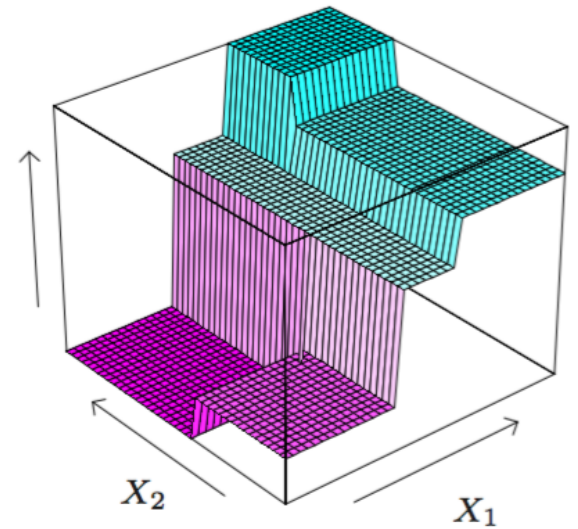
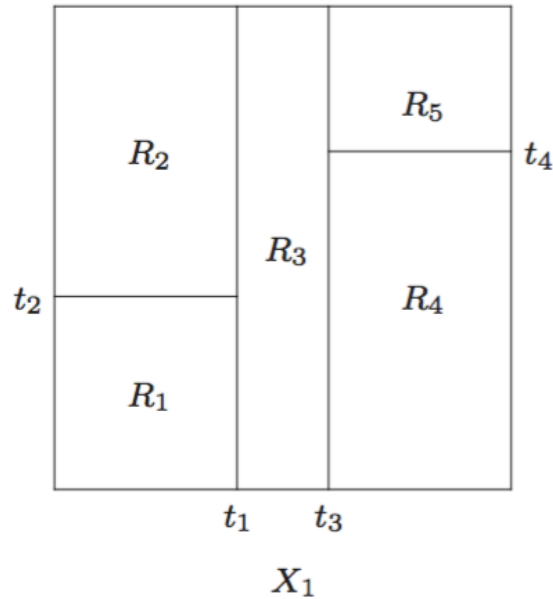
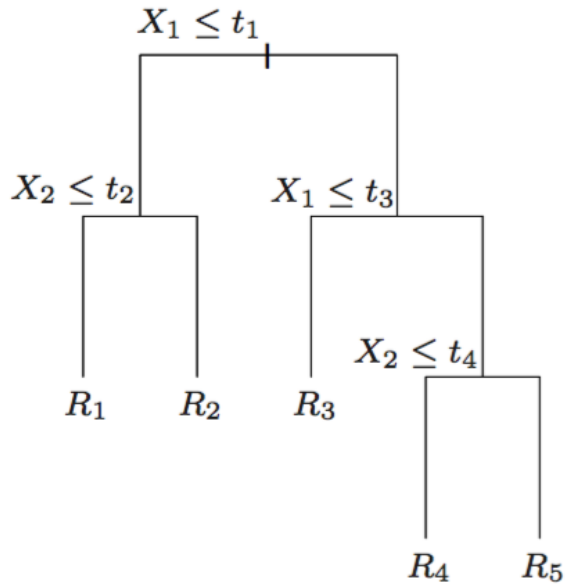


# Trees

---

# Trees

Example: binary tree with splits along axes



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

# Regression Trees

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

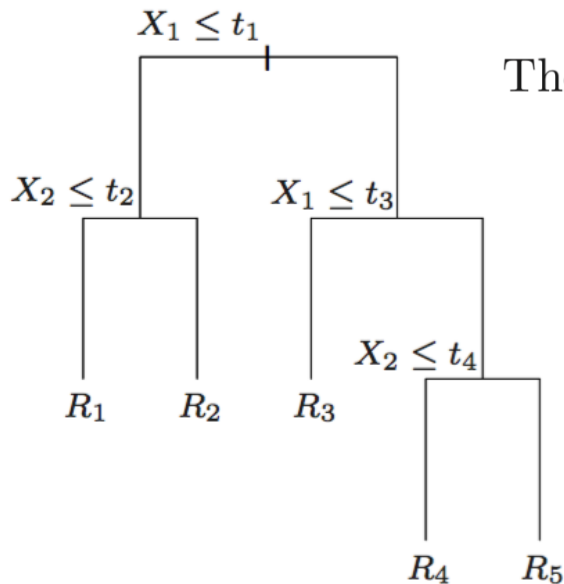
Binary tree with splits along axes.

