Lecture 22

XGBoost

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 of gradient boosting for classification	2 min
• Credit data example	6 min
• Shrinkage, depth, and validation	8 min
• Leaf values: one Newton step	10 min
• XGBoost $=$ quadratic for splits too $+$ penalty	15 min
• Scoring a split	5 min

Recall: 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 a least-squares regression tree to the (x_i, r_i); 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 $p_M(x) = \sigma(F_M(x))$.

Plan for today

• Recap of gradient boosting for classification	2 min
• Credit data example	6 min
• Shrinkage, depth, and validation	8 min
• Leaf values: one Newton step	10 min
• XGBoost $=$ quadratic for splits too $+$ penalty	15 min
• Scoring a split	5 min

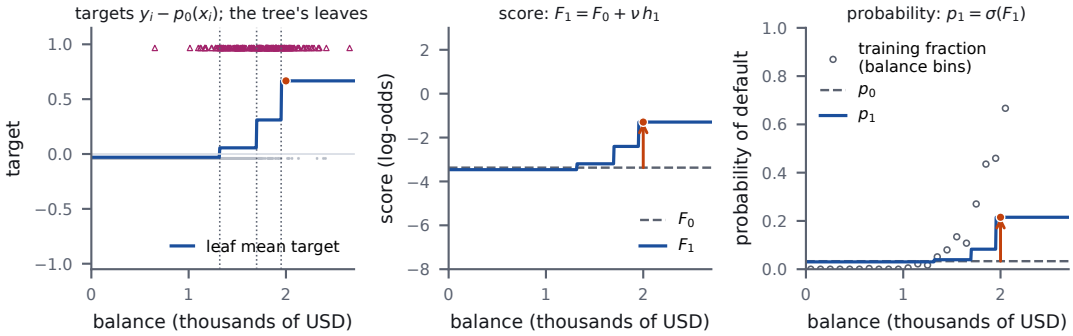
Aside: Reading the next three slides

Example: Credit default data, balance only, $5{,}000$ training accounts. Trees with two levels of splits; $\nu = 0.1$. 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

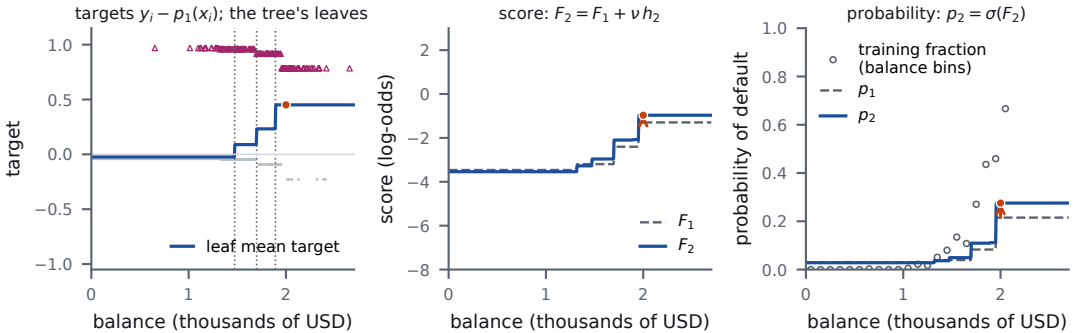
Example: Default, round 1

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



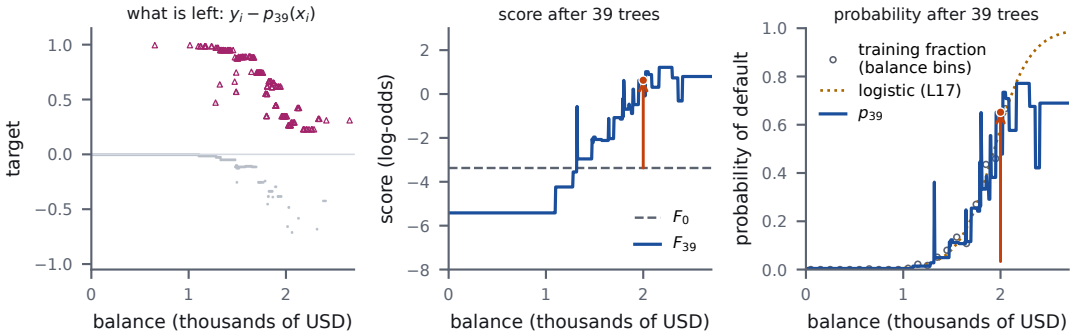
At $2,000: mean target is $0.667 \rightarrow$ leads to correction of $\nu\gamma = 2.08$, so score jumps from -3.37 to $-1.29 \rightarrow$ probability jumps from 0.033 to 0.215

