Announcements

- Evaluations
- Concerns over grades
- Google form sent out after class (for feedback, and incomplete requests)
- Future Offerings Discussion
- Lecture

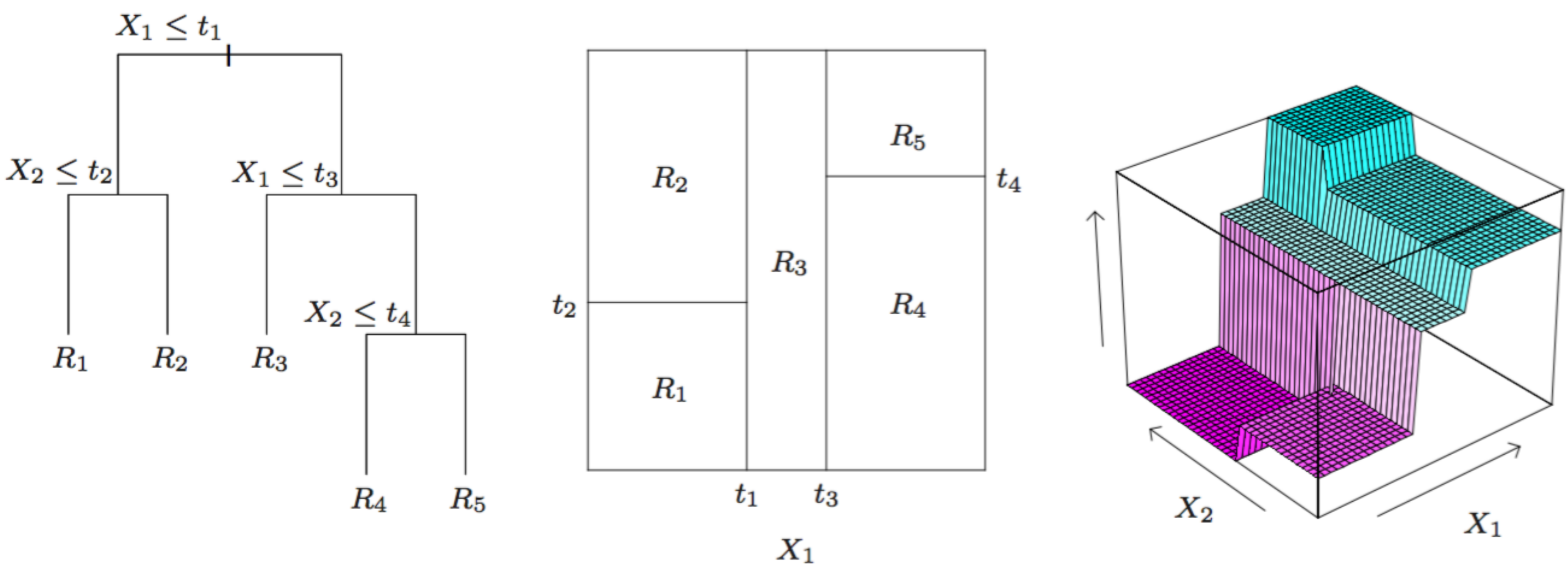
Trees

Regression Trees

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m).$$

Build a binary tree, splitting along axes

$$\hat{c}_m = \text{ave}(y_i | x_i \in R_m).$$



Regression Trees

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m).$$

$$\hat{c}_m = \text{ave}(y_i | x_i \in R_m).$$

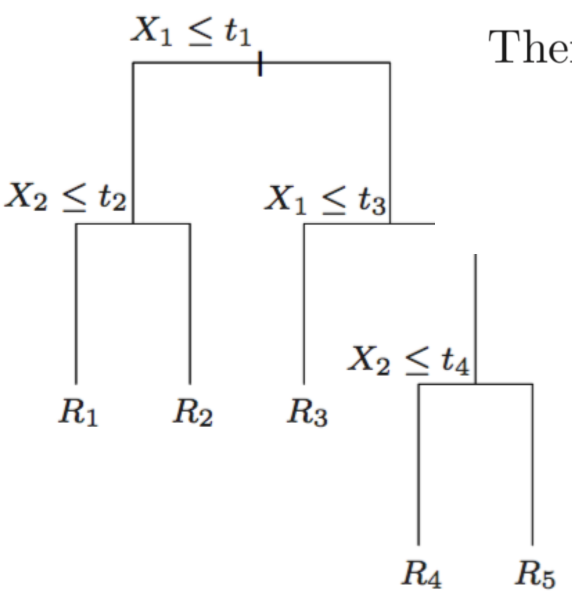
Build a binary tree, splitting along axes