How do you build the tree / find the splits?

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

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

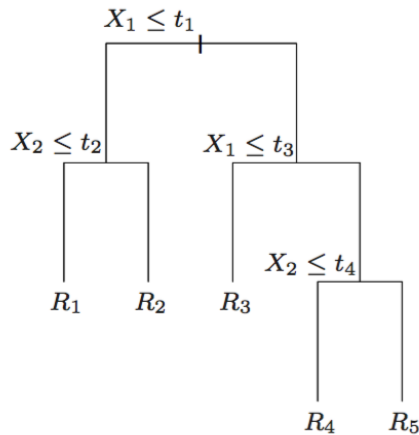
Then we seek the splitting variable  $j$  and split point  $s$  that solve



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

# Learning decision trees

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



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



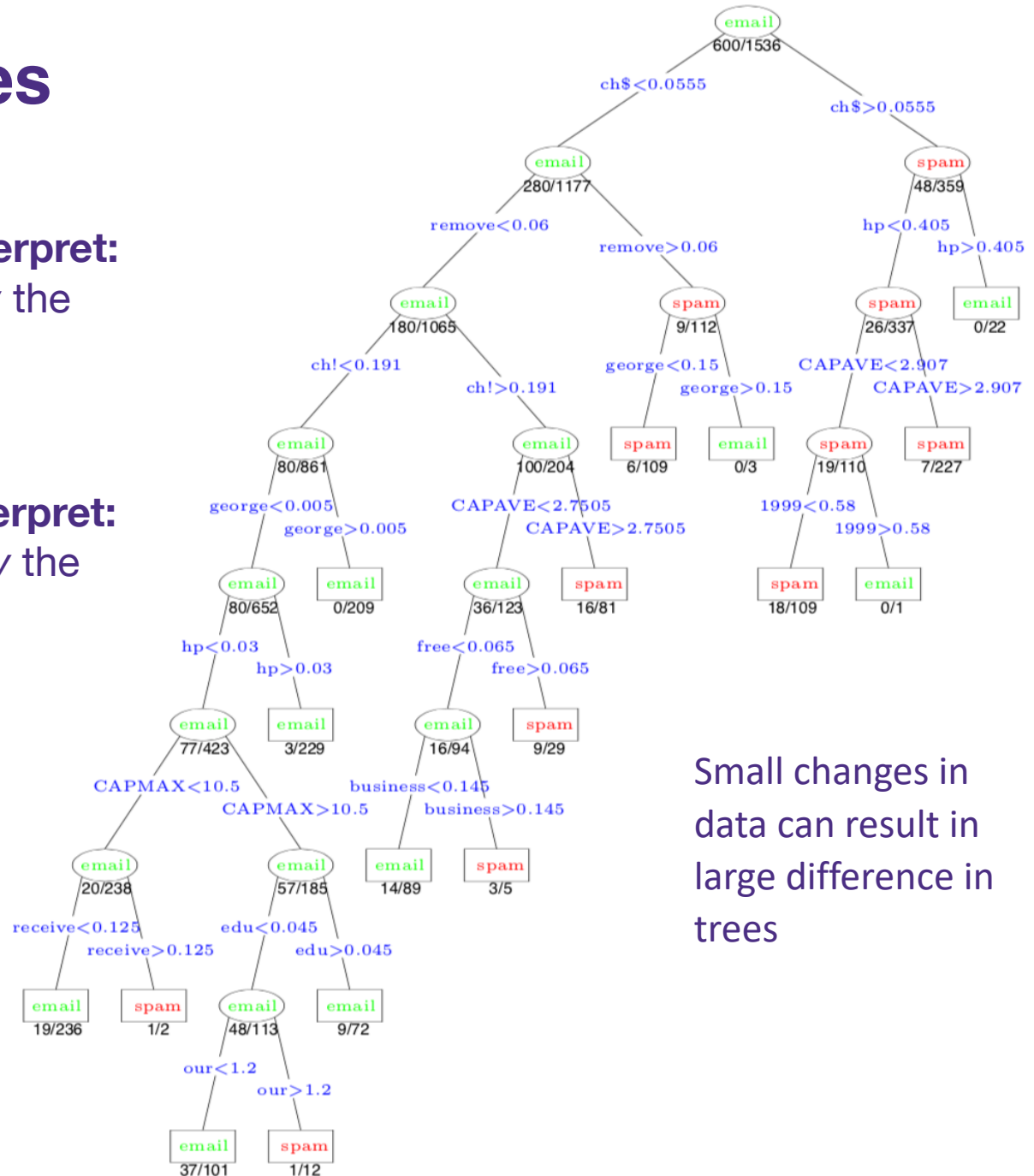
# Decision Trees

## 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

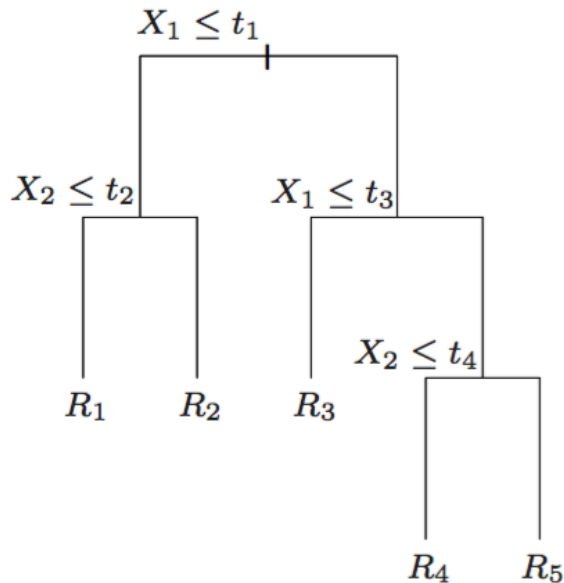


Small changes in data can result in large difference in trees

# Trees

---

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