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



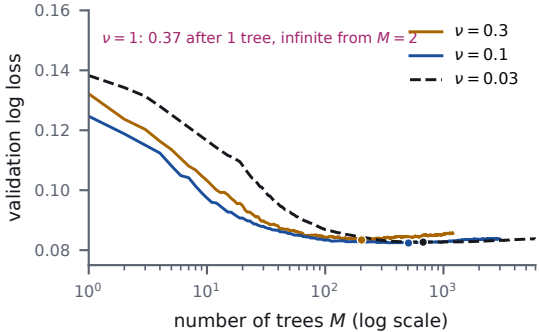
Validation chooses 39 trees. The corrections approximate L17's logistic curve (dotted) without assuming the log-odds are linear in balance.

Plan for today

• Recap of gradient boosting for classification	2 min
• Credit data example	6 min
• Shrinkage, depth, and validation	8 min
• Leaf values: one Newton step	10 min
• XGBoost $=$ quadratic for splits too $+$ penalty	15 min
• Scoring a split	5 min

Shrinkage: the learning rate ν

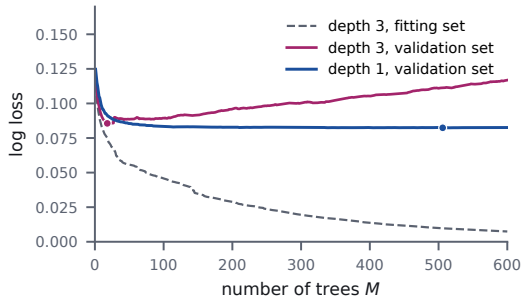
Example: Credit data using all three features. Boosting with stumps. Fit on $4{,}000$ accounts. Log loss on the $1{,}000$ validation accounts after each tree:



$\nu = 1$: 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

Example (Depth $d = 1$)

A sum of stumps is additive. For any number of trees M, we can write $F_M(x) = c + f_1(x_1) + \cdots + f_p(x_p)$, where each f_j a staircase-like function of *one* of p features.

Key idea: Depth limits interactions

Example (Depth $d = 1$)

A sum of stumps is additive. For any number of trees M, we can write $F_M(x) = c + f_1(x_1) + \cdots + f_p(x_p)$, where each f_j a staircase-like function of *one* of p features.

Key idea: A tree of depth d involves at most d features along any path, which limits interactions to at most d features.

Key idea: Depth limits interactions

Example (Depth $d = 1$)

A sum of stumps is additive. For any number of trees M, we can write $F_M(x) = c + f_1(x_1) + \cdots + f_p(x_p)$, where each f_j a staircase-like function of *one* of p features.

Key idea: A tree of depth d involves at most d features along any path, which limits interactions to at most d features.

Question (Depth versus degree)

Recall the advertising dataset. Can I fit TV^2 with polynomial features of degree 1? Can I fit TV^2 with stumps, i.e., depth 1?

Key idea: Depth limits interactions

Example (Depth $d = 1$)

A sum of stumps is additive. For any number of trees M, we can write $F_M(x) = c + f_1(x_1) + \cdots + f_p(x_p)$, where each f_j a staircase-like function of *one* of p features.

Key idea: A tree of depth d involves at most d features along any path, which limits interactions to at most d features.

Question (Depth versus degree)

Recall the advertising dataset. Can I fit TV^2 with polynomial features of degree 1? Can I fit TV^2 with stumps, i.e., depth 1?

Watch out: Depth is like degree of polynomial features, but not exactly!

