COMS 4721: Machine Learning for Data Science
Lecture 12, 2/28/2019

Prof. John Paisley

Department of Electrical Engineering
& Data Science Institute

Columbia University

Decision Trees

DECISION TREES

A *decision tree* maps input $x \in \mathbb{R}^d$ to output y using binary decision rules:

- Each node in the tree has a *splitting rule*.
- Each leaf node is associated with an output value (outputs can repeat).

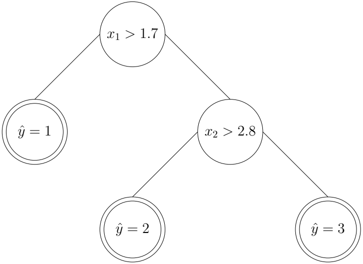
Each splitting rule is of the form

$$h(x) = \mathbb{1}\{x_j > t\}$$

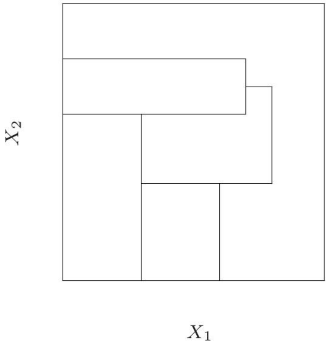
for some dimension j of x and $t \in \mathbb{R}$.

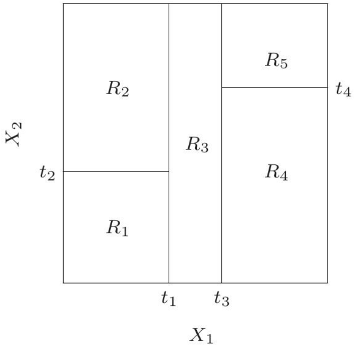
Using these transition rules, a path to a *leaf node* gives the prediction.

(One-level tree = *decision stump*)



Regression Trees



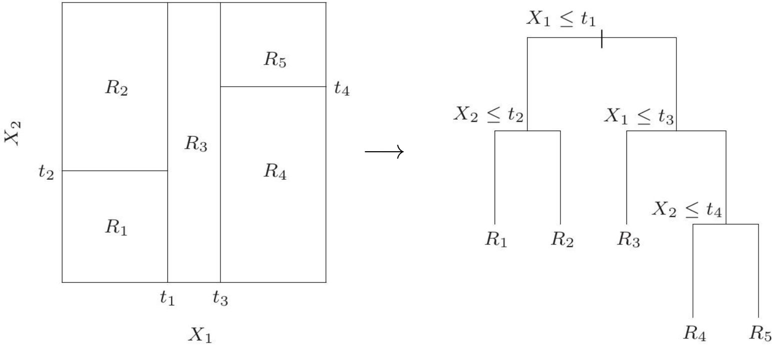


Motivation: Partition the space so that data in a region have same prediction

Left: Difficult to define a “rule”.

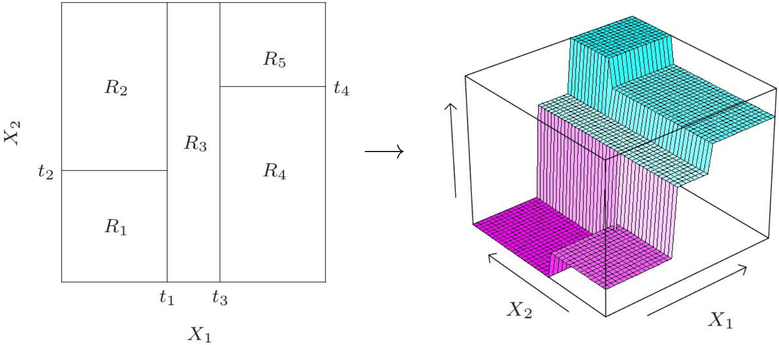
Right: Easy to define a recursive splitting rule.

