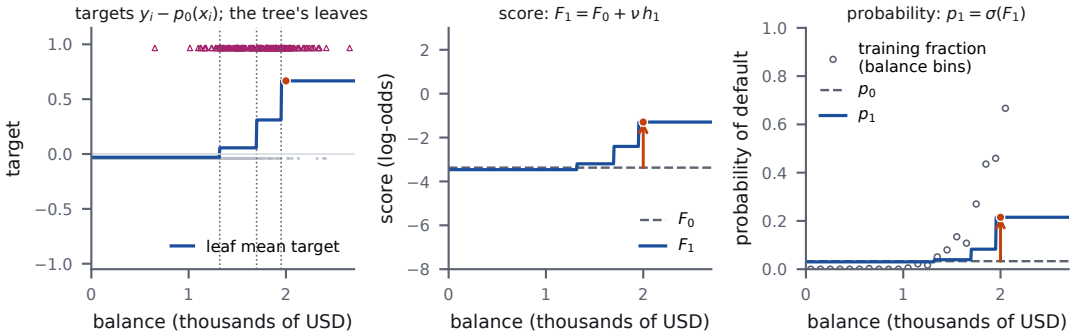
Aside: Reading the next three slides

Example: Credit default data, using balance only. Instead of stumps, use trees with two levels of splits. Learning rate $\nu = 0.1$. Each slide is one round.

left panel		**middle panel**		**right panel**
targets $y_i - p_{m-1}(x_i)$	$\xrightarrow{\text{tree, } \gamma_{jm}, \times\nu}$	score $F_m(x)$	$\xrightarrow{\sigma}$	probability $p_m(x)$
one per account		log-odds		between 0 and 1

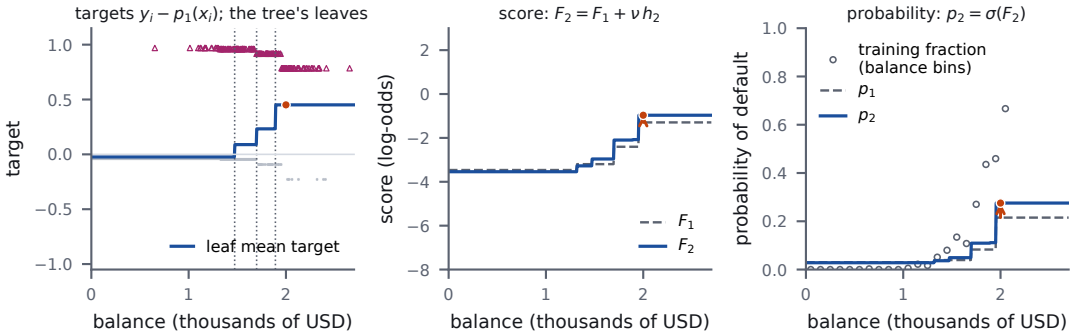
Example: Default, round 1

$p_0 = 0.033$ for every account, so each target is 0.967 (default) or -0.033



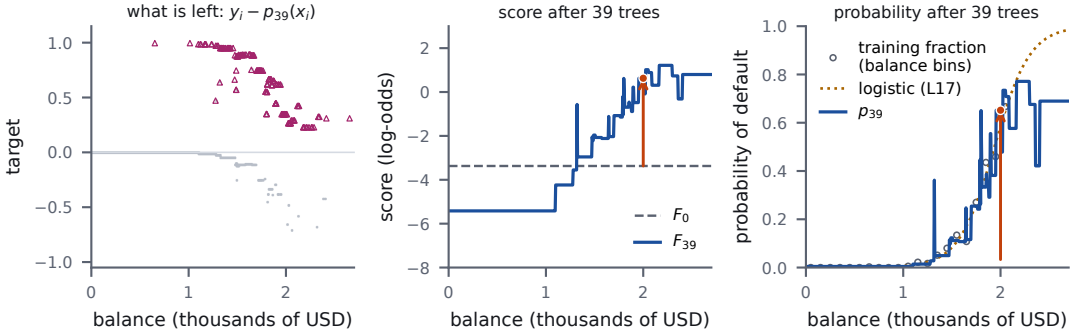
At $2,000: mean target is $0.667 \rightarrow$ leads to correction of $\nu\gamma = 2.08$, so score jumps from -3.37 to $-1.29 \rightarrow$ probability jumps from 0.033 to 0.215

Example: Default, round 2



The targets have changed, because $p_1 \neq p_0$: new targets give a new tree, with different splits.

Example: Default, after 39 trees



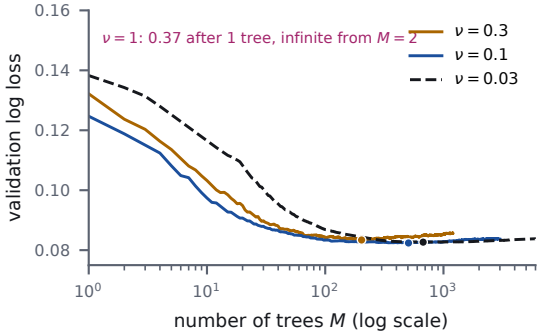
Validation chooses 39 trees. The corrections approximate L17's logistic curve (dotted) without assuming the log-odds are linear in balance.