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

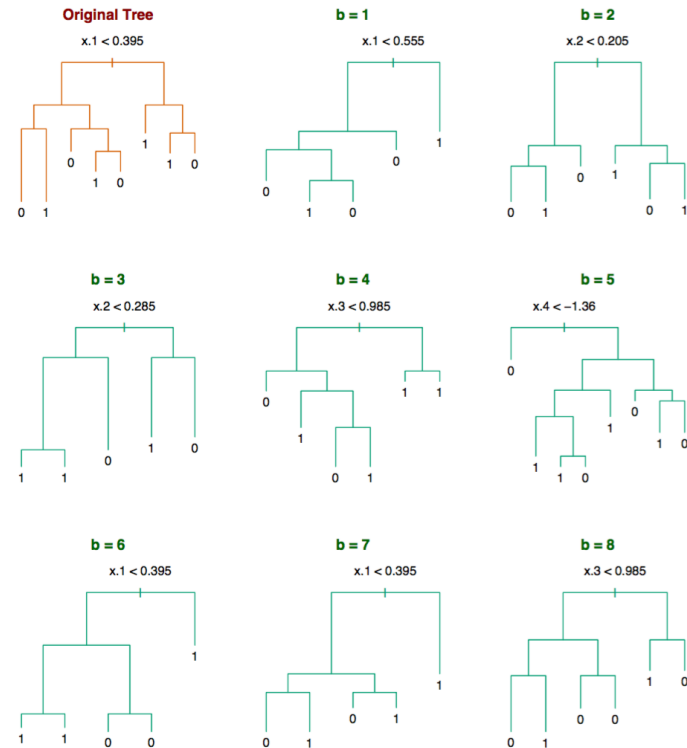
# Random Forests

---

# Random Forests

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 Forests

---

---

**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  $\{T_b\}_1^B$ .

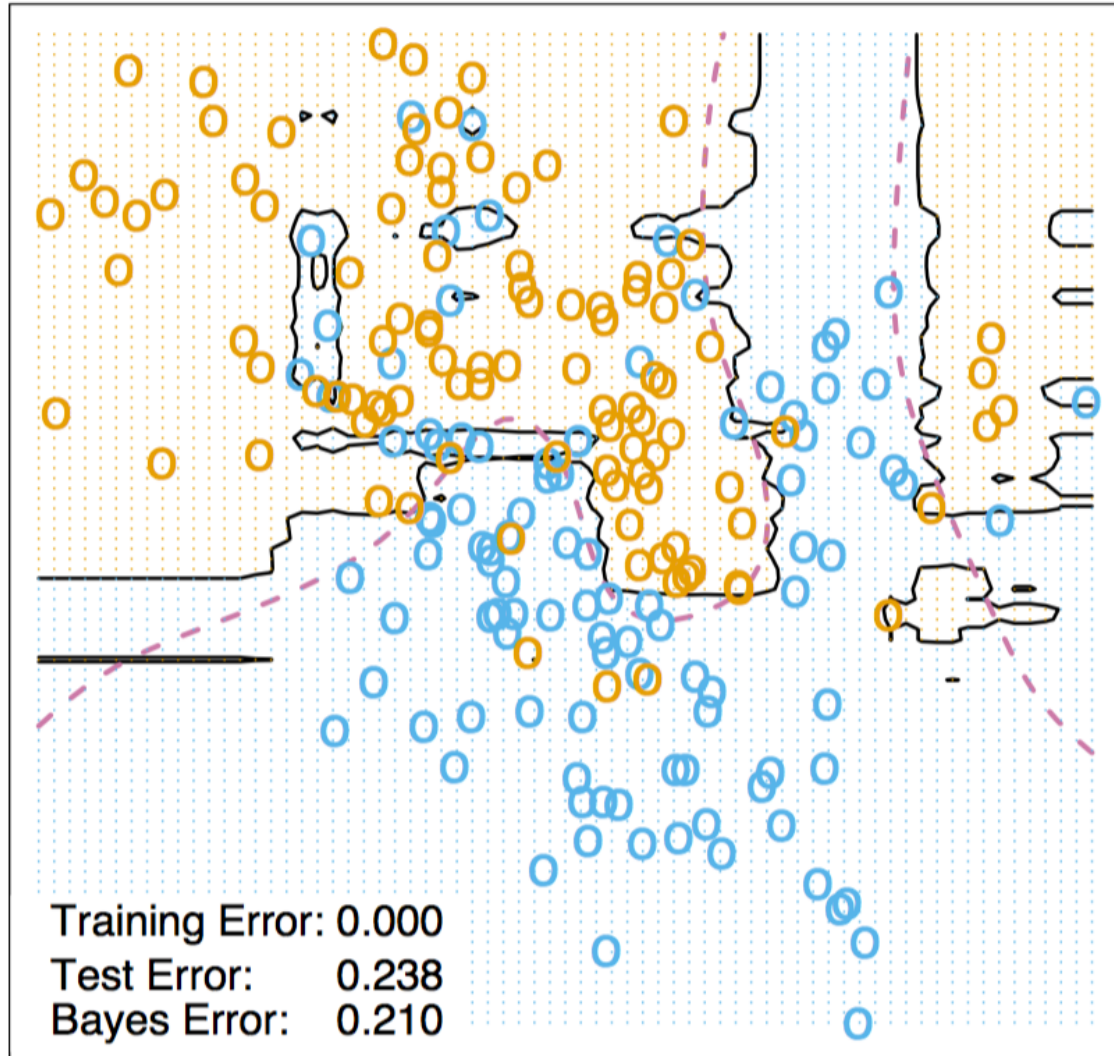
To make a prediction at a new point  $x$ :