Regression Trees



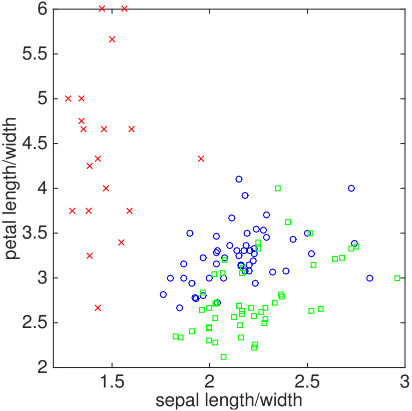
If we think in terms of trees, we can define a simple rule for partitioning the space. The left and right figures represent the same regression function.

REGRESSION TREES



Adding an output dimension to the figure (right), we can see how regression trees can learn a step function approximation to the data.

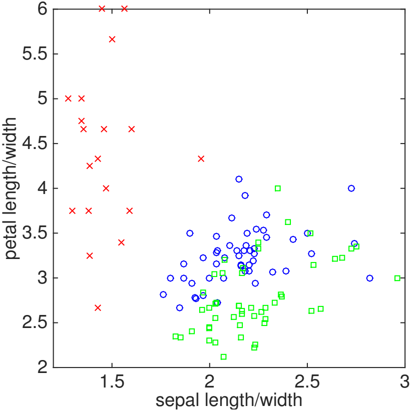
Classification Trees (Example)



Classifying irises using sepal and petal measurements:

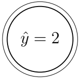
- $x \in \mathbb{R}^2$, $y \in \{1, 2, 3\}$
- x_1 = ratio of sepal length to width
- x_2 = ratio of petal length to width

Classification Trees (Example)

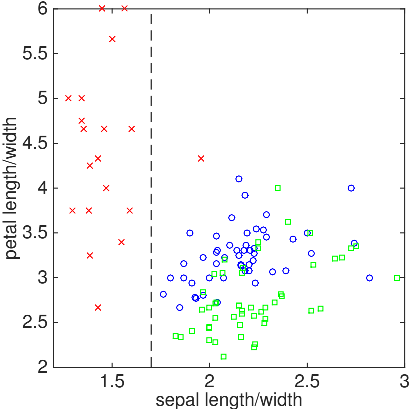


Classifying irises using sepal and petal measurements:

- $x \in \mathbb{R}^2$, $y \in \{1, 2, 3\}$
- x_1 = ratio of sepal length to width
- x_2 = ratio of petal length to width

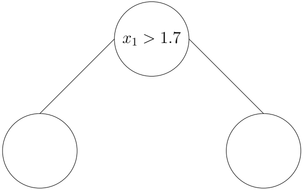


Classification Trees (Example)

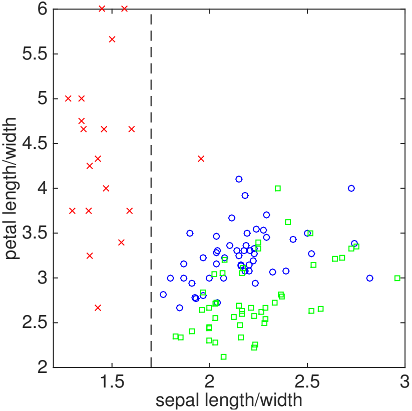


Classifying irises using sepal and petal measurements:

- $x \in \mathbb{R}^2$, $y \in \{1,2,3\}$
- x_1 = ratio of sepal length to width
- x_2 = ratio of petal length to width

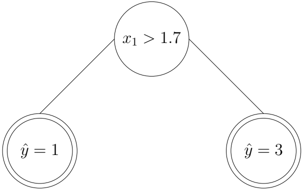


Classification Trees (Example)

