Random Forests

Pattern Recognition & Machine Learning Course, EPFL

December 2015

Carlos Becker

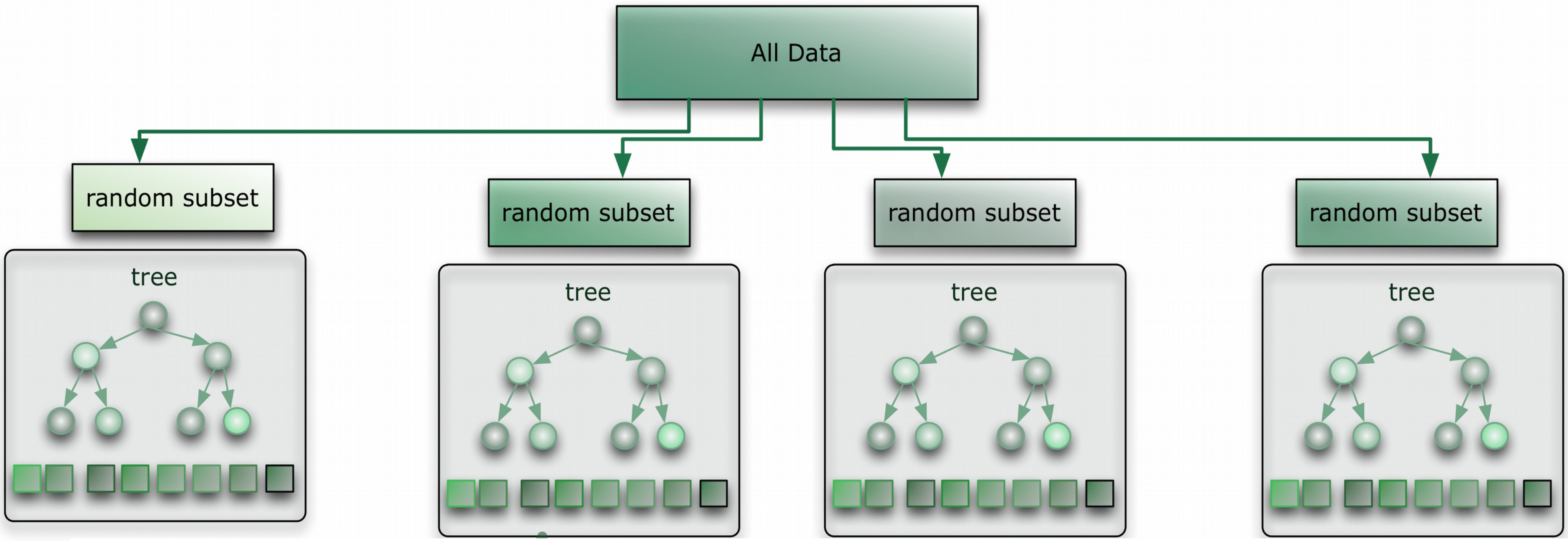
Quick overview

Overview

Trees are very flexible models

… but they may lead to overfitting (high variance)

Quick Look at a Random Forest



[citizennet.com]

- Training: Learn M trees on different subsets of training data
- Prediction: Average of prediction of each tree

Model Averaging

Key Concept: Model Averaging

We can learn multiple predictors

$f_1, f_2, \ldots, f_M$: predictions of M different models we trained

If we take the f_i to be identically distributed, with

$$V(f_i) = \mathbb{E}\left[f_i^2\right] = \sigma^2$$

$$C(f_i, f_j) = \mathbb{E}\left[f_i\, f_j\right] = \rho\, \sigma^2 \qquad \text{if } i \neq j$$

