Lecture 20

Bagging and 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

Announcements & Reminders

- This lecture is being recorded.
- I'll release midterm scores this week.

Plan for today

• Recap: decision trees	3 min
• Bagging	11 min
• Random forests	6 min
• An application	4 min
• Fitting what is left (regression)	12 min
• Gradient boosting	8 min

Recall: Decision trees

- tree $f(x) = \sum_m c_m \mathbf{1}\{x \in R_m\}$ is constant on each leaf

Recall: Decision trees

- tree $f(x) = \sum_m c_m \mathbf{1}\{x \in R_m\}$ is constant on each leaf
- leaf $=$ a rectangle R_m in feature space

Recall: Decision trees

- tree $f(x) = \sum_m c_m \mathbf{1}\{x \in R_m\}$ is constant on each leaf
- leaf $=$ a rectangle R_m in feature space
- regression: c_m is the mean of the y_i in R_m

Recall: Decision trees

- tree $f(x) = \sum_m c_m \mathbf{1}\{x \in R_m\}$ is constant on each leaf
- leaf $=$ a rectangle R_m in feature space
- regression: c_m is the mean of the y_i in R_m
- classification: same, but now c_m is a probability estimate

Recall: Decision trees

- tree $f(x) = \sum_m c_m \mathbf{1}\{x \in R_m\}$ is constant on each leaf
- leaf $=$ a rectangle R_m in feature space
- regression: c_m is the mean of the y_i in R_m
- classification: same, but now c_m is a probability estimate
- greedy recursive splitting: at each node, choose the feature–threshold pair (j, s) that minimizes impurity $n_L Q(\text{left}) + n_R Q(\text{right})$

Recall: Decision trees

- tree $f(x) = \sum_m c_m \mathbf{1}\{x \in R_m\}$ is constant on each leaf
- leaf $=$ a rectangle R_m in feature space
- regression: c_m is the mean of the y_i in R_m
- classification: same, but now c_m is a probability estimate
- greedy recursive splitting: at each node, choose the feature–threshold pair (j, s) that minimizes impurity $n_L Q(\text{left}) + n_R Q(\text{right})$
- impurity Q: mean squared deviation (regression); Gini impurity, which ranks splits as least squares on the $0/1$ labels (classification)

Recall: Decision trees

- tree $f(x) = \sum_m c_m \mathbf{1}\{x \in R_m\}$ is constant on each leaf
- leaf $=$ a rectangle R_m in feature space
- regression: c_m is the mean of the y_i in R_m
- classification: same, but now c_m is a probability estimate
- greedy recursive splitting: at each node, choose the feature–threshold pair (j, s) that minimizes impurity $n_L Q(\text{left}) + n_R Q(\text{right})$
- impurity Q: mean squared deviation (regression); Gini impurity, which ranks splits as least squares on the $0/1$ labels (classification)
- stopping rules: maximum depth, minimum leaf size, chosen on a validation set

Recall: Deep trees have large variance

Example (Credit data)

Using balance, income, student indicator with no stopping rule results in 190 leaves, every training account correct. But on the $5{,}000$ test accounts:

- every estimate is 0 or 1
- 109 defaulters get $\hat{\eta} = 0$, 105 non-defaulters get $\hat{\eta} = 1$
- Brier score 0.043; predicting 0.033 for everyone gives 0.032
- leaves of at least 200, chosen on a validation set: 0.023

Key idea: At fixed x, with $\eta = \mathbb{P}(Y = 1 \mid X = x)$:

$$\mathbb{E}\big[(\hat{\eta}(x) - Y)^2\big] = \eta(1-\eta) + \big(\mathbb{E}\hat{\eta}(x) - \eta\big)^2 + \operatorname{Var}\hat{\eta}(x).$$

Deep trees have small bias, large variance.

Plan for today

• Recap: decision trees	3 min
• Bagging	11 min
• Random forests	6 min
• An application	4 min
• Fitting what is left (regression)	12 min
• Gradient boosting	8 min

Deep trees are unstable

Example: Split $5{,}000$ training accounts into two sets of $2{,}500$. Grow a tree on each half with no stopping rule. Both halves come from the same population, so *ideally* the two trees would give similar estimates.

Key idea: Deep trees are unstable

Example: Split $5{,}000$ training accounts into two sets of $2{,}500$, called A and B. Grow a tree on each with no stopping rule. Both come from the same population, so *ideally* the A and B trees would give similar estimates.

- compare estimates $\hat{\eta}^{(A)}(x)$ and $\hat{\eta}^{(B)}(x)$ on the test accounts

Key idea: Deep trees are unstable

Example: Split $5{,}000$ training accounts into two sets of $2{,}500$, called A and B. Grow a tree on each with no stopping rule. Both come from the same population, so *ideally* the A and B trees would give similar estimates.

- compare estimates $\hat{\eta}^{(A)}(x)$ and $\hat{\eta}^{(B)}(x)$ on the test accounts
- for 242 test accounts, one tree says 0 and the other says 1

Key idea: Deep trees are unstable

Example: Split $5{,}000$ training accounts into two sets of $2{,}500$, called A and B. Grow a tree on each with no stopping rule. Both come from the same population, so *ideally* the A and B trees would give similar estimates.

