# Learning from trees: growing boosted trees with XGBoost

Lambert Moyon

Machine learning Journal Club 10/25/2017

# Reminder: supervised learning

In supervised learning, you want to learn the **relationship** between a set of input objects, described by their **features**, and their **label**.

Either a class (classification),
or a value (regression)

# Reminder: supervised learning

In supervised learning, you want to learn the **relationship** between a set of input objects, described by their **features**, and their **label**.

Either a class (classification),
or a value (regression)

Many applications, from SPAM classifier, to fraud detection in bank transactions, or sales predictions for a company.

# Supervised learning: a decision tree

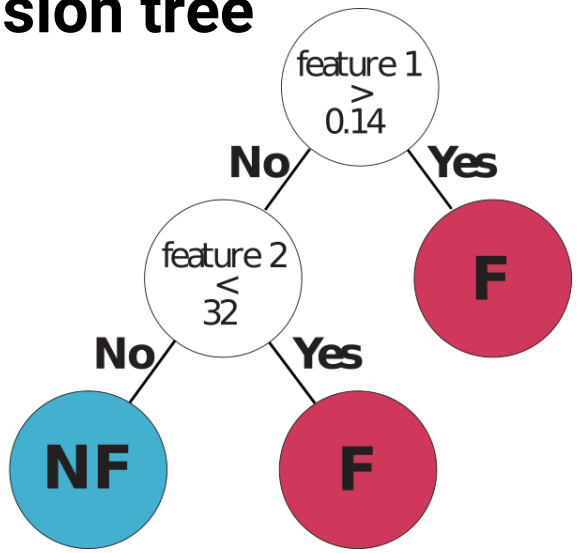
Conservation score

mutation

Dataset

| index | feature 1 | feature 2 | class |
|---|---|---|---|
| #1 | 0.12 | 51 | Functional |
| #2 | 0.09 | 91 | Functional |
| #3 | 0.11 | 76 | Functional |
| #4 | 0.10 | 13 | Non Functional |
| #5 | 0.16 | 67 | Non Functional |
| #6 | 0.18 | 47 | Non Functional |

**A decision tree**



# Supervised learning: a decision tree

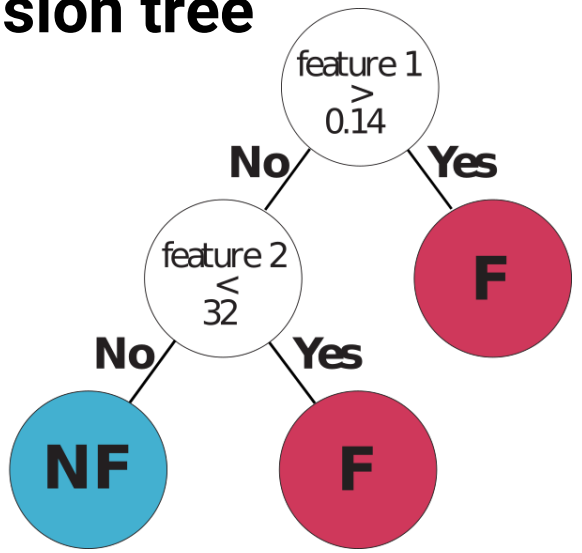
Conservation score

mutation

Dataset

| index | feature 1 | feature 2 | class |
|---|---|---|---|
| #1 | 0.12 | 51 | Functional |
| #2 | 0.09 | 91 | Functional |
| #3 | 0.11 | 76 | Functional |
| #4 | 0.10 | 13 | Non Functional |
| #5 | 0.16 | 67 | Non Functional |
| #6 | 0.18 | 47 | Non Functional |

**A decision tree**




```
if node is not "pure":
  Select feature and threshold
  that separate in the best way
  the samples

  Add node to the tree with the
  selected threshold.

  Separate samples in the child
  nodes

else:
  Set as leaf
```