Single Predictor

$$z_1 = f_1$$

$$V(z_1) = \sigma^2$$

Averaged Predictor

$$z_M = \frac{1}{M} \sum_{i=1}^{M} f_i$$

$$V(z_M) = \tfrac{1}{M}\sigma^2 + \rho\, \tfrac{M-1}{M}\sigma^2$$

Model Averaging

Key Concept: Model Averaging

Single Predictor

$$z_1 = f_1$$

$$V(z_1) = \sigma^2$$

Averaged Predictor

$$z_M = \frac{1}{M} \sum_{i=1}^{M} f_i$$

$$V(z_M) = \tfrac{1}{M}\sigma^2 \; + \; \rho \, \tfrac{M-1}{M}\sigma^2$$

Variance reduction ratio: $\dfrac{V(z_1)}{V(z_M)} = \dfrac{M}{1 + \rho\,(M-1)}$

Model Averaging

Key Concept: Model Averaging

$$\text{Variance reduction ratio:} \quad \frac{V(z_1)}{V(z_M)} = \frac{M}{1+\rho\,(M-1)}$$

$$\text{If} \quad M \to \infty \quad \text{then} \quad \frac{V(z_1)}{V(z_M)} \to \frac{1}{\rho}$$

Therefore, if we use model averaging we want

- Large number M of predictors
- Low correlation between them

Model Averaging

Random Forests

Train an ensemble of trees on the training data

- Provide a mechanism to help decorrelate trees

  $\longrightarrow$ reduce prediction variance

- Output is average of all trees

Model Averaging

Random Forests – 'Just' a mix of two procedures

Decorrelating trees

1) Randomize training data: Bagging
2) Randomize feature space: Randomized feature selection

Model Averaging

Random Forests

Decorrelating trees

1) Randomize training data: Bagging

Bagging Model Construction

Input: Training samples $X = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$

Predictor learning procedure $\mathcal{L} : \{\mathcal{X}\} \to \mathcal{H}$ (e.g. tree learning function)

Number of learners M

1: **for** $i = 1$ to M **do**

2: Generate X^i by randomly sampling N samples with replacement from X

3: Learn $f_i(\cdot) = \mathcal{L}(X^i)$

4: **end for**

5: **return** prediction function $z(\mathbf{x}) = \frac{1}{M} \sum_{i=1}^{M} f_i(\mathbf{x})$

Model Averaging

Random Forests

Decorrelating trees

1) Randomize training data: **Bagging**

Generate X^i by randomly sampling N samples with replacement from X

- Also known as *Bootstrapping*: simulates different draws of data from the original training data.
- Probability of choosing a sample at least once = 63%

Model Averaging

Random Forests

Decorrelating trees

1) Randomize <u>training data</u>: Bagging

Generate X^i by randomly sampling N samples with replacement from X

- Also known as *Bootstrapping*: simulates different draws of data from the original training data.
- Probability of choosing a sample at least once = 63%

Example: all training data $X = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5\}$

$\longrightarrow$ Bootstrapped sets: $\{\mathbf{x}_1, \mathbf{x}_5, \mathbf{x}_5, \mathbf{x}_3, \mathbf{x}_5\}$ $\{\mathbf{x}_4, \mathbf{x}_1, \mathbf{x}_5, \mathbf{x}_5, \mathbf{x}_4\}$

$\{\mathbf{x}_4, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_1\}$ $\{\mathbf{x}_5, \mathbf{x}_4, \mathbf{x}_2, \mathbf{x}_5, \mathbf{x}_1\}$

Model Averaging

Random Forests – 'Just' a mix of two procedures

Decorrelating trees

1) Randomize <u>training data</u>: Bagging
2) Randomize <u>feature space</u>: Randomized feature selection

Model Averaging

Random Forests

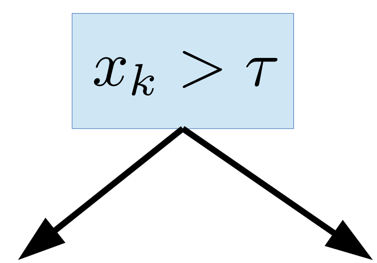
Decorrelating trees

2) Randomize <u>feature space</u>: Randomized feature selection

When learning a split:

instead of searching for k over all D possible features

$\longrightarrow$ search on a reduced random subset



Model Averaging

Random Forests

Decorrelating trees

2) Randomize <u>feature space</u>: Randomized feature selection

---

Learn split on training data X, with random subspace search

Input: Training samples $X = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}, \quad \mathbf{x}_i \in \mathbf{R}^D$