How do you split?

$$R_1(j, s) = \{X | X_j \le s\} \text{ and } R_2(j, s) = \{X | X_j > s\}.$$

Then we seek the splitting variable j and split point s that solve

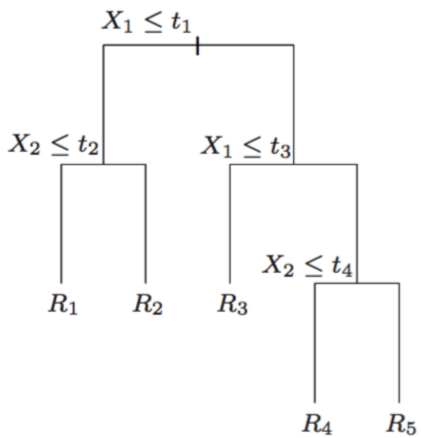
$$\min_{j,\, s} \Big[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \Big].$$



When do you stop?

Decision Trees

- Start from empty decision tree
- Split on **next best attribute (feature)**
  - Use, for example, information gain to select attribute
  - Split on $\arg\max_i IG(X_i) = \arg\max_i H(Y) - H(Y \mid X_i)$
- Recurse
- Prune

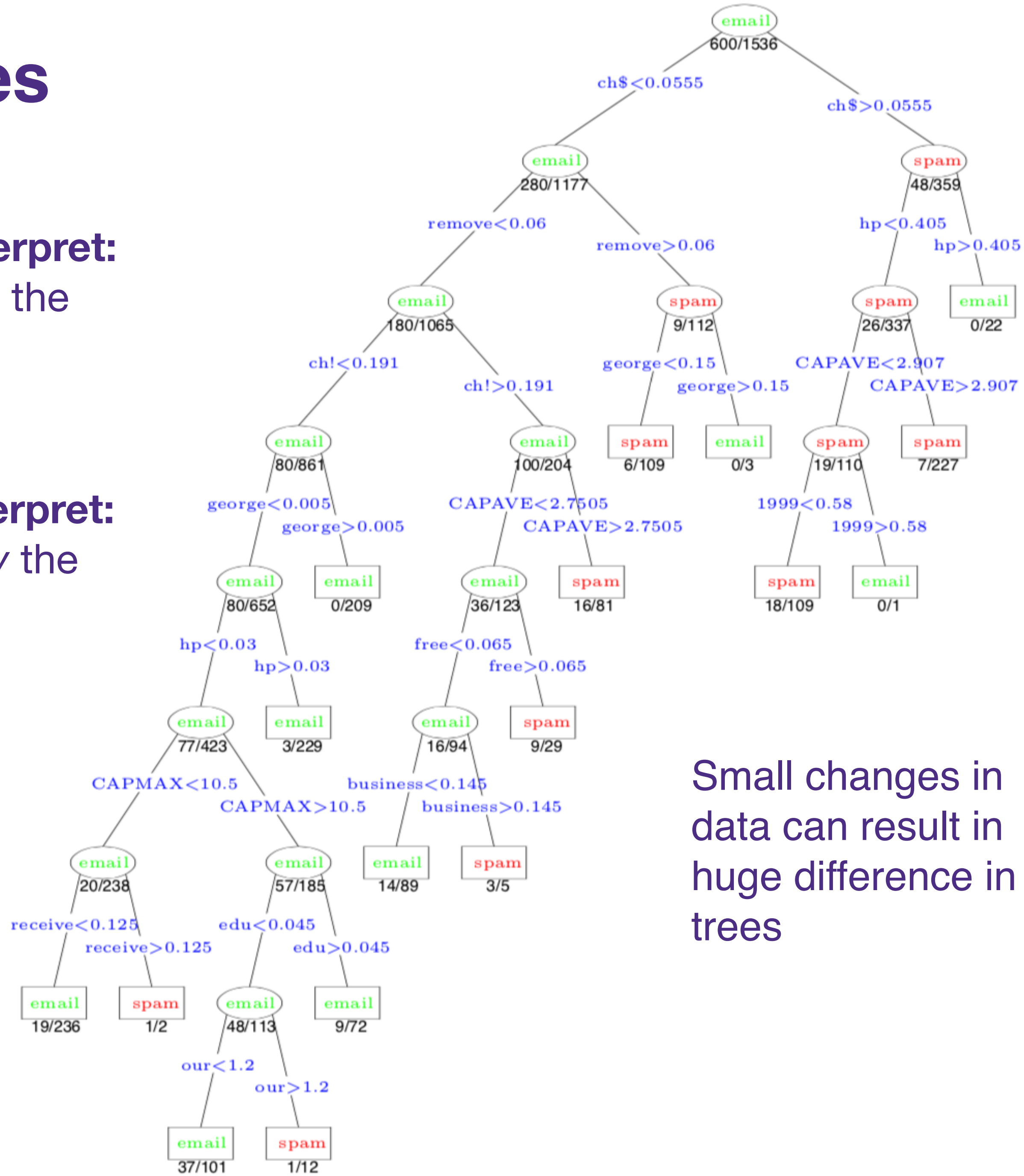


$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m).$$

Decision Trees

Trees are easy to interpret:
- You can explain *how* the classifier came to the conclusion it did

