CS6375: Machine Learning

Gautam Kunapuli

Ensemble Methods: Bagging

THE UNIVERSITY OF TEXAS AT DALLAS
Erik Jonsson School of Engineering and Computer Science

The Bias-Variance Decomposition Revisited

Example: Consider the underlying **true labels and model** $y = h(x)$, and a machine learning model that makes predictions as $f(x)$.

Our goal is to **minimize** the squared loss using a sample S:
$$E\left[\!\left[\left(y - f(x)\right)^2\right]\!\right]$$

Lemma: $Var[\![z]\!] = E[\![(z - E[\![z]\!])^2]\!] = E[\![z^2]\!] - E[\![z]\!]^2$

squared mean, μ^2

$$E\left[\!\left[\left(y - f(x)\right)^2\right]\!\right] = E[\![y^2 - 2yf(x) + f(x)^2]\!]$$

$$= E[\![y^2]\!] - 2E[\![y]\!]E[\![f(x)]\!] + E[\![f(x)^2]\!]$$

using the lemma for first and last terms

$$= Var[\![y^2]\!] + E[\![y]\!]^2 - 2E[\![y]\!]E[\![f(x)]\!] + Var[\![f(x)]\!] + E[\![f(x)]\!]^2$$

using $y = h(x)$, the true model

$$= \epsilon^2 + E[\![h(x)]\!]^2 - 2E[\![h(x)]\!]E[\![f(x)]\!] + E[\![f(x)]\!]^2 + Var[\![f(x)]\!]$$

$$= \epsilon^2 + h(x)^2 - 2h(x)E[\![f(x)]\!] + E[\![f(x)]\!]^2 + Var[\![f(x)]\!]$$

$$= \epsilon^2 + (h(x) - E[\![f(x)]\!])^2 + Var[\![f(x)]\!]$$

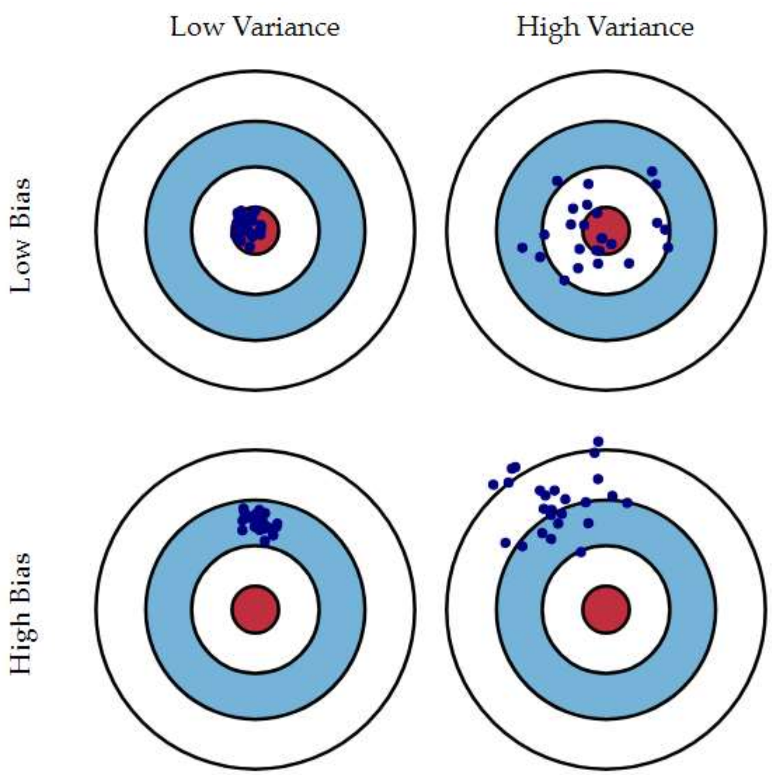
$$= \textbf{noise} + \textbf{bias}^2 + \textbf{variance}$$