Number of features to search $m_{\text{try}} \leq D$

1: $Q =$ sample m_{try} values without replacement from $\{1...D\}$

2: **for** $k \in Q$ **do**

3: Find best split for feature k: $\tau_k^* = \underset{\tau}{\operatorname{argmin}}\, I_{\text{split}}(X, k, \tau)$

4: Compute cost of this split: $I_k = I_{\text{split}}(X, k, \tau^*)$

5: **end for**

6: **return** k and τ_k that got the minimum impurity $I_{\text{split}}(\cdot)$

Model Averaging

Random Forests – 'Just' a mix of two procedures

Decorrelating trees

1) Randomize training data: Bagging
2) Randomize feature space: Randomized feature selection

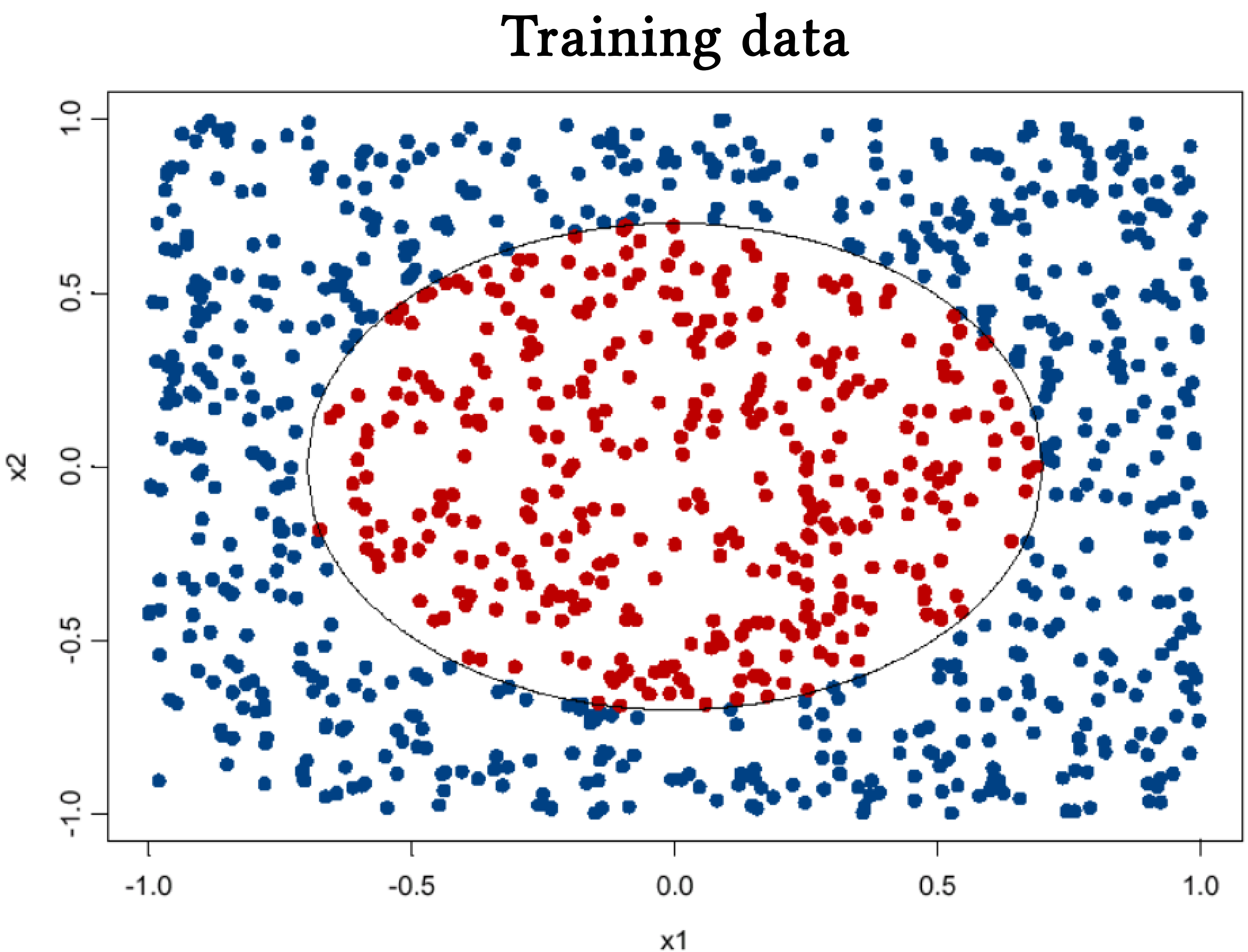
That's all RFs are about.