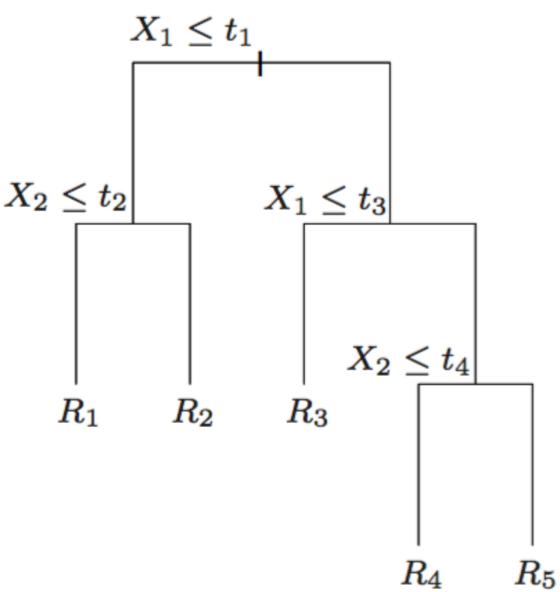
Trees are hard to interpret:
- Tough to explain *why* the classifier came to the conclusion it did



Small changes in data can result in huge difference in trees

Decision Trees

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m).$$



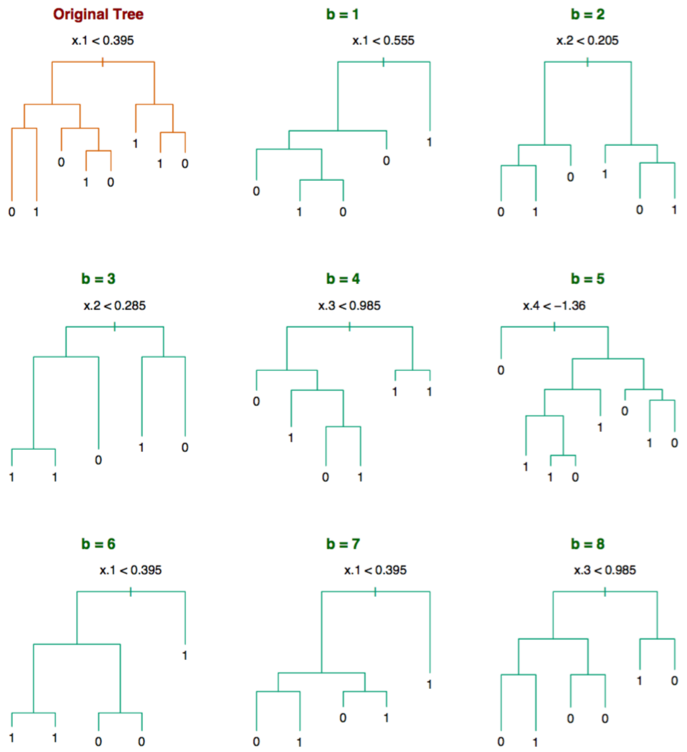
- Trees
  - **have low bias, high variance**
  - deal with categorial variables well
  - intuitive, interpretable (maybe)
  - good software exists
  - Some theoretical guarantees

Random Forests

Random Forest

Tree methods have **low bias** but **high variance**.

One way to reduce variance is to construct a lot of "lightly correlated" trees and average them:



"Bagging:" Bootstrap aggregating

Random Forest

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:
   (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.
      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.
2. Output the ensemble of trees $\{T_b\}_1^B$.

