

Gradient Boosting

Decision trees, one correction at a time

AdaBoost recap

Iterative error correction with weak learners.

- Each new tree sees the **same events**, reweighted.
- Misclassified events get **larger weights**.
- The final model is a **weighted vote** of all trees.
- Sensitive to outliers and label noise: they keep growing in weight.

Famous use: the Viola–Jones face detector (2001).

Today: gradient boosting

Instead of reweighting events, each new tree is trained on **what is still missing**:

new target = residual = $y - \text{current prediction}$.

The same idea works for **any differentiable loss**, which covers regression and classification.

A gradual approach: crocodile and turtle

A turtle visits a crocodile. Every day she covers only $\eta = 5\%$ of the **remaining** distance.

After d days the remaining fraction is $(1 - \eta)^d$, so the covered fraction is

$$1 - (1 - \eta)^d \approx 1 - e^{-\eta d}.$$

- She never arrives exactly, but gets arbitrarily close.
- 95% covered when $(0.95)^d \leq 0.05$, i.e. after $d = 59$ days.
- Only for $\eta d \ll 1$ is the covered fraction $\approx \eta d$.

Boosting does the same: each tree covers a fraction η of the remaining residual.



Notation

Symbol	Meaning
$i = 1, \dots, N$	training events, with features $x_i = (x_{i1}, \dots, x_{in}) \in \mathbb{R}^n$ and label y_i
$F(x)$	model output (“score”) for an event with features x
$L(y, F)$	loss of one event; training minimizes $\sum_{i=1}^N L(y_i, F(x_i))$
$m = 1, \dots, M$	boosting round; F_m is the model after m trees, F_0 a constant
$\gamma_m(x)$	output of tree m ; its leaves are regions R_{mj} , $j = 1, \dots, J_m$, with values γ_{mj}
$\eta \in (0, 1]$	learning rate (“shrinkage”): the fraction of each tree that is added

Start with regression

Features $x \in \mathbb{R}^n$, label $y \in \mathbb{R}$.

Squared-error loss per event:

$$L(y, F) = \frac{1}{2}(y - F)^2.$$

Its average over events is (half) the **MSE**. The factor $\frac{1}{2}$ only simplifies derivatives.

The **residual** of event i is what the model still misses:

$$r_i = y_i - F(x_i).$$

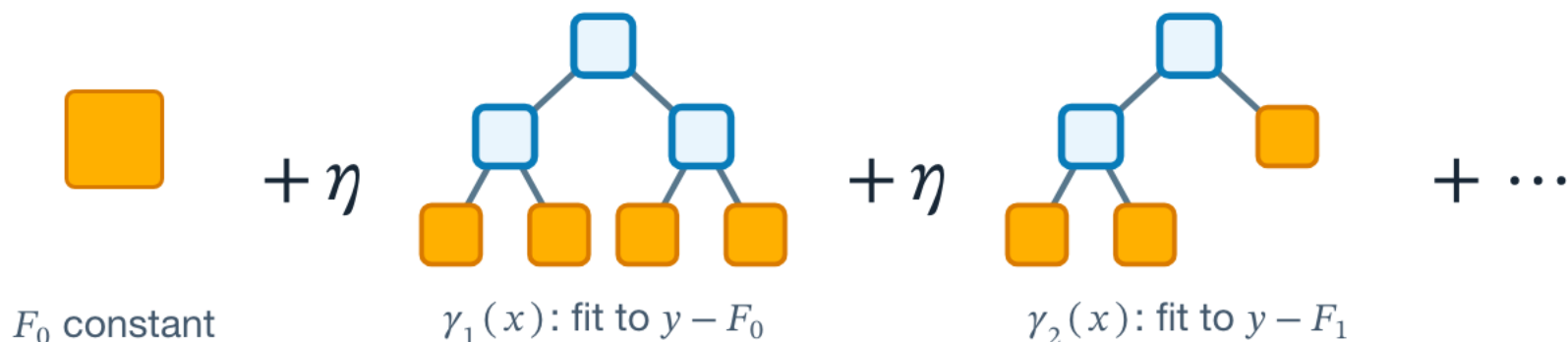
It is also minus the slope of the loss:

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

This second form generalizes to other losses.

Reaching the target in steps

Each tree is a **function** of x . The model is their sum:



$$F_M(x) = F_0 + \eta \sum_{m=1}^M \gamma_m(x), \quad F_m(x) = F_{m-1}(x) + \eta \gamma_m(x).$$

Tree m is trained on the residuals **left over** by the previous $m - 1$ trees.

Algorithm for squared error

J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine", Ann. Stat. 29 (2001) 1189.