# Supervised learning: a decision tree

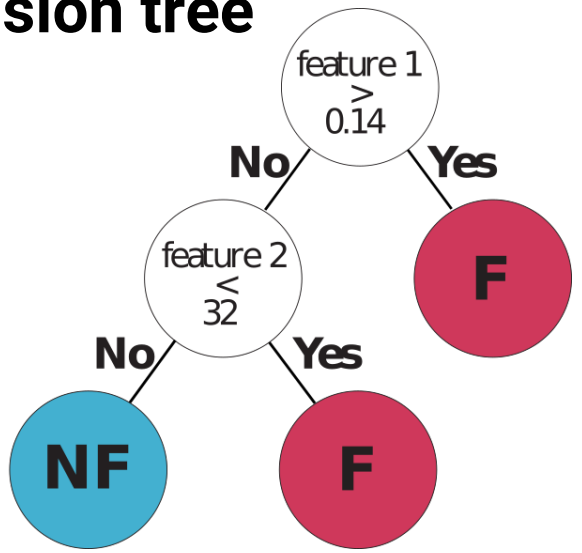
Conservation score

mutation

Dataset

| index | feature 1 | feature 2 | class |
|---|---|---|---|
| #1 | 0.12 | 51 | Functional |
| #2 | 0.09 | 91 | Functional |
| #3 | 0.11 | 76 | Functional |
| #4 | 0.10 | 13 | Non Functional |
| #5 | 0.16 | 67 | Non Functional |
| #6 | 0.18 | 47 | Non Functional |

**A decision tree**



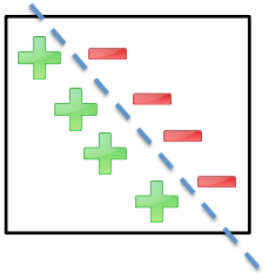

```
if node is not "pure":
  Select feature and threshold
  that separate in the best way
  the samples

  Add node to the tree with the
  selected threshold.

  Separate samples in the child
  nodes

else:
  Set as leaf
```

Decision Trees are AXIS-ALIGNED!

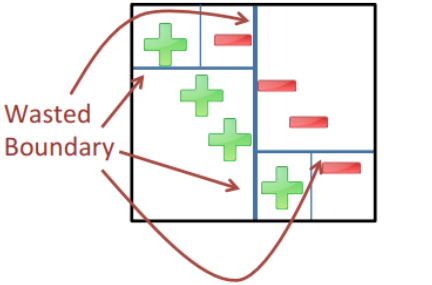
– Cannot easily model diagonal boundaries

Example:

Simple Linear SVM can Easily Find Max Margin



Decision Trees Require Complex Axis-Aligned Partitioning



# How to grow a decision tree

The impurity reduction will be our goal when growing the tree: we want to find the tree with the lowest impurity over its nodes.

# How to grow a decision tree

The impurity reduction will be our goal when growing the tree: we want to find the tree with the lowest impurity over its nodes.

Two measures are usually used for classification:

- **Entropy:**

$$L(Node) = -|Node| \, (p_{c1} \log(p_{c1}) + p_{c2} \log(p_{c2}))$$

**⇒Information Gain from a split**

$$IG(A,B/Node) = L(Node) - L(A) - L(B)$$

- **Gini Index:**

$$L(Node) = |Node| \, (1 - \Sigma \, p^2)$$

# How to grow a decision tree

**Stopping conditions**

- Minimum number of samples per child node
- Maximum depth
- Maximum number of nodes
- Minimum reduction in impurity