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]
```

Python: The same model, with XGBoost

```
from xgboost import XGBClassifier
model = XGBClassifier(n_estimators=97, max_depth=1,
                      learning_rate=0.1)
model.fit(X_train, y_train)
p_hat = model.predict_proba(X_test)[:, 1]
```

- same interface as scikit-learn: `n_estimators` is M, `learning_rate` is ν
- $M = 97$ chosen on validation set, as before
- rest of the lecture is about how XGBoost differs from gradient boosting

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?

Plan for today

• Recap of gradient boosting for classification	2 min
• Credit data example	6 min
• Shrinkage, depth, and validation	8 min
• Leaf values: one Newton step	10 min
• XGBoost $=$ quadratic for splits too $+$ penalty	15 min
• Scoring a split	5 min

Recall: Where do these leaf values come from?

Definition (Gradient boosting with log loss)

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

1. compute $p_i = \sigma(F_{m-1}(x_i))$ and the targets $r_i = y_i - p_i$;
2. fit a least-squares regression tree to the (x_i, r_i); 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 $p_M(x) = \sigma(F_M(x))$.

Newton's method

To minimize a function $\phi(w)$ of one number w, starting from $w = 0$:

> **Key idea:** Taylor-expand ϕ to 2nd order around the current guess, then minimize that instead. Repeat.

Newton's method

To minimize a function $\phi(w)$ of one number w, starting from $w = 0$:

> **Key idea:** Taylor-expand ϕ to 2nd order around the current guess, then minimize that instead. Repeat.

Near $w = 0$, ϕ is approximately a parabola in w:

$$\phi(w) \approx \phi(0) + \phi'(0)\, w + \tfrac{1}{2}\, \phi''(0)\, w^2.$$

Newton's method

To minimize a function $\phi(w)$ of one number w, starting from $w = 0$:

> **Key idea:** Taylor-expand ϕ to 2nd order around the current guess, then minimize that instead. Repeat.

Near $w = 0$, ϕ is approximately a parabola in w:

$$\phi(w) \approx \phi(0) + \phi'(0)\, w + \tfrac{1}{2}\, \phi''(0)\, w^2.$$

If $\phi''(0) > 0$, the parabola is smallest where its derivative $\phi'(0) + \phi''(0)\, w$ is zero: $w = -\phi'(0)/\phi''(0)$.

Newton's method

To minimize a function $\phi(w)$ of one number w, starting from $w = 0$:

> **Key idea:** Taylor-expand ϕ to 2nd order around the current guess, then minimize that instead. Repeat.

Near $w = 0$, ϕ is approximately a parabola in w:

$$\phi(w) \approx \phi(0) + \phi'(0)\, w + \tfrac{1}{2}\, \phi''(0)\, w^2.$$

If $\phi''(0) > 0$, the parabola is smallest where its derivative $\phi'(0) + \phi''(0)\, w$ is zero: $w = -\phi'(0)/\phi''(0)$.

If ϕ is quadratic (e.g., squared loss), this is the exact minimizer.

A leaf's correction is one Newton step

Notation: Consider one leaf R in round m. Write $F_i = F_{m-1}(x_i)$ for the ith example's current score, and $\ell_i(F_i) = L(y_i, F_i)$ for its loss as a function of this score.

Every $x_i \in R$ gets the same correction w:

$$\phi(w) = \sum_{x_i \in R} \ell_i(F_i + w).$$

A leaf's correction is one Newton step

Notation: Consider one leaf R in round m. Write $F_i = F_{m-1}(x_i)$ for the ith example's current score, and $\ell_i(F_i) = L(y_i, F_i)$ for its loss as a function of this score.

Every $x_i \in R$ gets the same correction w:

