Boosting
and
Gradient Boosting

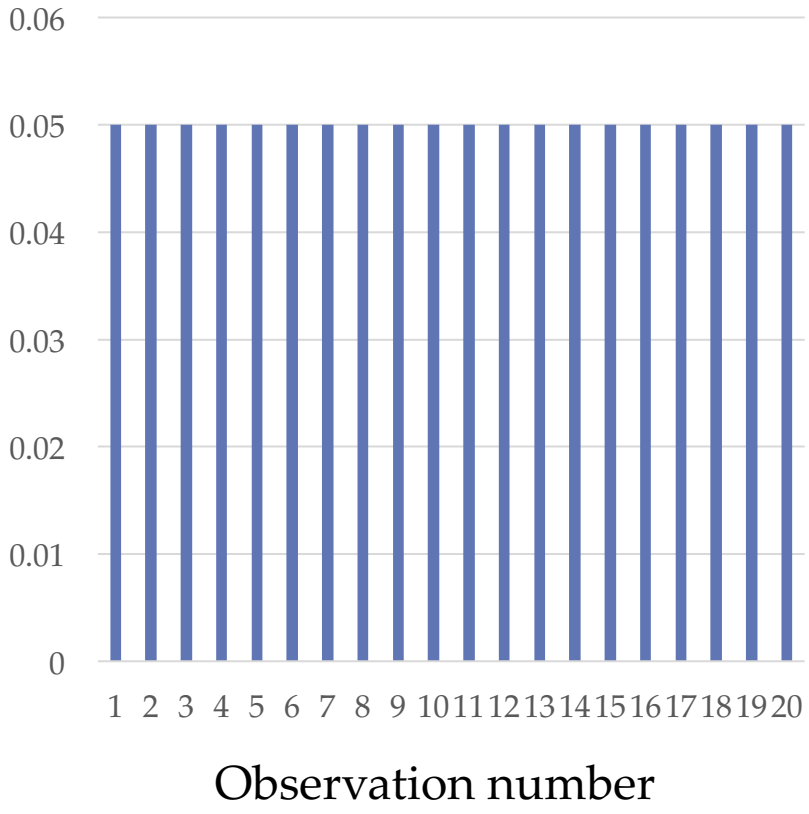
and xgboost

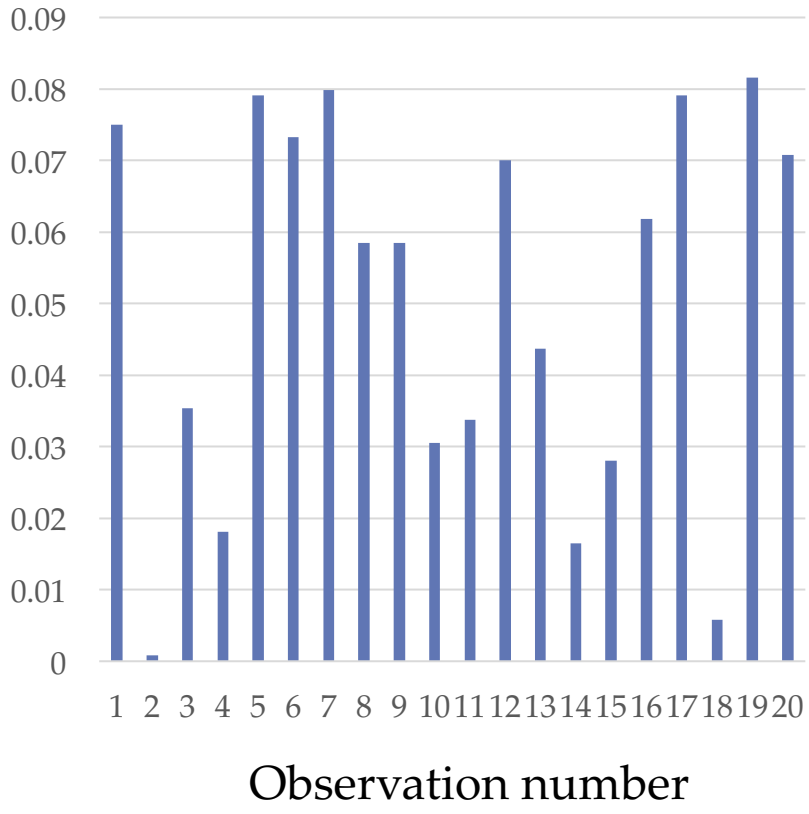
Boosting Overview

- Like bagging, going to draw a sample of the observations from our data with replacement
- Unlike bagging, the observations not sampled randomly
- Boosting assigns a weight to each training observation and uses that weight as a sampling distribution
  - Higher weight observations more likely to be chosen.
- May adaptively change that weight in each round
- **The weight is higher for examples that are harder to classify**

