STEVENS INSTITUTE of TECHNOLOGY
School of Business

FE 582

Lecture 6: Tree-Based Methods & Clustering

Dragos Bozdog

Decision Trees

Can be applied to:

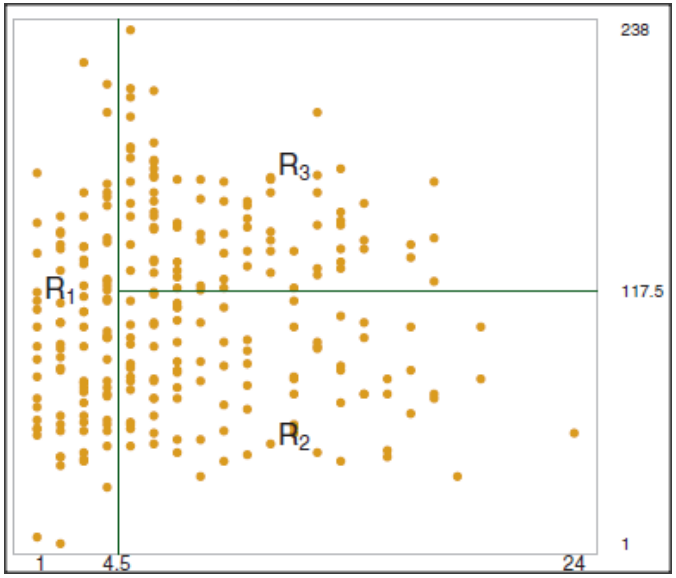
- Regression problems.
- Classification problems.

Regression Trees

Process of building a regression tree:

- We divide the predictor space—that is, the set of possible values for $X_1, X_1, \cdots X_p$—into *J* distinct and non-overlapping regions, $R_1, R_2, \cdots R_J$.
- For every observation that falls into the region R_J, we make the same prediction, which is simply the mean of the response values for the training observations in R_J .

How do we construct the regions $R_1, R_2, \cdots R_J$?



Regression Trees

In theory, the regions could have any shape.

- However, we choose to divide the predictor space into high-dimensional rectangles, or *boxes*, for simplicity and for ease of interpretation of the resulting predictive model.

- The goal is to find boxes $R_1, R_2, \cdots R_J$ that minimize the RSS (residual sum of squares), given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j} \right)^2$$

where:

- $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box

Issue: computational infeasible to consider all possible partitions of feature space into J boxes

Regression Trees

Partitioning:

- *recursive binary splitting* (or *top-down*, *greedy* approach)

Begins at the top of the tree

- all observations belong to a single region

Then successively splits the predictor space

- each split is indicated via two new branches further down on the tree.

It is *greedy* because at each step of the tree-building process, the *best* split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

Regression Trees

Select the predictor X_j and the cutpoint s such that splitting the predictor space into the regions $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ leads to the greatest possible reduction in RSS.

for any j and s, we define the pair of half-planes

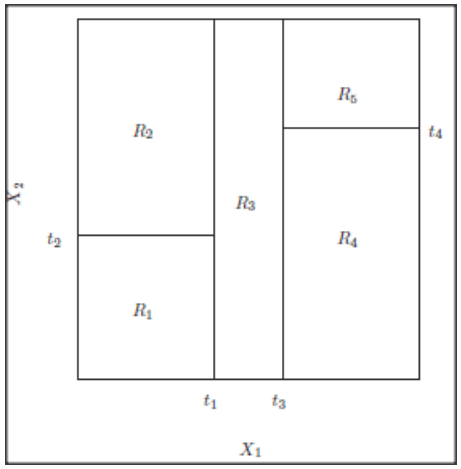
- $R1(j, s) = \{X|X_j < s\}$
- $R2(j, s) = \{X|X_j \geq s\}$,

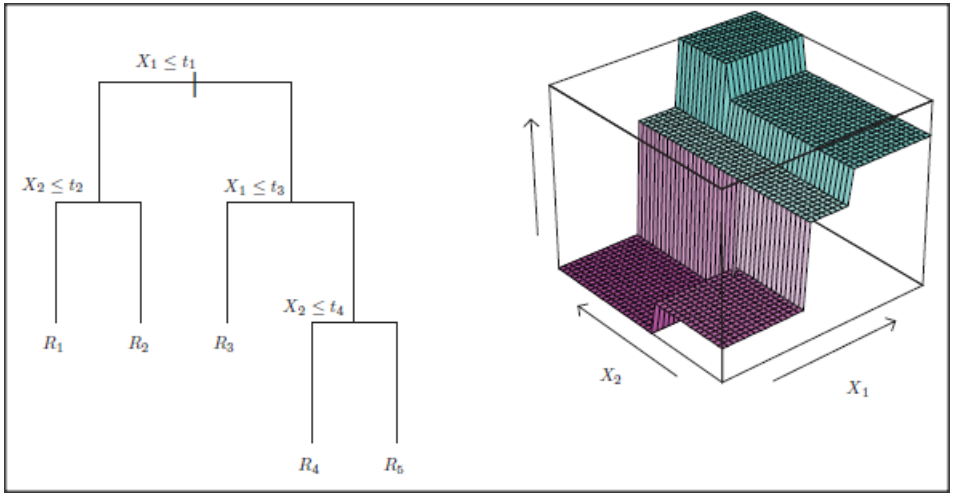
Seek the value of j and s that minimize the equation

$$\sum_{i:\, x_i \in R_1(j,s)} \left(y_i - \hat{y}_{R_1}\right)^2 + \sum_{i:\, x_i \in R_2(j,s)} \left(y_i - \hat{y}_{R_2}\right)^2$$

Regression Trees

Example of a two-dimensional feature space





Regression Trees

The process described before may produce good predictions on the training set, but is likely to overfit the data, leading to poor test set performance because of the complexity of the tree.

