Non-parametric methods
Random Forests, Bagging

Natasha Jaques

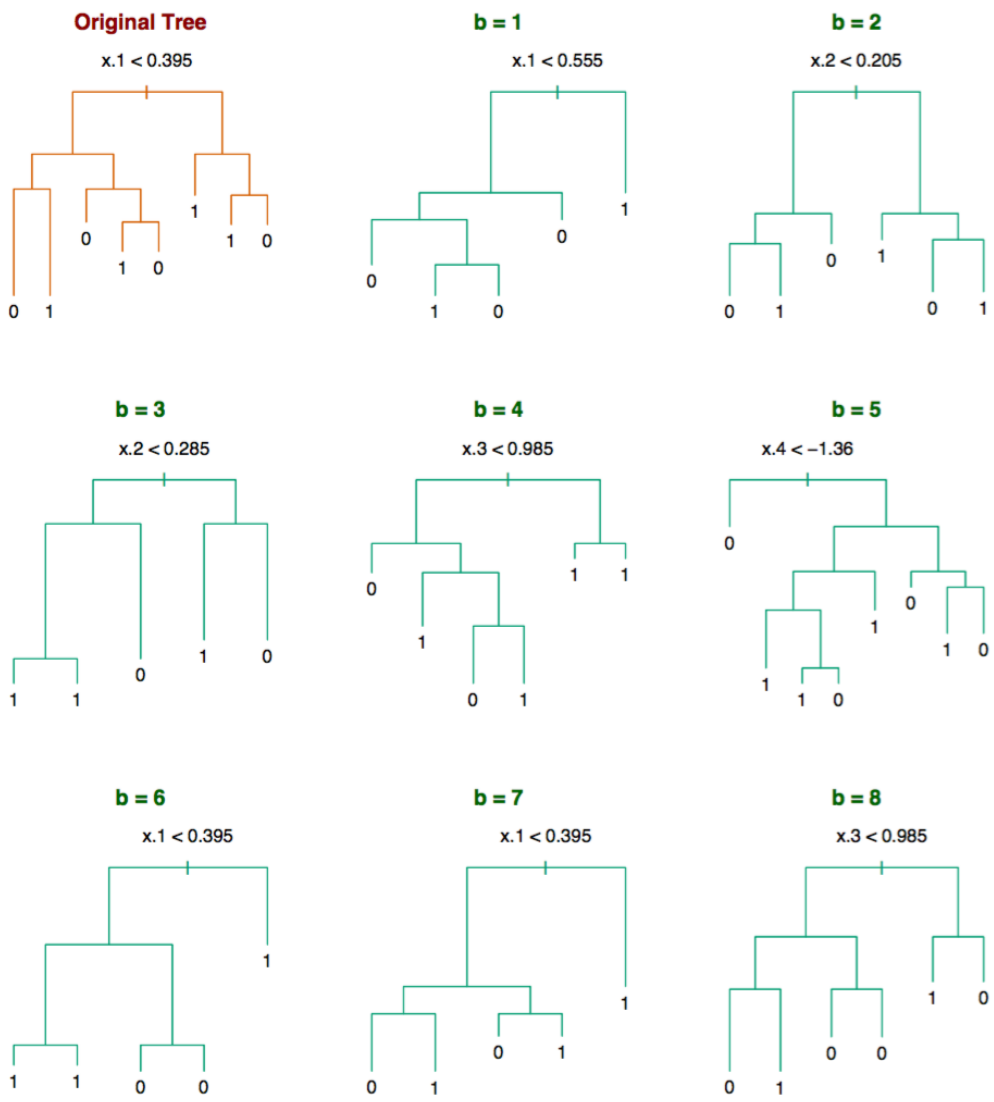
UNIVERSITY of WASHINGTON

Random forests

- Forest = many trees
- Tree methods have low bias but high variance
- We can reduce variance by constructing many “lightly correlated” trees and averaging them

- Bagging: Bootstrap aggregating

Each tree is trained on new bootstrap sample of data



Random forests

Random forests

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:

   # with replacement

   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.

   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

   # b = train tree

      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.