1. **Start** with a constant: $F_0 = \bar{y} = \frac{1}{N} \sum_i y_i$.
2. **For** $m = 1, \dots, M$:
 - a. residuals $r_i = y_i - F_{m-1}(x_i)$ for every event;
 - b. grow a regression tree γ_m that predicts r_i from x_i ;
 - c. update $F_m(x) = F_{m-1}(x) + \eta \gamma_m(x)$.
3. **Predict** $\hat{y}(x) = F_M(x)$.

Step 2b hides the two technical questions: **which regions** R_{mj} (splitting) and **which values** γ_{mj} (leaves).

Why a constant $\bar{y}$? It is the best single number:

$$\frac{\partial}{\partial c} \sum_i \frac{1}{2} (y_i - c)^2 = 0 \Rightarrow c = \bar{y}.$$

Why $\eta < 1$? Small steps average over many trees and fit the noise more slowly.

A tree is a piecewise-constant function

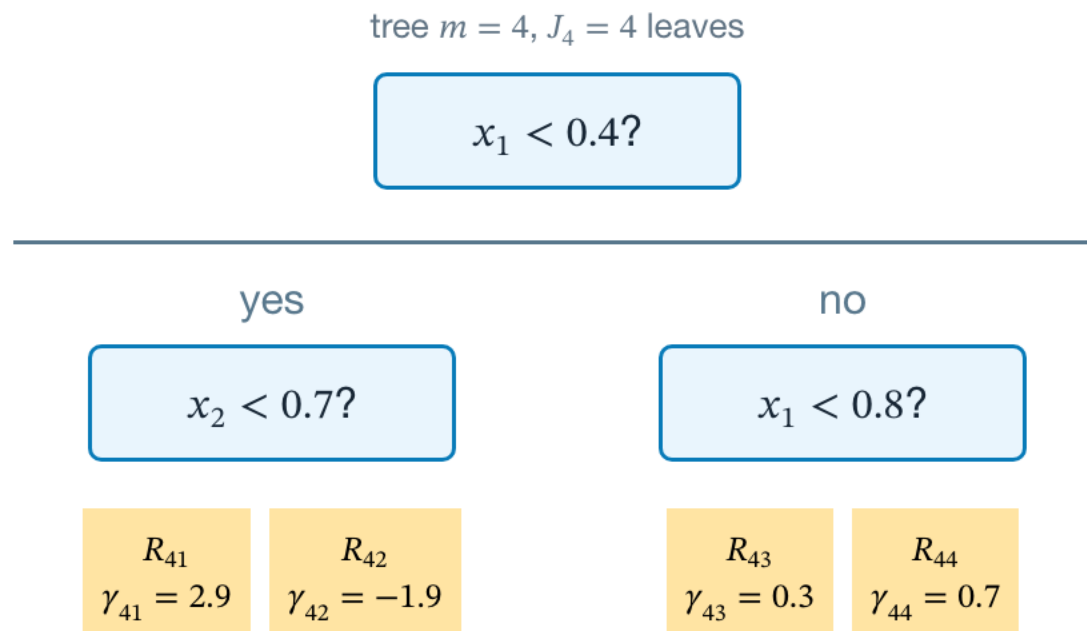
Tree m cuts feature space into J_m disjoint leaves $R_{m1}, \dots, R_{mJ_m}$. In each leaf it returns a constant γ_{mj} :

$$\gamma_m(x) = \sum_{j=1}^{J_m} \gamma_{mj} \mathbf{1}[x \in R_{mj}],$$

where $\mathbf{1}[\cdot] = 1$ if the condition holds and 0 otherwise.

So the update reads

$$F_m(x) = F_{m-1}(x) + \eta \sum_{j=1}^{J_m} \gamma_{mj} \mathbf{1}[x \in R_{mj}].$$



How leaf values are computed

Take the regions R_{mj} as given. Choose each leaf value to **minimize the loss of the events in that leaf**:

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

Leaves do not interact: every leaf is a separate one-number problem.

For $L = \frac{1}{2}(y - F)^2$, with N_j events in the leaf:

$$\frac{\partial}{\partial \gamma} \sum_{i \in R_{mj}} \frac{1}{2}(r_i - \gamma)^2 = 0 \Rightarrow \gamma_{mj} = \frac{1}{N_j} \sum_{i \in R_{mj}} r_i.$$

The leaf value is the mean residual in the leaf.

Example: residuals 1.2, 0.8, 1.0 give $\gamma = 1.0$;
with $\eta = 0.3$ these events move up by 0.3.