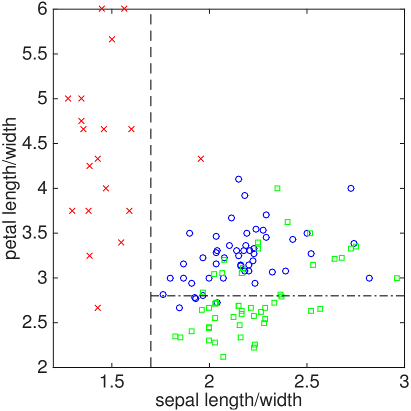


Classifying irises using sepal and petal measurements:

- $x \in \mathbb{R}^2$, $y \in \{1, 2, 3\}$
- x_1 = ratio of sepal length to width
- x_2 = ratio of petal length to width

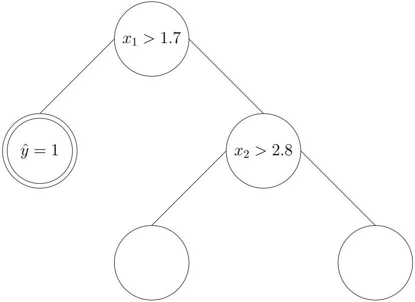


Classification Trees (Example)



Classifying irises using sepal and petal measurements:

- $x \in \mathbb{R}^2$, $y \in \{1, 2, 3\}$
- x_1 = ratio of sepal length to width
- x_2 = ratio of petal length to width

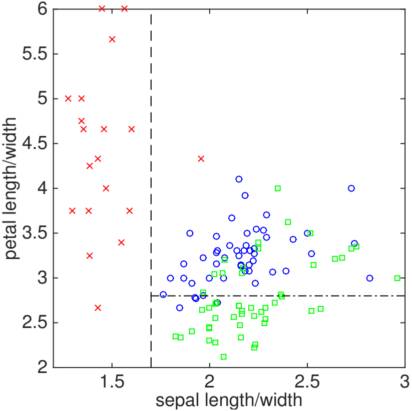


$x_1 > 1.7$

$\hat{y} = 1$

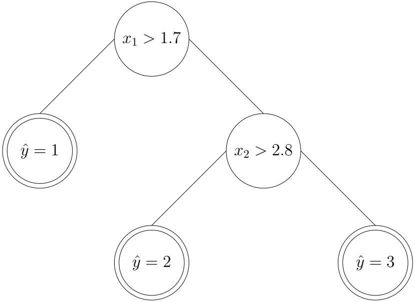
$x_2 > 2.8$

Classification Trees (Example)

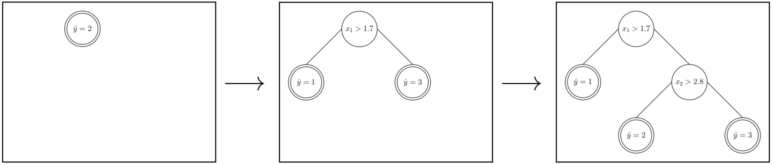


Classifying irises using sepal and petal measurements:

- $x \in \mathbb{R}^2$, $y \in \{1, 2, 3\}$
- x_1 = ratio of sepal length to width
- x_2 = ratio of petal length to width



BASIC DECISION TREE LEARNING ALGORITHM



The basic method for learning trees is with a top-down greedy algorithm.

- Start with a single leaf node containing all data
- Loop through the following steps:
  - Pick the leaf to split that reduces uncertainty the most.
  - Figure out the $\lessgtr$ decision rule on one of the dimensions.
- Stopping rule discussed later.

Label/response of the leaf is majority-vote/average of data assigned to it.

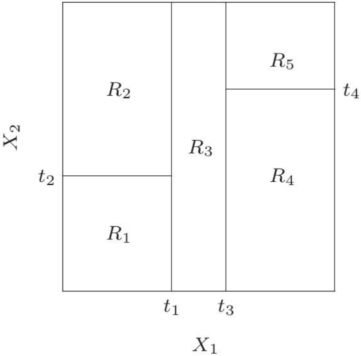
GROWING A REGRESSION TREE

How do we grow a regression tree?

