CSCI 4/5588
ML-II

Chapter #6:

Random Forest

Md Tamjidul Hoque

Overview

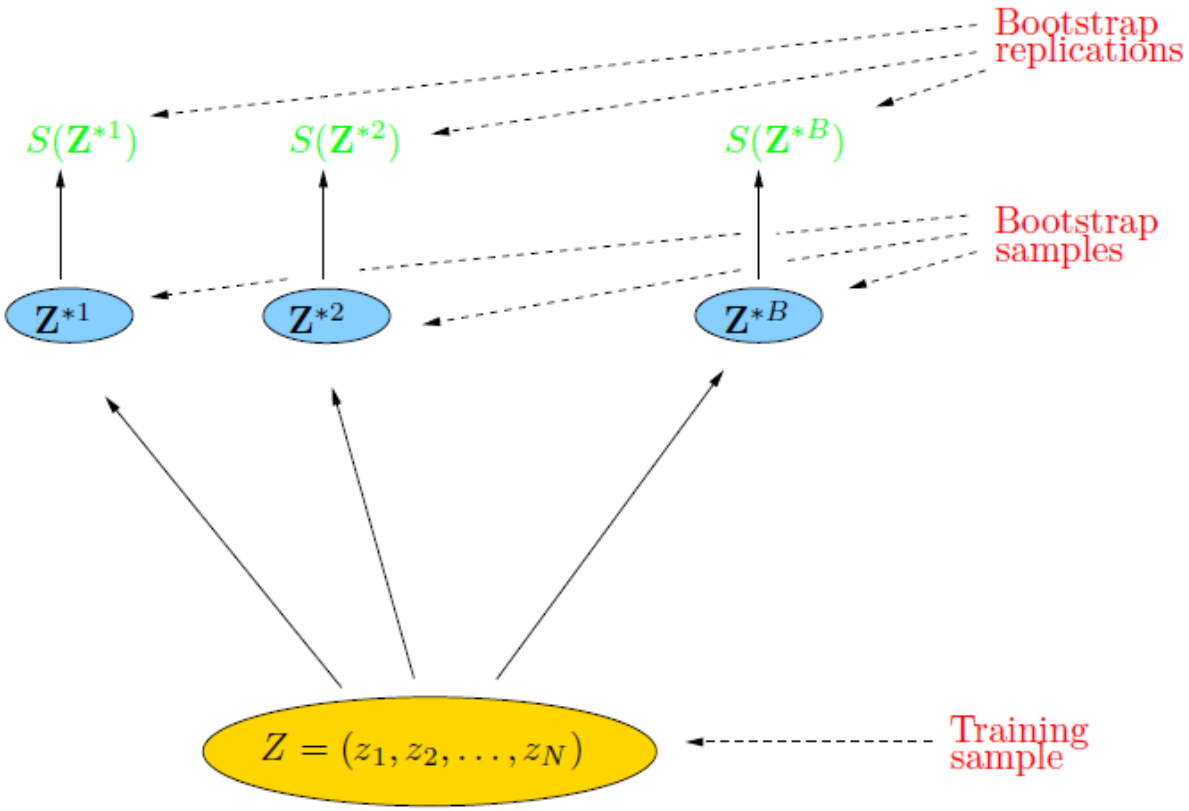
- We will be studying
  - Bootstrap methods
  - Bagging
  - Boosting methods
  - Decision Trees
  - Random Forest
  - Ensemble Learning

Bootstrap Methods

- The bootstrap is a general tool for ***assessing statistical accuracy***.

- Suppose we have a model fit to a set of training data. We denote the training set by $Z = (z_1, z_2, \ldots, z_N)$ where $z_i = (x_i, y_i)$. The basic idea is to
  - randomly draw datasets ***with replacement*** from the training data,
  - this is done B times (B = 100 say), producing B bootstrap datasets, as shown in Figure (next Slide).
  - Then we refit the model to each of the bootstrap datasets, and examine the behavior of the fits over the B replications.

Bootstrap Methods



Schematic of the bootstrap process:

> We wish to assess the statistical accuracy of a quantity $S(Z)$ computed from our dataset.

> B training sets Z^{*b}, $b = 1, \ldots, B$ each of size N are drawn with replacement from the original dataset.