   # but you're using a random subset of the features

2. Output the ensemble of trees $\{T_b\}_1^B$.

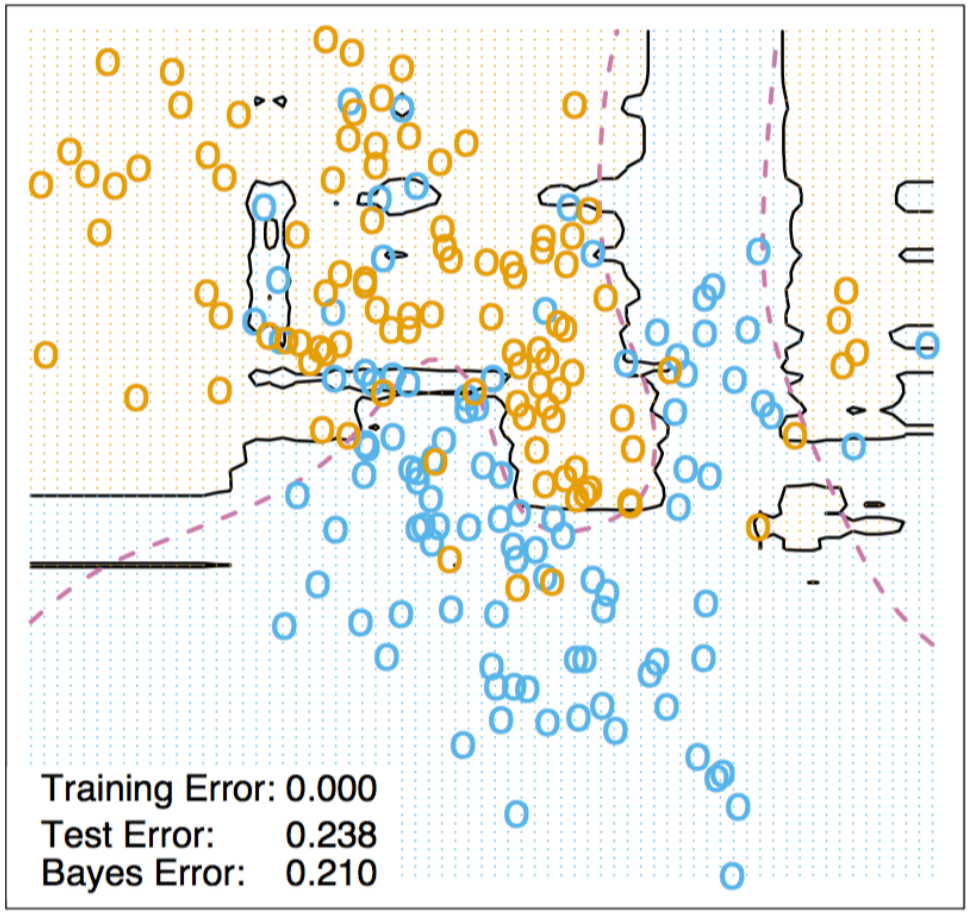
To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Average

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Random forests: Decision boundary example



Dashed line is the Bayes optimal classifier

Random forests theoretical motivation

We will show the average of lightly correlated variables (predictors) has lower variance

Given random variables $Y_1, Y_2, \ldots, Y_B$ with
$\mathbb{E}[Y_i] = y$, $\mathbb{E}[(Y_i - y)^2] = \sigma^2$, $\mathbb{E}[(Y_i - y)(Y_j - y)] = \rho\sigma^2$

correlation coefficient

Unbiased — Variance — Correlation b/w predictors

No relationship $0 \leq \rho \leq 1$ Duplicate

$$\mathbb{E}[(\frac{1}{B}\sum_{i=1}^{B} Y_i - y)^2] = \mathbb{E}[\frac{1}{B^2}\sum_{i=1}^{B}(Y_i - y)^2] + \mathbb{E}[\frac{1}{B^2}\sum_{i \neq j}(Y_i - y)(Y_j - y)] = \frac{\sigma^2}{B} + \frac{B-1}{B}\rho\sigma^2$$

if $\rho = 1$

$$\frac{\sigma^2}{B} + \frac{(B-1)}{B}\sigma^2 = \frac{B}{B}\sigma^2 = \sigma^2$$

but if $\rho < 1$

Can reduce variation by increasing B!