- compare estimates $\hat{\eta}^{(A)}(x)$ and $\hat{\eta}^{(B)}(x)$ on the test accounts
- for 242 test accounts, one tree says 0 and the other says 1
- among the 456 test accounts with balance above $\$1{,}500$: 167, or 36.6%

Key idea: Deep trees are unstable

Example: Split $5{,}000$ training accounts into two sets of $2{,}500$, called A and B. Grow a tree on each with no stopping rule. Both come from the same population, so *ideally* the A and B trees would give similar estimates.

- compare estimates $\hat{\eta}^{(A)}(x)$ and $\hat{\eta}^{(B)}(x)$ on the test accounts
- for 242 test accounts, one tree says 0 and the other says 1
- among the 456 test accounts with balance above $\$1{,}500$: 167, or 36.6%
- the estimates therefore depend heavily on which accounts happen to be in A and B

Key idea: Deep trees are unstable

Example: Split $5{,}000$ training accounts into two sets of $2{,}500$, called A and B. Grow a tree on each with no stopping rule. Both come from the same population, so *ideally* the A and B trees would give similar estimates.

- compare estimates $\hat{\eta}^{(A)}(x)$ and $\hat{\eta}^{(B)}(x)$ on the test accounts
- for 242 test accounts, one tree says 0 and the other says 1
- among the 456 test accounts with balance above $\$1{,}500$: 167, or 36.6%
- the estimates therefore depend heavily on which accounts happen to be in A and B

Key idea: Deep trees are unstable

Example: Split $5{,}000$ training accounts into two sets of $2{,}500$, called A and B. Grow a tree on each with no stopping rule. Both come from the same population, so *ideally* the A and B trees would give similar estimates.

- compare estimates $\hat{\eta}^{(A)}(x)$ and $\hat{\eta}^{(B)}(x)$ on the test accounts
- for 242 test accounts, one tree says 0 and the other says 1
- among the 456 test accounts with balance above $\$1{,}500$: 167, or 36.6%
- the estimates therefore depend heavily on which accounts happen to be in A and B

Idea: Suppress variance by averaging many such trees, each grown on different subsets of the training data.

Aside: The bootstrap

Definition (Bootstrap sample (Efron, 1979))

Draw n points with replacement from the n training points.

- training set $\{1, 2, 3, 4\}$; bootstrap samples, e.g., $\{2, 2, 3, 3\}$ or $\{1, 1, 1, 3\}$

Aside: The bootstrap

Definition (Bootstrap sample (Efron, 1979))

Draw n points with replacement from the n training points.

- training set $\{1, 2, 3, 4\}$; bootstrap samples, e.g., $\{2, 2, 3, 3\}$ or $\{1, 1, 1, 3\}$
- treats training set as if it were the entire population

Aside: The bootstrap

Definition (Bootstrap sample (Efron, 1979))

Draw n points with replacement from the n training points.

- training set $\{1, 2, 3, 4\}$; bootstrap samples, e.g., $\{2, 2, 3, 3\}$ or $\{1, 1, 1, 3\}$
- treats training set as if it were the entire population
- bootstrap samples imitate collecting a new training set

Aside: The bootstrap

Definition (Bootstrap sample (Efron, 1979))

Draw n points with replacement from the n training points.

- training set $\{1, 2, 3, 4\}$; bootstrap samples, e.g., $\{2, 2, 3, 3\}$ or $\{1, 1, 1, 3\}$
- treats training set as if it were the entire population
- bootstrap samples imitate collecting a new training set
- recompute any statistic/fitted model on each bootstrap sample

Aside: The bootstrap

Definition (Bootstrap sample (Efron, 1979))

Draw n points with replacement from the n training points.

- training set $\{1, 2, 3, 4\}$; bootstrap samples, e.g., $\{2, 2, 3, 3\}$ or $\{1, 1, 1, 3\}$
- treats training set as if it were the entire population
- bootstrap samples imitate collecting a new training set
- recompute any statistic/fitted model on each bootstrap sample
- the spread across samples shows how much it depends on data

Aside: The bootstrap

Definition (Bootstrap sample (Efron, 1979))

Draw n points with replacement from the n training points.

- training set $\{1, 2, 3, 4\}$; bootstrap samples, e.g., $\{2, 2, 3, 3\}$ or $\{1, 1, 1, 3\}$
- treats training set as if it were the entire population
- bootstrap samples imitate collecting a new training set
- recompute any statistic/fitted model on each bootstrap sample
- the spread across samples shows how much it depends on data

Aside: The bootstrap

Definition (Bootstrap sample (Efron, 1979))

Draw n points with replacement from the n training points.

- training set $\{1, 2, 3, 4\}$; bootstrap samples, e.g., $\{2, 2, 3, 3\}$ or $\{1, 1, 1, 3\}$
- treats training set as if it were the entire population
- bootstrap samples imitate collecting a new training set
- recompute any statistic/fitted model on each bootstrap sample
- the spread across samples shows how much it depends on data

> **Note:** Every bootstrap sample has the same number of points as the original training set.