> The quantity of interest $S(Z)$ is computed from each bootstrap training set, and the values $S(Z^{*1}) , \ldots , S(Z^{*B})$ are used to assess the statistical accuracy of $S(Z)$.

Bootstrap Methods

- In the figure, $S(Z)$ is any quantity computed from the data Z, for example, the prediction at some input point.
- From the bootstrap sampling we can estimate any aspect of the distribution (say, distribution function F for the data ($z_1, z_2, \ldots, z_N$)) of $S(Z)$, for example, its variance,
- $$Var[S(Z)] = \frac{1}{B-1}\sum_{b=1}^{B}(S(Z^{*b}) - \bar{S}^{*})^2$$
- where mean, $$\bar{S}^{*} = \sum_{b=1}^{B} S(Z^{*b})/B.$$
- The larger the number of B of bootstrap samples, the more satisfactory is the estimate of a statistic and its variance. One of the benefits of bootstrap estimation is that B can be adjusted to the computational resources.

Bootstrap Methods

- How can we apply the bootstrap to estimate prediction error?
- One approach would be to fit the model in question on a set of bootstrap samples, and then keep track of how well it predicts the original training set.
- If $\hat{f}^{*b}(x_i)$ is the predicted value at x_i, from the model fitted to the b^{th} bootstrap dataset, our estimate is

$$Err_{boot} = \frac{1}{B}\frac{1}{N}\sum_{b=1}^{B}\sum_{i=1}^{N} L(y_i, \hat{f}^{*b}(x_i))$$

Bagging

- *Bagging* is basically *bootstrap aggregation*.
- Bootstrap has been shown to be a way of assessing the accuracy of a parameter estimate or a prediction.
- Consider first the regression problem;
  - Suppose we fit a model to our training data $Z = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, obtaining the prediction $\hat{f}(x)$ at input x.
  - *Bootstrap aggregation* or bagging averages this prediction over a collection of bootstrap samples, thereby reducing its variance.
  - For each bootstrap sample Z^{*b}, $b = 1, 2, \ldots, B$, we fit our model, giving prediction $\hat{f}^{*b}(x)$. The bagging estimate is defined by

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x)$$

Boosting Methods

- Boosting is one of the most powerful learning ideas introduced in the last twenty years. It was originally designed for classification problems, but it can be used for regression as well.
- The motivation for boosting was a procedure that combines the outputs of many "weak" classifiers to produce a powerful "committee."
- We will see "AdaBoost.M1", the most popular boosting algorithm:
  - Consider a two-class problem, with the output variable coded as $Y \in \{-1, 1\}$. Given a vector of predictor variables X, a classifier $G(X)$ produces a prediction taking one of the two values $\{-1, 1\}$. The error rate on the training sample is

$$\overline{err} = \frac{1}{N}\sum_{i=1}^{N} I(y_i \neq G(x_i)),$$

and the expected error rate on future predictions is $E_{XY} I(Y \neq G(X))$.

Boosting Methods

- A weak classifier is one whose error rate is only slightly better than random guessing.
- The purpose of boosting is to **sequentially** apply the weak classification algorithm to repeatedly modified versions of the data, thereby producing a sequence of weak classifiers $G_m(x)$, $m = 1, 2, \ldots, M$.
- The predictions from all of them are then combined through a **weighted majority vote** to produce the final prediction:

$$G(x) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$$

- Here $\alpha_1, \alpha_2, \ldots, \alpha_M$ are computed by the boosting algorithm, and weight the contribution of each respective $G_m(x)$. Their effect is to give higher influence to the more accurate classifiers in the sequence.