Ensembles are powerful

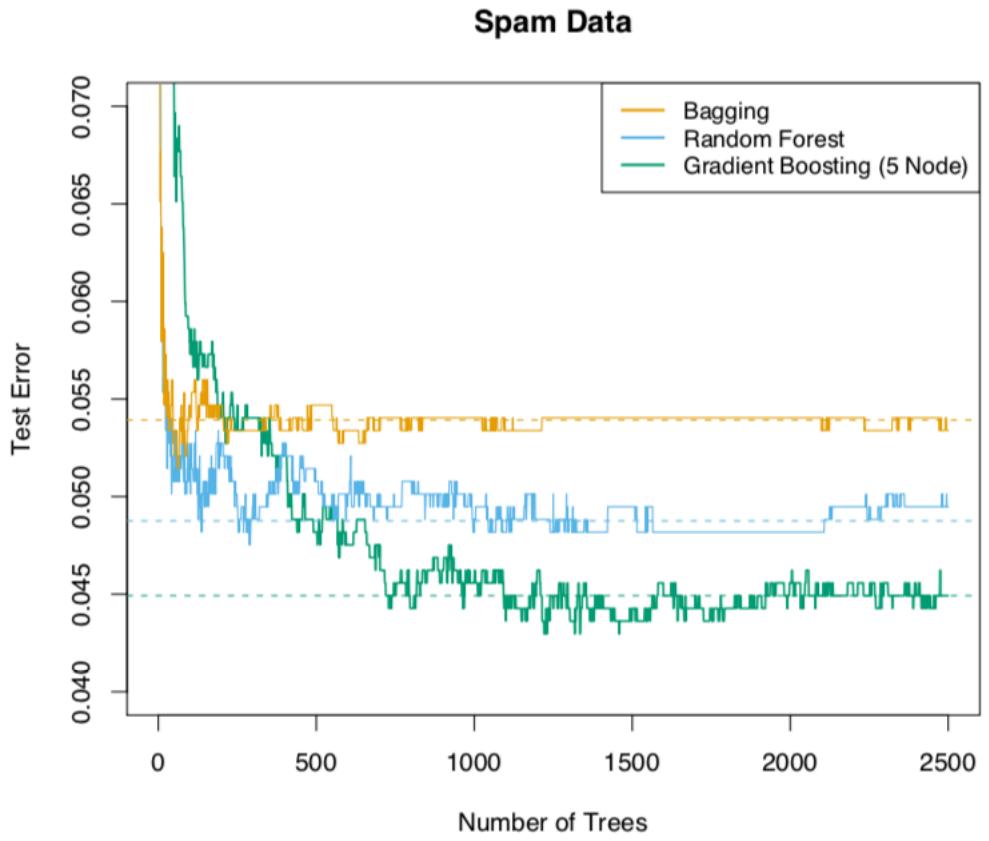
For binary classification, if you have N weak classifiers, but each one is slightly better than random chance (gets right answer with $p > 0.5$), what happens if you take the majority vote?

As $N \to \infty$, what is the probability that the majority vote gets the right answer?

100%!!

- Marquis de Condorcet, "Essay on the Application of Analysis to the Probability of Majority Decisions" (1785). Known as the Condorcet Jury Theorem.
- Schapire, "**The Strength of Weak Learnability**" (1990)

The power of weakly correlated predictors



Bagging: Averaged trees on bootstrapped datasets using all d features

Random forest: Averaged trees on bootstrapped datasets using m randomly selected features

Takeaways:

- Reducing correlation improves performance
- Ensembles are powerful

Summary so far

- Random forests have low bias, low variance
- Deal with categorial variables well
- Not that intuitive nor “interpretable”
- Gives some notion of confidence estimates
- Good software exists
- Some theoretical guarantees

Boosting and additive models

Instead of ensembling bootstrapped models, can we:

- Keep the idea of ensembling / combining simpler models, but
- Not necessarily have the models be identically distributed?

Key idea: Given a current collection of models, add a new model that focuses on what the previous models got wrong

Additive models

Given data $\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, \quad y_i \in \{+1, -1\}$ # We're doing classification

$b_m : \mathbb{R}^d \to \mathbb{R}, \quad m = 1 \ldots M$ # There are M classifiers

$x \to y$

$\beta_m \in \mathbb{R}$ # Weight we will use to combine each classifier

$$\hat{\beta}_1, \hat{b}_1 \ldots \hat{\beta}_M, \hat{b}_M = \arg\min \sum_{i=1}^{n} \ell(y_i, \sum_{m=1}^{M} \beta_m b_m(x_i))$$

Ideally we would fit a bunch of models simultaneously and combine together

Buuuut, this is way too hard.