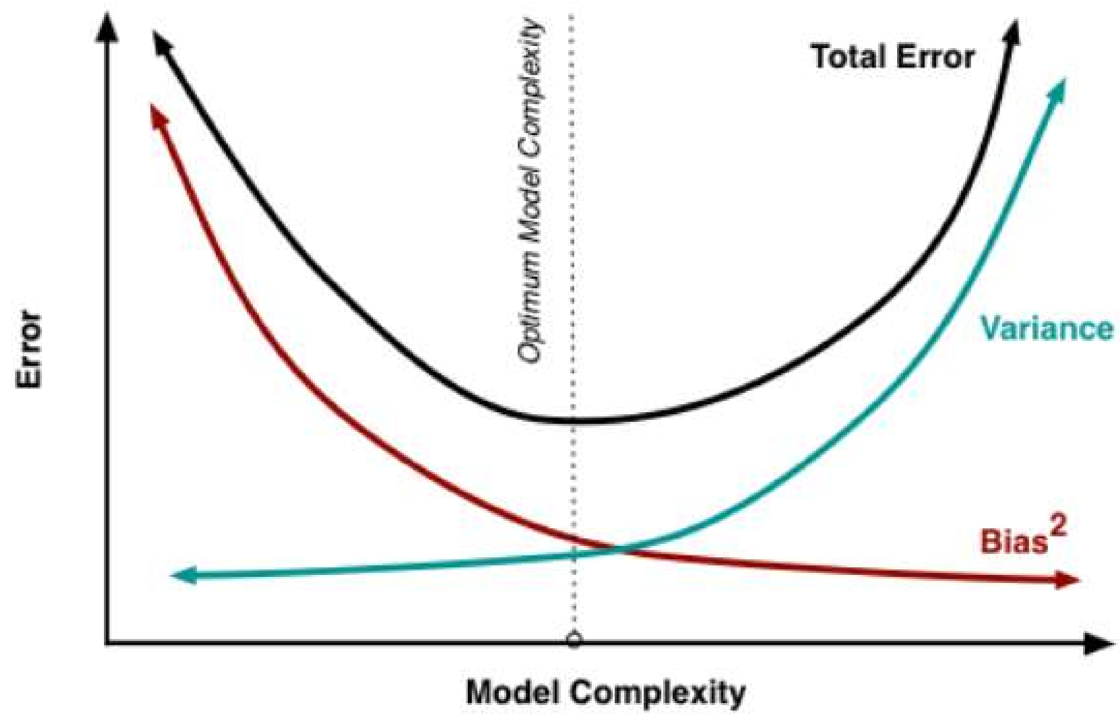
The Bias-Variance Decomposition Revisited

Example: Consider the underlying **true labels and model** $y = h(x)$, and a machine learning model that makes predictions as $f(x)$.

Our goal is to **minimize** the squared loss using a sample S:

$$E\left[\!\left[\left(y - f(x)\right)^2\right]\!\right] = \textbf{noise} + \textbf{bias}^2 + \textbf{variance}$$





The Bias-Variance Decomposition Revisited

Example: Consider the underlying **true labels and model** $y = h(x)$, and a machine learning model that makes predictions as $f(x)$.

Our goal is to **minimize** the squared loss using a sample S:

$$E\left[\!\left[\left(y - f(x)\right)^2\right]\!\right] = \mathbf{noise} + \mathbf{bias}^2 + \mathbf{variance}$$

variance: describes how much $f(x)$ varies from one training set to another
(sensitivity to small fluctuations in the data set)
bias: describes the average error of $f(x)$
(result of erroneous algorithmic assumptions)
noise: describes how much y varies from $h(x)$
(irreducible error on unseen samples)