Boosting Methods

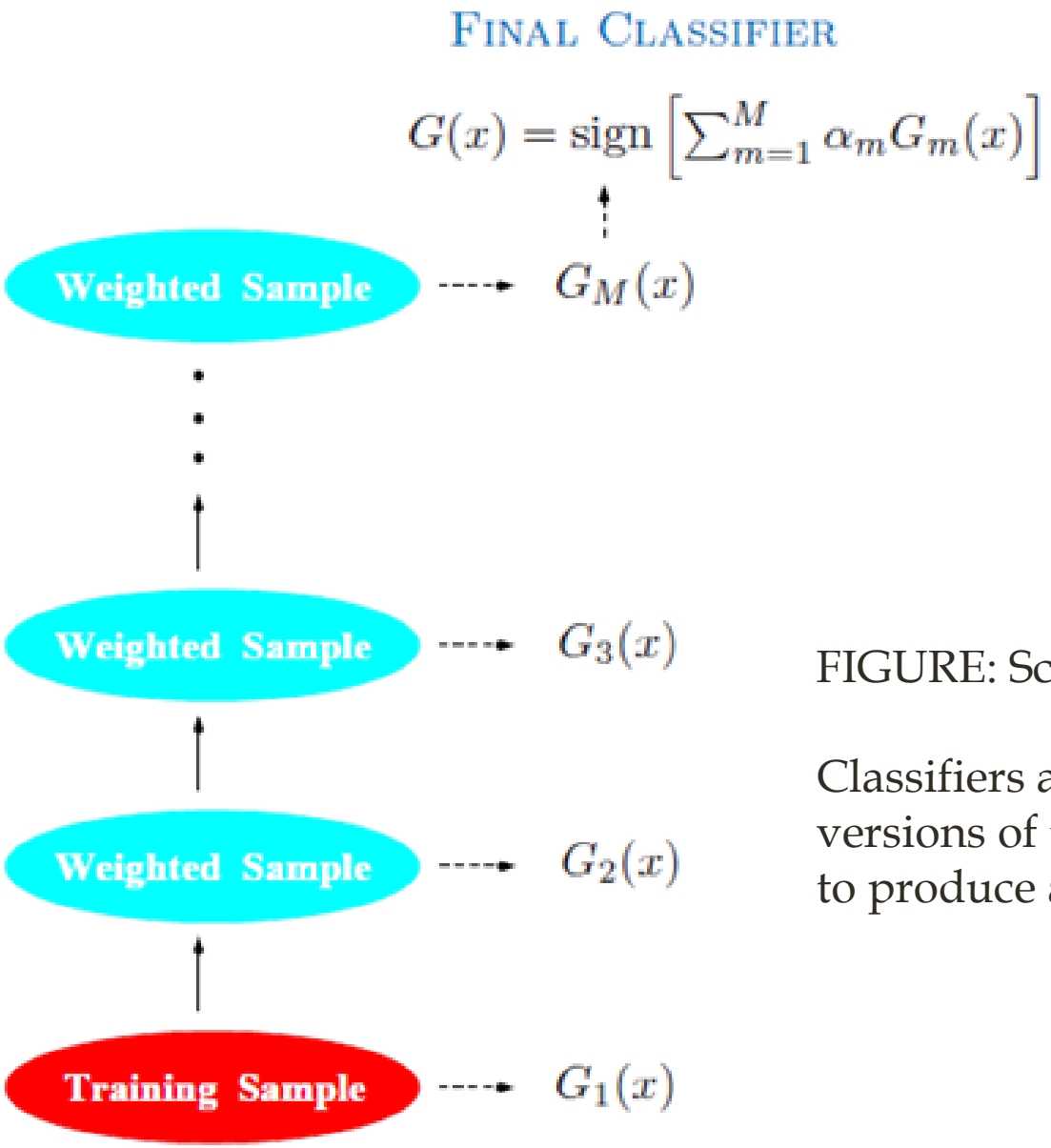


FIGURE: Schematic of AdaBoost.

Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.