So instead we sequentially learn each new model (greedily) one at a time

Forward stagewise additive models

Algorithm 10.2 *Forward Stagewise Additive Modeling.*

1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to M: # For all models in the ensemble
   (a) Compute

$$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)).$$

   (b) Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$. # f_{m-1} is the previous ensemble

E.g. Adaboost

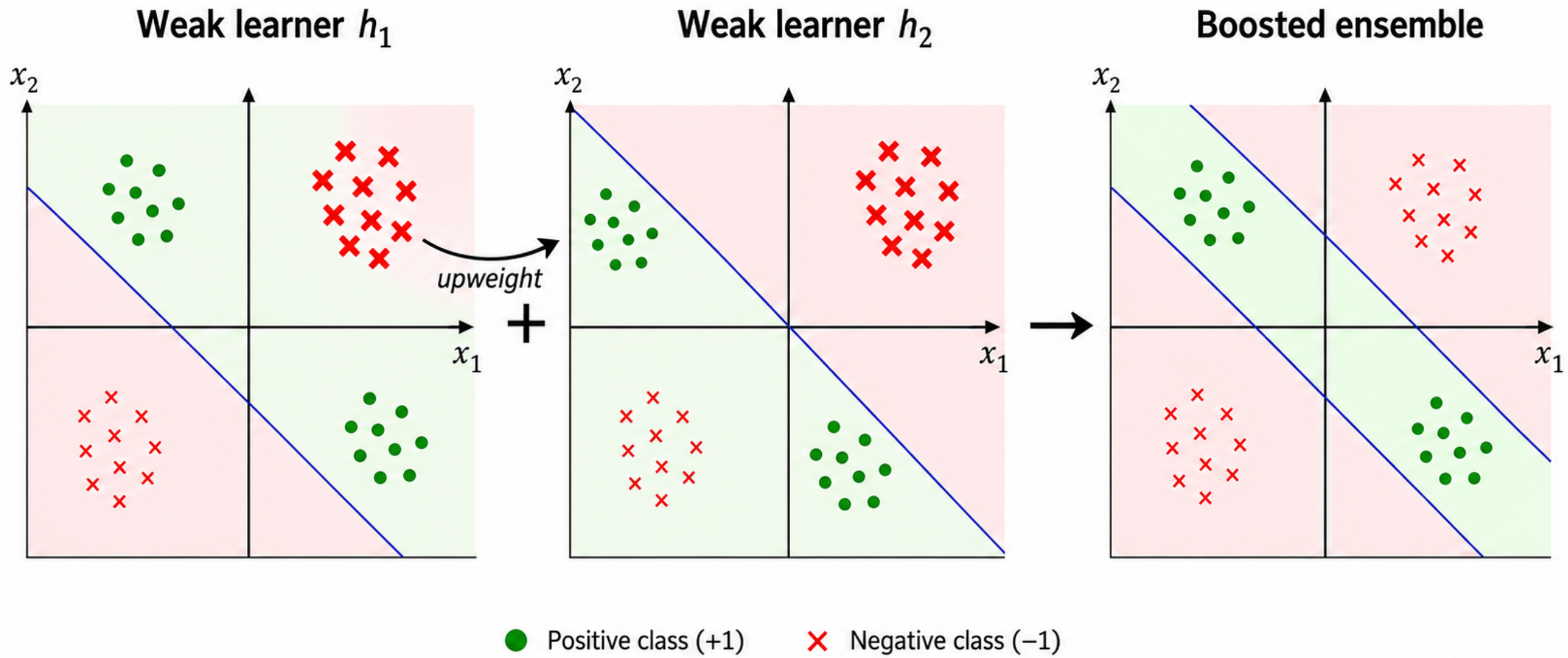
$$b(x, \gamma): \mathbb{R}^d \to \{-1, 1\}$$

$$\ell(y, f_m(x)) = e^{-y f_m(x)}$$

Loss exponentially upweights previous classification errors

Key idea: "boost" weight on incorrect points, try to fit them with a new classifier

Forward stagewise additive models



Forward stagewise additive models

Algorithm 10.2 *Forward Stagewise Additive Modeling.*

1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to M: # For all models in the ensemble
   (a) Compute

$$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)).$$

   (b) Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$. # f_{m-1} is the previous ensemble

