Session 5

49-781 Data Analytics for Product Managers

Spring 2018

Topics

1. Nonlinear Regression
    a. Regression Splines
    b. Local Regression
    c. Generalized Additive Models
2. Tree-based Methods
    a. Regression Trees
    b. Classification Trees
    c. Bagging
    d. Random Forests
3. Unsupervised Learning
    a. Principal Component Analysis
    b. K-means Clustering
    c. Hierarchical Clustering

Nonlinear Regression

Limitations of Linear Models

Linear models are

- Relatively simple to describe and implement
- Easy to interpret.

Standard linear regression can have limitations in terms of predictive power.

Linearity assumption is almost always an approximation, and sometimes a poor one.

Relaxing Linearity

We can relax the linearity assumption while still attempting to maintain as much interpretability as possible.

1. We can use very simple extensions of linear models like polynomial regression and step functions
2. Incorporate more sophisticated approaches such as splines, local regression, and generalized additive models.

Polynomial Regression

The standard way is to replace the standard linear model

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

with a polynomial function

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \ldots + \beta_d x_i^d + \epsilon_i$$

This approach is known as polynomial regression (We worked on this earlier as Numerical Transformations)

Polynomial Regression

For large regression enough degree d, a polynomial regression allows us to produce an extremely non-linear curve.

The coefficients in the equation can be easily estimated using least squares linear regression because this is just a standard linear model with predictors

$$x_i, x_i^2, x_i^3, \ldots, x_i^d$$

It is unusual to use d greater than 3 or 4 because for large values of d, the polynomial curve can become overly flexible and can take on some very strange shapes. This is especially true near the boundary of the X variable. (You can recall the examples from earlier classes)

Step Functions

Using polynomial functions of the features as predictors in a linear model imposes a *global structure* on the *non-linear* function of X.

We can instead use *step functions* in order to avoid imposing such a global structure.

We break the range of X into *bins*, and fit a different *constant* in each bin. This amounts to converting a continuous variable into an ordered categorical variable.

Step Function

We create cutpoints c_1, c_2,...,c_K in the range of X, and then construct $K + 1$ new variables

$$
\begin{aligned}
C_0(X) &= I(X < c_1), \\
C_1(X) &= I(c_1 \leq X < c_2), \\
C_2(X) &= I(c_2 \leq X < c_3), \\
&\vdots \\
C_{K-1}(X) &= I(c_{K-1} \leq X < c_K), \\
C_K(X) &= I(c_K \leq X),
\end{aligned}
$$

where $I(\cdot)$ is an indicator function that returns a 1 if the condition is true, and returns a 0 otherwise. X must be in exactly one of the $K + 1$ intervals.

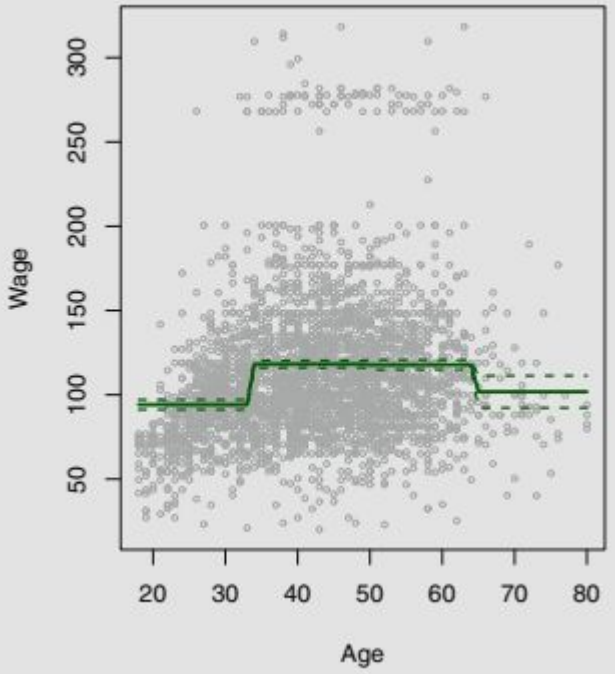
$$y_i = \beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + \ldots + \beta_K C_K(x_i) + \epsilon_i.$$

Step Function

Unless there are natural breakpoints in the predictors, piecewise-constant functions can miss the action.