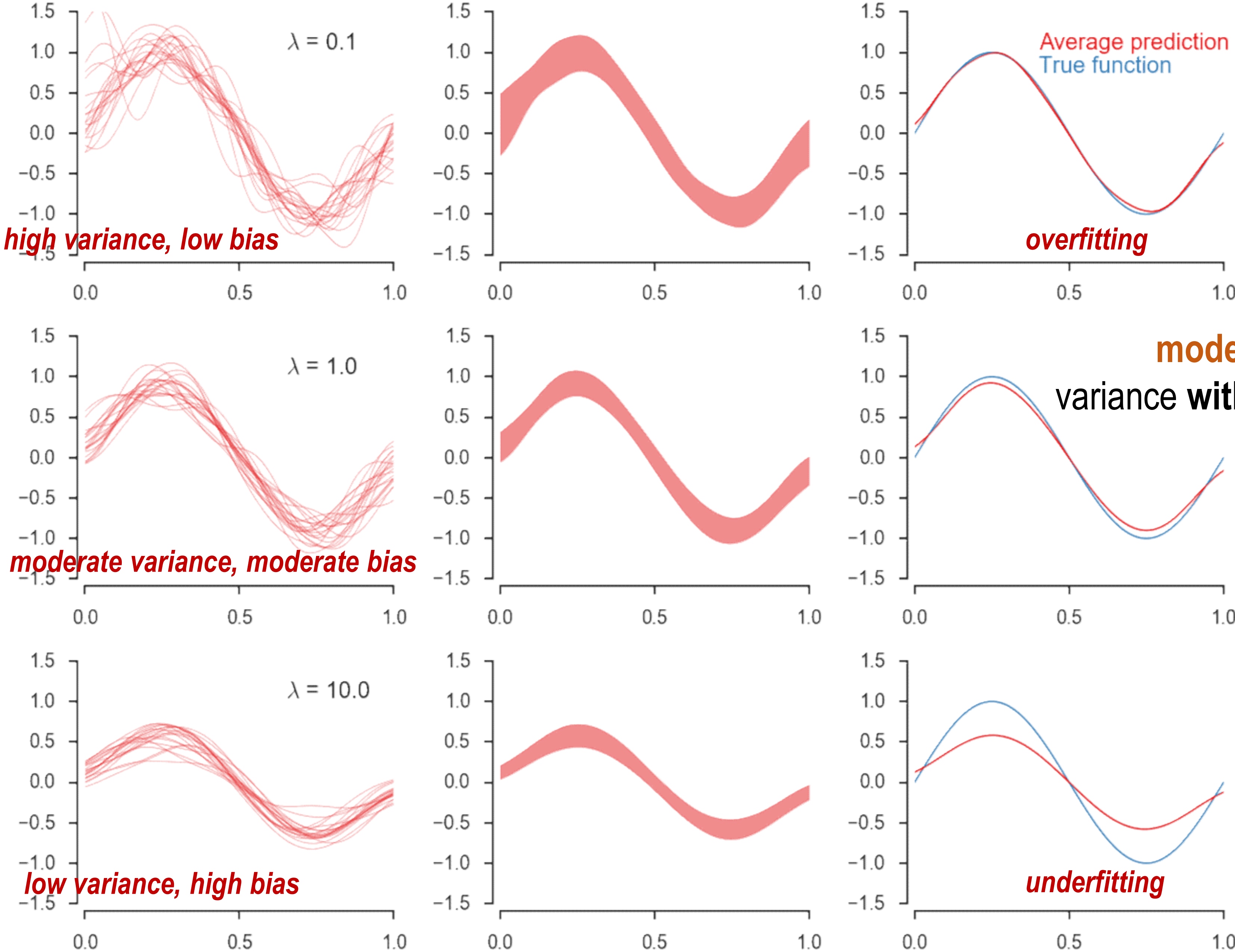
What causes bias?

- Inability to represent certain decision boundaries
  - e.g., linear hyperplanes, Naïve Bayes, decision trees
- Incorrect assumptions
  - e.g, failure of independence assumption in naïve Bayes
- Classifiers that are "too global" (or too smooth)
  - for example, a single linear separator, a small decision tree, a large number of nearest neighbors
- If the bias is high, the model is **underfitting** the data

What causes variance?