- For M regions of the space, $R_1, \ldots, R_M$, the prediction function is

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

  So for a fixed M, we need R_m and c_m.



Goal: Try to minimize $\sum_i (y_i - f(x_i))^2$.

1. Find c_m given R_m: Simply the average of all y_i for which $x_i \in R_m$.
2. How do we find regions? Consider splitting region R at value s of dim j:
   - Define $R^-(j,s) = \{x_i \in R | x_i(j) \leq s\}$ and $R^+(j,s) = \{x_i \in R | x_i(j) > s\}$
   - For each dimension j, calculate the best splitting point s for that dimension.
   - Do this for each region (leaf node). Pick the one that reduces the objective most.

Growing a classification tree

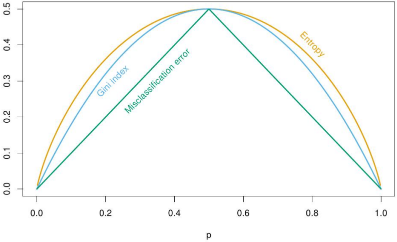
For regression: Squared error is a natural way to define the splitting rule.

For classification: Need some measure of how badly a region classifies data and how much it can improve if it's split.

K-class problem: For all $x \in R_m$, let p_k be empirical fraction labeled k.

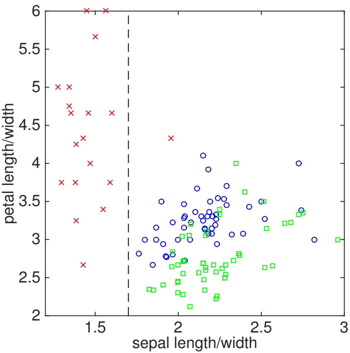
Measures of quality of R_m include

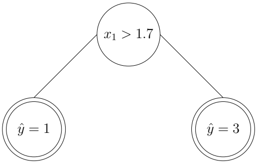
1. Classification error: $1 - \max_k p_k$
2. Gini index: $1 - \sum_k p_k^2$
3. Entropy: $-\sum_k p_k \ln p_k$



- These are all *maximized* when p_k is uniform on the K classes in R_m.
- These are *minimized* when $p_k = 1$ for some k (R_m only contains one class)

Growing a classification tree





Search R_1 and R_2 for splitting options.

1. R_1: $y = 1$ leaf classifies perfectly
2. R_2: $y = 3$ leaf has Gini index

$$\begin{aligned} u(R_2) &= 1 - \left(\frac{1}{101}\right)^2 - \left(\frac{50}{101}\right)^2 - \left(\frac{50}{101}\right)^2 \\ &= 0.5098 \end{aligned}$$

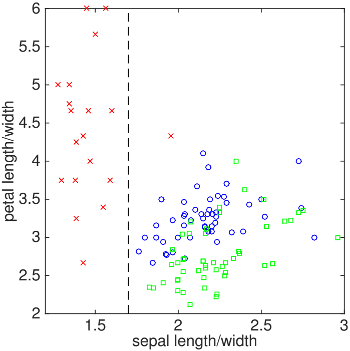
Gini improvement from split R_2 to R_2^- & R_2^+ at point t in dimension j (not written):

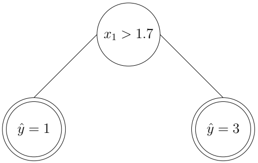
$$u(R_2) - \left(p_{R_2^-} \cdot u(R_2^-) + p_{R_2^+} \cdot u(R_2^+)\right)$$

$p_{R_m^+}$: Fraction of data in R_2 split into R_2^+.

$u(R_2^+)$: New quality measure in region R_2^+.