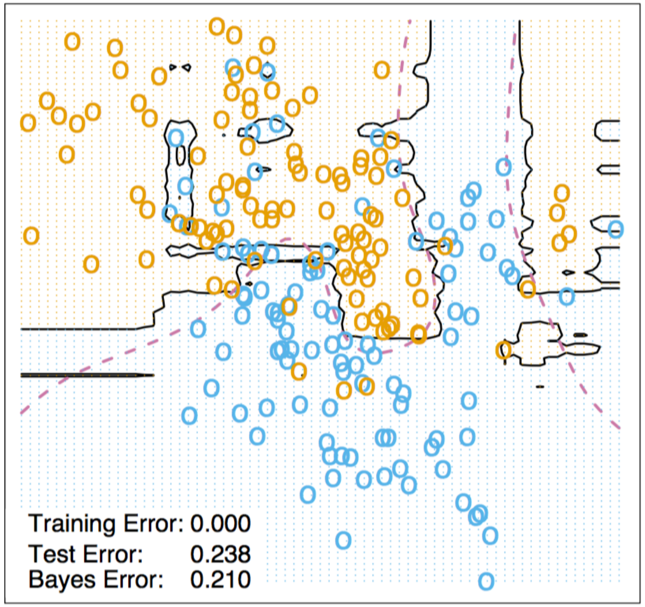
To make a prediction at a new point x:

Regression: $\hat{f}_{\mathrm{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$. m~p/3

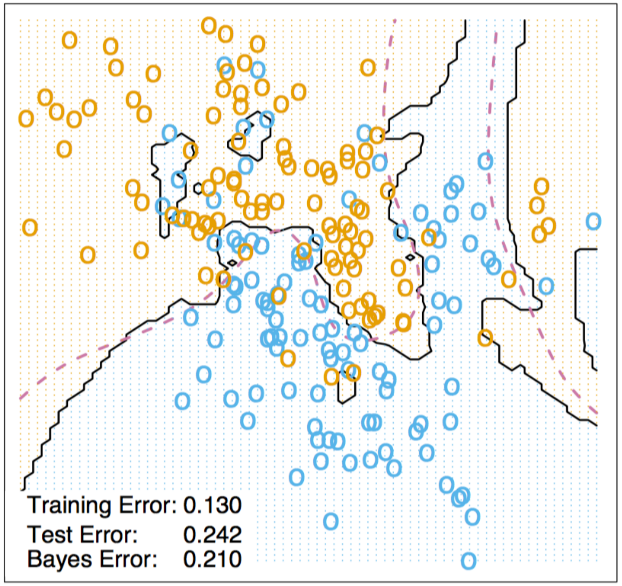
Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\mathrm{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$. m~sqrt(p)

Random Forest

Random forrest



3 nearest neighbor



Random Forest

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

σ^2 Variance of individual predictor

Assume bias = 0

$\rho\sigma^2$ Correlation between predictors

The Yi's are identically distributed but **not** independent

$$\mathbb{E}[(\frac{1}{B}\sum_{i=1}^{B} Y_i - y)^2] =$$

Random Forest

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

σ^2 Variance of individual predictor

Assume bias = 0

$\rho\sigma^2$ Correlation between predictors

The Yi's are identically distributed but **not** independent

$$\mathbb{E}[(\frac{1}{B}\sum_{i=1}^{B} Y_i - y)^2] = \underbrace{\frac{1}{B}\sigma^2}_{\text{Goes to 0 as } B \to \infty} + \underbrace{(1 - \frac{1}{B})\rho\sigma^2}_{\text{Error dominated by correlation}}$$