# “A single tree does not make a forest”

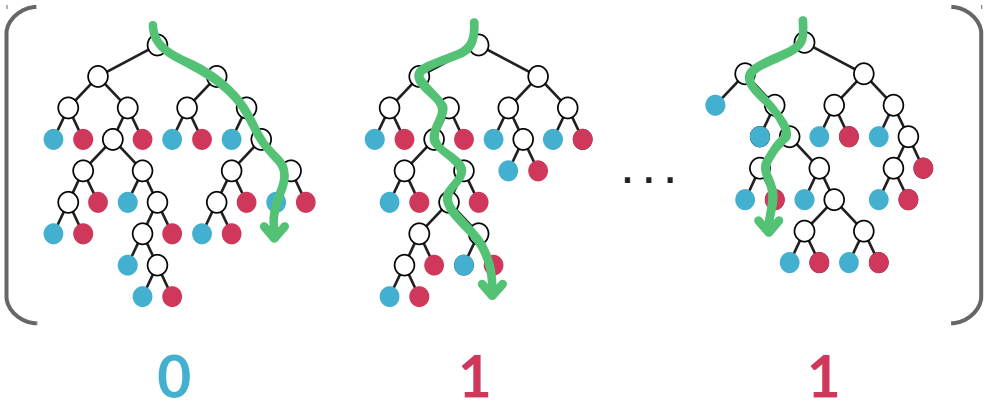
A single decision tree will correctly learn to predict from the dataset its given...
... But will most likely fail when given a new dataset

# “A single tree does not make a forest”

A single decision tree will correctly learn to predict from the dataset its given...
... But will most likely fail when given a new dataset

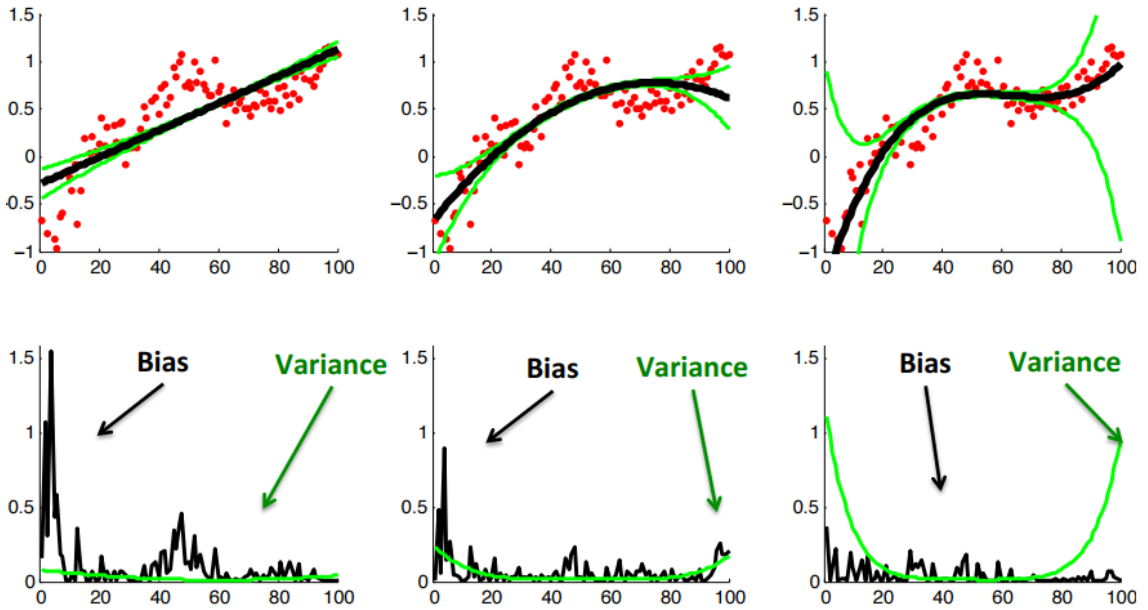
**⇒ The solution is to grow multiple trees!**

To avoid correlation between trees: **boostrap and feature selection**




**Random forest classifier**
The class of a variant is given by the average vote from all trees

# The bias-variance trade-off

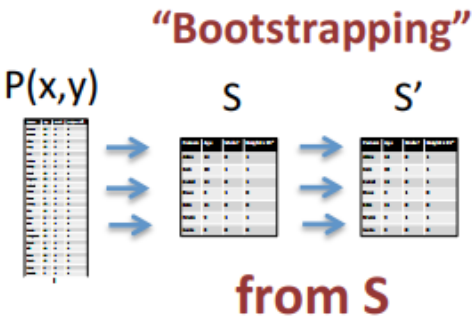



The **bias** is error from **erroneous assumptions in the learning algorithm.** High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).