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

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

# Random Forest - Decision Boundary Example

---



# Random Forest

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

$\sigma^2$  Variance of individual predictor

Assume bias = 0

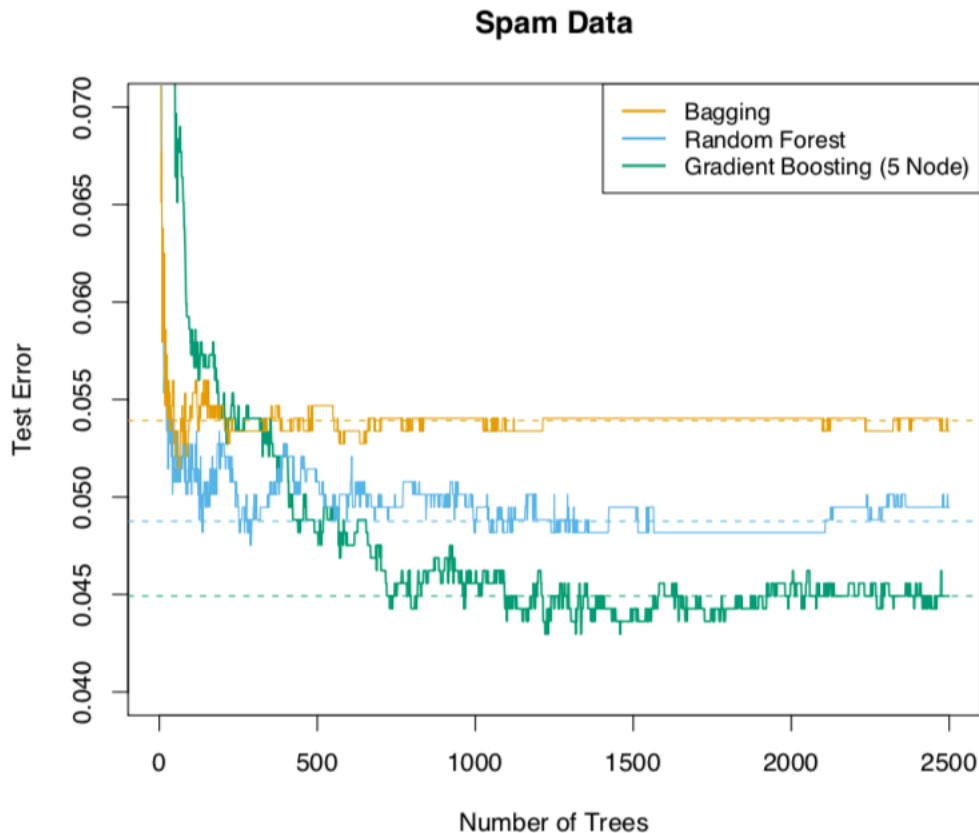
$\rho\sigma^2$  Correlation between predictors

The  $Y_i$ 's are identically distributed but **not** independent

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

# 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 used  **$m < d$  random variables**

Gradient boosting: ignore for now

**Takeaway: reducing correlation improves performance!**

# Random Forests

---

- Random Forests
  - **have low bias, low variance**
  - deal with categorical variables well
  - not that intuitive or interpretable
  - Notion of confidence estimates
  - good software exists
  - Some theoretical guarantees
  - **works well with default hyperparameters**

# Boosting and Additive Models

---

# 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

# Additive models

---

- **Given:**  $\{(x_i, y_i)\}_{i=1}^n$   $x_i \in \mathbb{R}^d, y_i \in \{-1, 1\}$
- **Generate random functions:**  $\phi_t : \mathbb{R}^d \rightarrow \mathbb{R}$   $t = 1, \dots, p$
- **Learn some weights:**  $\hat{w} = \arg \min_w \sum_{i=1}^n \text{Loss} \left( y_i, \sum_{t=1}^p w_t \phi_t(x_i) \right)$
- **Classify new data:**  $f(x) = \text{sign} \left( \sum_{t=1}^p \hat{w}_t \phi_t(x) \right)$