How a split is chosen

A node holds events I . A candidate split uses **feature** k and **cut** c : $I_L = \{i \in I: x_{ik} < c\}$, $I_R = I \setminus I_L$.

With the optimal leaf value (the mean), a node's loss is $\sum_{i \in I} \frac{1}{2} (r_i - \bar{r}_I)^2 = \frac{1}{2} \sum_{i \in I} r_i^2 - \frac{1}{2} \frac{(\sum_{i \in I} r_i)^2}{N_I}$. The $\sum r_i^2$ terms cancel in the loss reduction:

$$\text{Gain}(k, c) = \frac{1}{2} \left[\frac{(\sum_{i \in I_L} r_i)^2}{N_L} + \frac{(\sum_{i \in I_R} r_i)^2}{N_R} - \frac{(\sum_{i \in I} r_i)^2}{N_I} \right] \geq 0.$$

1. For each feature k , **sort** the events by x_{ik} ; candidate cuts are midpoints between neighbouring distinct values.
2. Scan left to right, updating $\sum_{i \in I_L} r_i$ and N_L with running sums: all cuts in $\mathcal{O}(N)$ after sorting.
3. Take the (k, c) with the largest gain; recurse into both children until the depth limit, a minimum leaf size, or no positive gain.

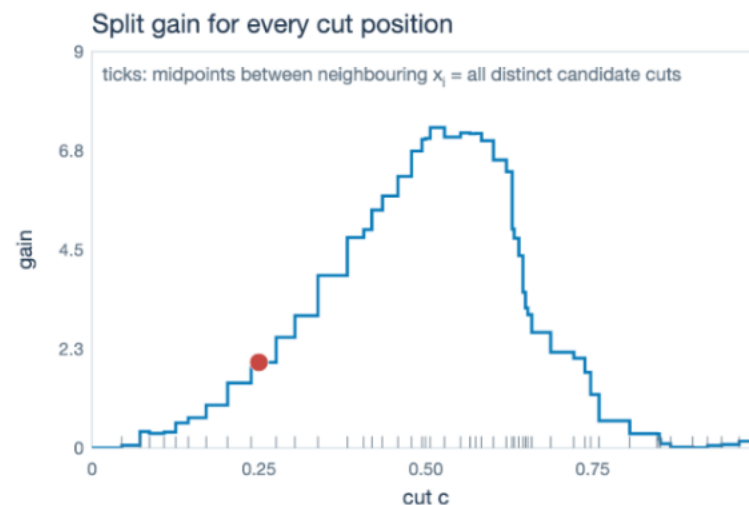
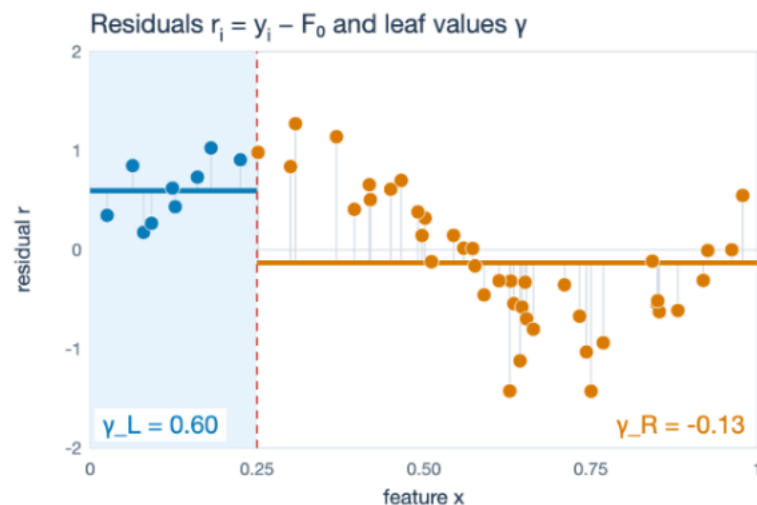
Where does the first tree cut?

Where does the first tree cut?

• left child $x < c$ • right child $x \geq c$

Round 1 of MSE boosting: start from the constant $F_0 = \bar{y}$ and fit a depth-1 tree (a “stump”) to the residuals $r_i = y_i - F_0$.