Plan for today

- Recap: bagging and random forests — 2 min
- Boosting (regression) — 12 min
- Gradient boosting — 8 min
- Boosting (classification) — 14 min
- Shrinkage, depth, and validation — 8 min

Shrinkage: the learning rate ν

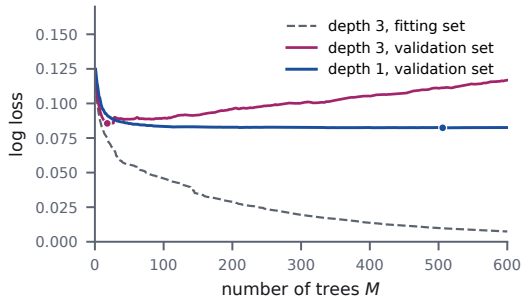
Example: Credit data using all three features. Boosting with stumps. Fit on $4{,}000$ accounts. Log loss on the $1{,}000$ validation accounts after each tree:



$\nu = 1$: one full step in a leaf of very confident accounts sends scores to $\pm 10^5$. Smaller ν: smaller steps, more trees, about the same best value.

How many trees? Use validation set

M, the depth, and ν are hyperparameters: choose them on validation data.
Example: Credit data gives $\nu = 0.1$, depths 1 and 3:



Depth 3 overfits after about 20 trees. Depth 1 wins: stumps give an additive log-odds, and default's log-odds look nearly additive (L17).

Key idea: Depth limits interactions

A sum of stumps is additive: $F(x) = c + f_1(x_1) + \cdots + f_p(x_p)$, each f_j a nonlinear step function, and no term involves two features.

- a tree of depth d involves at most d features along any path: interactions of at most d features (for classification, in the log-odds)

Key idea: Depth limits interactions

A sum of stumps is additive: $F(x) = c + f_1(x_1) + \cdots + f_p(x_p)$, each f_j a nonlinear step function, and no term involves two features.

- a tree of depth d involves at most d features along any path: interactions of at most d features (for classification, in the log-odds)
- L11's polynomial features of degree 2 on Advertising: TV^2 (curvature in one feature) and TV $\times$ radio (an interaction)

Key idea: Depth limits interactions

A sum of stumps is additive: $F(x) = c + f_1(x_1) + \cdots + f_p(x_p)$, each f_j a nonlinear step function, and no term involves two features.

- a tree of depth d involves at most d features along any path: interactions of at most d features (for classification, in the log-odds)
- L11's polynomial features of degree 2 on Advertising: TV^2 (curvature in one feature) and TV $\times$ radio (an interaction)
- depth is like the number of different features in a product, not the power: curvature in one feature costs nothing

Key idea: Depth limits interactions

A sum of stumps is additive: $F(x) = c + f_1(x_1) + \cdots + f_p(x_p)$, each f_j a nonlinear step function, and no term involves two features.

- a tree of depth d involves at most d features along any path: interactions of at most d features (for classification, in the log-odds)
- L11's polynomial features of degree 2 on Advertising: TV^2 (curvature in one feature) and TV $\times$ radio (an interaction)
- depth is like the number of different features in a product, not the power: curvature in one feature costs nothing
- Advertising, test MSE: stumps 2.70; depth 2: 0.91; depth 3: 0.70. Least squares: budgets 2.83, degree 2: 0.50, degree 3: 0.42

Example: Credit data, chosen on validation

Depth 1 and $M = 506$, refit on all $5{,}000$ training accounts, evaluated once on the test half:

	log loss	AUC	cost at $t = 0.1$
boosted stumps	0.0790	0.944	609
logistic regression (L17)	0.0758	0.949	624

- cost $= \mathrm{FP} + 9\,\mathrm{FN}$ (L17: a missed default costs 9, so flag when $\hat{\eta} \geq 0.1$)

Example: Credit data, chosen on validation

Depth 1 and $M = 506$, refit on all $5{,}000$ training accounts, evaluated once on the test half:

	log loss	AUC	cost at $t = 0.1$
boosted stumps	0.0790	0.944	609
logistic regression (L17)	0.0758	0.949	624

- cost $= \mathrm{FP} + 9\,\mathrm{FN}$ (L17: a missed default costs 9, so flag when $\hat{\eta} \geq 0.1$)
- which would you deploy if cost were the objective? Decide the criterion before looking at the test set; a difference of 15 is small

Python: Choose M, then refit

```
import numpy as np
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import log_loss
# depth 1 was already chosen the same way
kw = dict(max_depth=1, learning_rate=0.1, random_state=0)
gb = GradientBoostingClassifier(n_estimators=1000, **kw)
gb.fit(X_fit, y_fit)
loss = [log_loss(y_val, p[:, 1])
        for p in gb.staged_predict_proba(X_val)]
M = int(np.argmin(loss)) + 1          # 506 here
best = GradientBoostingClassifier(n_estimators=M, **kw)
best.fit(X_train, y_train)    # fitting + validation
p_hat = best.predict_proba(X_test)[:, 1]
```

A question

Question

Two ensembles both contain 500 stumps. In one, every stump is fitted to the original responses on a bootstrap sample. In the other, each stump is fitted to the errors left by the previous updates. Why can their predictions differ substantially? What limitation do both have when the fitted function requires interactions?

Summary

- Averaging cannot remove bias that every tree shares. Boosting instead adds small trees one at a time, each fitted to what earlier ones missed.
- Gradient boosting fits each tree to the negative gradient of the loss, then sets each leaf's value by the loss.
- For classification, each tree corrects the estimated log-odds, giving a probability estimate of $p = \sigma(F)$.
- Choose the number of trees, the depth and ν on a validation set; depth limits the interactions in the score.