- Making decision based on small subsets of the data
  - e.g., decision tree splits near the leaves
- Computational reasons
  - e.g., randomization in the learning algorithm such as bad initial weights in gradient descent
- Classifiers that are "too local" (or too nonlinear) and can easily fit noisy data
  - e.g., a small number of nearest neighbors, large decision trees
- Learners that make sharp decisions can be **unstable**
  - e.g. the decision boundary can change if one training example changes)
- If the variance is high, the model is **overfitting** the data

Can We Reduce Variance Without Increasing Bias?



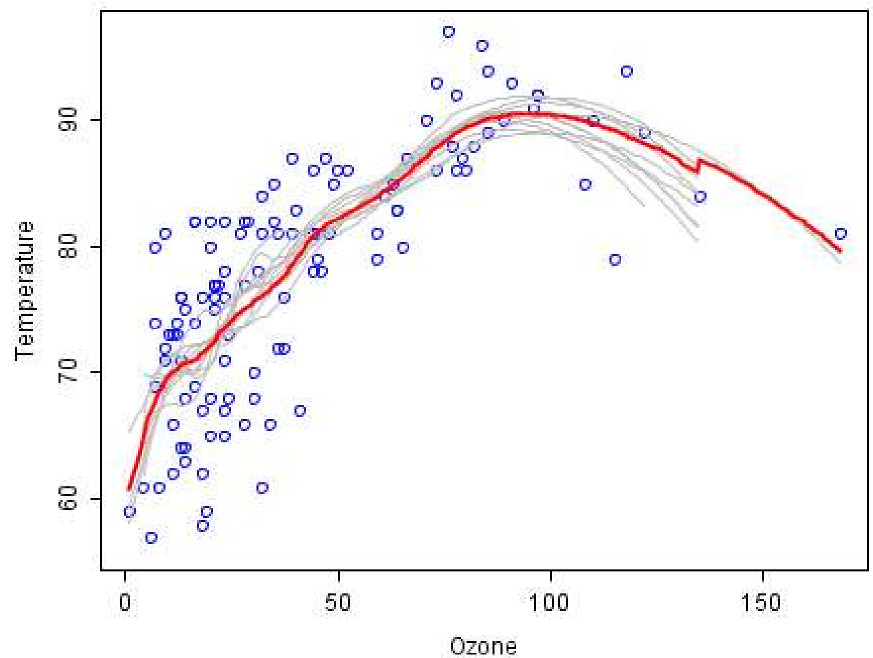
model averaging reduces variance **without changing bias**!

Ensemble Methods

Idea: Train models on different data set samples to reduce the model variance
Problem: Only one training set; where do multiple models come from?
Solution: Take a **single** learning algorithm and generate multiple variations called **ensembles**

Why Ensembles?

- When combining multiple independent and diverse decisions, random errors cancel each other out, correct decisions are reinforced
  - decision can come from weak learners:**at least** more accurate than random guessing
- **Human ensembles** are demonstrably better
  - How many jelly beans in the jar? individual estimates vs. group average
  - *Who Wants to be a Millionaire*: expert friend v. audience vote
  - crowd-sourcing
- Theoretically: they serve to reduce variance (and/or bias)