Bagging vs. Boosting

Probability of an observation being chosen for the sample at each round

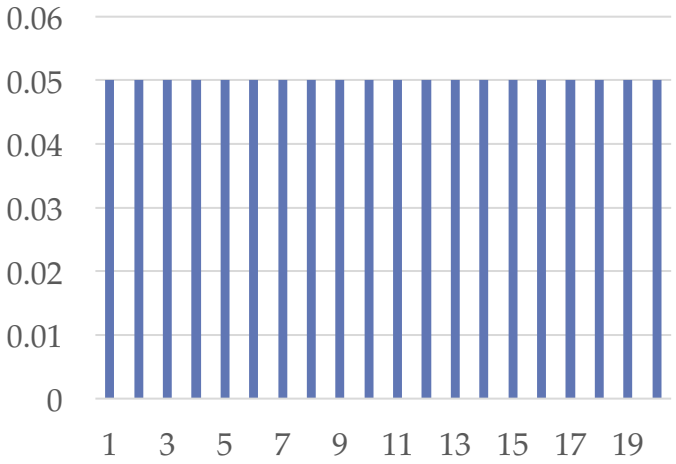




Bagging vs. Boosting

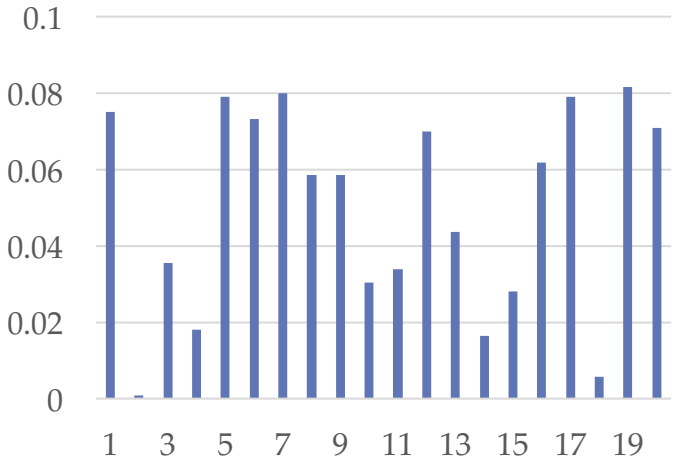
Only trying to create variability in the models by using training set variation.

Ensemble models built simultaneously, no time to evaluate accuracy.



Points with higher sampling probability were harder to predict accurately.

Want a chance to improve predictions sequentially



Boosting Example

input variable

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
y	1	1	1	-1	-1	-1	-1	1	1	1

target

- Same dataset used to illustrate bagging
- Boosting typically requires fewer rounds of sampling and classifier training.
- Start with equal weights for each observation
- Update weights each round based on the classification errors

Boosting Example

Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

(a) Training records chosen during boosting

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

(b) Weights of training records

Boosting: Weighted Ensemble

- Unlike Bagging, Boosted Ensembles usually weight the votes of each classifier by a function of their accuracy.
- If a classifier gets the higher weight observations wrong, it has a higher error rate.
- More accurate classifiers get higher weight in the prediction.

Boosting: Classifier weights

Errors made: First 3 observations

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Errors made: Middle 4 observations

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Errors made: Last 3 observations

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

(a) Training records chosen during boosting

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

(b) Weights of training records

Boosting: Classifier weights

Errors made: First 3 observations

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Errors made: Middle 4 observations

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Errors made: Last 3 observations

x	0.2	0.2	0.4	0.4	0.4	0.4
y	1	1	-1	-1	-1	-1

Lowest weighted error.
Highest weighted model.