Cut c 0.250 Leaf penalty λ 0 Find best cut Reset



Left: $N_L, \Sigma r$	$\gamma_L = \Sigma r / (N_L + \lambda)$	Right: $N_R, \Sigma r$	$\gamma_R = \Sigma r / (N_R + \lambda)$	Gain
9, 5.36	0.595	41, -5.36	-0.131	1.943

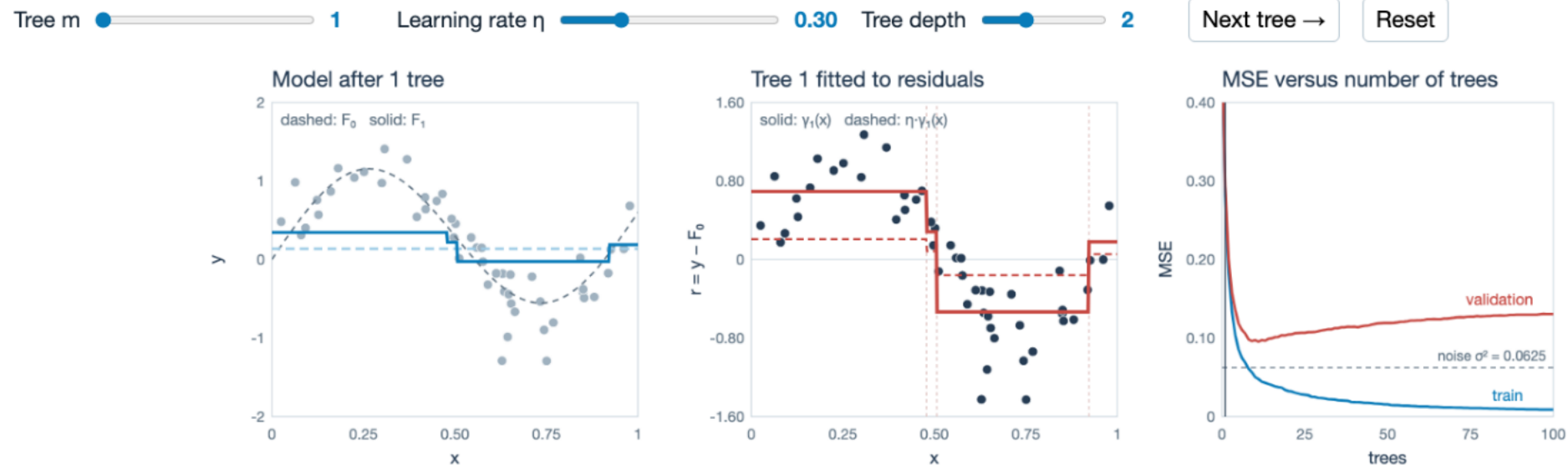
Predict the best cut before pressing the button. Then raise λ : leaf values shrink toward 0, small leaves most.

Boosting a regression, one tree at a time

Boosting a regression, one tree at a time

— F_m — tree γ_m - - truth

MSE loss. Tree m is fitted to the residuals of F_{m-1} ; the model moves by $\eta \gamma_m(x)$. 50 training events, 400 validation events.



Leaf values of tree 1	Train MSE	Validation MSE	Best validation
0.69, 0.28, -0.53, 0.18	0.2817	0.2933	0.0955 at $m = 11$

Step through the first trees at $\eta = 1$ and $\eta = 0.3$. Then drag m to 100 with depth 4: the training MSE keeps falling, the validation MSE does not.

Other losses: pseudo-residuals

For MSE the residual is minus the gradient.

For **any** differentiable loss define the **pseudo-residual**

$$r_{im} = - \left[\frac{\partial L(y_i, F)}{\partial F} \right]_{F = F_{m-1}(x_i)} .$$

Treat the N values $F(x_1), \dots, F(x_N)$ as parameters. $-\partial L / \partial F$ is the direction of steepest descent. The tree **generalizes** that direction to all x .

Gradient boosting = gradient descent in function space.

Loss $L(y, F)$	Pseudo-residual r
$\frac{1}{2}(y - F)^2$	$y - F$
$ y - F $	$\text{sign}(y - F)$
Huber, width δ	$y - F$, clipped to $[-\delta, \delta]$
BCE, $y \in \{0, 1\}$	$y - \sigma(F)$

$\sigma(F) = 1 / (1 + e^{-F})$ is defined on the classification slides.