Boosting Methods

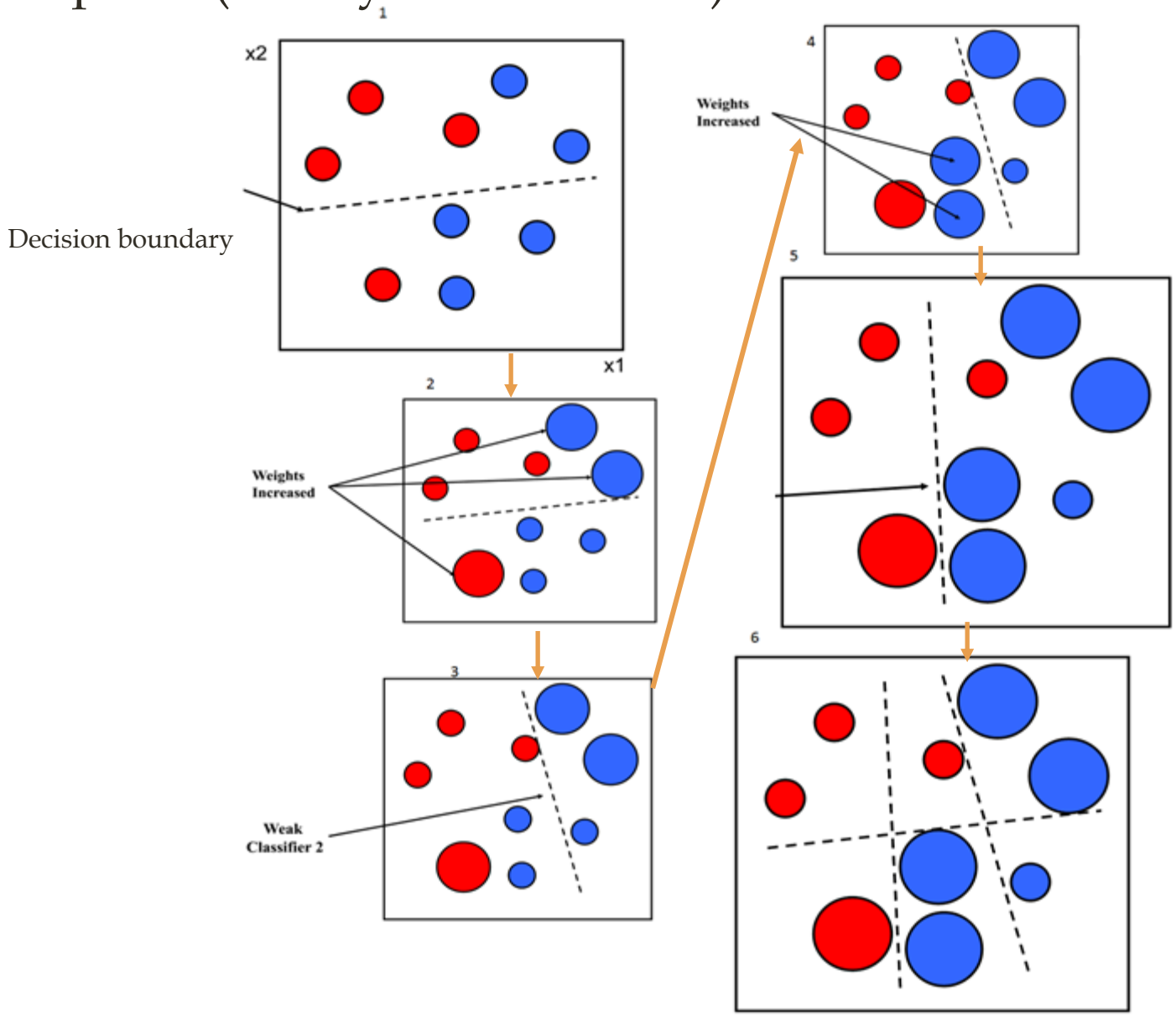
Algorithm: AdaBoost.M1.

1. Initialize the observation weights $w_i = 1/N$, i = 1, 2, …, N.
2. For $m = 1$ to M
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute

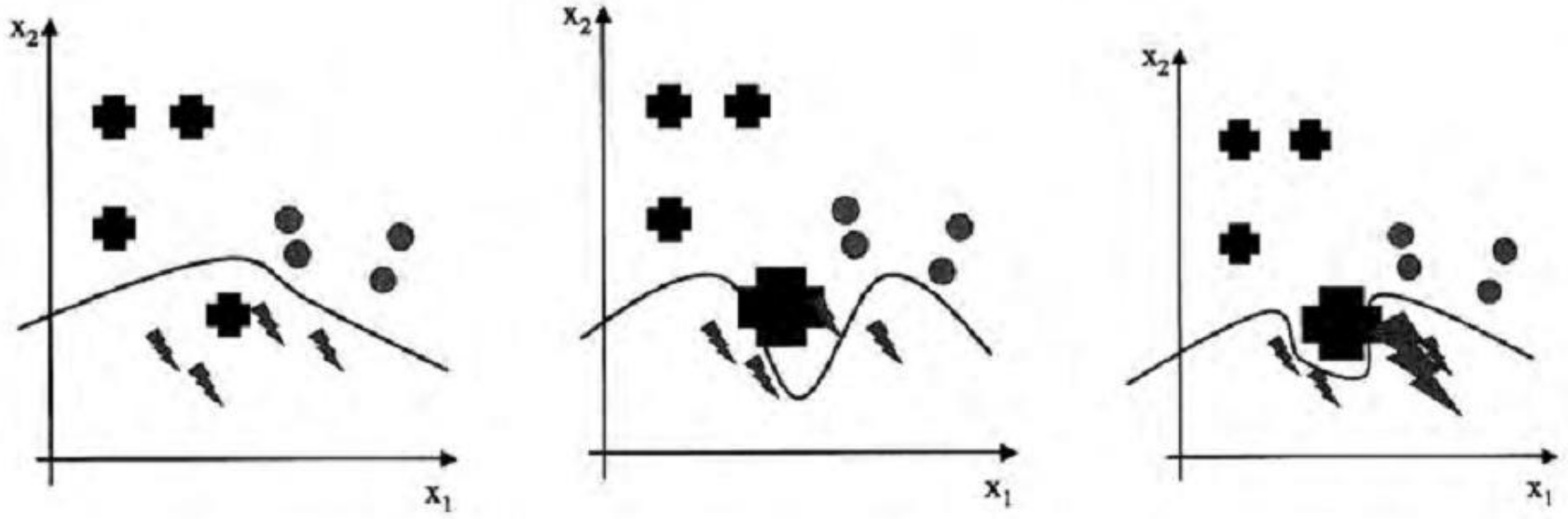
$$err_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$$

   (c) Compute $\alpha_m = \log((1 - err_m)/err_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$, $i = 1,2,...,N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Example 1: (Binary-classification) How does the boosting work?



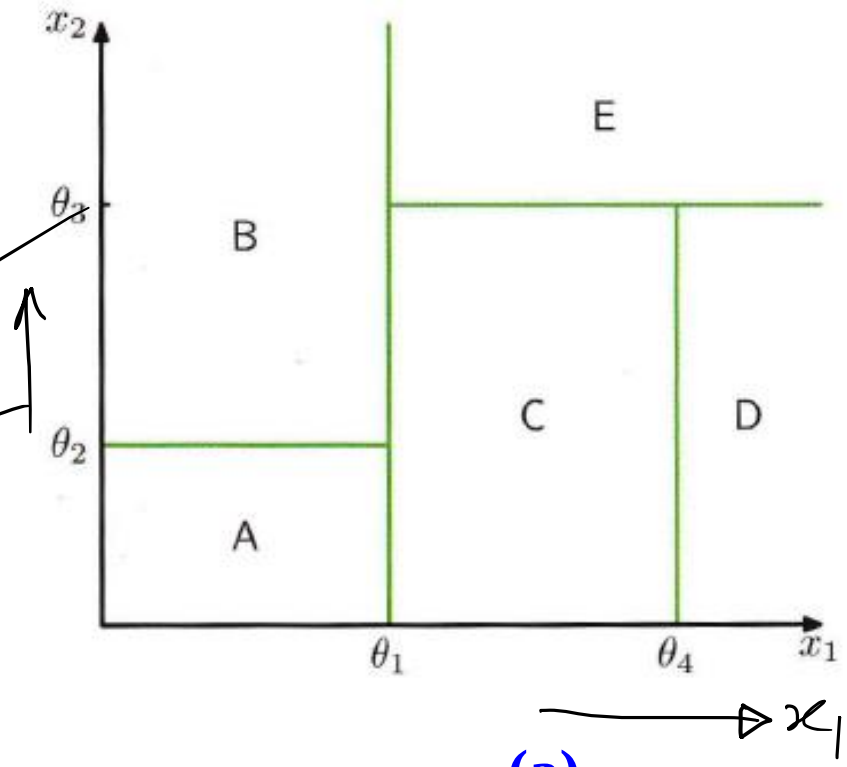
Example 2: (Multiclass) How does the boosting work?