Therefore, the parameters of a RF are:

- Maximum tree depth
- Number of trees (Forest size)
- Value of m_{try}. Typically m_{try} = sqrt(D)

Model Averaging

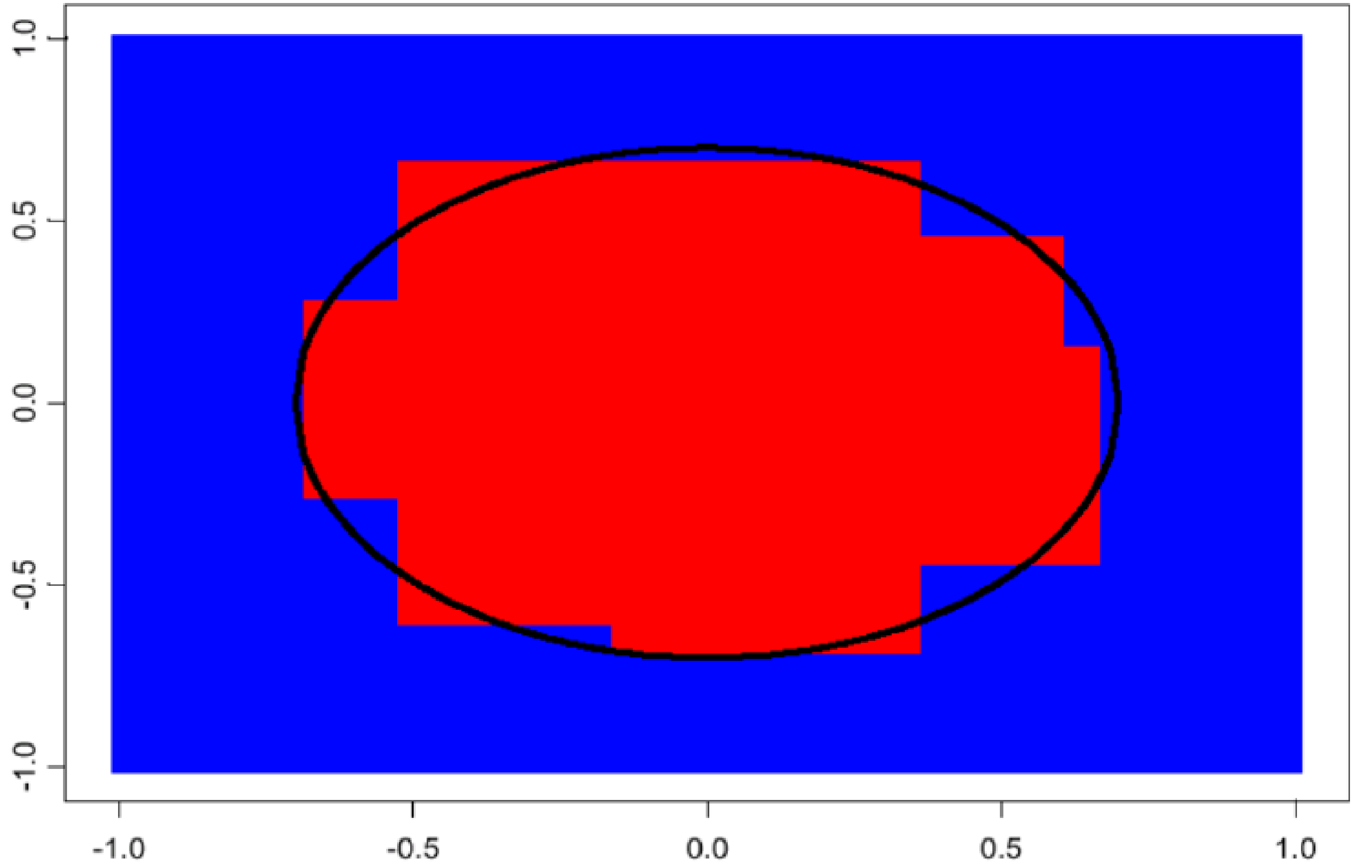
Random Forests – Toy example [from Jessie Li's slides from Penn State University]