Definition: Bagging

Definition (Bootstrap aggregation (Breiman, 1996))

For $b = 1, \ldots, B$: draw a bootstrap sample and grow a tree $\hat{\eta}^{(b)}$ on it. Estimate

$$\hat{\eta}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{\eta}^{(b)}(x).$$

- we average the trees' estimated probabilities, as scikit-learn does (another version takes a majority vote of their labels)

Definition: Bagging

Definition (Bootstrap aggregation (Breiman, 1996))

For $b = 1, \dots, B$: draw a bootstrap sample and grow a tree $\hat{\eta}^{(b)}$ on it. Estimate

$$\hat{\eta}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{\eta}^{(b)}(x).$$

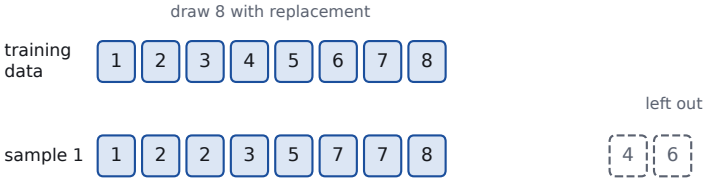
- we average the trees' estimated probabilities, as scikit-learn does (another version takes a majority vote of their labels)
- the trees are grown without a stopping rule: each has small bias and large variance

Bagging, visualized



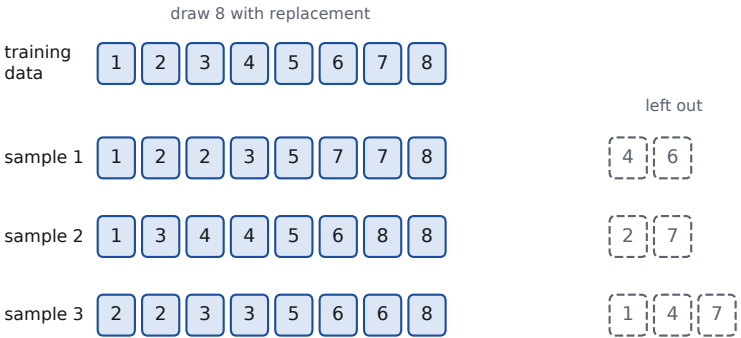
Eight training points.

Bagging, visualized



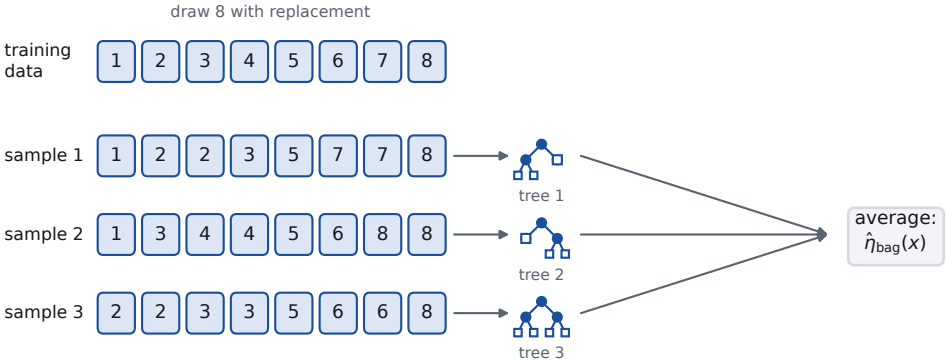
A bootstrap sample: 8 draws with replacement. Points 2 and 7 appear twice; 4 and 6 are left out.

Bagging, visualized



Each bootstrap sample repeats some points and leaves out others, at random.

Bagging, visualized



Grow one tree on each sample, then average the trees' estimates.

Averaging lowers the variance

Proposition

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

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

- $T_b = \hat{\eta}^{(b)}(x)$ at a fixed x; random through the training data and the bootstrap draw

Averaging lowers the variance

Proposition

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

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

- $T_b = \hat{\eta}^{(b)}(x)$ at a fixed x; random through the training data and the bootstrap draw
- $\mathrm{Var}(\sum_b T_b) = B\sigma^2 + B(B-1)\rho\sigma^2$ (Check this.)

Averaging lowers the variance

Proposition

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

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

- $T_b = \hat{\eta}^{(b)}(x)$ at a fixed x; random through the training data and the bootstrap draw
- $\mathrm{Var}(\sum_b T_b) = B\sigma^2 + B(B-1)\rho\sigma^2$ (Check this.)
- more trees shrink the first term, but the trees share one training set, so $\rho > 0$ and $\rho\sigma^2$ remain

Out-of-bag estimates

- each bootstrap sample leaves out some training points

Out-of-bag estimates

- each bootstrap sample leaves out some training points
- so every training point was not used by some of the trees

Out-of-bag estimates

- each bootstrap sample leaves out some training points
- so every training point was not used by some of the trees
- those trees can predict it as if it were new data

Out-of-bag estimates

- each bootstrap sample leaves out some training points
- so every training point was not used by some of the trees
- those trees can predict it as if it were new data

Out-of-bag estimates

- each bootstrap sample leaves out some training points
- so every training point was not used by some of the trees
- those trees can predict it as if it were new data

Definition (Out-of-bag estimate)

For training point i, average $\widehat{\eta}^{(b)}(x_i)$ over only the trees b whose bootstrap sample did not contain i. Comparing these estimates with the y_i estimates the error on new data, with no validation set.