E.g. Boosted regression trees

$b(x, \gamma)$ is a regression tree

$$\ell(y, f(x)) = (y - f(x))^2$$

$$\ell(y_i, f_{m-1}(x_i) + \beta b(x_i, \gamma)) = (y_i - f_{m-1}(x_i) - \beta b(x_i, \gamma))^2$$

Previous ensemble

New model

Residual

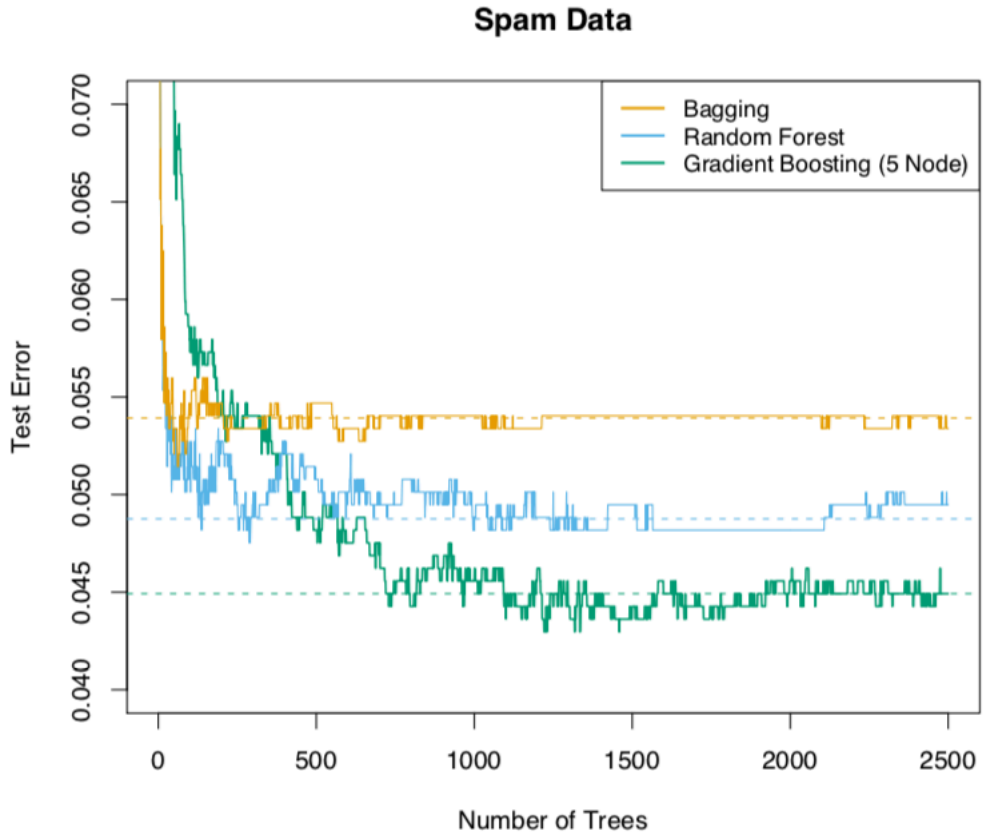
$$= (r_{im} - \beta b(x_i, \gamma))^2$$

A brief history of boosting

- 1988 Kearns and Valiant: “Can weak learners be combined to create a strong learner?”
- 1990 Schapire: “Yup, in theory”
- 1995 Schapire and Freund: “Practical for 0/1 loss” -> AdaBoost
- 2001 Friedman: “Practical for arbitrary losses”
- 2014 Tianqi Chen: “Scale it up!” -> XGBoost

François Chollet @fchollet · Apr 3, 2019

What machine learning tools do Kaggle champions use? We ran a survey among teams that ranked in the *top 5* of a competition since 2016.



Bagging: Averaged trees on bootstrapped datasets using all d features

Random forest: Averaged trees on bootstrapped datasets using m randomly selected features

Boosting: Learned combinations of trees

Takeaways

- Single trees: low bias, high variance
- Ensembles: low bias, (relatively) low variance
- Bagging averages many lightly dependent models to reduce variance
  - Random forests: same but with random subset of features
- Boosting learns a linear combination of high bias, highly dependent classifiers to reduce error
- Gradient boosted trees are commonly used for categorical data