Smaller tree might lead to

- Smaller variance
- Better interpretation

Alternative

- Build the tree only so long as the decrease in the RSS due to each split exceeds some (high) threshold.

- Short-sighted since a seemingly worthless split early on in the tree might be followed by a very good split—that is, a split that leads to a large reduction in RSS later on.

Tree Pruning

Better strategy:

- grow a very large tree T_0, and then
- *prune* it back in order to obtain a *subtree*.

Algorithm *Building a Regression Tree*

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:
   - Repeat Steps 1 and 2 on all but the kth fold of the training data.
   - Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.
   - Average the results for each value of α, and pick α to minimize the average error.
4. Return the subtree from Step 2 that corresponds to the chosen α.

Tree Pruning

For each value of α there corresponds a subtree $T \subset T_0$ such that

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} \left(y_i - \hat{y}_{R_m}\right)^2 + \alpha|T|$$

is as small as possible

- $|T|$ indicates the number of terminal nodes of the tree T ,
- R_m is the rectangle (i.e. the subset of predictor space) corresponding to the m^{th} terminal node,
- $\hat{y}_{R_m}$ is the predicted response associated with R_m—that is, the mean of the training observations in R_m.
- The tuning parameter α controls a trade-off between the subtree's complexity and its fit to the training data.

Classification Trees

A *classification tree* is very similar to a regression tree, except that it is classification used to predict a qualitative response rather than a quantitative one.

- We predict that each observation belongs to the *most commonly occurring class* of training observations in the region to which it belongs.

- In interpreting the results of a classification tree, we are often interested not only in the class prediction corresponding to a particular terminal node region, but also in the *class proportions* among the training observations that fall into that region.

Classification Trees

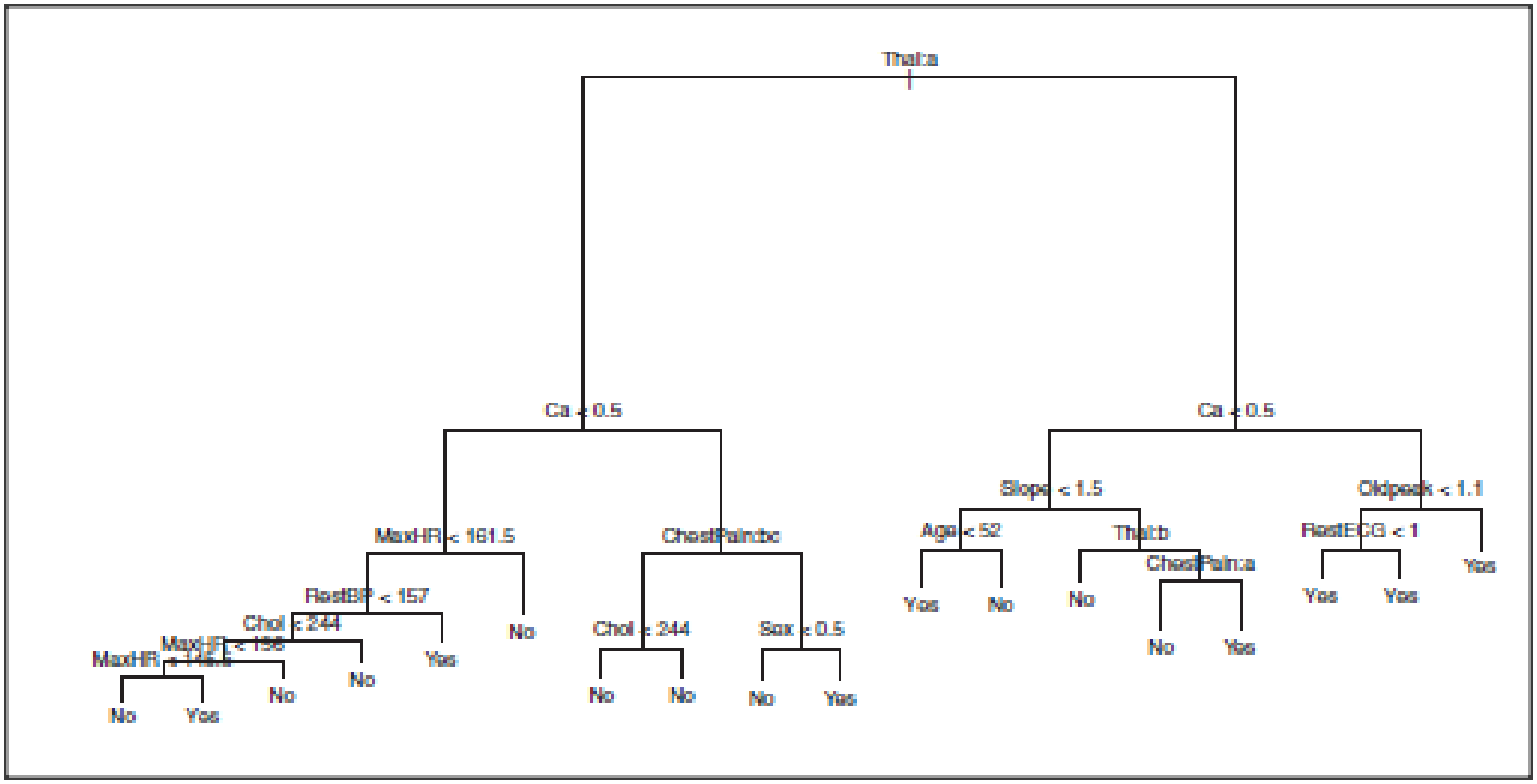
The task of growing a classification tree is quite similar to the task of growing a regression tree.