Out-of-bag estimates

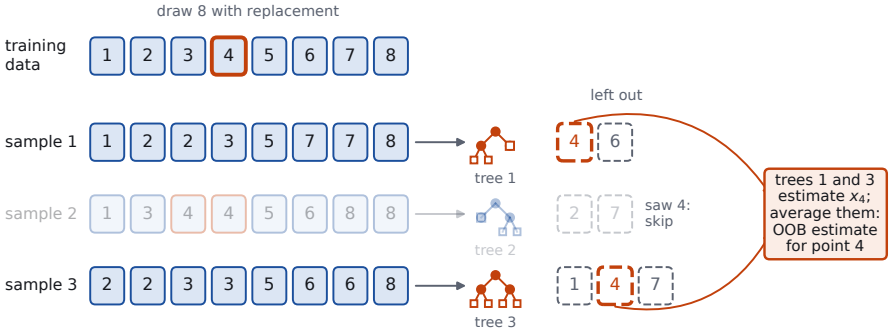
- each bootstrap sample leaves out some training points
- so every training point was not used by some of the trees
- those trees can predict it as if it were new data

Definition (Out-of-bag estimate)

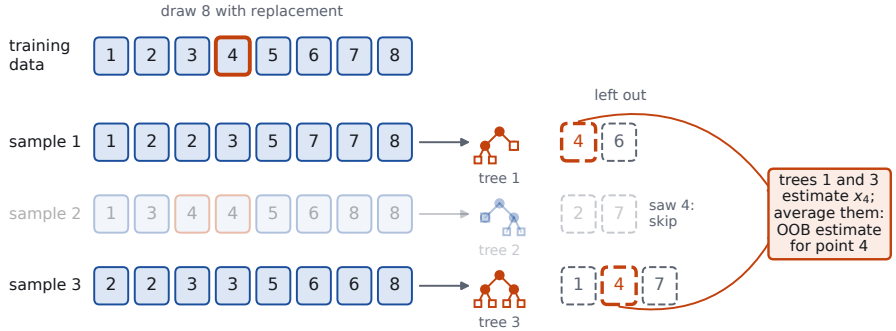
For training point i, average $\hat{\eta}^{(b)}(x_i)$ over only the trees b whose bootstrap sample did not contain i. Comparing these estimates with the y_i estimates the error on new data, with no validation set.

Note: Point i is left out with probability $(1 - 1/n)^n \to e^{-1} \approx 0.37$ (why?): about a third of the trees. Used to choose settings, it is part of model selection, not a final test.

Out-of-bag estimates, visualized



Out-of-bag estimates, visualized



Key idea: OOB error is almost identical to cross-validation error, but you can calculate it as you go! No refitting needed.

Plan for today

• Recap: decision trees	3 min
• Bagging	11 min
• Random forests	6 min
• An application	4 min
• Fitting what is left (regression)	12 min
• Gradient boosting	8 min

Reminder: Averaging lowers the variance

Proposition

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

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

Key idea: Variance is our major concern. Increasing B shrinks the first term in the variance, but the trees share one training set, hence $\rho > 0$.

Reminder: Averaging lowers the variance

Proposition

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

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

Key idea: Variance is our major concern. Increasing B shrinks the first term in the variance, but the trees share one training set, hence $\rho > 0$.

Question

How can we reduce the correlation ρ?

Definition: Random forest

Definition (Random forest (Breiman, 2001))

Bagging, except that each split chooses among only m of the p features, drawn at random for that split.

- Pro: lowering correlation ρ between trees will reduce variance

Definition: Random forest

Definition (Random forest (Breiman, 2001))

Bagging, except that each split chooses among only m of the p features, drawn at random for that split.

- Pro: lowering correlation ρ between trees will reduce variance
- Con: forcing splits on weak features can make trees biased

Definition: Random forest

Definition (Random forest (Breiman, 2001))

Bagging, except that each split chooses among only m of the p features, drawn at random for that split.

- Pro: lowering correlation ρ between trees will reduce variance
- Con: forcing splits on weak features can make trees biased
- $m = p$ is bagging; Breiman suggests $m \approx \sqrt{p}$ for classification

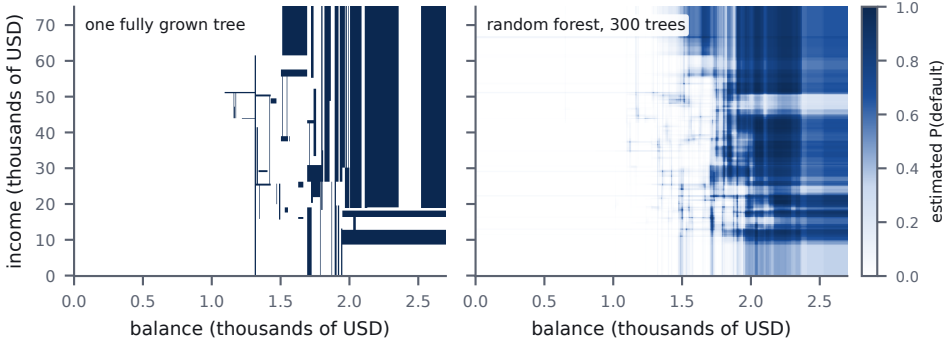
Definition: Random forest

Definition (Random forest (Breiman, 2001))

Bagging, except that each split chooses among only m of the p features, drawn at random for that split.