Leaf values for any loss

T. Chen, C. Guestrin, "XGBoost: A Scalable Tree Boosting System", KDD 2016, arXiv:1603.02754.

Expand each event's loss around the current prediction, to second order in the small update γ :

$$L(y_i, F_{m-1}(x_i) + \gamma) \approx L(y_i, F_{m-1}(x_i)) + g_i \gamma + \frac{1}{2} h_i \gamma^2, \quad g_i = \frac{\partial L}{\partial F} \Big|_{F_{m-1}(x_i)}, \quad h_i = \frac{\partial^2 L}{\partial F^2} \Big|_{F_{m-1}(x_i)}.$$

In leaf j sum the derivatives: $G_j = \sum_{i \in R_{mj}} g_i$, $H_j = \sum_{i \in R_{mj}} h_i$. Add a penalty $\frac{1}{2} \lambda \gamma^2$ ($\lambda \geq 0$) against large leaf values:

$$\text{leaf objective} = G_j \gamma + \frac{1}{2} (H_j + \lambda) \gamma^2 \quad \xrightarrow{\partial/\partial \gamma = 0} \quad \gamma_{mj}^* = -\frac{G_j}{H_j + \lambda}, \quad \text{loss change} = -\frac{1}{2} \frac{G_j^2}{H_j + \lambda}.$$

This is **one Newton step** per leaf. Check with MSE: $g_i = F - y_i = -r_i$, $h_i = 1$, so $\gamma_{mj}^* = \sum r_i / (N_j + \lambda)$, the mean residual for $\lambda = 0$.

Split gain for any loss

Insert the best leaf values into the objective. Splitting node I into I_L and I_R reduces the loss by

$$\text{Gain} = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{G_I^2}{H_I + \lambda} \right] - \tau,$$

with $G_L = \sum_{i \in I_L} g_i$, $H_L = \sum_{i \in I_L} h_i$, and likewise for I_R and the parent I .

Term	Meaning
$\frac{G^2}{H + \lambda}$	how much a node could lower the loss with its best single value
$\lambda \geq 0$	leaf-value penalty: shrinks γ^* , disfavours leaves with little curvature H
$\tau \geq 0$	cost of one extra leaf: split only if the improvement exceeds τ (gamma in XGBoost)

Same search as before: sort, scan cuts with running sums G_L, H_L , keep the best. For MSE with $\lambda = \tau = 0$ this is the gain of the regression slide.

The general algorithm

1. $F_0 = \arg \min_c \sum_i L(y_i, c)$.
2. **For** $m = 1, \dots, M$:
 - a. compute g_i and h_i at $F_{m-1}(x_i)$ for every event;
 - b. **grow** a tree: at each node choose the (k, c) with the largest Gain; stop at the depth limit or if no gain is positive;
 - c. **leaf values** $\gamma_{mj} = -G_j / (H_j + \lambda)$;
 - d. **update** $F_m(x) = F_{m-1}(x) + \eta \sum_j \gamma_{mj} \mathbf{1}[x \in R_{mj}]$.
3. Output $F_M(x)$, or $p(x) = \sigma(F_M(x))$ for classification.

Only g_i and h_i depend on the loss.

Changing the task means changing **two derivatives**, not the tree code.

The trees in round m never see the labels directly, only g_i and h_i .