Example: The first bin clearly misses the increasing trend of wage with age.

Still, step function approaches are very popular in biostatistics and epidemiology, among other disciplines. For example, 5-year age groups are often used to define the bins.



Basis Functions

Basis functions is a generalization of polynomial and step functions. You can apply a family of functions to a variable X

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \beta_3 b_3(x_i) + \ldots + \beta_K b_K(x_i) + \epsilon_i.$$

For polynomial regression, the basis functions are $b_j(x_i) = x_i^j$, and for piecewise constant functions they are $b_j(x_i) = I(c_j \leq x_i < c_{j+1})$.

Nonlinear Regression

Regression Splines

Piecewise Polynomials

Instead of fitting a high-degree polynomial over the entire range of X, piecewise polynomial regression involves fitting separate low-degree polynomials piecewise polynomial regression over different regions of X.

For example, a piecewise cubic polynomial works by fitting a cubic regression model of the form

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \epsilon_i,$$

where the coefficients β_0, β_1, β_2, *and* β_3 differ in different parts of the range of X. The points where the coefficients change are called *knots*.

Piecewise Polynomials

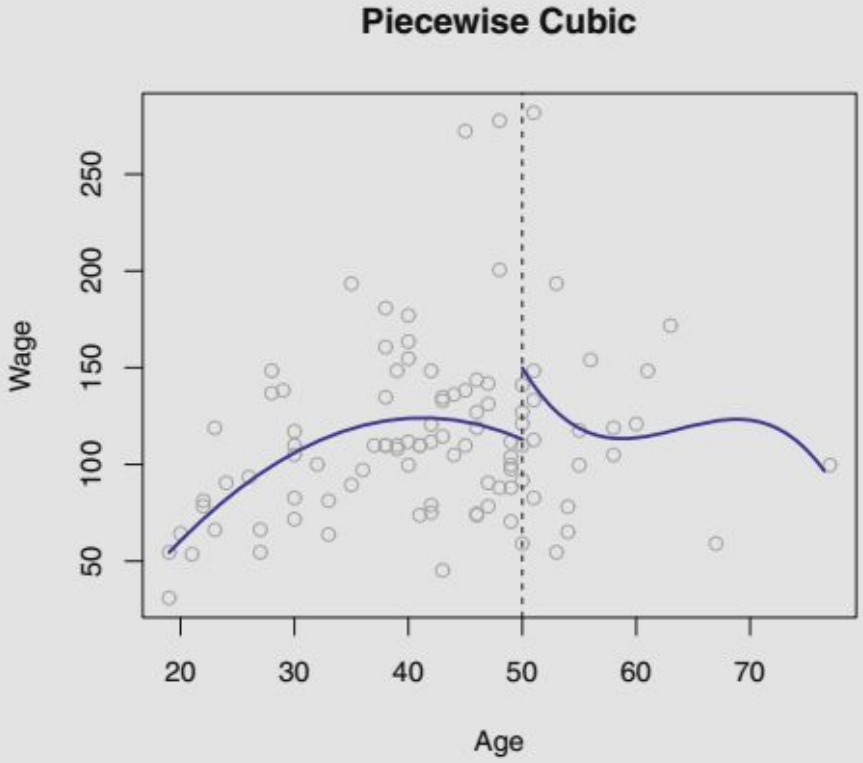
For example, a piecewise *cubic* with no knots is just a standard cubic polynomial. A piecewise cubic polynomial with a single knot at a point c takes the form

$$y_i = \begin{cases} \beta_{01} + \beta_{11} x_i + \beta_{21} x_i^2 + \beta_{31} x_i^3 + \epsilon_i & \text{if } x_i < c; \\ \beta_{02} + \beta_{12} x_i + \beta_{22} x_i^2 + \beta_{32} x_i^3 + \epsilon_i & \text{if } x_i \geq c. \end{cases}$$

Each of these polynomial functions can be fit using least squares applied to simple functions of the original predictor.