- Use recursive binary splitting to grow a classification tree. However, in the classification setting, RSS cannot be used as a criterion for making the binary splits.
- A natural alternative to RSS is the classification error rate.
- The classification error rate is simply the fraction of the training observations in that region that do not belong to the most common class:

$$E = 1 - \max_k(\hat{p}_{mk})$$

$\hat{p}_{mk}$ represents the proportion of training observations in the mth region that are from the kth class.

Classification Trees



Classification Trees

However, it turns out that classification error is not sufficiently sensitive for tree-growing, and in practice two other measures are preferable.

The *Gini index* is defined by

$$G = \sum_{k=1}^{K} \hat{p}_{mk}\,(1 - \hat{p}_{mk})$$

- measures the total variance across the K classes.
- the Gini index is referred to as a measure of node *purity*—a small value indicates that a node contains predominantly observations from a single class.

Classification Trees

An alternative to the Gini index is cross-entropy, given by

$$\mathrm{D} = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

- the cross-entropy will take on a value near zero if the $\hat{p}_{mk}$'s are all near zero or near one.

- like the Gini index, the cross-entropy will take on a small value if the *m*th node is pure.

Trees Versus Linear Models

- linear regression assumes a model of the form

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

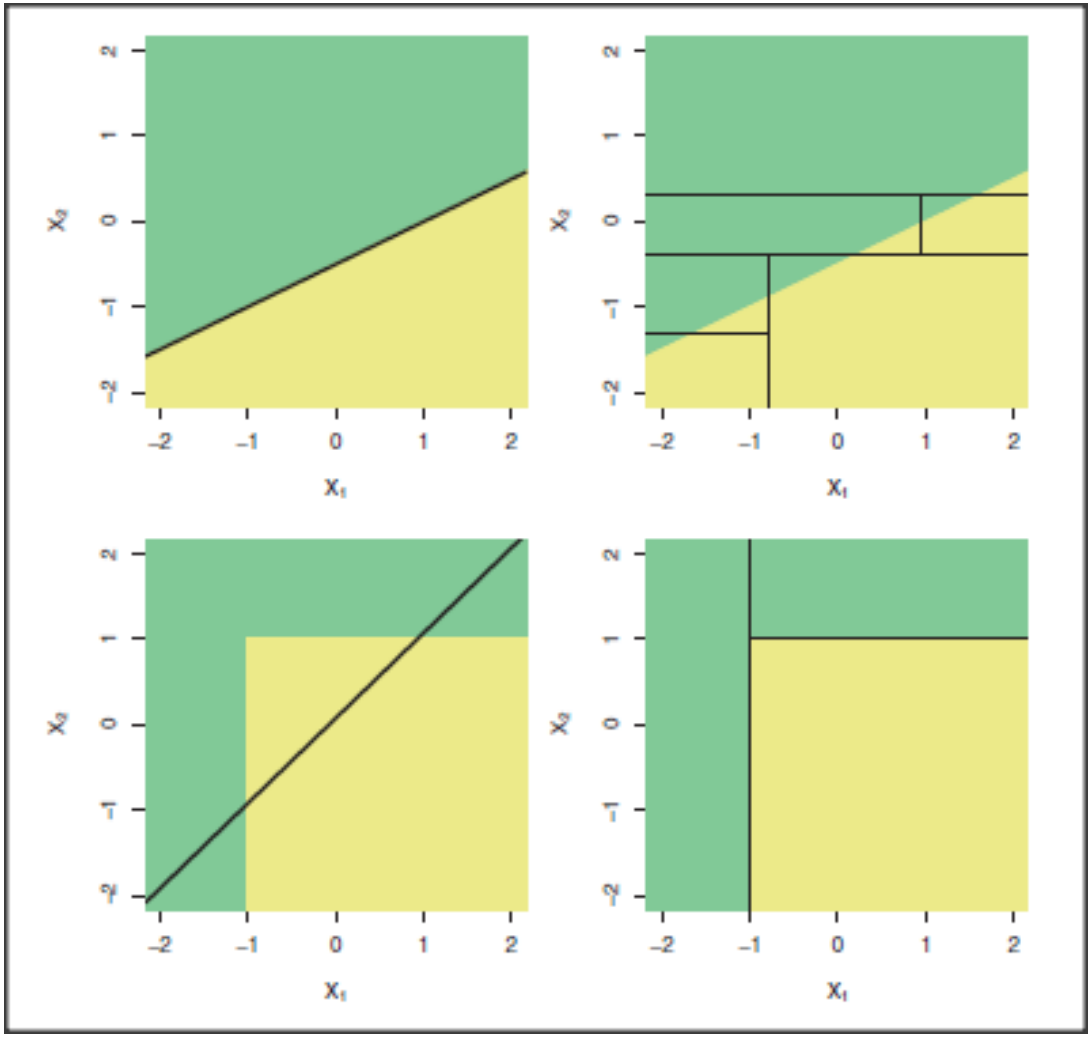
- regression trees assume a model of the form

$$f(X) = \sum_{m=1}^{M} c_m \cdot 1_{(X \in R_m)}$$

where $R_1, \ldots, R_m$ represent a partition of feature space

Trees Versus Linear Models

Which model is better? It depends on the problem!



Advantages and Disadvantages of Trees

Advantages:

- Trees are very easy to explain.
- Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches discussed in previous lectures.
- Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- Trees can easily handle qualitative predictors without the need to create dummy variables.

Disadvantages:

- Trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches.

Improving the predictive performance of trees

Bagging

The decision trees discussed so far suffer from high variance.

- if we split the training data into two parts at random, and fit a decision tree to both halves, the results that we get could be quite different. In contrast, a procedure with low variance will yield similar results if applied repeatedly to distinct data sets;
- linear regression tends to have low variance, if the ratio of n to p is moderately large.
- Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the bagging variance of a statistical learning method.