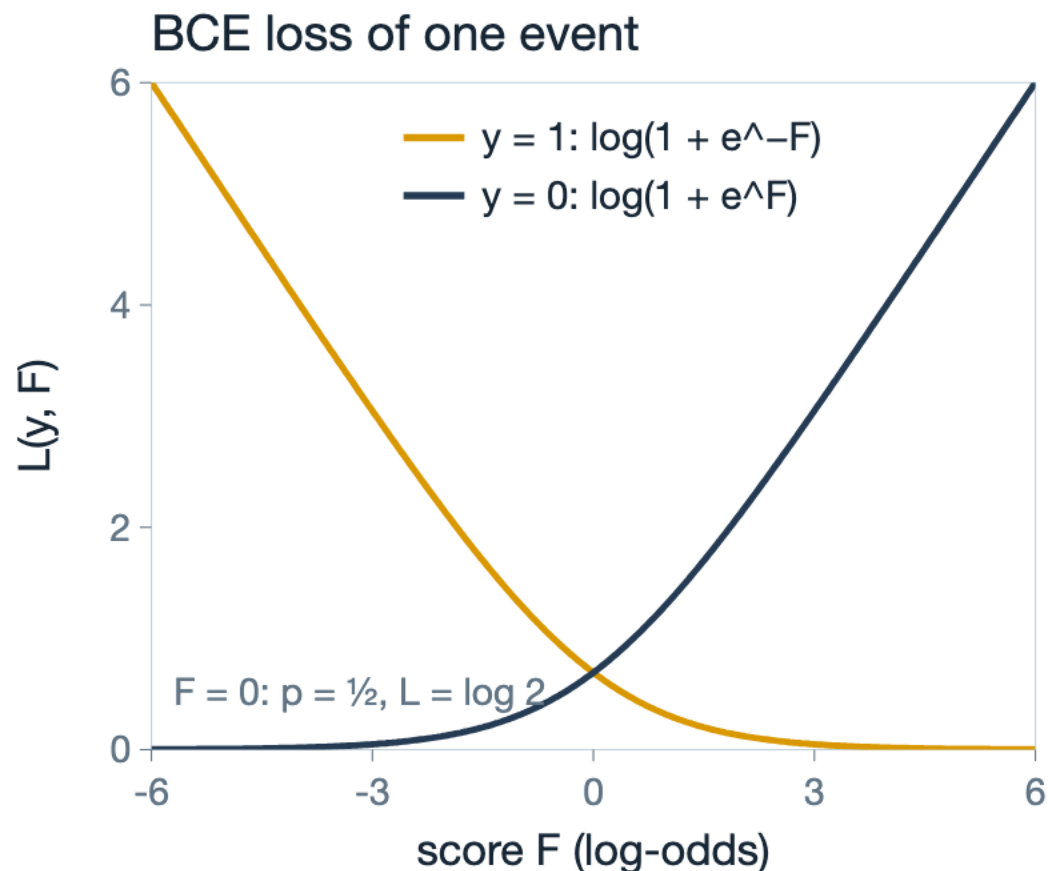
Classification: the score is a log-odds

Labels $y \in \{0, 1\}$. Trees add up to an unbounded score $F(x) \in \mathbb{R}$; map it to a probability with the **logistic function**

$$p(x) = \sigma(F(x)) = \frac{1}{1 + e^{-F(x)}}, \quad F = \log \frac{p}{1 - p}.$$

Loss: **binary cross-entropy** (negative log-likelihood)

$$\begin{aligned} L(y, F) &= -[y \log p + (1 - y) \log (1 - p)] \\ &= \log (1 + e^F) - yF. \end{aligned}$$



BCE: gradients and leaf values

With $p_i = \sigma(F_{m-1}(x_i))$, using $\frac{d\sigma}{dF} = \sigma(1 - \sigma)$:

$$g_i = \frac{\partial L}{\partial F} = p_i - y_i, \quad h_i = \frac{\partial^2 L}{\partial F^2} = p_i(1 - p_i).$$

Pseudo-residual: $r_i = y_i - p_i$, the **probability still missing**.

Newton leaf value:

$$\gamma_{mj} = \frac{\sum_{i \in R_{mj}} (y_i - p_i)}{\sum_{i \in R_{mj}} p_i(1 - p_i) + \lambda}.$$

Start: $F_0 = \log \frac{\bar{y}}{1 - \bar{y}}$, the log-odds of the signal fraction $\bar{y}$.

Worked example. Balanced sample, so $F_0 = 0$ and all $p_i = \frac{1}{2}$. One leaf holds $y = 1, 1, 1, 0$:

- $g = -\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}, +\frac{1}{2} \Rightarrow G = -1$
- $h = \frac{1}{4}$ each $\Rightarrow H = 1$
- $\lambda = 0$: $\gamma = 1$ (leaf log-odds $\log 3 = 1.10$)
- $\eta = 0.3$: $F = 0.3$, $p = \sigma(0.3) = 0.57$

One algorithm, two losses

	Regression: MSE	Classification: BCE
Label and output	$y \in \mathbb{R}$, prediction F	$y \in \{0, 1\}$, log-odds F , $p = \sigma(F)$
Loss $L(y, F)$	$\frac{1}{2}(y - F)^2$	$\log(1 + e^F) - yF$
Start F_0	$\bar{y}$	$\log(\bar{y}/(1 - \bar{y}))$
Gradient g_i	$F_{m-1}(x_i) - y_i$	$p_i - y_i$
Curvature h_i	1	$p_i(1 - p_i)$
Leaf value γ_{mj}	$\frac{\sum (y_i - F_{m-1}(x_i))}{N_j + \lambda}$	$\frac{\sum (y_i - p_i)}{\sum p_i(1 - p_i) + \lambda}$
Split gain	$\frac{1}{2}[G_L^2/(H_L + \lambda) + G_R^2/(H_R + \lambda) - G_I^2/(H_I + \lambda)] - \tau$	same formula
Prediction	$F_M(x)$	$p(x) = \sigma(F_M(x))$, cut on p