Using more knots leads to a more flexible piecewise polynomial. In general, if we place K different knots throughout the range of X, then we will end up fitting K + 1 different cubic polynomials.

Piecewise Cubic Example



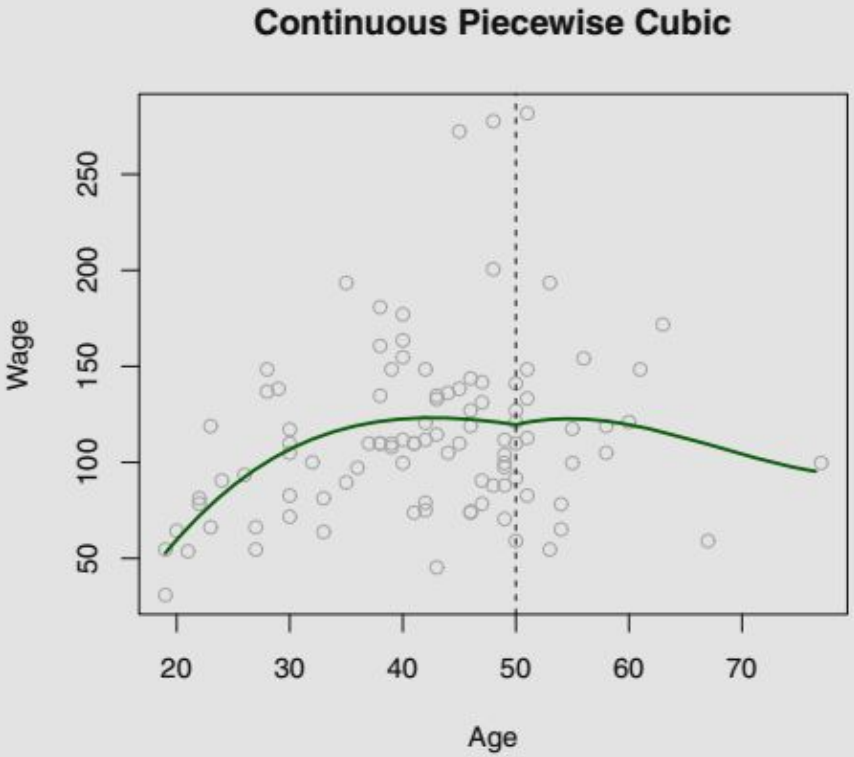
We use a knot at 50.

This fit is not continuous at 50 - a behavior that may not be desired.

Constrained Piecewise Cubic

We can fit a piecewise polynomial under the constraint that the fitted curve must be continuous.