$$\phi(w) = \sum_{x_i \in R} \ell_i(F_i + w).$$

Differentiate in w, then set $w = 0$:

$$\phi'(0) = \sum_{x_i \in R} \underbrace{\ell_i'(F_i)}_{g_i} = G, \qquad \phi''(0) = \sum_{x_i \in R} \underbrace{\ell_i''(F_i)}_{h_i} = H.$$

A leaf's correction is one Newton step

Notation: Consider one leaf R in round m. Write $F_i = F_{m-1}(x_i)$ for the ith example's current score, and $\ell_i(F_i) = L(y_i, F_i)$ for its loss as a function of this score.

Every $x_i \in R$ gets the same correction w:

$$\phi(w) = \sum_{x_i \in R} \ell_i(F_i + w).$$

Differentiate in w, then set $w = 0$:

$$\phi'(0) = \sum_{x_i \in R} \underbrace{\ell_i'(F_i)}_{g_i} = G, \qquad \phi''(0) = \sum_{x_i \in R} \underbrace{\ell_i''(F_i)}_{h_i} = H.$$

One Newton step from $w = 0$ gives $w = -G/H$.

Example: g_i and h_i for the log loss

Log loss of y_i, as a function of its score F, with $p = \sigma(F) = e^F/(e^F + 1)$:

$$\ell_i(F) = -\big[\, y_i \log p + (1 - y_i) \log(1 - p)\big] = \log(1 + e^F) - y_i F.$$

Example: g_i and h_i for the log loss

Log loss of y_i, as a function of its score F, with $p = \sigma(F) = e^F/(e^F + 1)$:

$$\ell_i(F) = -\big[\, y_i \log p + (1 - y_i) \log(1 - p)\big] = \log(1 + e^F) - y_i F.$$

Its derivatives in F:

$$\ell_i'(F) = \frac{e^F}{1 + e^F} - y_i = \sigma(F) - y_i, \qquad \ell_i''(F) = \frac{e^F}{(1 + e^F)^2} = \sigma(F)(1 - \sigma(F)).$$

Example: g_i and h_i for the log loss

Log loss of y_i, as a function of its score F, with $p = \sigma(F) = e^F/(e^F + 1)$:

$$\ell_i(F) = -\big[\, y_i \log p + (1 - y_i) \log(1 - p)\big] = \log(1 + e^F) - y_i F.$$

Its derivatives in F:

$$\ell_i'(F) = \frac{e^F}{1 + e^F} - y_i = \sigma(F) - y_i, \qquad \ell_i''(F) = \frac{e^F}{(1 + e^F)^2} = \sigma(F)(1 - \sigma(F)).$$

At the current score F_i, with $p_i = \sigma(F_i)$:

$$g_i = p_i - y_i, \qquad h_i = p_i(1 - p_i).$$

Example: g_i and h_i for the log loss

Log loss of y_i, as a function of its score F, with $p = \sigma(F) = e^F/(e^F + 1)$:

$$\ell_i(F) = -\big[\, y_i \log p + (1 - y_i) \log(1 - p)\big] = \log(1 + e^F) - y_i F.$$

Its derivatives in F:

$$\ell_i'(F) = \frac{e^F}{1 + e^F} - y_i = \sigma(F) - y_i, \qquad \ell_i''(F) = \frac{e^F}{(1 + e^F)^2} = \sigma(F)(1 - \sigma(F)).$$

At the current score F_i, with $p_i = \sigma(F_i)$:

$$g_i = p_i - y_i, \qquad h_i = p_i(1 - p_i).$$

So $-G/H = \sum(y_i - p_i)/\sum p_i(1 - p_i)$ is step 3 of gradient boosting.

Finally: Leaf values come from $-G/H$

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 a least-squares regression tree to the (x_i, r_i); 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 $p_M(x) = \sigma(F_M(x))$.

Plan for today

- Recap of gradient boosting for classification — 2 min
- Credit data example — 6 min
- Shrinkage, depth, and validation — 8 min
- Leaf values: one Newton step — 10 min
- XGBoost = quadratic for splits too + penalty — 15 min
- Scoring a split — 5 min

Gradient boosting: two approximations

Problem (Adding the mth tree)

Find the tree f that minimizes $\sum_i L(y_i, F_{m-1}(x_i) + f(x_i))$, then set $F_m = F_{m-1} + \nu f$.

- splits (step 2): least squares on the targets $-\partial L/\partial F$

Gradient boosting: two approximations

Problem (Adding the mth tree)

Find the tree f that minimizes $\sum_i L(y_i, F_{m-1}(x_i) + f(x_i))$, then set $F_m = F_{m-1} + \nu f$.