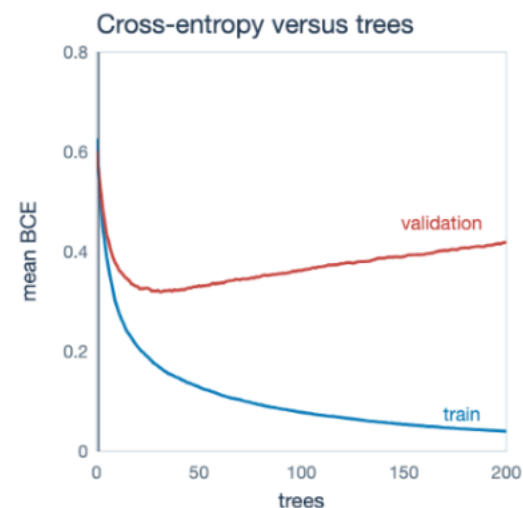
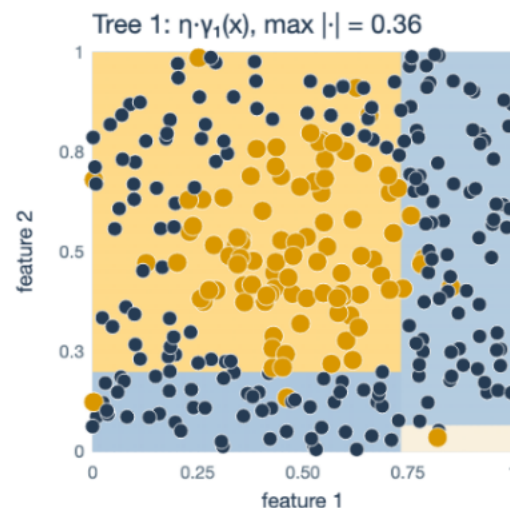
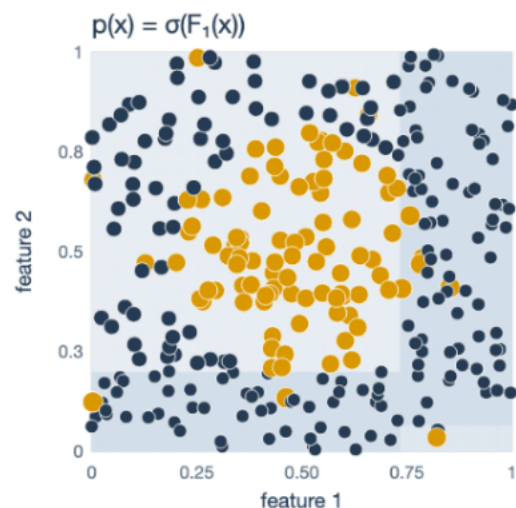
Boosting a classifier

Boosting a classifier with binary cross-entropy

● signal $y = 1$ ● background $y = 0$

Trees add to the log-odds $F(x)$; $p = \sigma(F)$. Point area $\propto |y - p|$, the size of the pseudo-residual. Leaf penalty $\lambda = 1$; 300 training, 1000 validation events; up to 200 trees.

Data Tree m η Depth Leaf



Train BCE	Validation BCE	Validation accuracy	Best validation BCE	$F_0 = \log(\bar{y}/(1-\bar{y}))$
0.556	0.563	71.4%	0.319 at m = 31	-0.754

Try “xor” with depth 1: a single cut cannot see the pattern, so stumps barely help. Depth 2 solves it.

Regularization

A boosted ensemble can always lower its **training** loss. These controls keep the **validation** loss low:

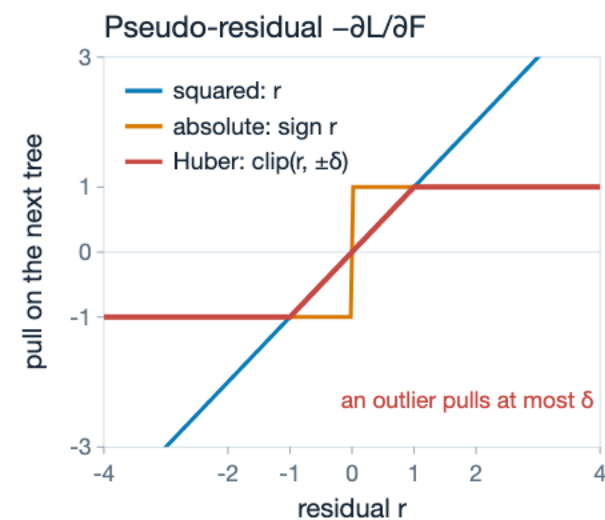
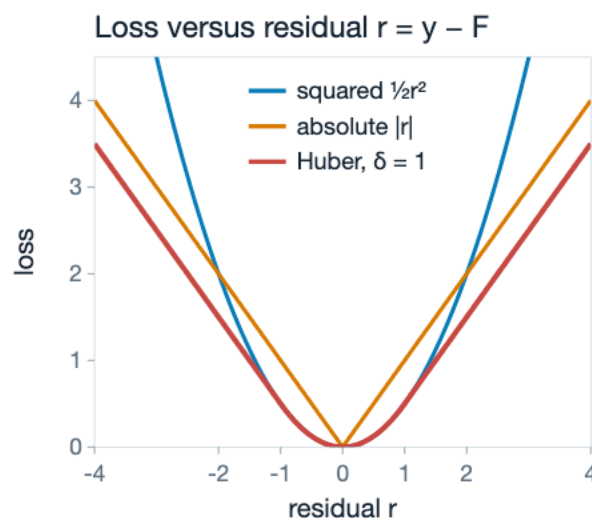
Control	Effect
Shrinkage η	small steps; smaller η needs more trees M but usually generalizes better
Tree depth (1–5 typical)	depth D allows interactions among up to D features; shallow trees are weak learners
Subsampling	each tree sees a random fraction of events (stochastic gradient boosting)
Leaf penalties λ , τ , minimum H per leaf	limit leaf values and forbid tiny or uninformative leaves
Early stopping	stop at the M with the lowest validation loss