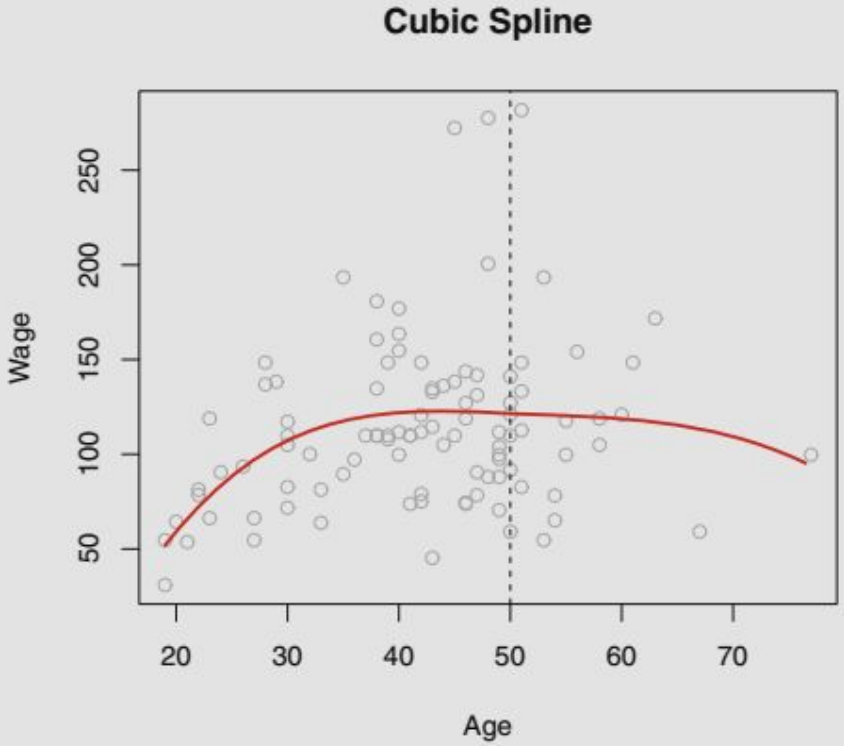
You still have a v-shape join - so we probably can improve this further.



Additional Constraints

We have added two additional constraints:
now both the first and second derivatives of the piecewise polynomials are continuous
derivative at age=50.

It's now continuous and smooth.



Placing the knots

The regression spline is most flexible in regions that contain a lot of knots, because in those regions the polynomial coefficients can change rapidly.

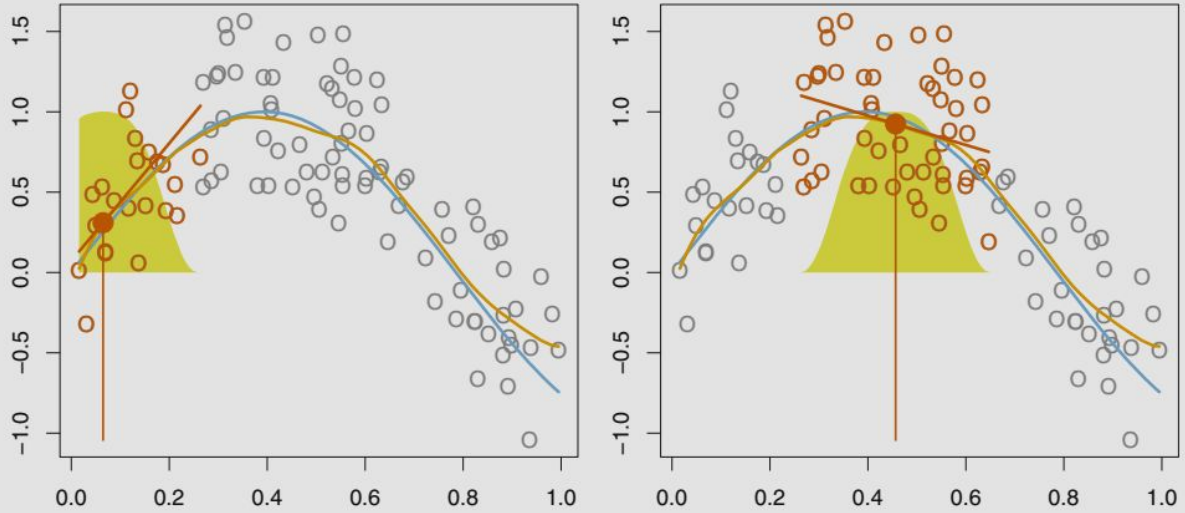
Hence, one option is to place more knots in places where we feel the function might vary most rapidly, and to place fewer knots where it seems more stable.