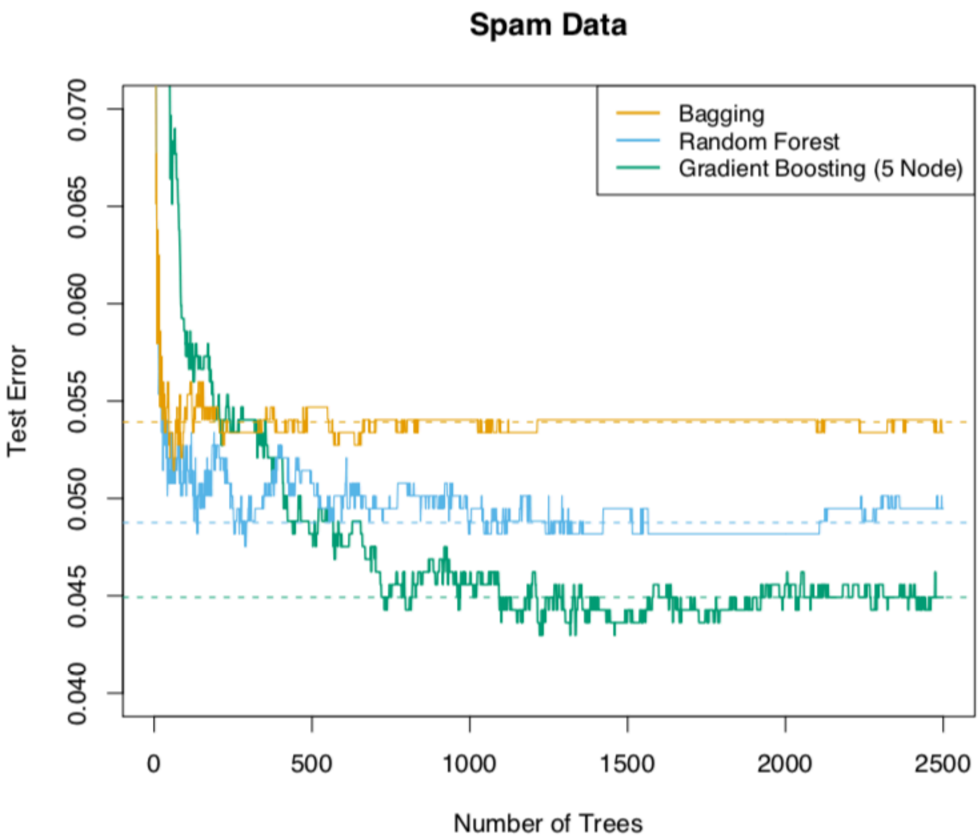
Goes to 0 as $B \to \infty$

Error dominated by correlation

Averaging weakly correlated models results in biggest gains

Random Forest

The power of weakly correlated predictors:



Bagging: Averaged trees trained on bootstrapped datasets that used **all d variables**

Random forest: Averaged trees trained on bootstrapped datasets that **m<d random variables**

Gradient boosting: ignore for now

Takeaway: reducing correlation improves performance!

Random Forests

- Random Forests
  - **have low bias, low variance**
  - deal with categorial variables well
  - not that intuitive or interpretable
  - good software exists
  - Some theoretical guarantees
  - Can still overfit
  - **Extremely effective in practice**

Boosting

Boosting

- 1988 Kearns and Valiant: “Can **weak learners** be combined to create a **strong learner?**”

Weak learner definition (informal):

An algorithm $\mathcal{A}$ is a *weak learner* for a hypothesis class $\mathcal{H}$ that maps $\mathcal{X}$ to $\{-1, 1\}$ if for all input distributions over $\mathcal{X}$ and $h \in \mathcal{H}$, we have that $\mathcal{A}$ correctly classifies h with error at most $1/2 - \gamma$

Boosting

- 1988 Kearns and Valiant: “Can **weak learners** be combined to create a **strong learner?**”

Weak learner definition (informal):

An algorithm $\mathcal{A}$ is a *weak learner* for a hypothesis class $\mathcal{H}$ that maps $\mathcal{X}$ to $\{-1, 1\}$ if for all input distributions over $\mathcal{X}$ and $h \in \mathcal{H}$, we have that $\mathcal{A}$ correctly classifies h with error at most $1/2 - \gamma$

- 1990 Robert Schapire: “Yup!”
- 1995 Schapire and Freund: “Practical for 0/1 loss” AdaBoost

Boosting

- 1988 Kearns and Valiant: "Can **weak learners** be combined to create a **strong learner?**"

Weak learner definition (informal):

An algorithm $\mathcal{A}$ is a *weak learner* for a hypothesis class $\mathcal{H}$ that maps $\mathcal{X}$ to $\{-1, 1\}$ if for all input distributions over $\mathcal{X}$ and $h \in \mathcal{H}$, we have that $\mathcal{A}$ correctly classifies h with error at most $1/2 - \gamma$

- 1990 Robert Schapire: "Yup!"
- 1995 Schapire and Freund: "Practical for 0/1 loss" AdaBoost
- 2001 Friedman: "Practical for arbitrary losses"

Boosting

- 1988 Kearns and Valiant: “Can **weak learners** be combined to create a **strong learner?**”

Weak learner definition (informal):

An algorithm $\mathcal{A}$ is a *weak learner* for a hypothesis class $\mathcal{H}$ that maps $\mathcal{X}$ to $\{-1, 1\}$ if for all input distributions over $\mathcal{X}$ and $h \in \mathcal{H}$, we have that $\mathcal{A}$ correctly classifies h with error at most $1/2 - \gamma$

- 1990 Robert Schapire: “Yup!”
- 1995 Schapire and Freund: “Practical for 0/1 loss” AdaBoost
- 2001 Friedman: “Practical for arbitrary losses”
- 2014 Tianqi Chen: “Scale it up!” XGBoost
- 2017 MSR: “We can go faster” LightGBM