Growing a classification tree



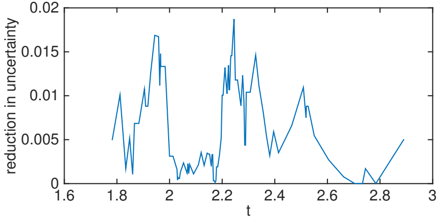


Search R_1 and R_2 for splitting options.

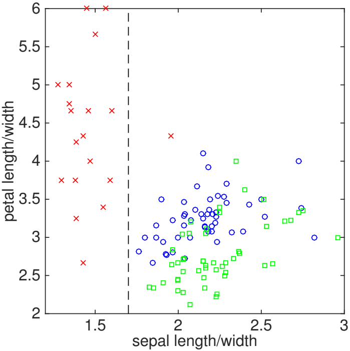
1. R_1: $y = 1$ leaf classifies perfectly
2. R_2: $y = 3$ leaf has Gini index

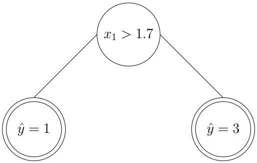
$$u(R_2) = 1 - \left(\frac{1}{101}\right)^2 - \left(\frac{50}{101}\right)^2 - \left(\frac{50}{101}\right)^2$$
$$= 0.5098$$

Check split R_2 with $\mathbb{1}\{x_1 > t\}$



Growing a classification tree



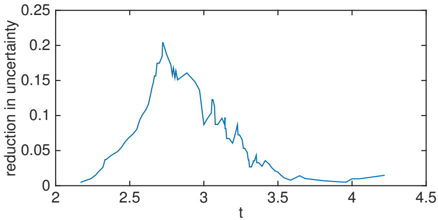


Search R_1 and R_2 for splitting options.

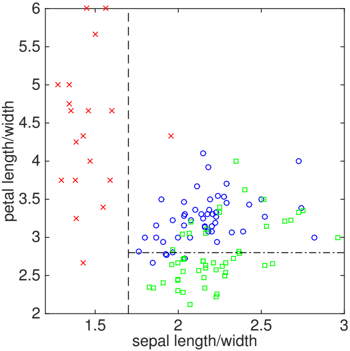
1. R_1: $y = 1$ leaf classifies perfectly
2. R_2: $y = 3$ leaf has Gini index

$$u(R_2) = 1 - \left(\frac{1}{101}\right)^2 - \left(\frac{50}{101}\right)^2 - \left(\frac{50}{101}\right)^2$$
$$= 0.5098$$

Check split R_2 with $\mathbb{1}\{x_2 > t\}$



Growing a classification tree

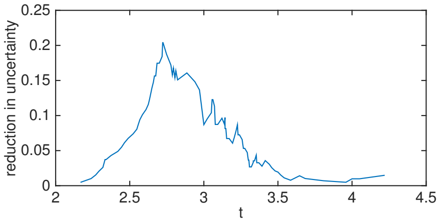


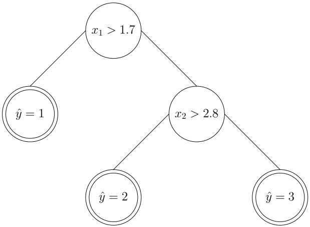
Search R_1 and R_2 for splitting options.

1. R_1: $y = 1$ leaf classifies perfectly
2. R_2: $y = 3$ leaf has Gini index

$$u(R_2) = 1 - \left(\frac{1}{101}\right)^2 - \left(\frac{50}{101}\right)^2 - \left(\frac{50}{101}\right)^2$$
$$= 0.5098$$

Check split R_2 with $\mathbb{1}\{x_2 > t\}$





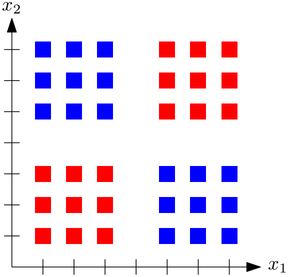
Pruning a tree

Q: When should we stop growing a tree?

A: Uncertainty reduction is not best way.