The **variance** is error from **sensitivity to small fluctuations in the training set.** High variance can cause overfitting: modeling the random noise in the training data, rather than the intended outputs.

# Bias-variance and random forests

## Bagging




- **Goal:** reduce variance
- **In practice:** resample S' with replacement
  - Train model using each S'
  - Average predictions

Variance reduces sub-linearly
(Because S' are correlated)
Bias often increases slightly

$$E_S[(h_S(x) - y)^2] = E_S[(Z-\check{z})^2] + \check{z}^2$$

Expected Error
On single (x,y)

Variance

Bias

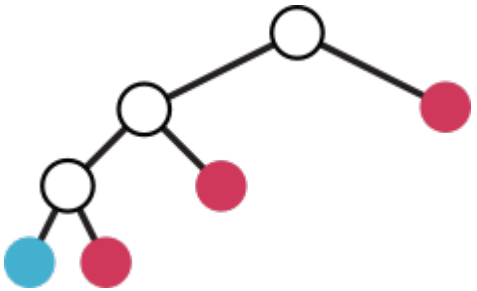
$$Z = h_S(x) - y$$
$$\check{z} = E_S[Z]$$

Bagging = Bootstrap Aggregation

**Bagging Predictors"** [Leo Breiman, 1994]

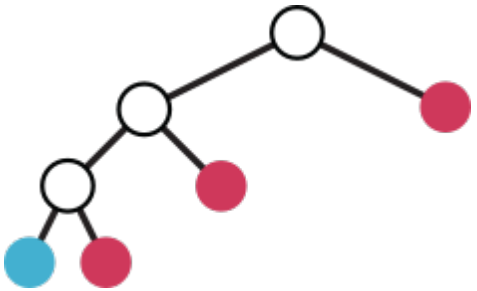
# Boosting trees: working on the bias

Instead of working with a large tree (with high variance), Boosting relies on **shallow trees**, with low variance, but high bias.

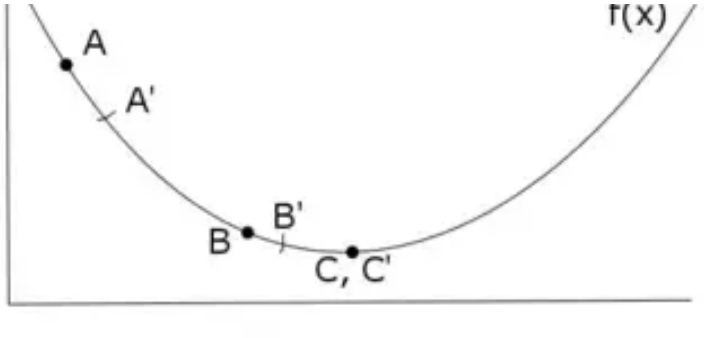


# Boosting trees: working on the bias

Instead of working with a large tree (with high variance), Boosting relies on **shallow trees**, with low variance, but high bias.



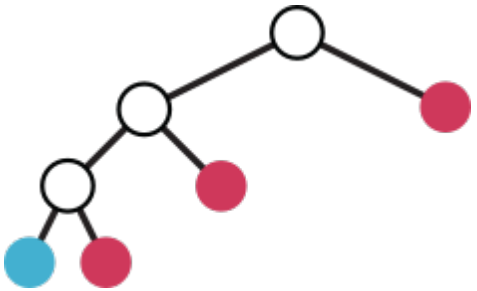
$$x^{(i)} = x^{(i-1)} - \eta \frac{df(x^{(i-1)})}{dx}$$