(a) Training records chosen during boosting

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

(b) Weights of training records

Boosting: Weighted Ensemble

Round	Split Point	Left Class	Right Class	Weight
1	0.75	-1	1	1.738
2	0.05	1	1	2.7784
3	0.3	1	-1	4.1195

Classifier Decision Rules and Classifier Weights

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	-1	-1	-1	-1	-1	-1	-1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
Sum	5.16	5.16	5.16	-3.08	-3.08	-3.08	-3.08	0.397	0.397	0.397
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Individual Classifier Predictions and Weighted Ensemble Predictions

Boosting: Weighted Ensemble

Round	Split Point	Left Class	Right Class	Weight
1	0.75	-1	1	1.738
2	0.05	1	1	2.7784
3	0.3	1	-1	4.1195

Classifier Decision Rules and Classifier Weights

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	-1	-1	-1	-1	-1	-1	-1	1	1	1
2	1	1					1	1	1	1
3	1	1					-1	-1	-1	-1
Sum	5.16	5.16	5.16	-3.08	-3.08	-3.08	-3.08	0.397	0.397	0.397
Sign	1	1	1	-1	-1	-1	-1	1	1	1

5.16 = -1.738+2.7784+4.1195

Individual Classifier Predictions and Weighted Ensemble Predictions

(Major) Boosting Algorithms

AdaBoost (This is sooo 2007)

Gradient Boosting [xgboost]
(Welcome to the New Age of learning)

(Self-Study)

AdaBoost Details: The Classifier Weights

- Let w_j be the weight of observation j entering into present round.
- Let $m_j = 1$ if observation j is misclassified, 0 otherwise
- The error of the classifier this round is

$$\epsilon_i = \frac{1}{N}\sum_{j=1}^{N} w_j \, m_j$$

- The voting weight for the classifier this round is then

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\epsilon_i}{\epsilon_i}\right)$$

(Self-Study)

AdaBoost Details: Updating observation Weights

To update the observation weights from the current round (round i) to the next round (round $i+1$):

$w_j^{(i+1)} = w_j^i e^{-\alpha_j}$ if observation j was correctly classified

$w_j^{(i+1)} = w_j^i e^{\alpha_j}$ if observation j was misclassified

The new weights are then normalized to sum to 1 so they form a probability distribution.

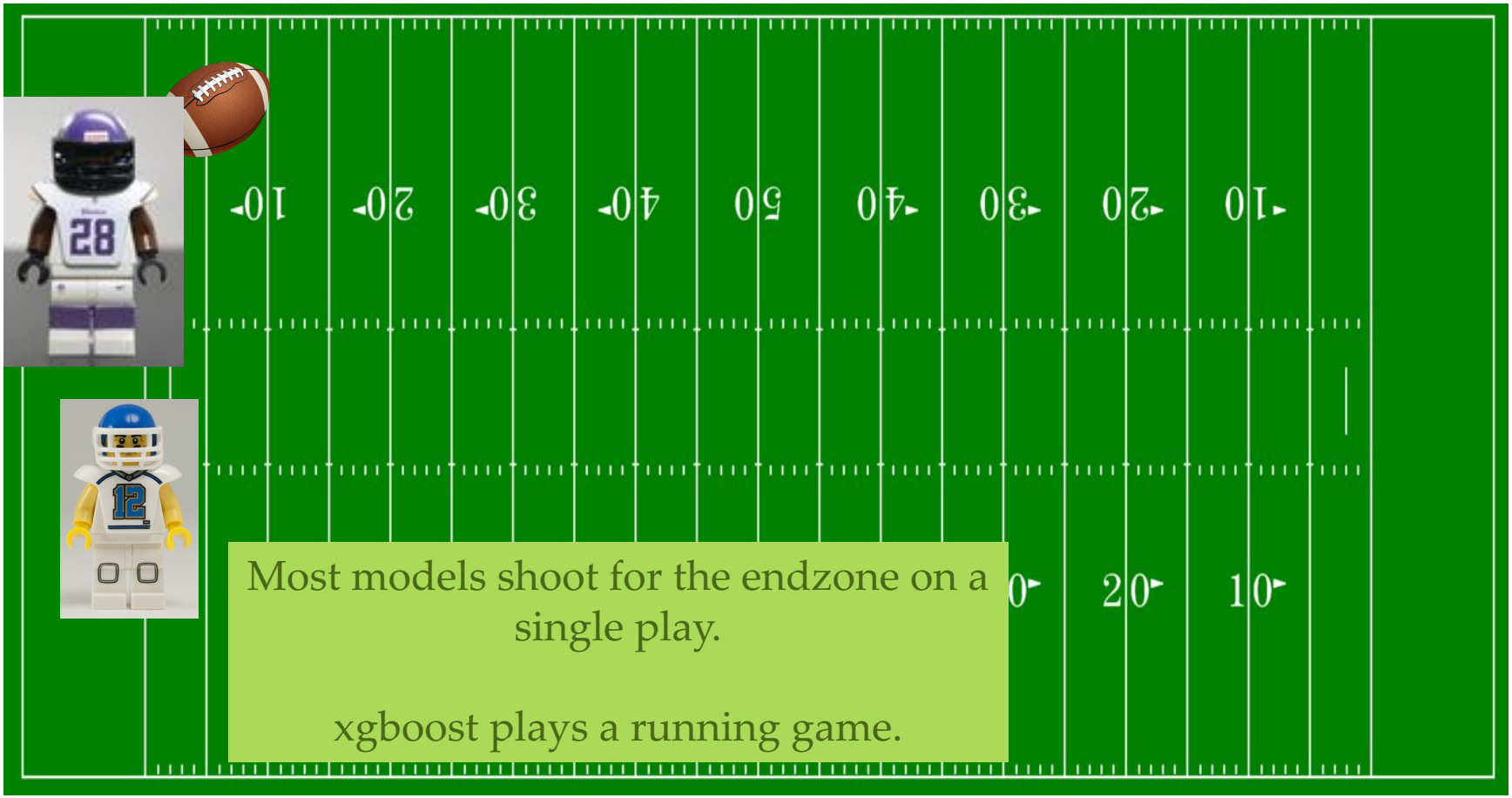
Gradient Boosting (xgboost)