Improving the predictive performance of trees

Bagging

- Recall: given a set of n independent observations $Z_1, Z_1, \ldots, Z_n$, each with variance σ^2, the variance of the mean $\bar{Z}$ of the observations is given by $\frac{\sigma^2}{n}$. In other words, averaging a set of observations reduces variance.
- Therefore, take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions.
- Calculate $\hat{f}^1(x), \hat{f}^2(x), \ldots, \hat{f}^K(x)$ using K separate training sets and

$$\hat{f}_{avg}(x) = \frac{1}{K}\sum_{k=1}^{K}\hat{f}^k(x)$$

Improving the predictive performance of trees

Bagging

- So far, we used bagging procedure in the regression context, to predict a quantitative outcome *Y*. How can bagging be extended to a classification problem where Y is qualitative?

- For a given test observation, we can record the class predicted by each of the K trees, and take a majority vote: the overall prediction is the most commonly occurring majority class among the B predictions.

Improving the predictive performance of trees

Random Forests

- Random forests provide an improvement over bagged trees by way of decorrelation of the trees. Consider a number forest of decision trees on bootstrapped training samples.

- When building these decision trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of *p* predictors.

- The split is allowed to use only one of those m predictors. A fresh sample of m predictors is taken at each split, and typically we choose $m \approx \sqrt{p}$, the number of predictors considered at each split is approximately equal to the square root of the total number of predictors.

Improving the predictive performance of trees

Random Forests

Why?

- Suppose that there is one very strong predictor in the data set, along with a number of other moderately strong predictors.
- In the collection of bagged trees, most or all of the trees will use this strong predictor in the top split. Consequently, all of the bagged trees will look quite similar to each other and the predictions from the bagged trees will be highly correlated.
- Averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities.

Random forests overcome this problem by forcing each split to consider only a subset of the predictors.

- On average $(p - m)/p$ of the splits will not even consider the strong predictor, and so other predictors will have more of a chance.
- We can think of this process as decorrelating the trees, thereby making the average of the resulting trees less variable and hence more reliable.

Improving the predictive performance of trees

Random Forests

- The main difference between bagging and random forests is the choice of predictor subset size m.

- If a random forest is built using $m = p$, then this amounts simply to bagging. Using a small value of m in building a random forest will typically be helpful when we have a large number of correlated predictors.

Boosting

- Boosting works in a similar way, except that the trees are grown sequentially: each tree is grown using information from previously grown trees. Boosting does not involve bootstrap sampling; instead each tree is fit on a modified version of the original data set.

Improving the predictive performance of trees

Boosting Algorithm

- Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
- For $k = 1, 2, \ldots, K$ repeat:

  - Fit a tree $\hat{f}^k(x)$ with d splits with ($d + 1$ terminal nodes) to the training data (X, r)

  - Update $\hat{f}(x)$ by adding a shrunken version of the new tree:

$$\hat{f}(x) = \hat{f}(x) + \lambda \hat{f}^k(x)$$

  - Update the residuals

$$r_i = r_i - \lambda \hat{f}^k(x_i)$$

- Output the boosted model

$$\hat{f}\ \ (x) = \sum_{k=1}^{K} \lambda \hat{f}^k(x)$$

Improving the predictive performance of trees

Boosting Algorithm

- Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead learns slowly.

- Given the current model, we fit a decision tree to the residuals from the model. That is, we fit a tree using the current residuals, rather than the outcome Y , as the response.

- We then add this new decision tree into the fitted function in order to update the residuals.

- By fitting small trees to the residuals, we slowly improve $\hat{f}$ in areas where it does not perform well. The shrinkage parameter λ slows the process down even further, allowing more and different shaped trees to attack the residuals.

Note: in boosting, unlike in bagging, the construction of each tree depends strongly on the trees that have already been grown.

Unsupervised Learning

Unsupervised learning, a set of statistical tools intended for the setting in which we have only a set of features $X1, X2, \ldots, Xp$ measured on n observations.

- We are not interested in prediction, because we do not have an associated response variable Y .
- The goal is to discover interesting things about the measurements on $X1, X2, \ldots, Xp$.

Focus on two particular types of unsupervised learning:

- *principal components analysis*, a tool used for data visualization or data pre-processing before supervised techniques are applied
- *clustering*, a broad class of methods for discovering unknown subgroups in data.

Principal Components Analysis

Principal component analysis (PCA) refers to the process by which principal components are computed, and the subsequent use of these components in understanding the data.

- PCA is an unsupervised approach, since it involves only a set of features $X1, X2, \ldots, Xp$, and no associated response Y .

What Are Principal Components?

- Suppose that we wish to visualize n observations with measurements on a set of p features, $X_1, X_2, \ldots, X_p$, as part of an exploratory data analysis.

- The idea is that each of the n observations lives in p-dimensional space, but not all of these dimensions are equally interesting.

- PCA seeks a small number of dimensions that are as interesting as possible, where the concept of *interesting* is measured by the amount that the observations vary along each dimension.

Principal Components Analysis

The *first principal component* of a set of features $X_1, X_2, \ldots, X_p$ is the normalized linear combination of the features

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \cdots + \phi_{p1}X_p$$

that has the largest variance.

By *normalized*, we mean that $\sum_{j=1}^{p} \phi_{j1}^2 = 1$

- the elements $\phi_{11}, \cdots, \phi_{p1}$ are the *loadings* of the first principal
- the loadings make up the principal component loading vector:

$$\phi_1 = \left(\phi_{11}\phi_{21} \cdots \phi_{p1}\right)^T$$

Principal Components Analysis

Given a $n \times p$ data set **X**, how do we compute the first principal component?