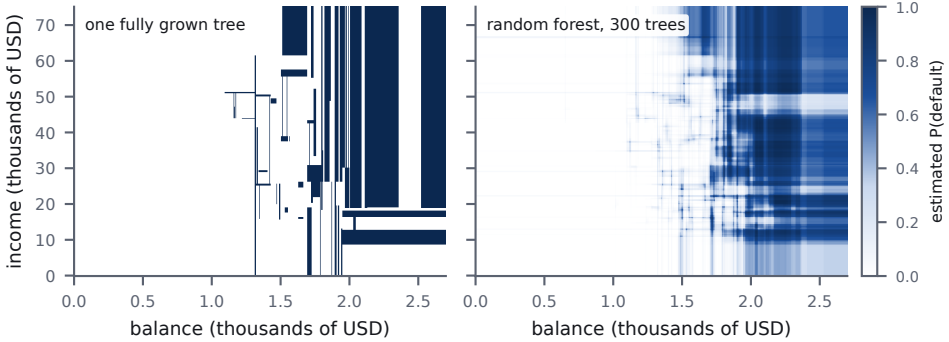
- Pro: lowering correlation ρ between trees will reduce variance
- Con: forcing splits on weak features can make trees biased
- $m = p$ is bagging; Breiman suggests $m \approx \sqrt{p}$ for classification
- Example: credit default dataset has $p = 3$ features, so we use $m = 1$

Example: tree versus forest



- left = grown until pure

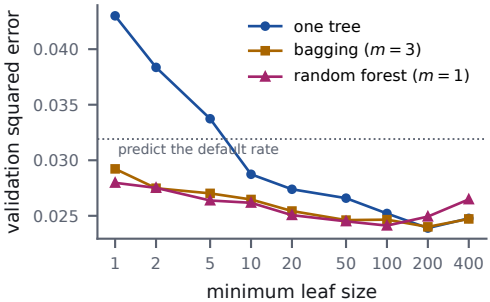
Example: tree versus forest



- left $=$ grown until pure
- right $=$ average of estimates from 300 trees ($m = 1$)

Example: Leaf size, and averaging

Each point $=$ squared error on $1{,}000$ validation accounts from fit on $4{,}000$ accounts with a given minimum leaf size.



Note: At large leaf size, each tree already averages many accounts, so averaging trees changes little.

Python: A random forest

```
from sklearn.ensemble import RandomForestClassifier
forest = RandomForestClassifier(n_estimators=300,
            max_features=1, oob_score=True,
            random_state=0)
forest.fit(X_train, y_train)  # all three features
p_hat = forest.predict_proba(X_test)[:, 1]  # average
p_oob = forest.oob_decision_function_[:, 1]
np.mean((p_oob - y_train) ** 2)  # OOB squared error
```

- `max_features` is m; the classifier's default is $\sqrt{p}$
- by default `oob_score_` is the OOB accuracy

A question

Question

In a forest of fully grown trees, each tree fits its own bootstrap sample almost exactly.

(a) Why can the forest estimate new accounts better than any one of its trees?

(b) Going from 100 to $1{,}000$ trees: what changes, and can it make the forest overfit?

One more time: Averaging lowers the variance

Proposition

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

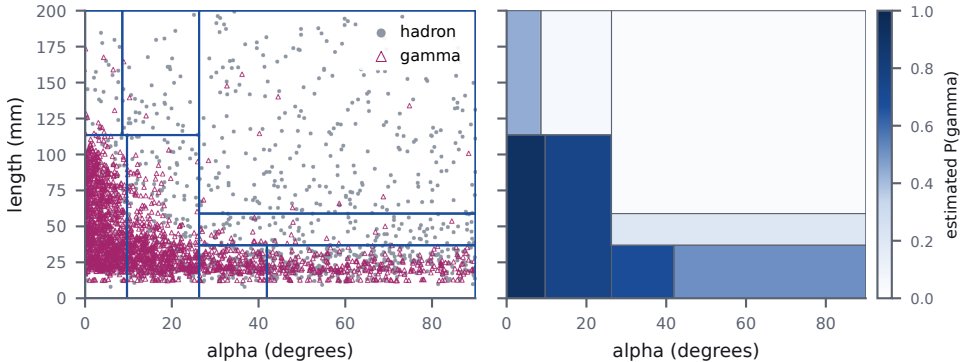
$$\mathbb{E}\left[\frac{1}{B}\sum_b T_b\right] = \mu, \qquad \mathrm{Var}\left(\frac{1}{B}\sum_b T_b\right) = \frac{1-\rho}{B}\sigma^2 + \rho\,\sigma^2.$$

Plan for today

• Recap: decision trees	3 min
• Bagging	11 min
• Random forests	6 min
• An application	4 min
• Fitting what is left (regression)	12 min
• Gradient boosting	8 min

Application: Gamma rays or background?

MAGIC telescopes image light from particle showers in the air: gamma ray (signal) or hadron (background)?



Application: Gamma rays or background?