• • •

The latest and greatest

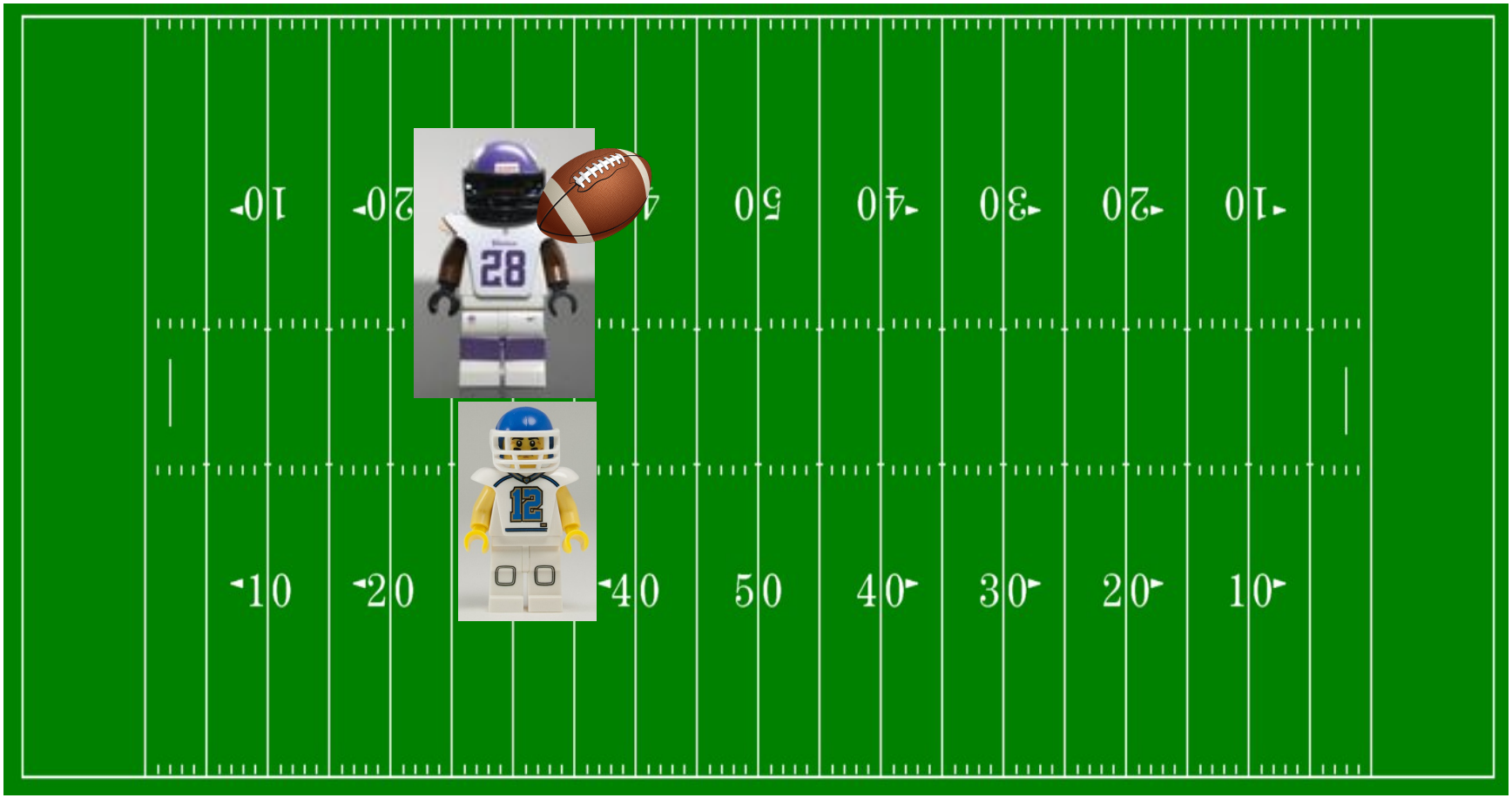
(Jerome H. Friedman 1999)