- Since we are only interested in variance, we assume that each of the variables in **X** has been centered to have mean zero (that is, the column means of **X** are zero). We then look for the linear combination of the sample feature values of the form

$$z_{i1} = \phi_{11} x_{i1} + \phi_{21} x_{i2} + \cdots + \phi_{p1} x_{ip}$$

that has largest sample variance, subject to the constraint that

$$\sum_{j=1}^{p} \phi_{j1}^2 = 1$$

Principal Components Analysis

The first principal component loading vector solves the optimization problem:

$$\underset{\phi_{11},\ldots,\phi_{p1}}{\text{maximize}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1} x_{ij} \right)^2 \right\} \text{ subject to } \sum_{j=1}^{p} \phi_{j1}^2 = 1.$$

After the first principal component *Z*1 of the features has been determined, we can find the second principal component *Z*2.

- it has maximal variance out of all linear combinations that are *uncorrelated* with Z_1

$$z_{i2} = \phi_{12} x_{i1} + \phi_{22} x_{i2} + \cdots + \phi_{p2} x_{ip}$$

and so on

See R Example

Clustering Methods

Clustering refers to a very broad set of techniques for finding *subgroups*, or *clusters*, in a data set.

- Both clustering and PCA seek to simplify the data via a small number of summaries, but their mechanisms are different:

- PCA looks to find a low-dimensional representation of the observations that explain a good fraction of the variance

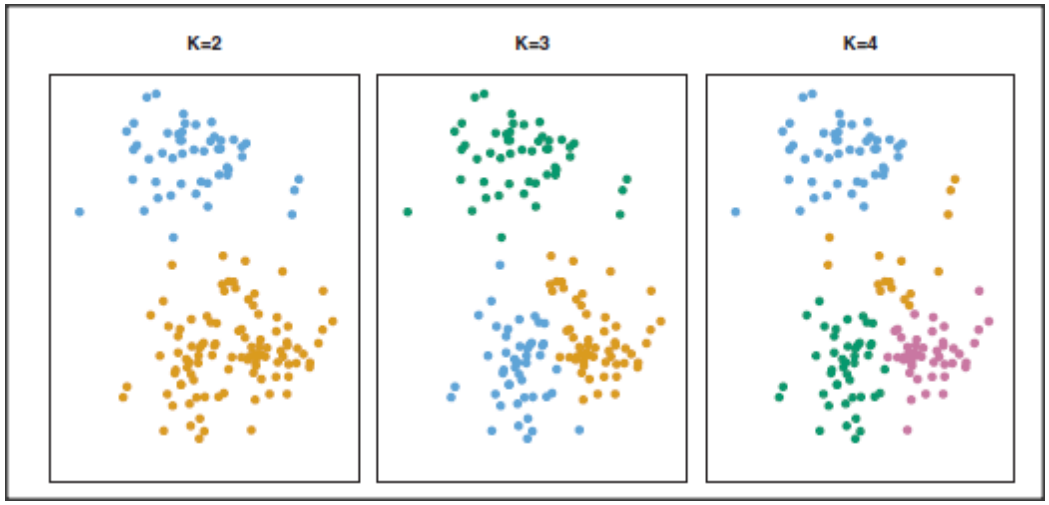
- Clustering looks to find homogeneous subgroups among the observations.

K-Means Clustering

K-means clustering is a simple and elegant approach for partitioning a data set into *K* distinct, non-overlapping clusters. To perform *K*-means clustering:

- we must first specify the desired number of clusters *K*
- then the *K*-means algorithm will assign each observation to exactly one of the *K* clusters.

The *K*-means clustering procedure results from a simple and intuitive mathematical problem. We begin by defining some notation. Let $C_1, \ldots, C_K$ denote sets containing the indices of the observations in each cluster. These sets satisfy two properties:



K-Means Clustering

1. $C_1 \cup C_2 \cup \quad \cdots C_k = \{1, \cdots, n\}$
each observation belongs to at least one of the *K* clusters.

2. $C_1 \cap C_2 = \emptyset$ for all $k \neq k'$.
the clusters are nonoverlapping: no observation belongs to more than one cluster.

- The idea behind *K*-means clustering is that a *good* clustering is one for which the *within-cluster variation* is as small as possible.
- The within-cluster variation for cluster C_k is a measure $W(C_k)$of the amount by which the observations within a cluster differ from each other.