Boosting and Additive Models

UNIVERSITY of WASHINGTON

Additive models

- Consider the first algorithm we used to get good classification for MNIST. Given: $\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, y_i \in \{-1, 1\}$
- Generate **random** functions: $\phi_t : \mathbb{R}^d \to \mathbb{R} \quad t = 1, \dots, p$
- Learn some weights: $\widehat{w} = \arg\min_w \sum_{i=1}^n \text{Loss}\left(y_i, \sum_{t=1}^p w_t \phi_t(x_i)\right)$
- Classify new data: $f(x) = \text{sign}\left(\sum_{t=1}^p \widehat{w}_t \phi_t(x)\right)$

Additive models

- Consider the first algorithm we used to get good classification for MNIST. Given: $\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, y_i \in \{-1, 1\}$
- Generate **random** functions: $\phi_t : \mathbb{R}^d \to \mathbb{R} \quad t = 1, \dots, p$
- Learn some weights: $$\widehat{w} = \arg\min_w \sum_{i=1}^n \text{Loss}\left(y_i, \sum_{t=1}^p w_t \phi_t(x_i) \right)$$
- Classify new data: $$f(x) = \text{sign}\left(\sum_{t=1}^p \widehat{w}_t \phi_t(x) \right)$$

An interpretation:
Each $\phi_t(x)$ is a classification rule that we are assigning some weight $\widehat{w}_t$

Additive models

- Consider the first algorithm we used to get good classification for MNIST. Given: $\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, y_i \in \{-1, 1\}$
- Generate **random** functions: $\phi_t : \mathbb{R}^d \to \mathbb{R} \quad t = 1, \dots, p$
- Learn some weights: $\widehat{w} = \arg\min_w \sum_{i=1}^n \text{Loss}\left(y_i, \sum_{t=1}^p w_t \phi_t(x_i) \right)$
- Classify new data: $f(x) = \text{sign}\left(\sum_{t=1}^p \widehat{w}_t \phi_t(x) \right)$

An interpretation:
Each $\phi_t(x)$ is a classification rule that we are assigning some weight $\widehat{w}_t$

$$\widehat{w}, \widehat{\phi}_1, \dots, \widehat{\phi}_t = \arg \min_{w, \phi_1, \dots, \phi_p} \sum_{i=1}^n \text{Loss}\left(y_i, \sum_{t=1}^p w_t \phi_t(x_i) \right)$$

is in general computationally hard

Forward Stagewise Additive models

$b(x, \gamma)$ is a function with parameters γ

Examples: $b(x, \gamma) = \dfrac{1}{1 + e^{-\gamma^T x}}$

$b(x, \gamma) = \gamma_1 \mathbf{1}\{x_3 \leq \gamma_2\}$

Algorithm 10.2 *Forward Stagewise Additive Modeling.*

1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to M:
   (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 $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.

Idea: greedily add one function at a time

Forward Stagewise Additive models

$b(x,\gamma)$ is a function with parameters γ

Examples: $$b(x,\gamma) = \frac{1}{1+e^{-\gamma^T x}}$$

$$b(x,\gamma) = \gamma_1 \mathbf{1}\{x_3 \leq \gamma_2\}$$

Algorithm 10.2 *Forward Stagewise Additive Modeling.*

1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to M:
   (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 $f_m(x) = f_{m-1}(x) + \beta_m b(x;\gamma_m)$.

Idea: greedily add one function at a time

AdaBoost: $b(x,\gamma)$: classifiers to $\{-1,1\}$

$$L(y,f(x)) = \exp(-yf(x))$$

Forward Stagewise Additive models

$b(x, \gamma)$ is a function with parameters γ

Examples: $b(x, \gamma) = \frac{1}{1 + e^{-\gamma^T x}}$

$$b(x, \gamma) = \gamma_1 \mathbf{1}\{x_3 \leq \gamma_2\}$$

Algorithm 10.2 *Forward Stagewise Additive Modeling.*

1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to M:
   (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 $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.

Idea: greedily add one function at a time

Boosted Regression Trees: $L(y, f(x)) = (y - f(x))^2$

$b(x, \gamma)$: regression trees

Forward Stagewise Additive models

$b(x, \gamma)$ is a function with parameters γ

Examples: $b(x,\gamma) = \dfrac{1}{1 + e^{-\gamma^T x}}$

$b(x,\gamma) = \gamma_1 \mathbf{1}\{x_3 \leq \gamma_2\}$

Algorithm 10.2 *Forward Stagewise Additive Modeling.*

1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to M:
   (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 $f_m(x) = f_{m-1}(x) + \beta_m b(x;\gamma_m)$.