- splits (step 2): least squares on the targets $-\partial L/\partial F$
- leaf values (step 3): one Newton step, which also uses $-\partial^2 L/\partial F^2$

Gradient boosting: two approximations

Problem (Adding the mth tree)

Find the tree f that minimizes $\sum_i L(y_i, F_{m-1}(x_i) + f(x_i))$, then set $F_m = F_{m-1} + \nu f$.

- splits (step 2): least squares on the targets $-\partial L/\partial F$
- leaf values (step 3): one Newton step, which also uses $-\partial^2 L/\partial F^2$

Gradient boosting: two approximations

Problem (Adding the mth tree)

Find the tree f that minimizes $\sum_i L(y_i, F_{m-1}(x_i) + f(x_i))$, then set $F_m = F_{m-1} + \nu f$.

- splits (step 2): least squares on the targets $-\partial L/\partial F$
- leaf values (step 3): one Newton step, which also uses $-\partial^2 L/\partial F^2$

Question

Why not choose the splits with the same quadratic approximation as the leaf values?

XGBoost: a penalized objective

Watch out: Notation changes ahead, to match XGBoost's.

Definition (XGBoost (Chen and Guestrin, 2016))

Round m chooses the tree f (splits and leaf values) to minimize

$$\sum_i L(y_i, F_{m-1}(x_i) + f(x_i)) + \Omega(f), \qquad \Omega(f) = \gamma T + \tfrac{1}{2}\lambda \textstyle\sum_{j=1}^{T} w_j^2,$$

where f has T leaves R_j with values w_j. Then, set $F_m = F_{m-1} + \nu f$.

- Notation: leaf values w_j were γ_{jm} before. Now, $\gamma \geq 0$ is a leaf cost. There's also a ridge penalty $\lambda \geq 0$!

XGBoost: a penalized objective

Watch out: Notation changes ahead, to match XGBoost's.

Definition (XGBoost (Chen and Guestrin, 2016))

Round m chooses the tree f (splits and leaf values) to minimize

$$\sum_i L(y_i, F_{m-1}(x_i) + f(x_i)) + \Omega(f), \qquad \Omega(f) = \gamma T + \tfrac{1}{2}\lambda \textstyle\sum_{j=1}^{T} w_j^2,$$

where f has T leaves R_j with values w_j. Then, set $F_m = F_{m-1} + \nu f$.

- Notation: leaf values w_j were γ_{jm} before. Now, $\gamma \geq 0$ is a leaf cost. There's also a ridge penalty $\lambda \geq 0$!
- One objective chooses both the splits and the leaf values.

The second-order approximation

Expand each term around $F_{m-1}(x_i)$, with g_i and h_i as before:

$$L(y_i, F_{m-1}(x_i) + f(x_i)) \approx L(y_i, F_{m-1}(x_i)) + g_i\, f(x_i) + \tfrac{1}{2} h_i\, f(x_i)^2.$$

The second-order approximation

Expand each term around $F_{m-1}(x_i)$, with g_i and h_i as before:

$$L(y_i, F_{m-1}(x_i) + f(x_i)) \approx L(y_i, F_{m-1}(x_i)) + g_i\, f(x_i) + \tfrac{1}{2} h_i\, f(x_i)^2.$$

For x_i in leaf R_j, $f(x_i) = w_j$, so summing over the leaf's points gives

$$\sum_{x_i \in R_j} L(y_i, F_{m-1}(x_i)) + G_j\, w_j + \tfrac{1}{2} H_j\, {w_j}^2, \qquad G_j = \sum_{x_i \in R_j} g_i, \quad H_j = \sum_{x_i \in R_j} h_i.$$

The second-order approximation

Expand each term around $F_{m-1}(x_i)$, with g_i and h_i as before:

$$L(y_i, F_{m-1}(x_i) + f(x_i)) \approx L(y_i, F_{m-1}(x_i)) + g_i\, f(x_i) + \tfrac{1}{2} h_i\, f(x_i)^2.$$

For x_i in leaf R_j, $f(x_i) = w_j$, so summing over the leaf's points gives

$$\sum_{x_i \in R_j} L(y_i, F_{m-1}(x_i)) + G_j\, w_j + \tfrac{1}{2} H_j\, {w_j}^2, \qquad G_j = \sum_{x_i \in R_j} g_i, \quad H_j = \sum_{x_i \in R_j} h_i.$$

Now, sum over the leaves and add $\Omega(f)$ to obtain