19,020 simulated events (UCI): fit 8,000; choose settings on 2,000; refit on the 10,000; test once on 9,020.

	accuracy	AUC	gammas kept*
logistic regression	79.0%	0.842	33%
one tree (leaves ≥ 20)	84.8%	0.886	—
random forest (300 trees)	87.6%	0.932	60%

*fraction of gammas flagged at the threshold that lets through 5% of hadrons

Note: On the credit data, logistic regression was better. The random forest, and even shallow trees on two features, win here.

Plan for today

• Recap: decision trees	3 min
• Bagging	11 min
• Random forests	6 min
• An application	4 min
• Fitting what is left (regression)	12 min
• Gradient boosting	8 min

JK, it's back: Averaging lowers variance, but not bias!

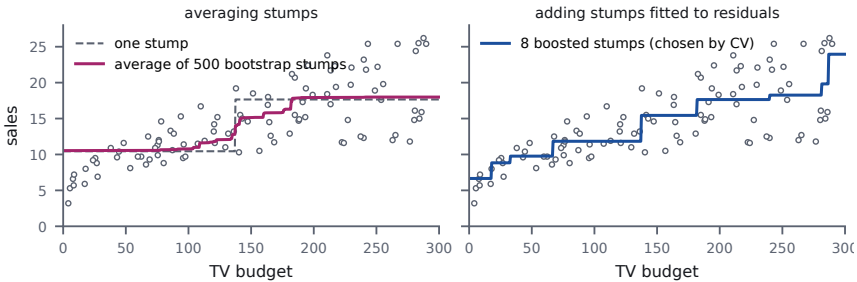
Proposition

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

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

Example: Averaging can't reduce bias on sales data

Example: Regress sales against TV using 100 training markets.

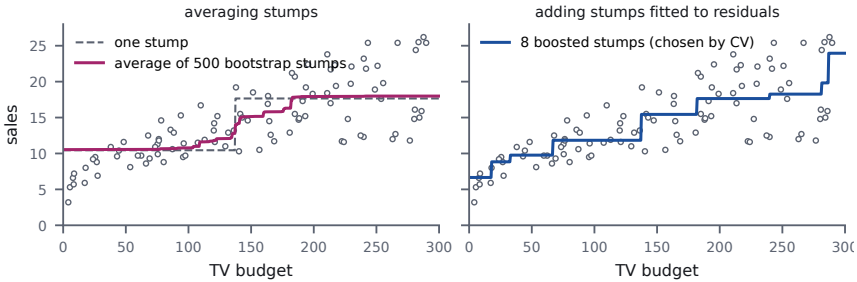


Stump $=$ a tree with one split.

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

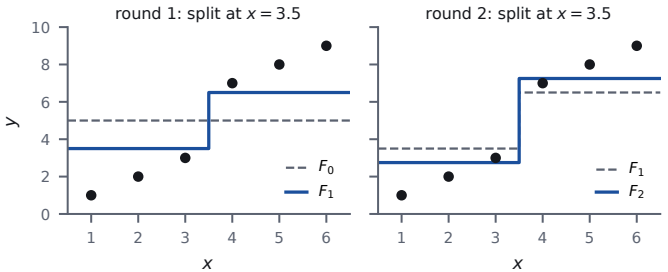
Problem

L19's six points, $x = 1, \ldots, 6$ and $y = 1, 2, 3, 7, 8, 9$. Stumps, $\nu = \frac{1}{2}$, $F_0 = 5$. Find F_1, F_2 and F_3.

Example: Boosting six points

Problem

L19's six points, $x = 1, \ldots, 6$ and $y = 1, 2, 3, 7, 8, 9$. Stumps, $\nu = \frac{1}{2}$, $F_0 = 5$. Find F_1, F_2 and F_3.



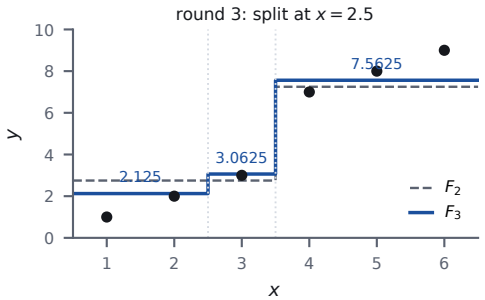
Rounds $1, 2$ both split at $x = 3.5$, as residuals jump between $x = 3$ and 4.

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

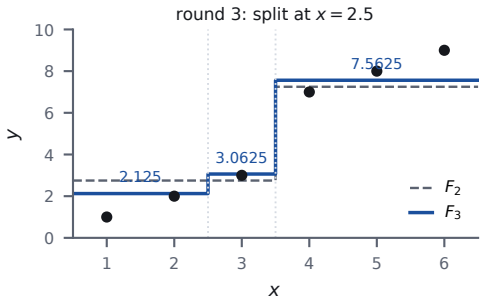


Round 3: the split changes

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)

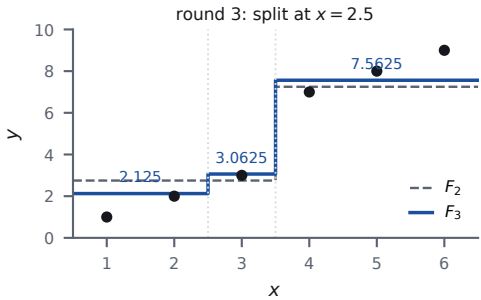


Round 3: the split changes

Residuals after round 2:

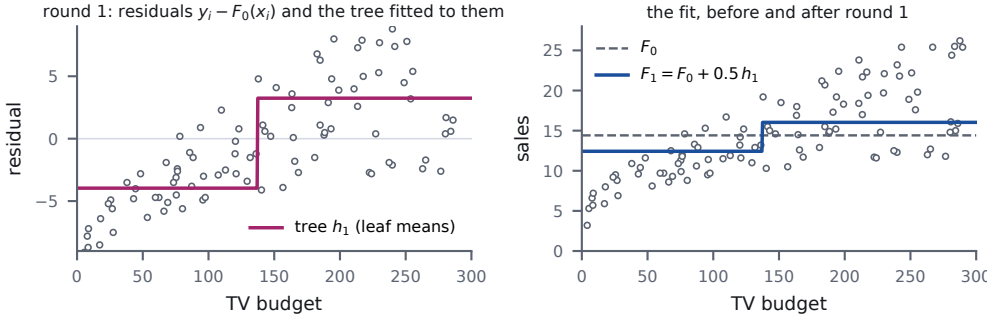
$$(-1.75,\ -0.75,\ 0.25,\ -0.25,\ 0.75,\ 1.75)$$

- 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

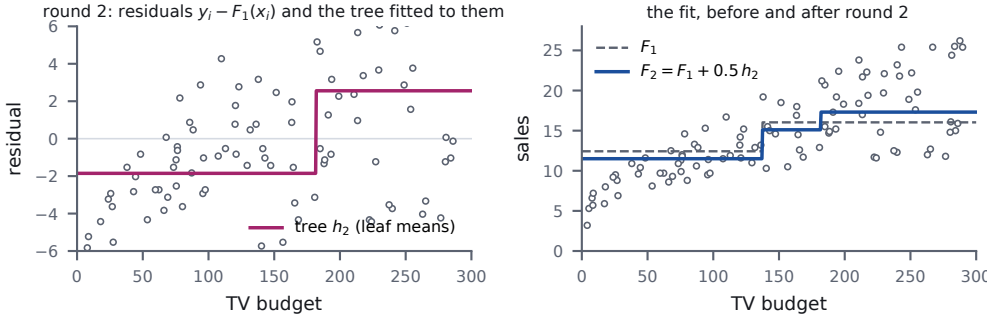
Example: Regress sales against TV using 100 training markets.



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

Example: Boosting on advertising data

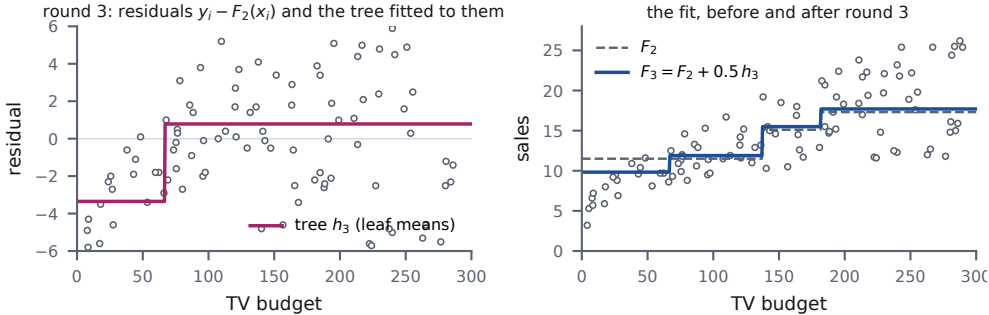
Example: Regress sales against TV using 100 training markets.



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

Example: Boosting on advertising data

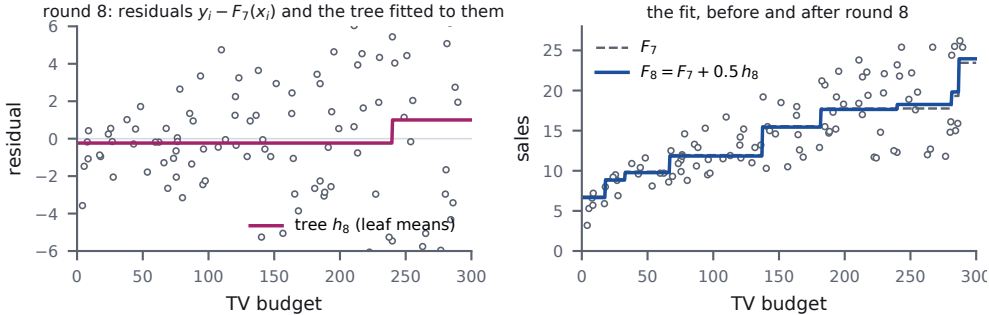
Example: Regress sales against TV using 100 training markets.



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

Example: Boosting on advertising data

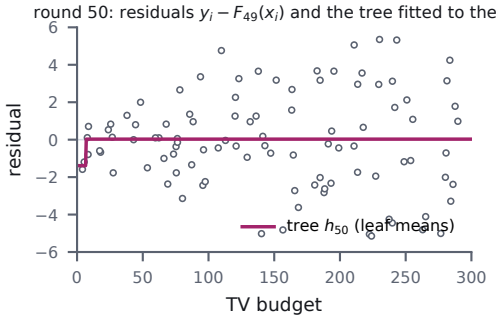
Example: Regress sales against TV using 100 training markets.