Example: Any split of x_1 or x_2 at right will show *zero* reduction in uncertainty.

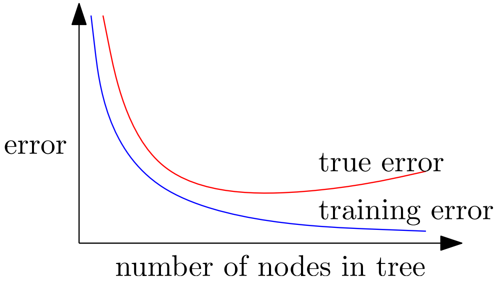
However, we can learn a perfect tree on this data by partitioning in quadrants.



Pruning is the method most often used. Grow the tree to a very large size. Then use an algorithm to trim it back.

(We won't cover the algorithm, but mention that it's non-trivial.)

OVERFITTING



- *Training error* goes to zero as size of tree increases.
- *Testing error* decreases, but then increases because of *overfitting*.

The Bootstrap

The Bootstrap: A Resampling Technique

We briefly present a technique called the *bootstrap*. This statistical technique is used as the basis for learning *ensemble classifiers*.

Bootstrap

Bootstrap (i.e., resampling) is a technique for improving estimators.

Resampling = Sampling from the empirical distribution of the data

Application to ensemble methods

- We will use resampling to generate many "mediocre" classifiers.
- We then discuss how "bagging" these classifiers improves performance.
- First, we cover the bootstrap in a simpler context.

BOOTSTRAP: BASIC ALGORITHM

Input

- A sample of data $x_1, \dots, x_n$.
- An estimation rule $\hat{S}$ of a statistic S. For example, $\hat{S} = \text{med}(x_{1:n})$ estimates the true median S of the unknown distribution on x.

Bootstrap algorithm

1. Generate *bootstrap samples* $\mathcal{B}_1, \dots, \mathcal{B}_B$.
   - Create $\mathcal{B}_b$ by picking points from $\{x_1, \dots, x_n\}$ randomly n times.
   - A particular x_i can appear in $\mathcal{B}_b$ many times (it's simply duplicated).
2. Evaluate the estimator on each $\mathcal{B}_b$ by pretending it's the data set:
$$\hat{S}_b := \hat{S}(\mathcal{B}_b)$$
3. Estimate the mean and variance of $\hat{S}$:
$$\mu_B = \frac{1}{B}\sum_{b=1}^{B}\hat{S}_b, \quad \sigma_B^2 = \frac{1}{B}\sum_{b=1}^{B}(\hat{S}_b - \mu_B)^2$$

Example: Variance estimation of the median

- The median of $x_1, \ldots, x_n$ (for $x \in \mathbb{R}$) is found by simply sorting them and taking the middle one, or the average of the two middle ones.

- How confident can we be in the estimate $\text{median}(x_1, \ldots, x_n)$?
  - Find it's variance.
  - But how? Answer: By bootstrapping the data.

1. Generate bootstrap data sets $\mathcal{B}_1, \ldots, \mathcal{B}_B$.

2. Calculate: (notice that $\hat{S}_{mean}$ is the mean of the sampled medians)

$$\hat{S}_{mean} = \frac{1}{B} \sum_{b=1}^{B} \text{median}(\mathcal{B}_b), \quad \hat{S}_{var} = \frac{1}{B} \sum_{b=1}^{B} \left(\text{median}(\mathcal{B}_b) - \hat{S}_{mean} \right)^2$$

- The procedure is remarkably simple, but has a lot of theory behind it.

Bagging and Random Forests

Bagging

Bagging uses the bootstrap for regression or classification:

Bagging = **B**ootstrap **agg**regation

Algorithm

For $b = 1, \ldots, B$:

1. Draw a bootstrap sample $\mathcal{B}_b$ of size n from training data.
2. Train a classifier or regression model f_b on $\mathcal{B}_b$.