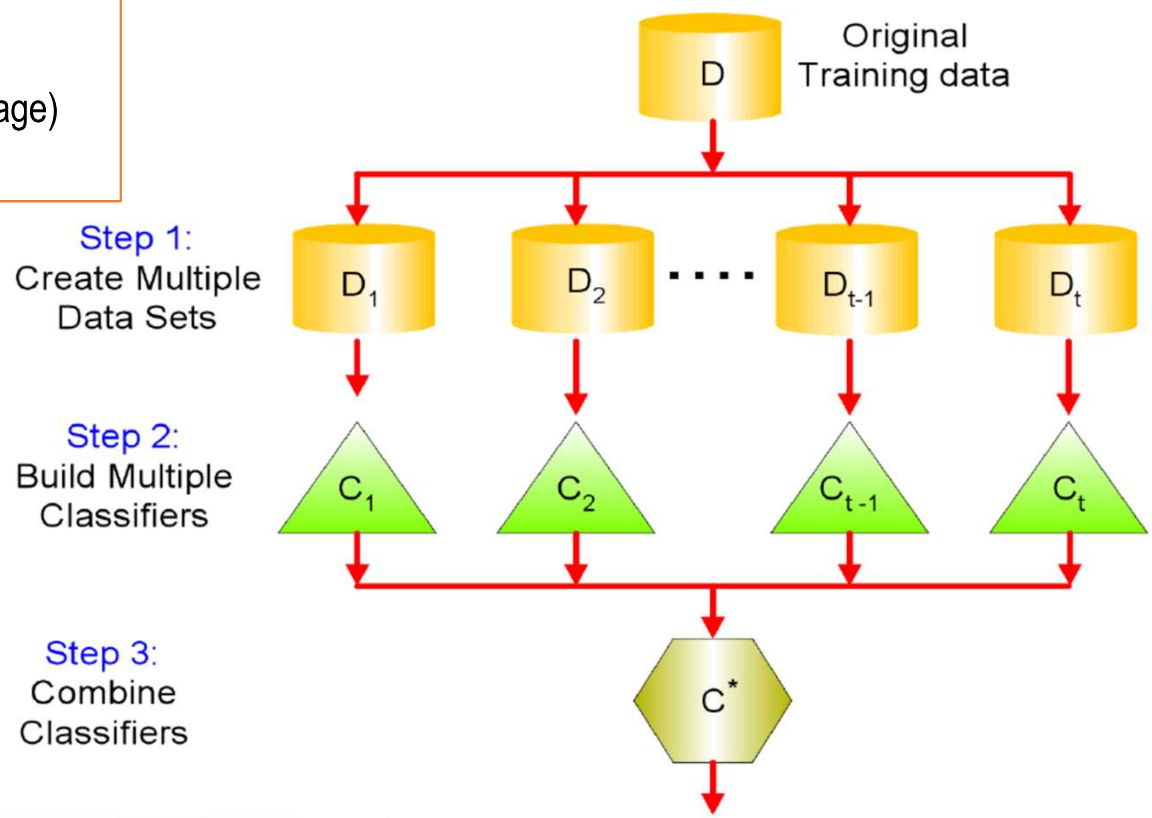


Bagging: Bootstrap Aggregation

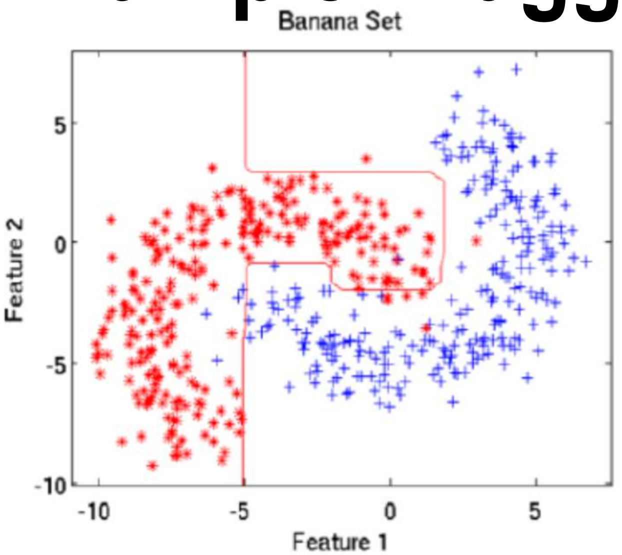
Bagging: Take repeated bootstrap samples from training set (Breiman, 1994)
Bootstrap sampling: Given set D containing n training examples, create a subset $\widehat{D}$ by drawing n samples from D **with replacement**

