XGBoost

$V_1(x)$ = WP if you have 1st and 10 at game-state x

dataset

i index of the i^{th} play

x_i game-state of the i^{th} play

y_i 1/0 win/loss corresponding to i^{th} play

learn relationship b/t y and x using Machine Learning.

- decision trees → overfit
- random forests → reduces overfitting by reducing variance

We can do better.

- boosting → also reduces bias

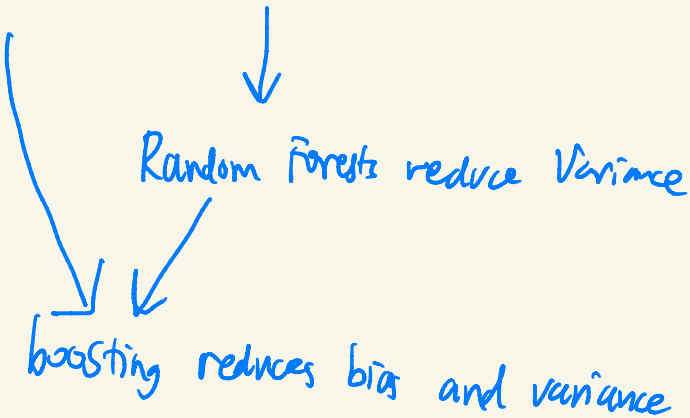
Bias-Variance Tradeoff:

out-of sample prediction error = bias + variance + irreducible error

bias: the model actually being wrong

variance: noise coming from randomness in the training set

irreducible error: fundamental difficulty of the problem



Boosting – train multiple models (e.g. decision trees) sequentially, train each successive model to optimize some loss function to improve the accuracy of the overall system.

Tree Boosting in a nutshell

dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$

n training examples $n = |\mathcal{D}|$

m features (columns) $x_i \in \mathbb{R}^m$

outcome $y_i \in \mathbb{R}$

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i)$$

f_k decision tree

To learn the set of functions (trees) $\{f_k\}$ used in this model, we minimize a Regularized objective function,

$$\mathcal{L}(\phi) = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k)$$

where $\Omega(f) = \gamma \cdot T + \frac{1}{2}\lambda \cdot \sum_j w_{jk}^2$

$y_i \in \{1, 0\}$ binary: $l(y_i, \hat{y}_i) =$ log loss $= y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)$

$y_i \in \mathbb{R}$ continuous: $l(y_i, \hat{y}_i) =$ squared error $= (y_i - \hat{y}_i)^2$

Regularization: $\Omega(f_k) =$ regularization term on the decision tree f_k

$T =$ # of leaves

$w_{jk} =$ weight value of tree k leaf j

→ making # leaves smaller and weights smaller

$$\Omega(f_k) = \gamma \cdot T_k + \frac{1}{2} \lambda \cdot \sum_j w_{jk}^2$$

$$\emptyset = \sum_k f_k = \text{argmin}_{\emptyset} \; \mathcal{L}(\emptyset)$$

$$\mathcal{L}(\emptyset) = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k)$$

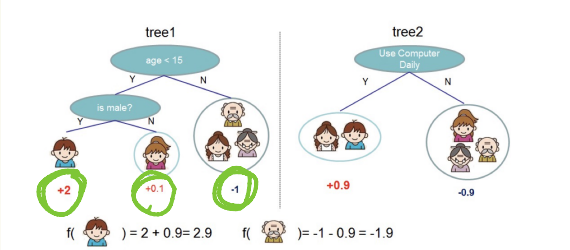


Figure 1: Tree Ensemble Model. The final prediction for a given example is the sum of predictions from each tree.

Ridge Regression $\hat{\beta} = \text{argmin}_{\beta} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + \lambda \sum_j \beta_j^2$

squared error

Regularized term

We wrote down a loss function $L(\emptyset) = L(\sum_k f_k)$ that we want to minimize in order to train our decision trees $\{f_k\}$. But how can we actually

minimize this?

$$L(\phi) = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k)$$

$\longrightarrow$ minimize in a sequential manner.

At iteration t, $\hat{y}_i^{(t)}$ prediction of i^{th} datapoint.
To fit the decision tree f_t at the t^{th} iteration, lets use a modified version of L,

$$L^{(t)}(f_t) = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t).$$

Assume we've already fit $t-1$ decision trees.
Then we have a prediction $\hat{y}_i^{(t-1)}$ from the aggregation of these $t-1$ decision trees.
Then the t^{th} aggregation looks like $\hat{y}_i^{(t-1)} + f_t(x_i)$.