$$\text{const} + \sum_{j=1}^{T} \left[G_j\, w_j + \tfrac{1}{2} (H_j + \lambda) {w_j}^2 \right] + \gamma T, \qquad \text{const} = \sum_i L(y_i, F_{m-1}(x_i)).$$

The best leaf values

Proposition

For fixed leaves $R_1, \dots, R_T$ with every $H_j + \lambda > 0$, the approximate objective is smallest at

$$w_j^* = -\frac{G_j}{H_j + \lambda} \qquad \text{with value} \qquad \text{const} - \frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T.$$

Proof. Take the ∂_{w_j} of the objective and set it to zero. □

- Note 1: $\lambda = 0$ gives the Newton step $-G/H$ from before

The best leaf values

Proposition

For fixed leaves $R_1, \dots, R_T$ with every $H_j + \lambda > 0$, the approximate objective is smallest at

$$w_j^* = -\frac{G_j}{H_j + \lambda} \qquad \text{with value} \qquad \text{const} - \frac{1}{2}\sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T.$$

Proof. Take the ∂_{w_j} of the objective and set it to zero. □

- Note 1: $\lambda = 0$ gives the Newton step $-G/H$ from before
- Note 2: λ matters most where H_j is small

The best leaf values

Proposition

For fixed leaves $R_1, \dots, R_T$ with every $H_j + \lambda > 0$, the approximate objective is smallest at

$$w_j^* = -\frac{G_j}{H_j + \lambda} \qquad \text{with value} \qquad \text{const} - \frac{1}{2}\sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T.$$

Proof. Take the ∂_{w_j} of the objective and set it to zero. □

- Note 1: $\lambda = 0$ gives the Newton step $-G/H$ from before
- Note 2: λ matters most where H_j is small
- Note 3: If L is squared loss instead, then $w_j^* = \sum_{x_i \in R_j} r_i/(n_j + \lambda)$, which is ridge regression of residuals on leaf indicators! (What?!)

Example: What λ does to the leaves

Example: Credit data using only balance. The four leaves of round 1åv√. Each account has $h_i = p(1-p) = 0.032$. Leaf values $w = -G/(H+\lambda)$:

leaf	H	$\lambda = 0$	$\lambda = 1$	$\lambda = 10$
$4{,}176$ accounts (15 defaults)	134.0	-0.92	-0.92	-0.86
604 accounts (54 defaults)	19.4	1.75	1.67	1.16
160 accounts (55 defaults)	5.1	9.68	8.10	3.28
60 accounts (42 defaults)	1.9	20.77	13.67	3.35

- $\lambda = 1$ barely moves the large leaf, but cuts the small one by a third

Example: What λ does to the leaves

Example: Credit data using only balance. The four leaves of round 1åv√. Each account has $h_i = p(1-p) = 0.032$. Leaf values $w = -G/(H+\lambda)$:

leaf	H	$\lambda = 0$	$\lambda = 1$	$\lambda = 10$
$4{,}176$ accounts (15 defaults)	134.0	-0.92	-0.92	-0.86
604 accounts (54 defaults)	19.4	1.75	1.67	1.16
160 accounts (55 defaults)	5.1	9.68	8.10	3.28
60 accounts (42 defaults)	1.9	20.77	13.67	3.35

- $\lambda = 1$ barely moves the large leaf, but cuts the small one by a third
- the exact correction for the small leaf is 4.22: the Newton step overshoots most where H is small

Example: Four points by hand

Recall: $F_0 = \log(\bar{y}/(1-\bar{y}))$; $p_i = \sigma(F_{m-1}(x_i))$; $g_i = p_i - y_i$, $h_i = p_i(1-p_i)$; G_j, H_j: their sums over leaf j; $w_j = -G_j/(H_j + \lambda)$.

Problem

$x = (-2, -1, 1, 2)$, $y = (0, 0, 1, 1)$, log loss, one split at $x = 0$. In round 1, find each g_i and h_i, and the leaf values for $\lambda = 1$ and $\lambda = 0$.

Example: Four points by hand

Recall: $F_0 = \log(\bar{y}/(1-\bar{y}))$; $p_i = \sigma(F_{m-1}(x_i))$; $g_i = p_i - y_i$, $h_i = p_i(1-p_i)$; G_j, H_j: their sums over leaf j; $w_j = -G_j/(H_j + \lambda)$.