Bagging: Bootstrap Aggregation

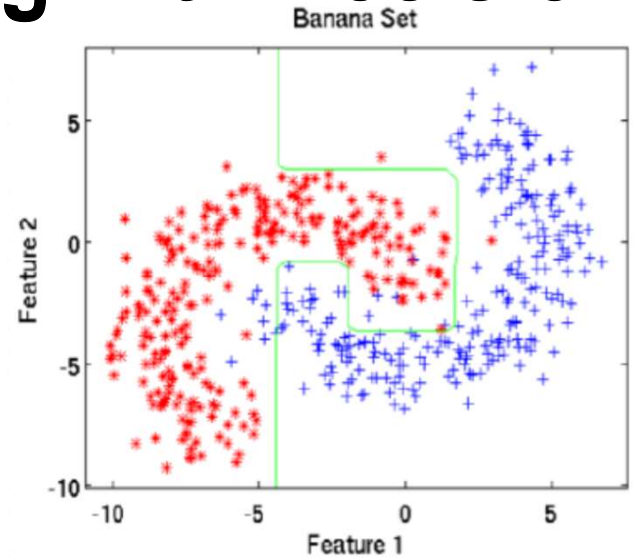
Given: data set D of size $|D|$
- Create T **bootstrap samples** $\{D_1, \dots, D_t, \dots, D_T\}$ as follows:
    - For each D_t: randomly draw $|D|$ training examples from D **with replacement**
- **for each** $t = 1, \dots, T$,
    - $f_t = \mathbf{Learn}(D_i)$
- Classify new instance by **ensembling** (majority vote/average)
    - $f_{\text{bag}} = \mathbf{Ensemble}(f_t)_{t=1}^{T}$