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} W(C_k) \right\}$$

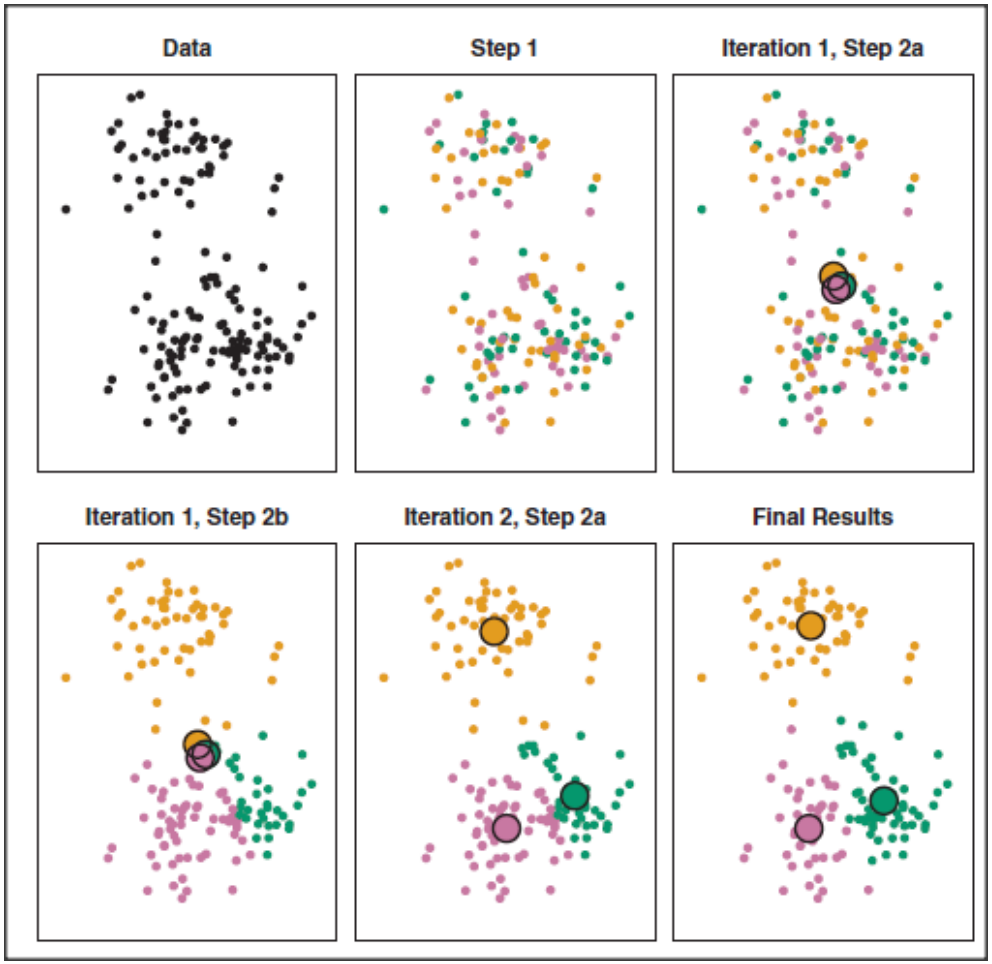
K-Means Clustering

Algorithm *K-Means Clustering*

1. Randomly assign a number, from 1 to *K*, to each of the observations. These serve as initial cluster assignments for the observations.
2. Iterate until the cluster assignments stop changing:
   - For each of the *K* clusters, compute the cluster *centroid*. The *k*th cluster centroid is the vector of the *p* feature means for the observations in the *k*th cluster.
   - Assign each observation to the cluster whose centroid is closest (where *closest* is defined using Euclidean distance).

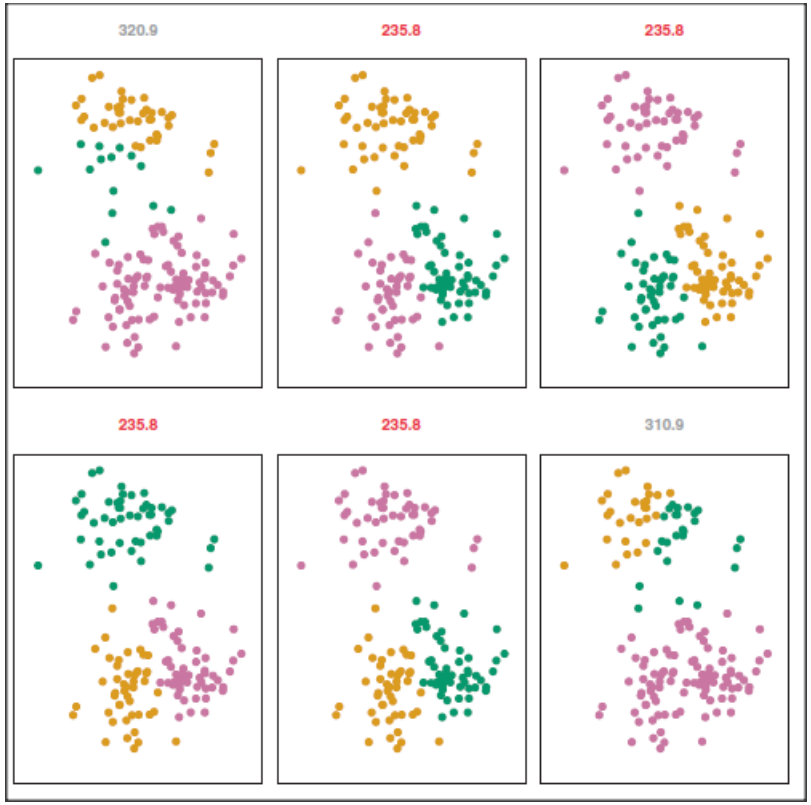
Because the *K*-means algorithm finds a local rather than a global optimum, the results obtained will depend on the initial (random) cluster assignment of each observation in Step 1 of Algorithm. For this reason, it is important to run the algorithm multiple times from different random initial configurations.

K-Means Clustering



K-Means Clustering

Figure shows the local optima obtained by running K-means clustering six times using six different initial cluster assignments, using the data from previous slides. In this case, the best clustering is the one with an objective value of 235.8.



Hierarchical Clustering

One potential disadvantage of *K*-means clustering is that it requires us to pre-specify the number of clusters *K*.

Hierarchical clustering is an alternative approach which does not require that we commit to a particular choice of *K*.