Football Strategy

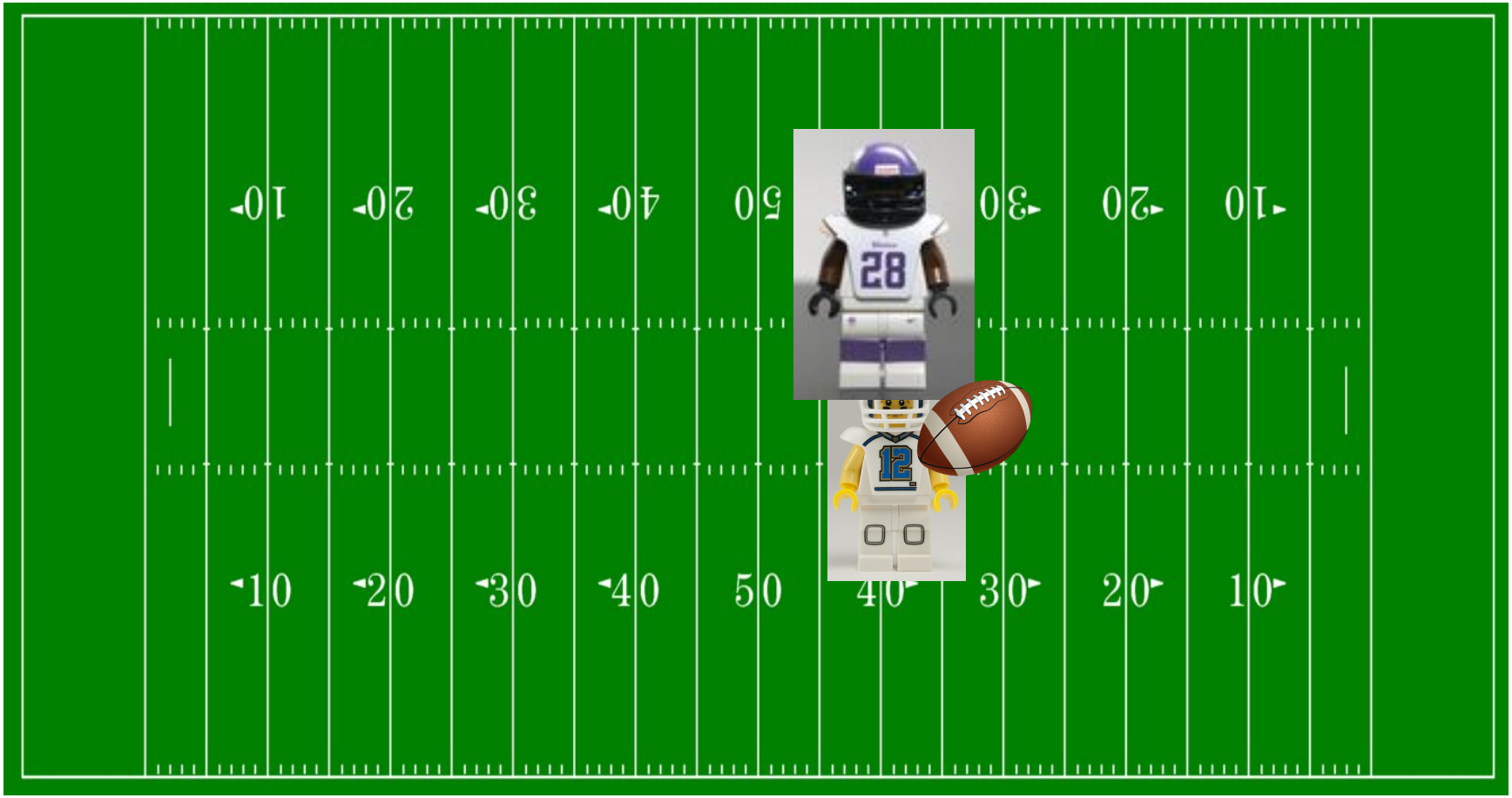


Football Strategy

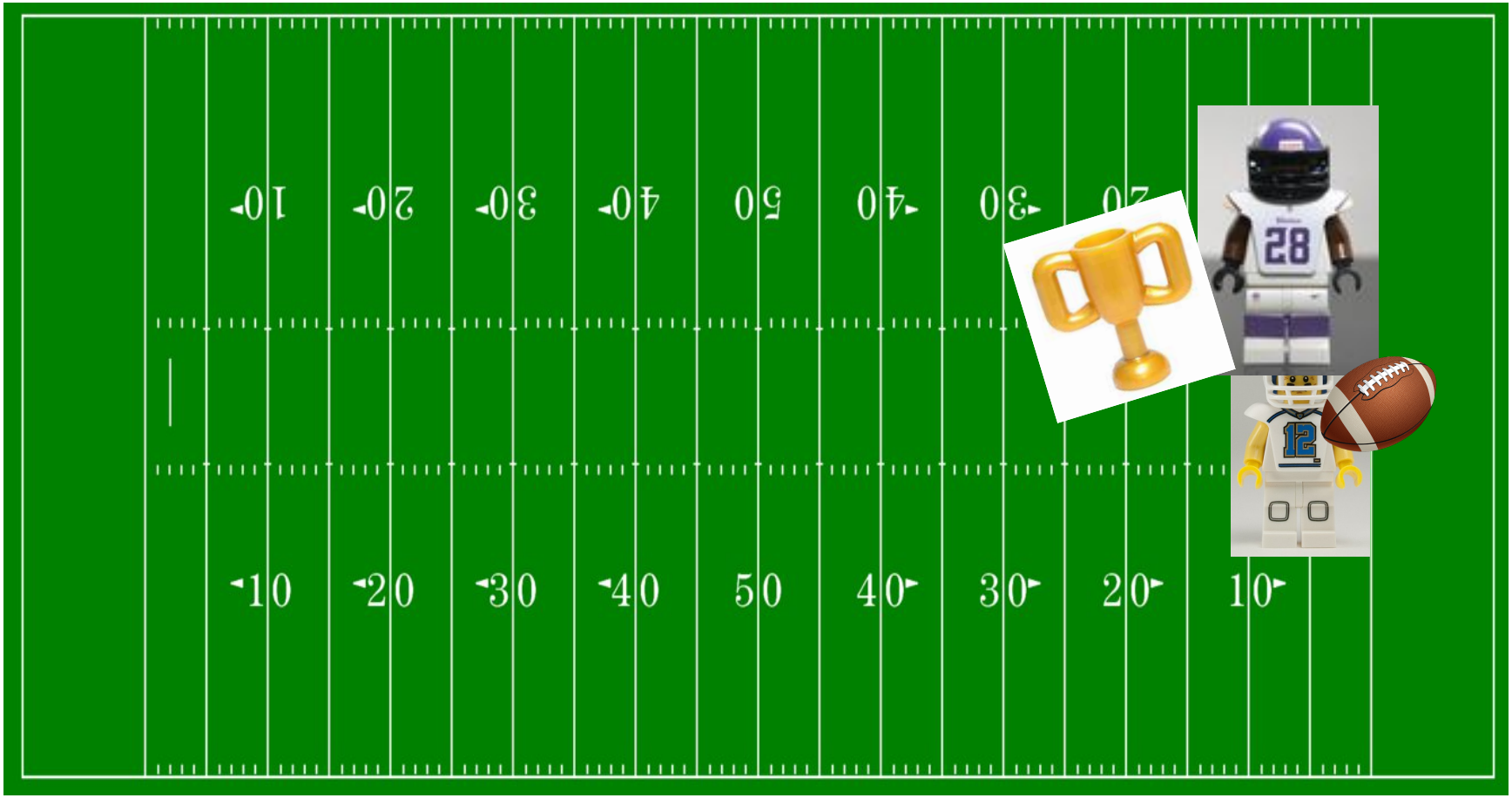
Football Strategy



Football Strategy



Football Strategy



Gradient Boosting Overview

- Build a **simple** model $f_1(x)$ trying to predict a target y
  - (i.e. try a running play)
- It has error.
  - (i.e. still some distance to the endzone)

$$y = f_1(x) + \epsilon_1$$

actual value *(initial yards to endzone)*

modeled value *(distance of run)*

error *(remaining distance to endzone)*

Gradient Boosting Overview

- Now, let's try to predict that error with another simple model, $f_2(x)$.
  - (Another running play)
- Unfortunately, it still has some error:
  - (Again, didn't make the endzone)

$$y = f_1(x) + f_2(x) + \epsilon_2$$

original modeled value *(line of scrimmage)*

predicting the residual, ϵ_1 *(yardage from second run)*

error *(remaining distance to endzone)*

Gradient Boosting Overview

- We could just continue to add model after model, trying to predict the residuals from the previous set of models.

$$y = f_1(x) + f_2(x) + f_3(x) + \cdots + f_k(x) + \epsilon_k$$

original modeled value *(first run yards)*

predicting the residual, ϵ_1 *(second run yards)*

predicting the residual, ϵ_2 *(third run yards)*