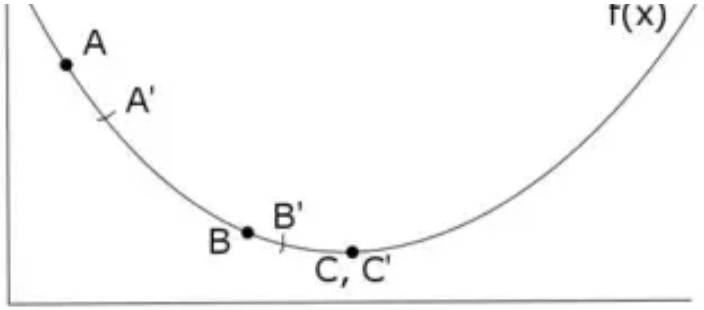

**Gradient descent**: finding the minimum of a function by iteratively updating the coordinates of a point.

# Boosting trees: working on the bias

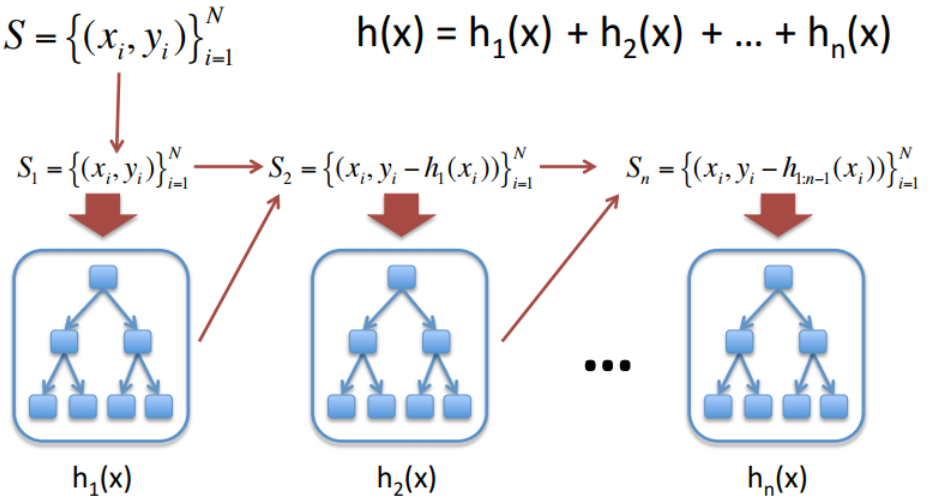
Instead of working with a large tree (with high variance), Boosting relies on **shallow trees**, with low variance, but high bias.



$$x^{(i)} = x^{(i-1)} - \eta \frac{df(x^{(i-1)})}{dx}$$




**Gradient descent**: finding the minimum of a function by iteratively updating the coordinates of a point.




Here, we are minimizing the **Loss function**, not the data.
Each new tree learns from the mistakes of the previous one.

# XGBoost: A Scalable Tree Boosting System

Tianqi Chen
University of Washington
tqchen@cs.washington.edu

Carlos Guestrin
University of Washington
guestrin@cs.washington.edu

# Overview of XGBoost


“Among the 29 challenge winning solutions published at Kaggle’s blog during 2015, 17 solutions used XGBoost.”

# Overview of XGBoost


“Among the 29 challenge winning solutions published at Kaggle’s blog during 2015, 17 solutions used XGBoost.”

**Several algorithmic optimizations:**

- novel tree learning algorithm, able to handle sparse data
- parallel construction of the trees
- out-of-core computation with cache-aware block structure.
- shrinkage (equivalent to a learning rate) that includes a measure of the complexity of the model

**Table 3: Comparison of Exact Greedy Methods with 500 trees on Higgs-1M data.**

| Method | Time per Tree (sec) | Test AUC |
|---|---|---|
| XGBoost | 0.6841 | 0.8304 |
| XGBoost (colsample=0.5) | 0.6401 | 0.8245 |
| scikit-learn | 28.51 | 0.8302 |
| R.gbm | 1.032 | 0.6224 |

# Take-home message

- **Bias-Variance Tradeoff!**
- **Bagging reduces variance of low-bias models**
  - Low-bias models are “complex” and unstable.
  - Bagging averages them together to create stability
- **Boosting reduces bias of low-variance models**
  - Low-variance models are simple with high bias
  - Boosting trains sequence of simple models
  - Sum of simple models is complex/accurate