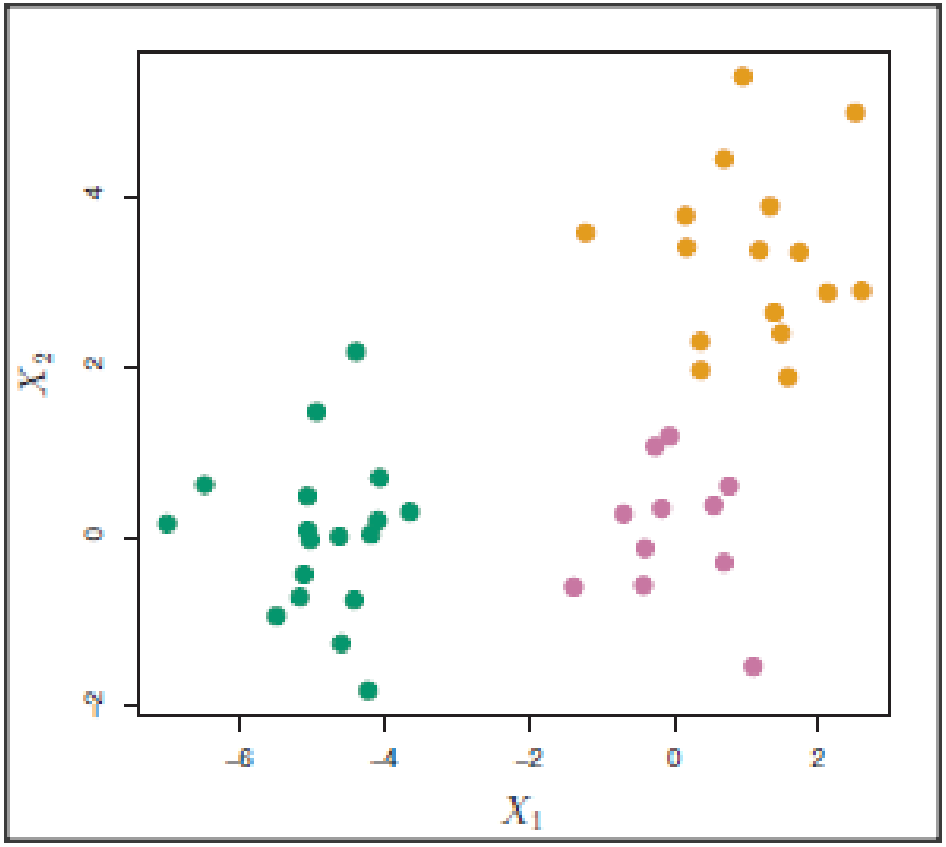
- Hierarchical clustering has an added advantage over *K*-means clustering in that it results in an attractive tree-based representation of the observations, called a *dendrogram*.

- In this section, we describe *bottom-up* or *agglomerative* clustering. The dendrogram (generally depicted as an upside-down tree; is built starting from the leaves and combining clusters up to the trunk).

Hierarchical Clustering

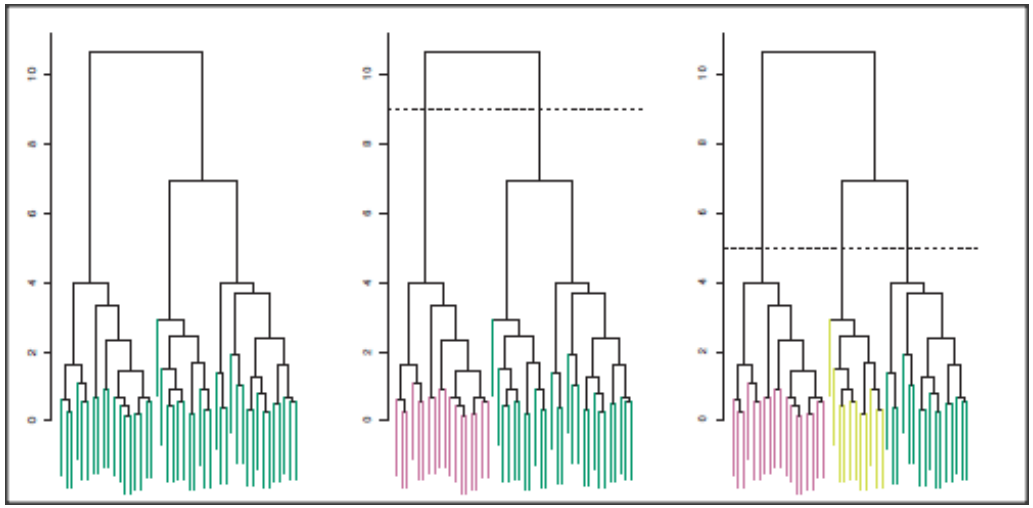
Interpreting a Dendrogram

Consider the data (45 observations in two-dimensional space.)



Hierarchical Clustering

Hierarchical clustering (with complete linkage, to be discussed later) yields the result shown in the left-hand panel of Figure.

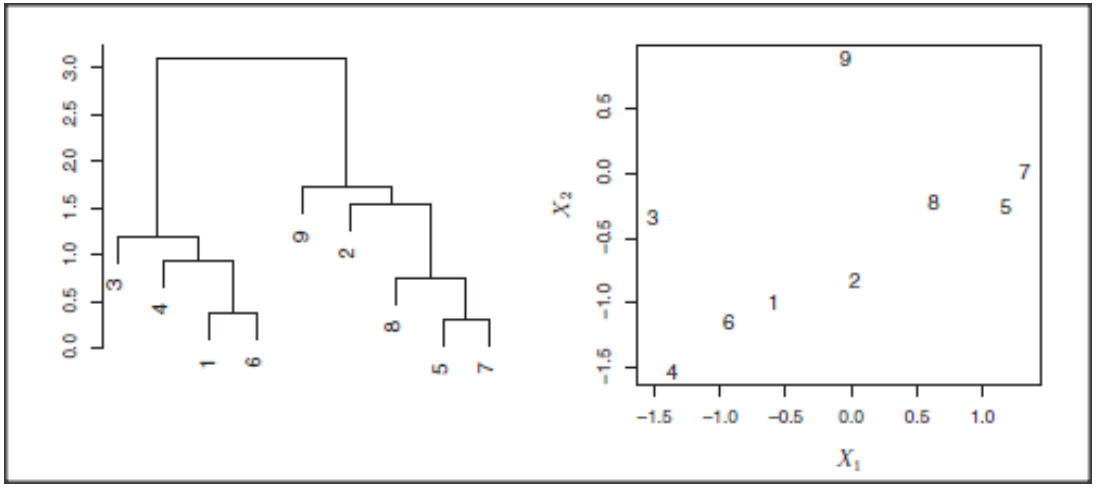


- Each *leaf* of the dendrogram represents one of the 45 observations. However, as we move up the tree, some leaves begin to *fuse* into branches.