Idea: greedily add one function at a time

Boosted Regression Trees: $L(y, f(x)) = (y - f(x))^2$

$$\begin{aligned} L(y_i, f_{m-1}(x_i) + \beta b(x_i;\gamma)) &= (y_i - f_{m-1}(x_i) - \beta b(x_i;\gamma))^2 \\ &= (r_{im} - \beta b(x_i;\gamma))^2, \end{aligned} \qquad r_{im} = y_i - f_{m-1}(x_i)$$

Efficient: No harder than learning regression trees!

Forward Stagewise Additive models

$b(x, \gamma)$ is a function with parameters γ

Examples: $b(x, \gamma) = \dfrac{1}{1 + e^{-\gamma^T x}}$

$$b(x, \gamma) = \gamma_1 \mathbf{1}\{x_3 \leq \gamma_2\}$$

Algorithm 10.2 *Forward Stagewise Additive Modeling.*

1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to M:
   (a) Compute
   $$\boxed{(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)).}$$
   (b) Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.

Idea: greedily add one function at a time

Boosted Regression Trees: $L(y, f(x)) = y \log(f(x)) + (1 - y) \log(1 - f(x))$

$b(x, \gamma)$: regression trees

Computationally hard to update

Gradient Boosting

Least squares, exponential loss easy. But what about cross entropy?

Algorithm 10.3 *Gradient Tree Boosting Algorithm.*

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2. For $m = 1$ to M:
   (a) For $i = 1, 2, \ldots, N$ compute

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

   (b) Fit a regression tree to the targets r_{im} giving terminal regions $R_{jm},\ j = 1, 2, \ldots, J_m$.

   (c) For $j = 1, 2, \ldots, J_m$ compute

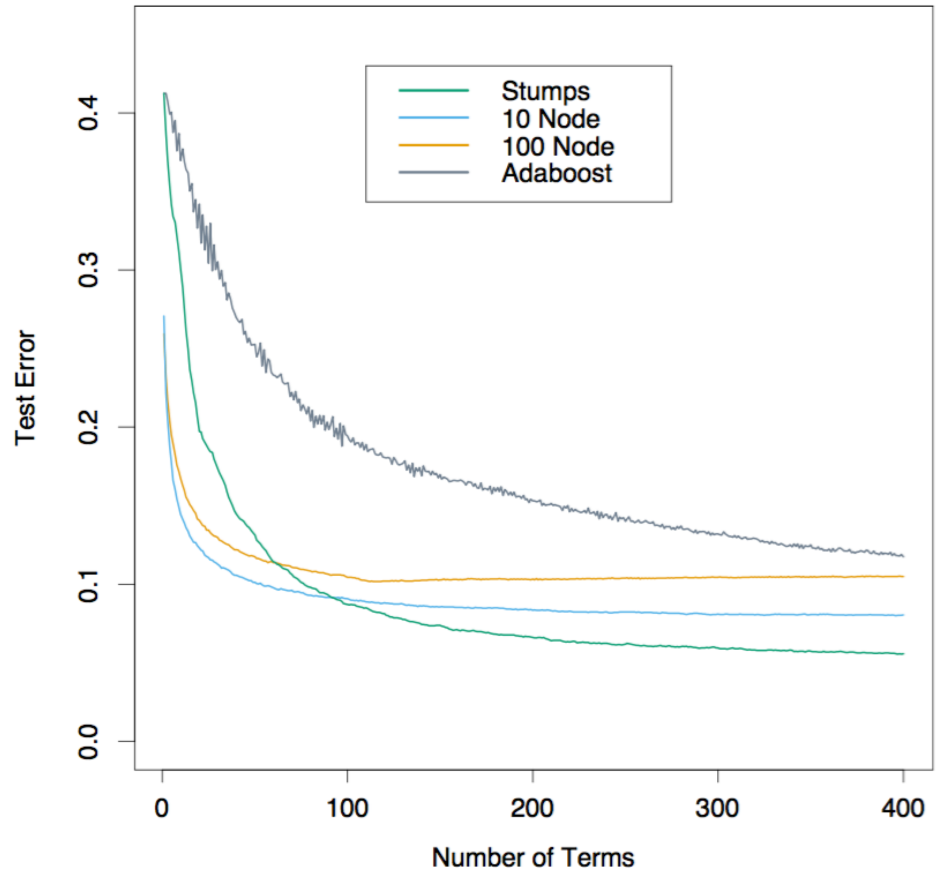
   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$

   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Output $\hat{f}(x) = f_M(x)$.

LS fit regression tree to n-dimensional gradient, take a step in that direction

Gradient Boosting

Least squares, exponential loss easy. But what about cross entropy?