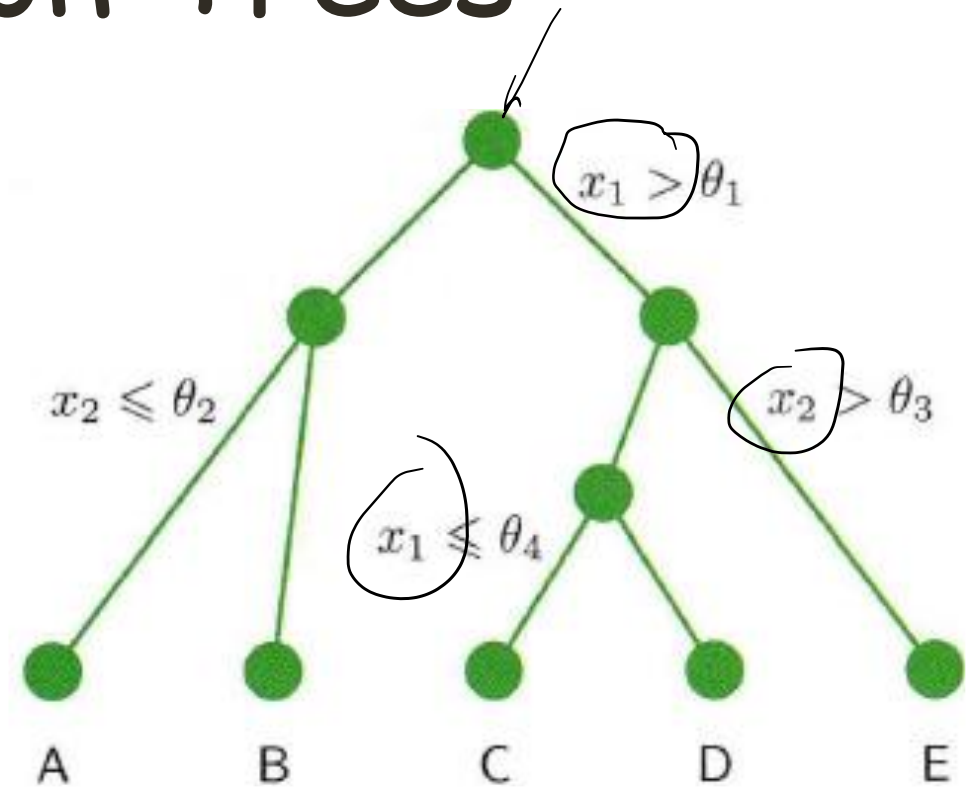
Decision Trees

- There are various simple, but widely used, models that work by partitioning the input space into cuboid regions, whose edges are aligned with the axes, and then assigning a simple model (for example, a constant) to each region.
- They can be viewed as a model combination method in which only one model is responsible for making predictions at any given point in input space.
- The process of selecting a specific model, given a new input x, can be described by a sequential decision-making process corresponding to the traversal of a binary tree (one that splits into two branches at each node).
- Here, we focus on a particular tree-based framework called **c**lassification **a**nd **r**egression **t**rees, or CART (Breiman *et al.*, 1984).

Decision Trees



(a)



(b)

Figure 1(a): Illustration of a two-dimensional input space that has been partitioned into five regions using axis-aligned boundaries.

1(b): Binary tree corresponding to the partitioning of input space shown in Figure 1(a).

Decision Trees

- Within each region, there is a separate model to predict the target variable.
- For instance, in **regression** we might simply predict a constant over each region, or in **classification** we might assign each region to a specific class.
- A key property of tree-based models, which makes them popular in fields such as medical diagnosis, for example, is that they are readily interpretable by humans because they correspond to a sequence of binary decisions applied to the individual input variables.
- For instance, to predict a patient's disease, we might first ask “is their temperature greater than some threshold?”. If the answer is yes, then we might next ask “is their blood pressure less than some threshold?” Each leaf of the tree is then associated with a specific diagnosis.

Decision Trees

- In order to learn such a model from a training set, we have to determine the structure of the tree:
  - including which input variable is chosen at each node to form the split criterion as well as
  - the value of the threshold parameter θ_1 for the split.
  - We also have to determine the values of the predictive variable within each region.