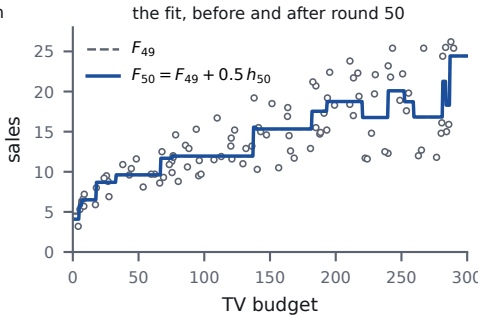


$M = 8$: the number of trees chosen by 5-fold cross-validation

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

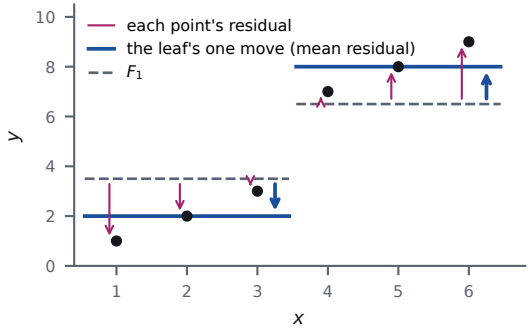
• Recap: decision trees	3 min
• Bagging	11 min
• Random forests	6 min
• An application	4 min
• Fitting what is left (regression)	12 min
• Gradient boosting	8 min

Residuals are negative gradients

Treat the fitted values $F(x_i)$ as unknowns; with $L(y, F) = \frac{1}{2}(y - F)^2$,

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

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

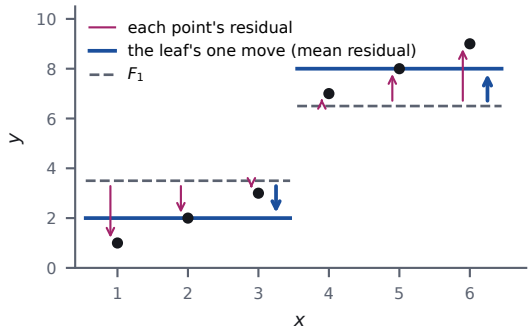


Residuals are negative gradients

Treat the fitted values $F(x_i)$ as unknowns; with $L(y, F) = \frac{1}{2}(y - F)^2$,

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

- gradient descent would move each $F(x_i)$ by its own residual (arrows)
- a tree cannot: a leaf has one value, so it moves by the mean residual

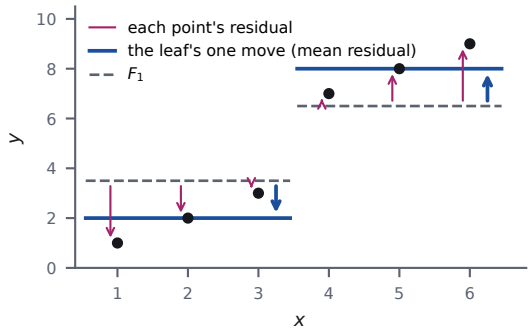


Residuals are negative gradients

Treat the fitted values $F(x_i)$ as unknowns; with $L(y, F) = \frac{1}{2}(y - F)^2$,

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

- gradient descent would move each $F(x_i)$ by its own residual (arrows)
- a tree cannot: a leaf has one value, so it moves by the mean residual

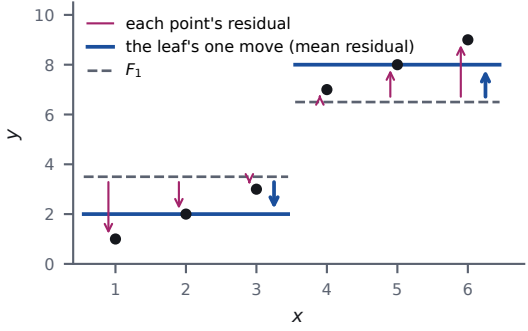


Residuals are negative gradients

Treat the fitted values $F(x_i)$ as unknowns; with $L(y, F) = \frac{1}{2}(y - F)^2$,

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

- gradient descent would move each $F(x_i)$ by its own residual (arrows)
- a tree cannot: a leaf has one value, so it moves by the mean residual



Note: Fit the trees to $-\partial L/\partial F$: the same recipe, for any loss.

Definition: Gradient boosting

Definition (Gradient boosting (Friedman, 2001; ESL Alg. 10.3))

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

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

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

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, 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, 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, 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)
- log loss: $y_i \in \{0, 1\}$, F a log-odds, and probabilities; next time

Summary

- deep trees have small bias, large variance; averaging keeps their mean and lowers their variance toward $\rho\sigma^2$
- bagging grows the trees on bootstrap samples; random forests also choose among m random features at each split, aiming to lower ρ
- averaging cannot remove a bias every tree shares; boosting adds small trees one at a time, each fitted to what the earlier ones left
- gradient boosting fits each tree to the negative gradient of the loss (the residuals, for squared loss), then sets each leaf's value by the loss