Model Averaging

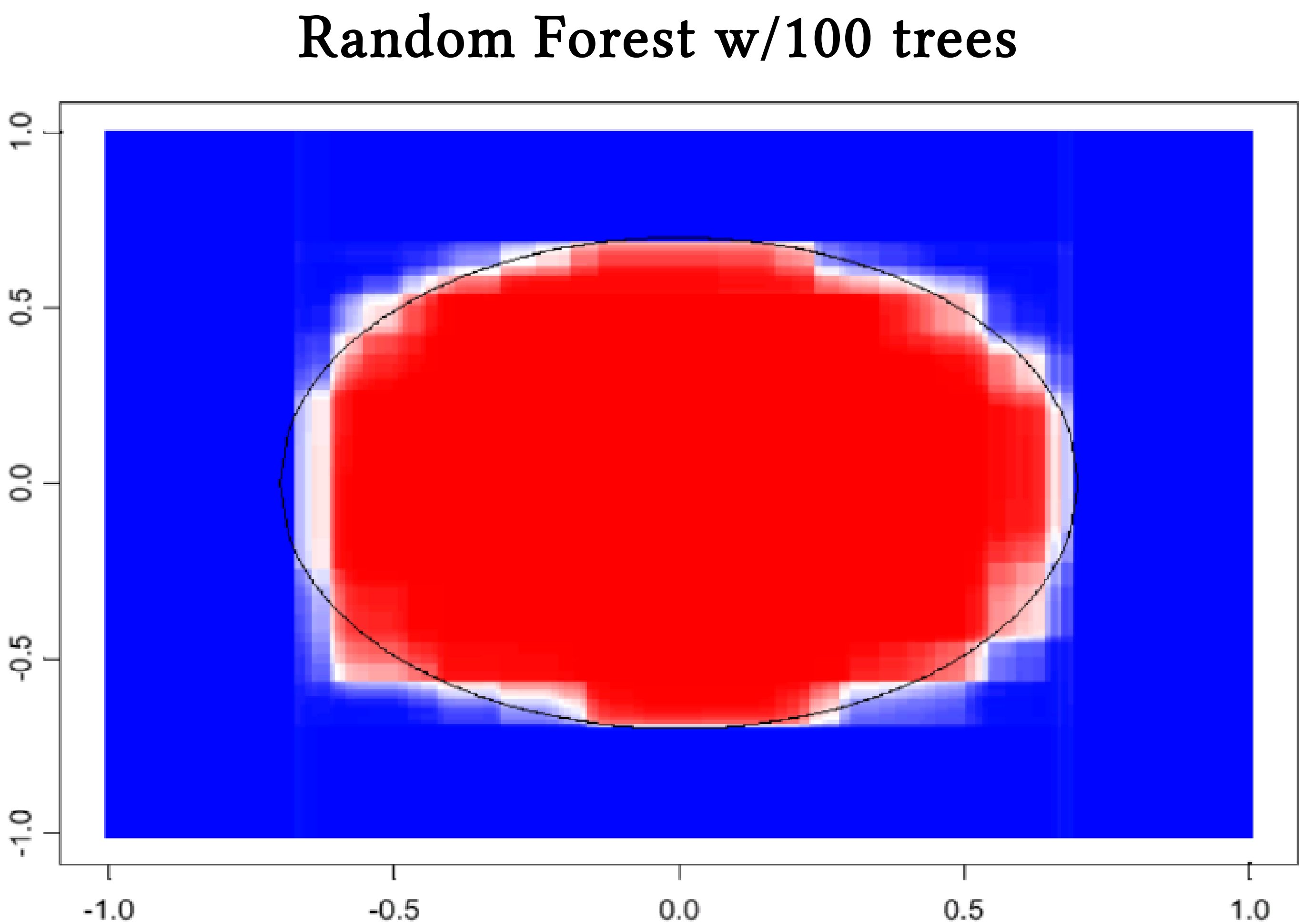
Random Forests – Toy example [from Jessie Li's slides from Penn State University]