Hierarchical Clustering

- The earlier (lower in the tree) fusions occur, the more similar the groups of observations are to each other.
- On the other hand, observations that fuse later (near the top of the tree) can be quite different.
- In fact, this statement can be made precise: for any two observations, we can look for the point in the tree where branches containing those two observations are first fused.
- Thus, observations that fuse at the very bottom of the tree are quite similar to each other, whereas observations that fuse close to the top of the tree will tend to be quite different.

Hierarchical Clustering



- Observations 5 and 7 are quite similar to each other, since they fuse at the lowest point on the dendrogram.
- Observations 1 and 6 are also quite similar to each other.

- It is tempting but incorrect to conclude from the figure that observations 9 and 2 are quite similar to each other on the basis that they are located near each other on the dendrogram.

- In fact, based on the information contained in the dendrogram, observation 9 is no more similar to observation 2 than it is to observations 8, 5, and 7.

Hierarchical Clustering

- The issue of identifying clusters on the basis of a dendrogram. In order to do this, we make a horizontal cut across the dendrogram, as shown in the center and right-hand panels of previous figure.

- The distinct sets of observations beneath the cut can be interpreted as clusters.

- One single dendrogram can be used to obtain any number of clusters.

Hierarchical Clustering

Algorithm *Hierarchical Clustering*

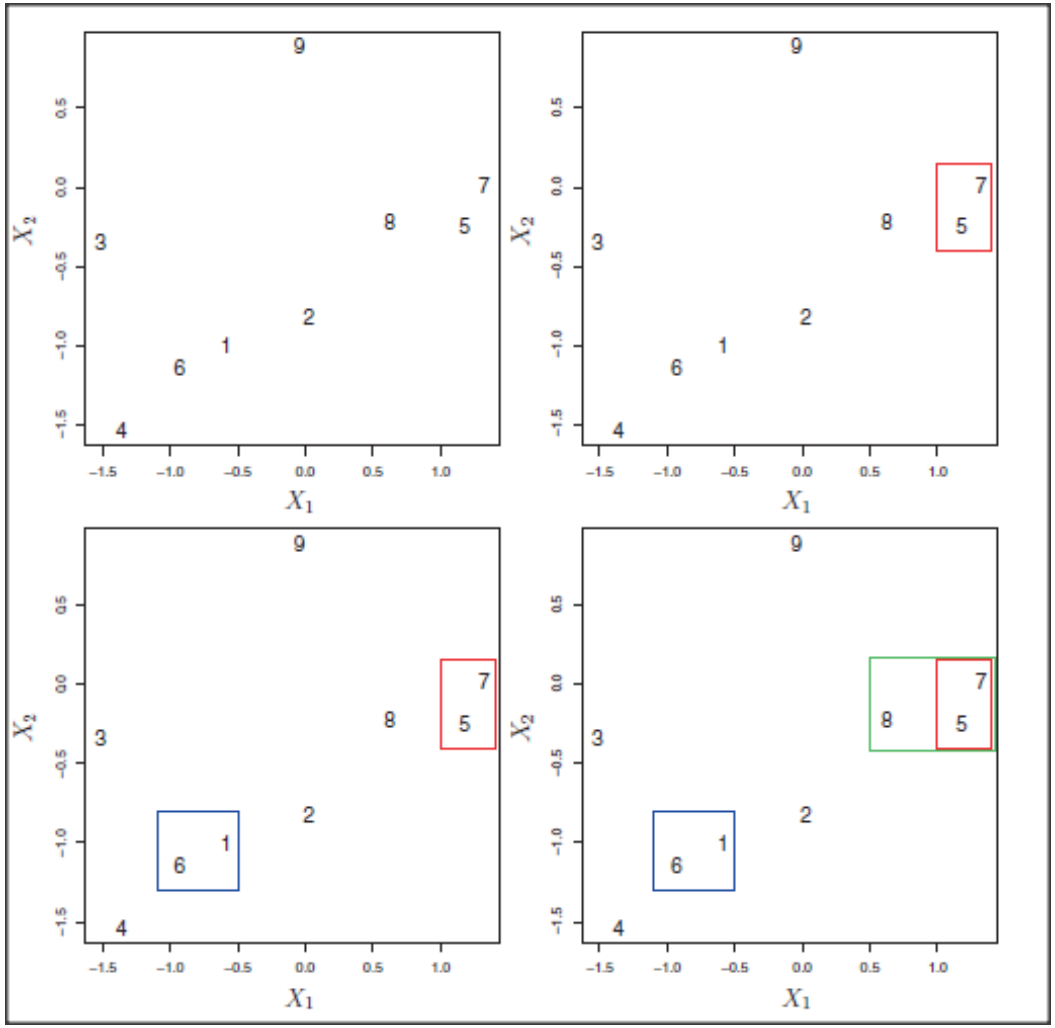
1. Begin with n observations and a measure (such as Euclidean distance) of all the $\binom{n}{2} = \frac{n(n-1)}{2}$ pairwise dissimilarities. Treat each observation as its own cluster.

2. For $i = n, n-1, \ldots, 2$:

- Examine all pairwise inter-cluster dissimilarities among the I clusters and identify the pair of clusters that are least dissimilar (that is, most similar). Fuse these two clusters. The dissimilarity between these two clusters indicates the height in the dendrogram at which the fusion should be placed.

- Compute the new pairwise inter-cluster dissimilarities among the $i-1$ remaining clusters.

Hierarchical Clustering



Thank You

Dragos Bozdog

For academic use only.