Recall $\sum_{k=1}^{t} f_k = \left(\sum_{k=1}^{t-1} f_k\right) + f_t.$

Boosting = sequential optimization.

$$f_t = \underset{f_t}{\operatorname{argmin}}\, L^{(t)}(f_t)$$

How to actually minimize $L^{(t)}$?

↳ consider many decision trees f_t and simply select the one which minimizes $L^{(t)}$.

Ben Baldwin

$V_1(x) = WP(x \mid 1^{st} \text{ down and } 10)$

X = yardline, score diff, timeouts, game sec. rem., point spread, Receive 2nd half kickoff.

i = index i^{th} play

y_i = 1/0 if team with possession on play i won the game

X_i = game-state on play i

$$\widehat{WP}_1 = XGBoost(y, X)$$

P_{FG}, $P_{conversion}$, $\mathbb{E}$ next yardline after punting, $\widehat{WP}_1$

→ $V_{GO}(x)$, $V_{fg}(x)$, $V_{punt}(x)$

Choose the decision which maximizes estimated value (with probability),

$$\underset{d \in \{Go, Fg, Punt\}}{\operatorname{argmax}} \; V_d(x).$$

Up 3, 4th & 1, 14 yards from opponent end zone

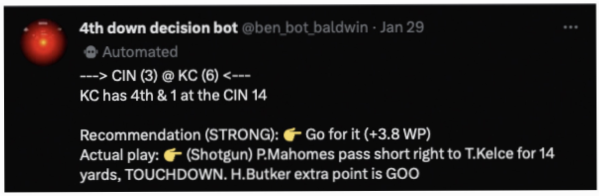
Qtr 2, 03:58 | Timeouts: Off 0, Def 3

	Win %	Success %[1]	Win % if Fail	Win % if Succeed
Go for it	**72**	68	64	76
Field goal attempt	**68**	94	62	69

[1] Likelihood of converting on 4th down or of making field goal

Source: @ben_bot_baldwin

(a)



(b)

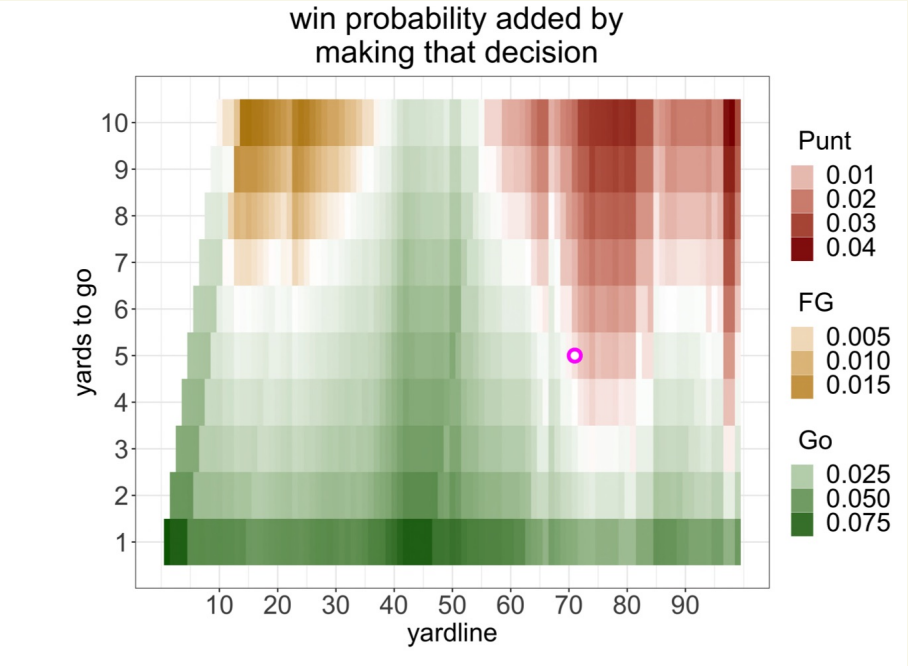
Figure 6: Baldwin's decision making for example play 1.

Gain = Value of Best − Value of 2nd best

Up 1, 4th & 5, 71 yards from opponent endzone

Qtr 3, 5:53 | Timeouts: Off 3, Def 3 | Point Spread: 3

decision	WP	success prob	WP if fail	WP if succeed	baseline coach %
Punt	0.440				0.934
Go for it	0.436	0.426	0.345	0.557	0.066
Field goal	0.345	0.000	0.345	0.548	0.000



Commanders have the ball against the Colts in week 8 of 2022

What if the model sucks?

Can we measure our confidence in the model's estimates?