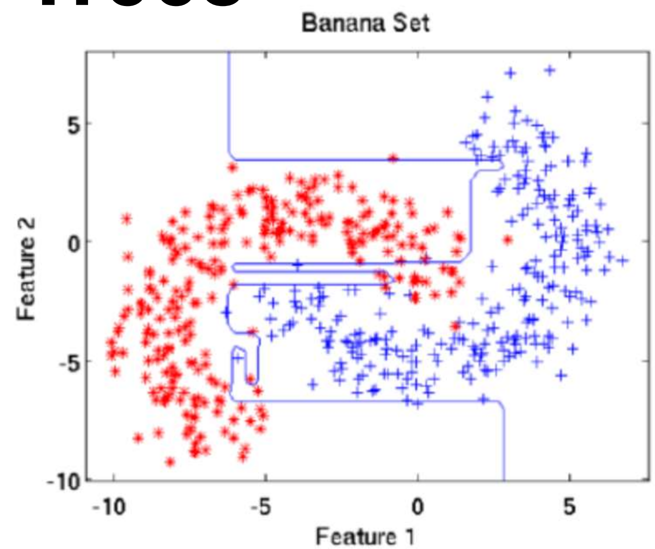
Example: Bagging with Decision Trees



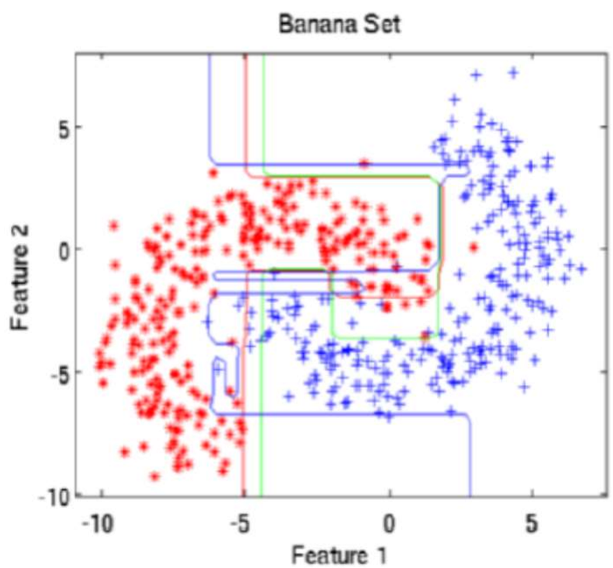
Decision boundary produced by one tree



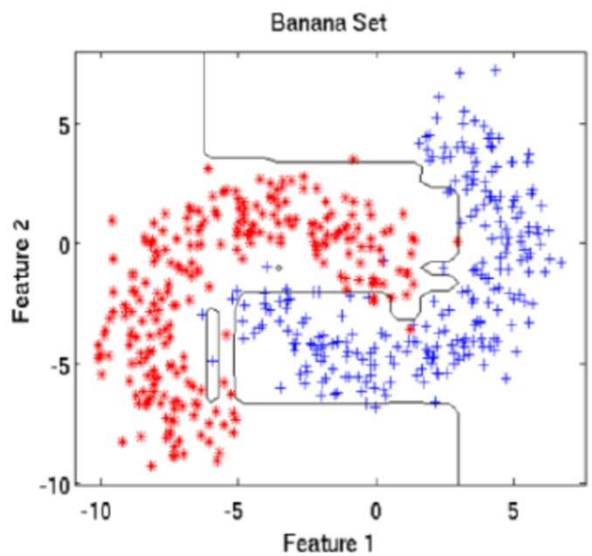
Decision boundary produced by a second tree



Decision boundary produced by a third tree



Three trees and final boundary overlaid



Final result from bagging all trees.

Bagging: Bootstrap Aggregation

Bagging: Take repeated bootstrap samples from training set (Breiman, 1994)
Bootstrap sampling: Given set D containing n training examples, create a subset $\widehat{D}$ by drawing n samples from D **with replacement**

How does bagging minimize error?

- Let bagging learn T models $(f_t)_{t=1}^T$ and ensemble them into a final model $f_{\text{bag}}(x) = \frac{1}{T}\sum_{t=1}^T f_t(x)$
- The bagging model approximates $f_{\text{bag}}(x) \approx E[\![f(x)]\!]$
- Recall (from the bias-variance decomposition and the definition of variance) that

$$\mathbf{bias}^2 + Var[\![f_{\text{bag}}(x)]\!] = \mathbf{bias}^2 + E\left[\!\!\left[f_{\text{bag}}(x) - E[\![f(x)]\!]\right]\!\!\right] \approx \mathbf{bias}^2 + 0$$

- bagging removes the variance while leaving bias unchanged; in reality, **bagging** only **reduces variance** and **tends to slightly increase bias**

When do we use Bagging?

- Depends on the **stability** of the **base-level classifiers**
  - A learner is unstable if a small change to the training set causes a large change in the output hypothesis
- If small changes in D cause large changes in the output, then there will **likely be an improvement in performance** with bagging
- Bagging helps unstable procedures, but could hurt the performance of stable procedures
  - decision trees are unstable
  - k-nearest neighbor is stable

Random Forests

Ensemble method specifically designed for **decision tree classifiers**
- Introduce **two sources of randomness**: “bagging” and “random input vectors”
- **Bagging method**: each tree is grown using a bootstrap sample of training data
- **Random vector method**: best split at each node is chosen from a random sample of m attributes instead of all attributes

for $t = 1, \dots, T$:
- Draw a **bootstrap sample** of size n from the data
- Grow a **decision tree** DT_t using the bootstrap sample:
    - Choose m attributes uniformly at random from the data
    - Choose the best attribute among the m to split on
    - Split on the best attribute and recurse (until partitions have fewer than s_{min} number of nodes)

Prediction for a new data point $\boldsymbol{x}$:
- **Regression**: $\sum_{t=1}^{T} DT_t(\boldsymbol{x})$
- **Classification**: choose the majority class label among $\{DT_1, \dots, DT_T\}$