# Additive models

- **Given:**  $\{(x_i, y_i)\}_{i=1}^n$   $x_i \in \mathbb{R}^d, y_i \in \{-1, 1\}$
- **Generate random functions:**  $\phi_t : \mathbb{R}^d \rightarrow \mathbb{R}$   $t = 1, \dots, p$
- **Learn some weights:**  $\hat{w} = \arg \min_w \sum_{i=1}^n \text{Loss} \left( y_i, \sum_{t=1}^p w_t \phi_t(x_i) \right)$
- **Classify new data:**  $f(x) = \text{sign} \left( \sum_{t=1}^p \hat{w}_t \phi_t(x) \right)$

An interpretation:

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

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

is in general computationally hard

# Forward Stagewise Additive models

---

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

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

---

**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

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

# Forward Stagewise Additive models

---

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

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

---

**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  $\gamma$

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

---

**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  $\gamma$       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:**  $L(y, f(x)) = (y - f(x))^2$

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

Efficient: No harder than learning regression trees!

# 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.
- Gradient boosting generalization with good software packages (e.g., *XGBoost*). Effective on Kaggle

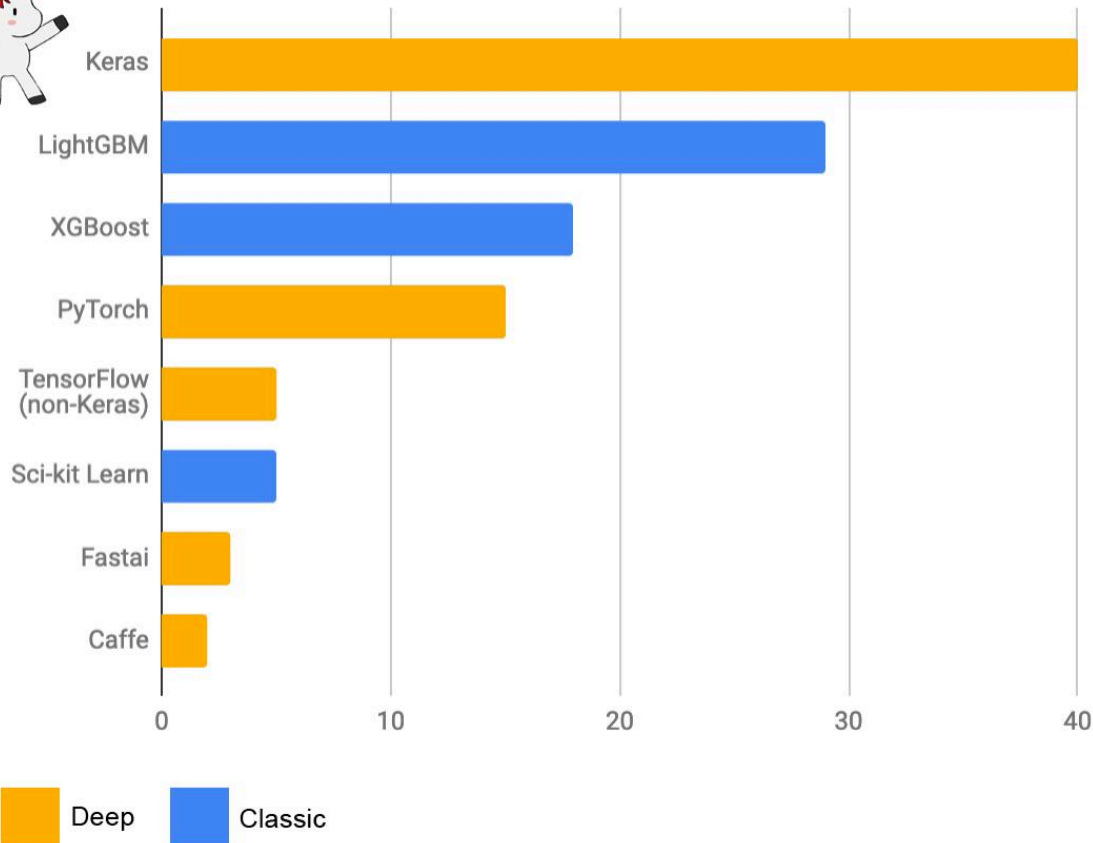
# Additive models



**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.

**Primary ML software tool used by top-5 teams on Kaggle in each competition (n=120)**



# 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