Decision Trees

- Consider first a regression problem in which the goal is to predict a single target variable $\boldsymbol{t}$ from a D-dimensional vector $\mathbf{x} = (x_1, \ldots, x_D)^T$ of input variables.
- The training data consists of input vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ along with the corresponding continuous labels $\{t_1, \ldots, t_N\}$.
- If the partitioning of the input space is given, and we minimize the sum-of-squares error function, then the optimal value of the predictive variable within any given region is just given by the average of the values of t_n for those data points that fall in that region.

Decision Trees

- How to determine the structure of the decision tree:
  - Even for a fixed number of nodes in the tree, the problem of determining the optimal structure (including choice of input variable for each split as well as the corresponding thresholds) to minimize the sum-of-squares error is usually computationally infeasible due to the combinatorially large number of possible solutions.
  - Instead, a greedy optimization is generally done by starting with a single root node, corresponding to the whole input space, and then growing the tree by adding nodes one at a time.
  - At each step there will be some number of candidate regions in input space that can be split, corresponding to the addition of a pair of leaf nodes to the existing tree.
  - For each of these, there is a choice of which of the D input variables to split, as well as the value of the threshold.
  - The joint optimization of the choice of region to split, and the choice of input variable and threshold, can be done efficiently by exhaustive search noting that, for a given choice of split variable and threshold, the optimal choice of predictive variable is given by the local average of the data, as noted earlier.
  - This is repeated for all possible choices of variable to be split, and the one that gives the smallest residual sum-of-squares error is retained.

Decision Trees

- When to stop growing the Tree:
  - stop when the reduction in residual error falls below some threshold
  - often none of the available splits produces a significant reduction in error, thus
  - it is common practice to grow a large tree, using a stopping criterion based on the number of data points associated with the leaf nodes, and then prune back the resulting tree.
- The pruning is based on a criterion that balances residual error against a measure of model complexity.
- Mostly, we use negative cross-entry and/or Gini-index … the main idea is, they encourage the formation of regions in which a high proportion of the data points are assigned to one class.

Random Forest

- The essential idea in bagging is to average many noisy but approximately unbiased models, and hence reduce the variance.
- Trees are ideal candidates for bagging, since they can capture complex interaction structures in the data, and if grown sufficiently deep, have relatively low bias.
- Since trees are notoriously noisy, they benefit greatly from the averaging.
- Moreover, since each tree generated in bagging is identically distributed (*i.d.*), the expectation of an average of B such trees is the same as the expectation of any one of them. This means the bias of bagged trees is the same as that of the individual trees, and the only hope of improvement is through variance reduction.

Random Forest

Algorithm: *Random Forest for Regression or Classification.*

1. For b = 1 to B:

   (a) Draw a bootstrap sample 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 $\{T_b\}_1^B$.

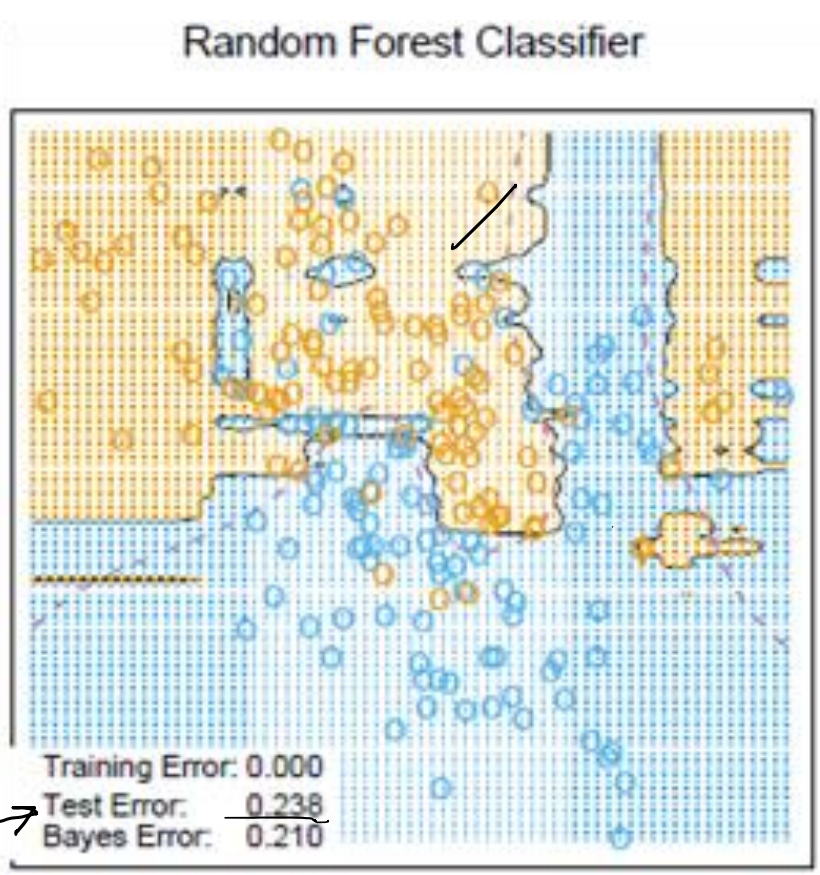
To make a new prediction at a new point x:

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

Classification: Let $\hat{C}_b(x)$ be the class prediction of the b^{th} random-forest tree.

Then $\hat{C}_{rf}^B(x)$ = majority vote $\{\hat{C}_b(x)\}_1^B$.

Random Forest



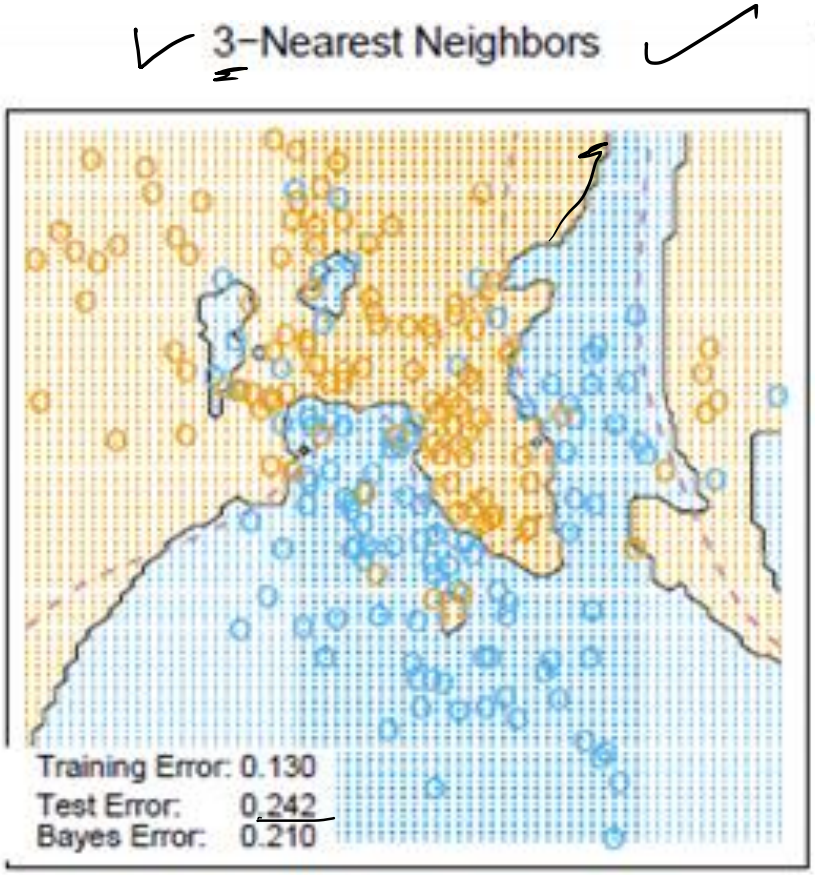


FIGURE: Random forests versus 3-NN on the mixture data. The axis-oriented nature of the individual trees in a random forest lead to decision regions with an axis-oriented flavor.

Ensemble Learning

- The idea of ensemble learning is to build a prediction model by combining the strengths of a collection of simpler base models.
- Bagging and random forests are ensemble methods for classification, where a *committee* of trees each cast a vote for the predicted class, ….
- Ensemble learning can be broken down into two tasks:
  - developing a population of base learners from the training data, and
  - then combining them to form the composite predictor.
- And the composite predictor is expected to perform better than the individual.

---- x -----