Problem

$x = (-2, -1, 1, 2)$, $y = (0, 0, 1, 1)$, log loss, one split at $x = 0$. In round 1, find each g_i and h_i, and the leaf values for $\lambda = 1$ and $\lambda = 0$.

- $\bar{y} = \frac{1}{2}$, so $F_0 = 0$ and every $p_i = \frac{1}{2}$; then $g = (\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, -\frac{1}{2})$ and every $h_i = \frac{1}{4}$

Example: Four points by hand

Recall: $F_0 = \log(\bar{y}/(1-\bar{y}))$; $p_i = \sigma(F_{m-1}(x_i))$; $g_i = p_i - y_i$, $h_i = p_i(1-p_i)$; G_j, H_j: their sums over leaf j; $w_j = -G_j/(H_j + \lambda)$.

Problem

$x = (-2, -1, 1, 2)$, $y = (0, 0, 1, 1)$, log loss, one split at $x = 0$. In round 1, find each g_i and h_i, and the leaf values for $\lambda = 1$ and $\lambda = 0$.

- $\bar{y} = \frac{1}{2}$, so $F_0 = 0$ and every $p_i = \frac{1}{2}$; then $g = (\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, -\frac{1}{2})$ and every $h_i = \frac{1}{4}$
- left: $G = 1$, $H = \frac{1}{2}$; right: $G = -1$, $H = \frac{1}{2}$. So $\lambda = 1$: $w = \mp\frac{1}{3/2} = \mp\frac{2}{3}$; $\lambda = 0$: $w = \mp 2$

Example: Four points by hand

Recall: $F_0 = \log(\bar{y}/(1-\bar{y}))$; $p_i = \sigma(F_{m-1}(x_i))$; $g_i = p_i - y_i$, $h_i = p_i(1-p_i)$; G_j, H_j: their sums over leaf j; $w_j = -G_j/(H_j + \lambda)$.

Problem

$x = (-2, -1, 1, 2)$, $y = (0, 0, 1, 1)$, log loss, one split at $x = 0$. In round 1, find each g_i and h_i, and the leaf values for $\lambda = 1$ and $\lambda = 0$.

- $\bar{y} = \frac{1}{2}$, so $F_0 = 0$ and every $p_i = \frac{1}{2}$; then $g = (\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, -\frac{1}{2})$ and every $h_i = \frac{1}{4}$
- left: $G = 1$, $H = \frac{1}{2}$; right: $G = -1$, $H = \frac{1}{2}$. So $\lambda = 1$: $w = \mp\frac{1}{3/2} = \mp\frac{2}{3}$; $\lambda = 0$: $w = \mp 2$
- each leaf holds one class, so the exact correction is infinite, but Newton step is finite

Small H: `min_child_weight`

Example: XGBoost with default settings would not split the four points on last slide because each child has $H = \frac{1}{2}$.

Definition (`min_child_weight` (default 1))

A split is allowed only if each child has $H \geq$ `min_child_weight`.

- Example: for squared loss, every $h_i = 1$, so H is the number of points $\equiv$ a minimum leaf size

Small H: `min_child_weight`

Example: XGBoost with default settings would not split the four points on last slide because each child has $H = \frac{1}{2}$.

Definition (`min_child_weight` (default 1))

A split is allowed only if each child has $H \geq$ `min_child_weight`.

- Example: for squared loss, every $h_i = 1$, so H is the number of points $\equiv$ a minimum leaf size
- Example: for log loss, $h_i = p_i(1 - p_i)$ is small when p_i is near 0 or 1, so H measure how uncertain the model is about the leaf's examples

Plan for today

• Recap of gradient boosting for classification	2 min
• Credit data example	6 min
• Shrinkage, depth, and validation	8 min
• Leaf values: one Newton step	10 min
• XGBoost $=$ quadratic for splits too $+$ penalty	15 min
• Scoring a split	5 min

Recall: The best leaf values

Proposition

For fixed leaves $R_1, \dots, R_T$ with every $H_j + \lambda > 0$, the approximate objective is smallest at