Single Decision Tree Prediction



Model Averaging

Random Forests – Toy example [from Jessie Li's slides from Penn State University]



Random Forests – 'Just' a mix of two procedures

Effect of m_{try}

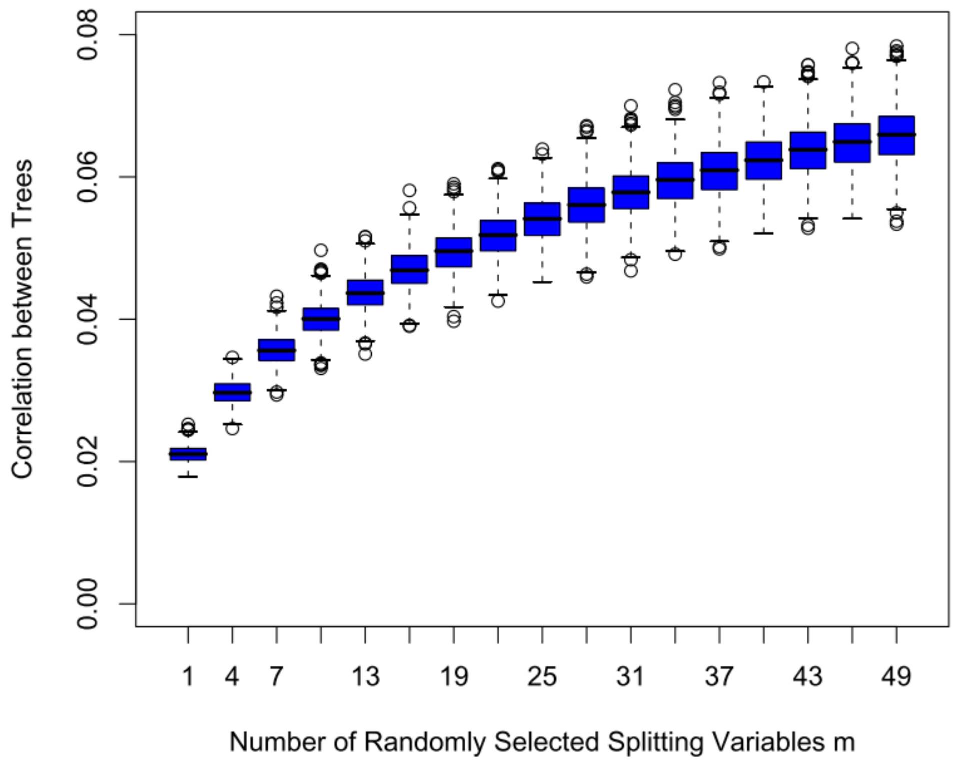


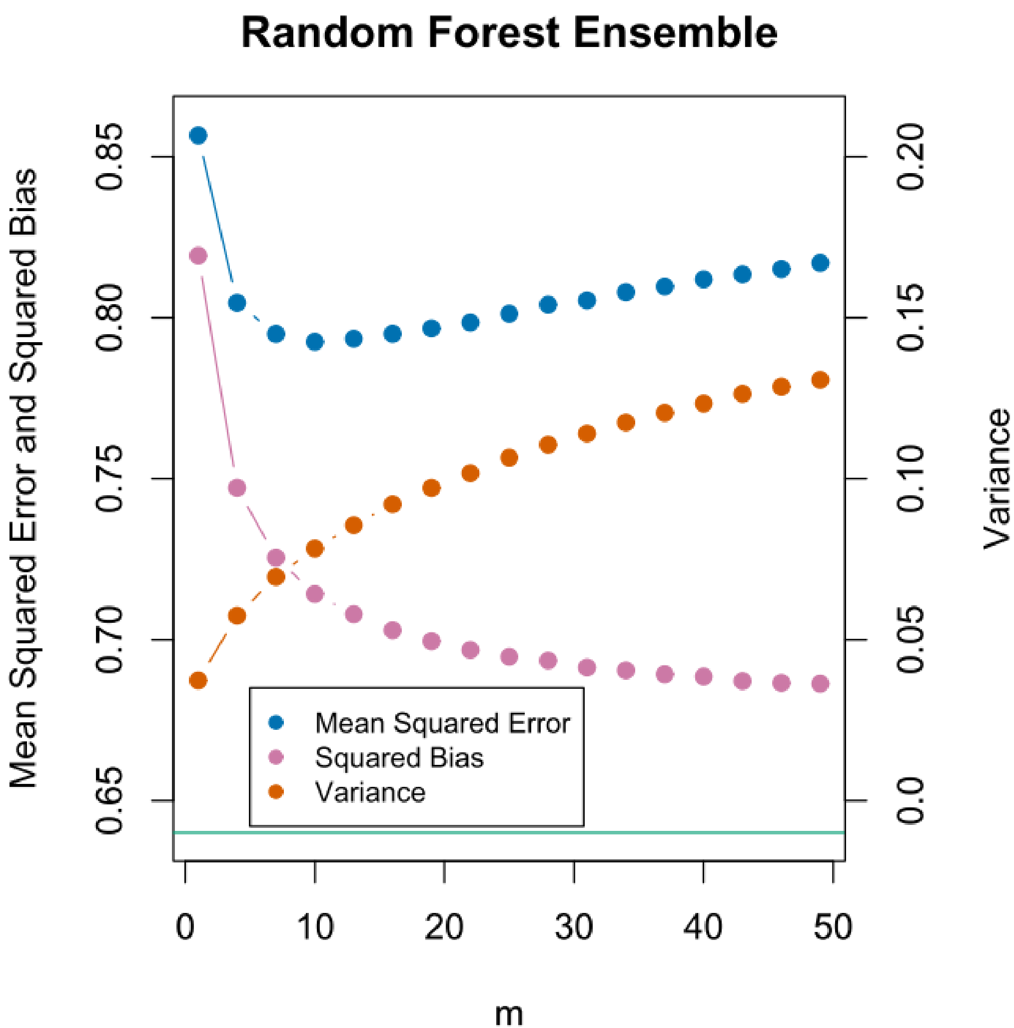
FIGURE 15.9. *Correlations between pairs of trees drawn by a random-forest regression algorithm, as a function of m. The boxplots represent the correlations at 600 randomly chosen prediction points x.*

[HTF]

Model Averaging

Random Forests – 'Just' a mix of two procedures

Effect of m_{try}



[HTF]

Model Averaging

Conclusion

Model Averaging aims at reducing variance through averaging

- Random Forests is one example, which trains multiple trees
  - Bagging & random sub-space search improves stability

- If you are interested, a few more cool things about RFs:
  - Out-of-bag examples/cross-validation [HTF 15.3.1]
  - Computing Variable/Feature Importance [HTF 15.3.2]
  - Partial Dependency Plots [HTF 15.4.3]
  - Adaptive Nearest Neighbors [HTF 10.13.2]