presumably very small error *(if we didn't get touchdown, hopefully at least in red-zone)*

Gradient Boosting Summary

- At each round, we create a model to predict the residual from the previous round.

- If we're just going to continue to model error until it vanishes, what's the obvious problem we should be aware of?

Gradient Boosting and Overfitting

- Gradient Boosting uses two forms of ***regularization*** to prevent overfitting:
  1. Control the number of trees/classifiers used in the prediction
     - Larger number of trees => More prone to overfitting
     - Choose a number of trees by observing out-of-sample error
  2. Use a *shrinkage parameter* ("learning rate") to effectively lessen the step-size taken at each step. Often called eta, η
     - $y = f_1(x) + \eta\, f_2(x) + \eta\, f_3(x) + \cdots + \eta\, f_k(x) + \epsilon_k$
     - Smaller values of eta => Less prone to overfitting
     - eta = 1 => no regularization

Gradient Boosted Trees

- Gradient boosting yields a *additive* ensemble model
  - There is no voting or averaging of individual models.
  - The predictions from each model get added together for final prediction.

- The key to gradient boosting is using **"weak learners"**
  - Typically simple, shallow decision/regression trees
  - Computationally fast and efficient
  - Alone, make poor predictions but ensembled in this additive fashion provide superior results

Gradient Boosting Summary

- Advantages
  - Exceptional model – one of most accurate available, generally superior to Random Forests when well trained
  - Can provide information on variable importance for the purposes of variable selection
- Disadvantages
  - Model lacks interpretability in the classical sense aside from variable importance
  - The trees must be trained sequentially so computationally this method is slower than Random Forest
  - Extra tuning parameter over Random Forests, the regularization or shrinkage parameter, eta.
  - Can be hard to optimize tuning parameters (time/complexity)

Notes about EM

- EM has node for Random Forest (HP tab=> HP Forest)
  - Uses CHAID unlike other implementations
  - Does not perform bootstrap sampling
  - Does not appear to work as well as the randomForest package in R
- EM has node for gradient boosting
  - Personally I recommend the "extreme gradient boosting" implementation of this method, which is called *xgboost* both in R and python.
  - This implementation appears to be stronger and faster than the one in SAS