$$w_j^* = -\frac{G_j}{H_j + \lambda} \qquad \text{with value} \qquad \text{const} - \frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T.$$

XGBoost: The gain of a split

Split a leaf with sums (G, H) into children with (G_L, H_L) and (G_R, H_R). By the proposition, the approximate objective falls by

$$\text{Gain} = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{G^2}{H + \lambda}\right] - \gamma.$$

- the tree is grown greedily (L19), each time with the split of largest gain

XGBoost: The gain of a split

Split a leaf with sums (G, H) into children with (G_L, H_L) and (G_R, H_R). By the proposition, the approximate objective falls by

$$\text{Gain} = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{G^2}{H + \lambda}\right] - \gamma.$$

- the tree is grown greedily (L19), each time with the split of largest gain
- a split is made only if $\text{Gain} > 0$, so γ is the smallest gain worth a leaf

XGBoost: The gain of a split

Split a leaf with sums (G, H) into children with (G_L, H_L) and (G_R, H_R). By the proposition, the approximate objective falls by

$$\text{Gain} = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{G^2}{H + \lambda}\right] - \gamma.$$

- the tree is grown greedily (L19), each time with the split of largest gain
- a split is made only if $\text{Gain} > 0$, so γ is the smallest gain worth a leaf
- the gain uses only sums of g_i and h_i over the points in each child

Example: Six points, scored two ways

L19's six points: $x = 1, \dots, 6$ and $y = 1, 2, 3, 7, 8, 9$. Squared loss, $F_0 = 5$, $\lambda = \gamma = 0$. Before splitting, $\mathrm{RSS} = 58$.

split	RSS after the split (L19)	gain
$x \leq 1.5$	38.8	
$x \leq 2.5$	21.25	
$x \leq 3.5$	4	
$x \leq 4.5$	21.25	
$x \leq 5.5$	38.8	

Example: Six points, scored two ways

L19's six points: $x = 1, \dots, 6$ and $y = 1, 2, 3, 7, 8, 9$. Squared loss, $F_0 = 5$, $\lambda = \gamma = 0$. Before splitting, $\mathrm{RSS} = 58$.

split	RSS after the split (L19)	gain
$x \leq 1.5$	38.8	9.6
$x \leq 2.5$	21.25	18.375
$x \leq 3.5$	4	27
$x \leq 4.5$	21.25	18.375
$x \leq 5.5$	38.8	9.6

Example: Six points, scored two ways

L19's six points: $x = 1, \dots, 6$ and $y = 1, 2, 3, 7, 8, 9$. Squared loss, $F_0 = 5$, $\lambda = \gamma = 0$. Before splitting, $\mathrm{RSS} = 58$.

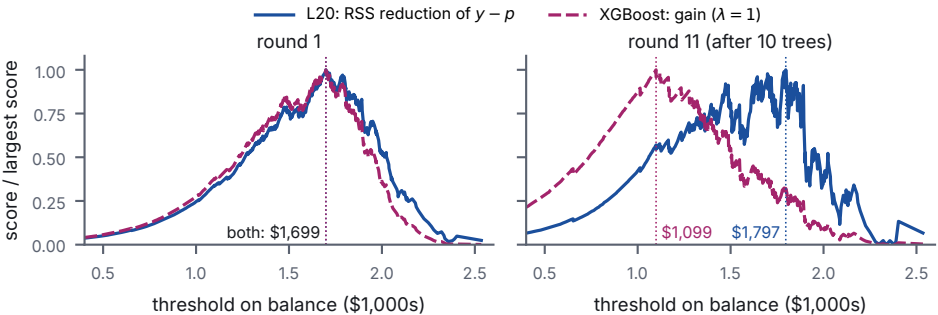
split	RSS after the split (L19)	gain
$x \leq 1.5$	38.8	9.6
$x \leq 2.5$	21.25	18.375
$x \leq 3.5$	4	27
$x \leq 4.5$	21.25	18.375
$x \leq 5.5$	38.8	9.6

Note: For squared loss with $\lambda = \gamma = 0$,

$$\mathrm{Gain} = \tfrac{1}{2}(\mathrm{RSS}_{\text{parent}} - \mathrm{RSS}_L - \mathrm{RSS}_R); \qquad \text{at } x = 3.5\text{: } \tfrac{1}{2}(58 - 4) = 27.$$

Example: One split, scored two ways

Example: Credit data using balance only. Gradient boosting and XGBoost, both with log loss. Plots show every possible first split, chosen either by least squares on the targets (gradient boosting) or by gain (XGBoost).



Summary

- For classification, each tree corrects the estimated log-odds. Small steps and validation determine the number of trees.
- XGBoost expands the loss to second order, setting leaf values to $-G/(H+\lambda)$. The splits are scored by the gain.
- Depth limits the interactions in the score. Choose ν, the number of trees, and the depth or leaves on a validation set.