- For a new point x_0, compute:

$$f_{\text{avg}}(x_0) = \frac{1}{B} \sum_{b=1}^{B} f_b(x_0)$$

  - For regression, $f_{\text{avg}}(x_0)$ is the prediction.
  - For classification, view $f_{\text{avg}}(x_0)$ as the distribution of B votes. Pick the majority.

Example: Bagging Trees

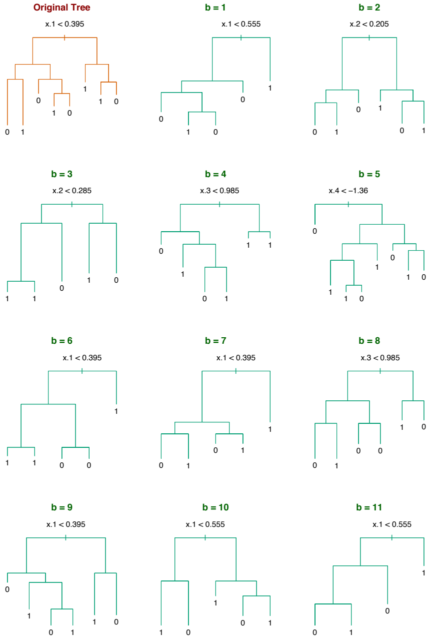
- Binary classification, $x \in \mathbb{R}^5$.

- Note the variation among bootstrapped trees.

- Take-home message:

  With bagging, each tree doesn't have to be great, just "ok".

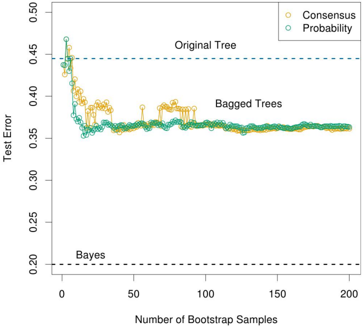
- Bagging often improves results *when the function is non-linear.*



Random Forests

Drawbacks of Bagging

- Bagging works on trees because of the bias-variance tradeoff (↑ bias, ↓ variance).
- However, the bagged trees are correlated.
- In general, when bootstrap samples are correlated, the benefit of bagging decreases.



Random Forests

Modification of bagging where trees are designed to reduce correlation.

- A very simple modification.
- Still learn a tree on each bootstrap set, $\mathcal{B}_b$.
- To split a region, only consider random subset of dimensions of $x \in \mathbb{R}^d$.

RANDOM FORESTS: ALGORITHM

Training

Input parameter: m — a positive integer with $m < d$, often $m \approx \sqrt{d}$

For $b = 1, \dots, B$:

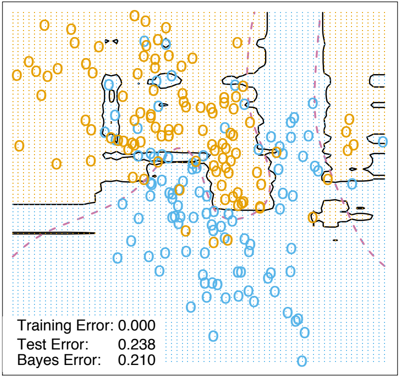
1. Draw a bootstrap sample $\mathcal{B}_b$ of size n from the training data.
2. Train a tree classifier on $\mathcal{B}_b$, where each split is computed as follows:
    - Randomly select m dimensions of $x \in \mathbb{R}^d$, newly chosen for each split.
    - Make the best split restricted to that subset of dimensions.

- Bagging for trees: Bag trees learned using the original algorithm.
- Random forests: Bag trees learned using algorithm on this slide.

RANDOM FORESTS

Example problem

- Random forest classification.
- Forest size: A few hundred trees.
- Notice there is a tendency to align decision boundary with the axis.


Training Error: 0.000
Test Error: 0.238
Bayes Error: 0.210