Huber loss: robust regression

With $r = y - F$ and a width $\delta > 0$:

$$L_{\delta}(r) = \begin{cases} \frac{1}{2}r^2, & |r| \leq \delta, \\ \delta(|r| - \frac{1}{2}\delta), & |r| > \delta. \end{cases}$$

- Quadratic near 0: as efficient as least squares for Gaussian errors.
- Linear in the tails: robust like absolute loss.
- Pseudo-residual: r clipped to $[-\delta, \delta]$.
An outlier's pull is capped.



Other losses for classification

Binomial deviance, $y \in \{-1, +1\}$

$$L(y, F) = \log(1 + e^{-2yF}), \quad r = -\frac{\partial L}{\partial F} = \frac{2y}{1 + e^{2yF}}.$$

Friedman's convention. It is the BCE of the previous slides with labels $\frac{1+y}{2} \in \{0, 1\}$ and score $2F$, so

$$F = \frac{1}{2} \log \frac{p}{1-p}.$$

Exponential loss, $y \in \{-1, +1\}$

$$L(y, F) = e^{-yF}, \quad g_i = -y_i e^{-y_i F(x_i)}.$$

The factor $e^{-y_i F(x_i)}$ is exactly the **AdaBoost event weight**: boosting with this loss reproduces AdaBoost.

It grows exponentially for badly misclassified events, hence the noise sensitivity.

Libraries and their parameter names

Symbol	XGBoost	ROOT TMVA (BDT)
M	<code>n_estimators</code>	<code>NTrees</code>
η	<code>learning_rate</code> (eta)	<code>Shrinkage</code>
depth	<code>max_depth</code>	<code>MaxDepth</code>
subsample	<code>subsample</code>	<code>BaggedSampleFraction</code>
λ	<code>reg_lambda</code>	—
τ	<code>gamma</code>	—
loss	<code>objective</code>	<code>BoostType=Grad</code>

Also: LightGBM, CatBoost; in Julia XGBoost.jl, EvoTrees.jl, JLBoost.jl.

```
1 import xgboost as xgb
2 clf = xgb.XGBClassifier(
3     objective="binary:logistic", # BCE
4     n_estimators=500, learning_rate=0.1,
5     max_depth=3, reg_lambda=1.0,
6     subsample=0.8, early_stopping_rounds=20)
7 clf.fit(X_train, y_train,
8         eval_set=[(X_val, y_val)])
9 p = clf.predict_proba(X_test)[: , 1]
```

Regression: `XGBRegressor` with `objective="reg:squarederror"`.

Impact of boosted trees

Kaggle statistic: Chen & Guestrin, arXiv:1603.02754.

- **State of the art on tabular data.** Of 29 winning Kaggle solutions published in 2015, 17 used XGBoost; 8 used it alone.
- **Industry:** risk modelling in finance, click-through-rate prediction, insurance, customer modelling. Accurate, fast, and comparatively interpretable.
- **High-energy physics:** for years the most common architecture for signal/background separation at the LHC, e.g. in the Higgs discoveries and flavour tagging.

BDT playground

Online: Alex Rogozhnikov's [gradient boosting playground](#) and [explanation](#).

Julia: a Pluto adaptation with JLBoost.jl: [static preview](#) · [source](#)

Live version, from this lecture's folder:

```
1 julia notebooks/launch.jl
```

Tasks

1. Reach a test loss below 0.05 on every dataset.
2. Which datasets need depth ≥ 2 ? Why?
3. Fix ηM and compare $\eta = 1$ with $\eta = 0.1$.
4. Compare Newton and gradient updates at equal M .

End

Thank you for the attention