AdaBoost uses 0/1 loss,
all other trees are minimizing
binomial deviance

Additive models

- Boosting is popular at parties: Invented by theorists, heavily adopted by practitioners.

Additive models

- Boosting is popular at parties: Invented by theorists, heavily adopted by practitioners.
- Computationally efficient with “weak” learners. But can also use trees! Boosting can scale.
- Kind of like sparsity?

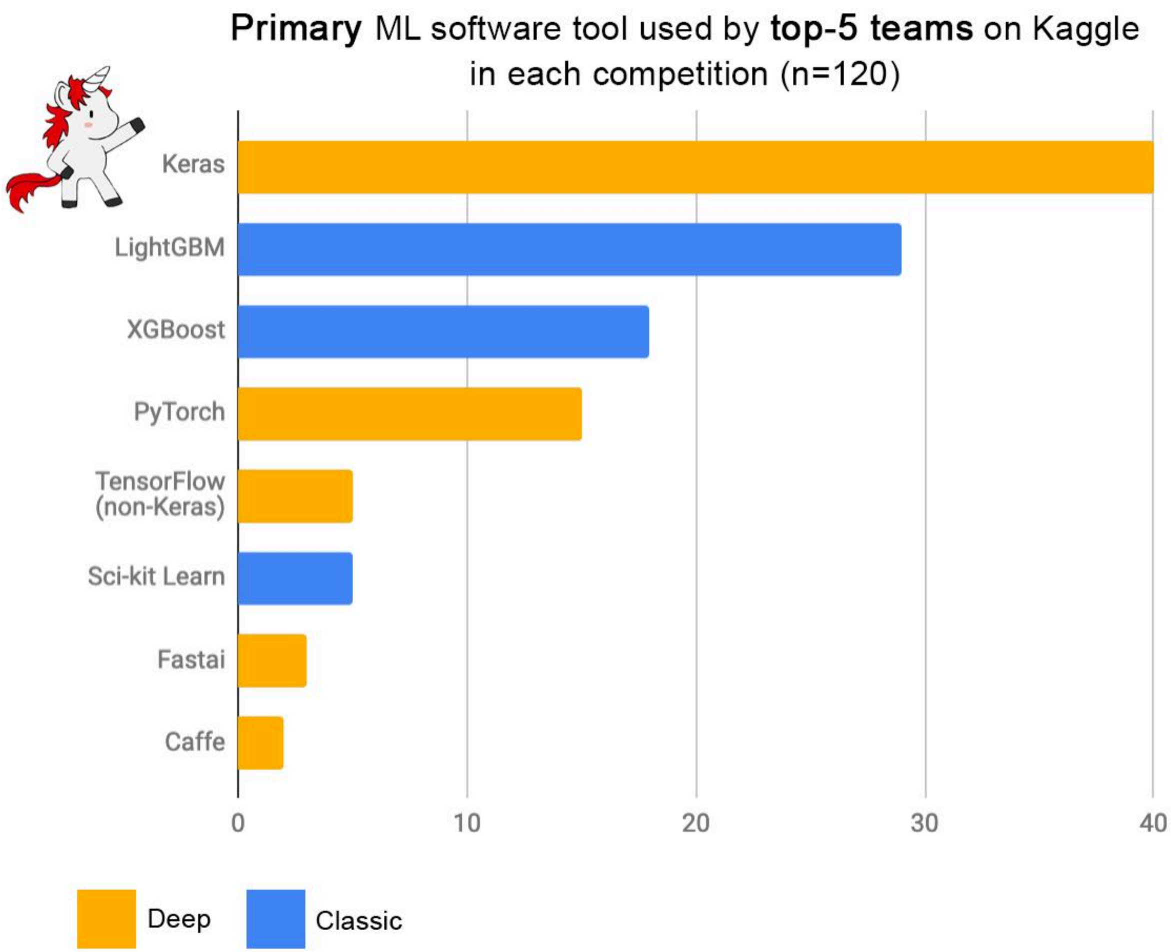
Additive models

- Boosting is popular at parties: Invented by theorists, heavily adopted by practitioners.
- Computationally efficient with “weak” learners. But can also use trees! Boosting can scale.
- Kind of like sparsity?
- Gradient boosting generalization with good software packages (e.g., *XGBoost, LGBM)*. Effective on Kaggle
- Robust to overfitting and can be dealt with with “shrinkage” and “sampling”

Additive models

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 versus Boosting

- Bagging *averages* many **low-bias**, **lightly dependent** classifiers to reduce the variance
- Boosting *learns* linear combination of **high-bias**, **highly dependent** classifiers to reduce error
- Empirically, boosting appears to outperform bagging