Nonlinear Regression

Local Regression

Local Regression

Local regression is a different approach for fitting flexible non-linear func- local tions, which involves *computing the fit at a target point* x_0 *using only the regression on nearby training observations.*



Local Regression Algorithm

1. Gather the fraction $s = k/n$ of training points whose x_i are closest to x_0.

2. Assign a weight $K_{i0} = K(x_i, x_0)$ to each point in this neighborhood, so that the point furthest from x_0 has weight zero, and the closest has the highest weight. All but these k nearest neighbors get weight zero.

3. Fit a *weighted least squares regression* of the y_i on the x_i using the aforementioned weights, by finding $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize

$$\sum_{i=1}^{n} K_{i0}(y_i - \beta_0 - \beta_1 x_i)^2.$$

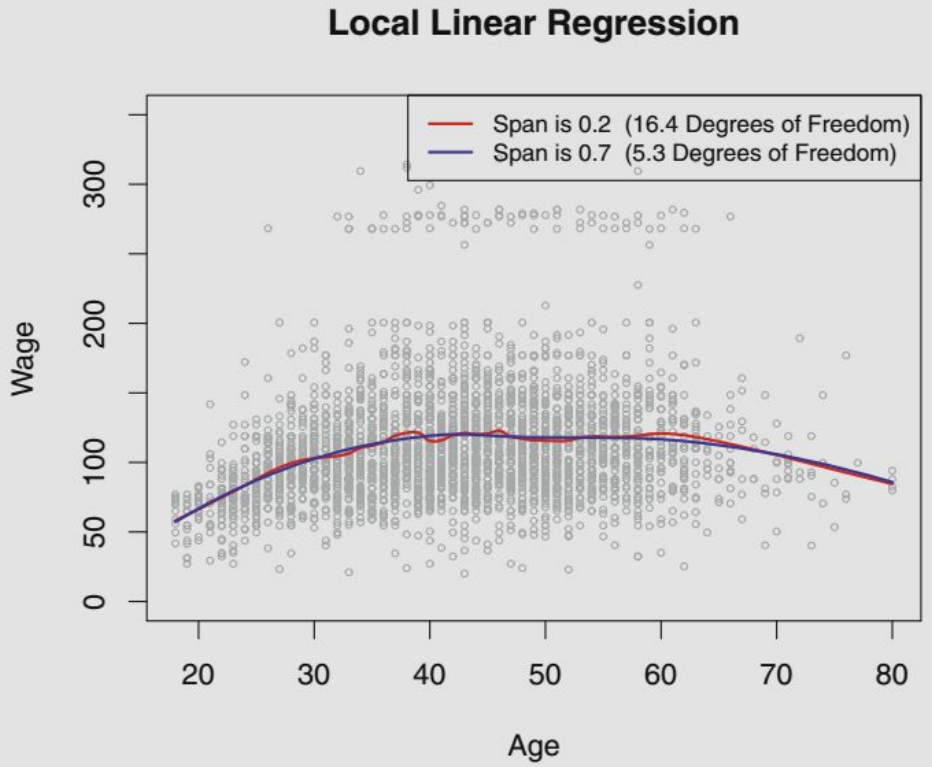
4. The fitted value at x_0 is given by $\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$.

Local Regression

In order to perform local regression, there are a number of choices to be made, such as how to define the weighting function K, and whether to fit a linear, constant, or quadratic regression in Step 3. While all of these choices make some difference, the most important choice is the span *s*,

Local Regression - Example

A smaller span gives a more flexible fit.



Nonlinear Regression

Generalized Additive Models

Generalized Additive Models

These approaches can be seen as extensions of simple linear regression. Here we explore the problem of flexibly predicting Y on the basis of several predictors, $X_1,...,X_p$. This amounts to an extension of multiple linear regression.

GAMs provide a general framework for extending a standard linear model by allowing non-linear functions of each of the variables, while maintaining additivity.

GAMs additivity can be applied with both quantitative and qualitative responses.

GAM

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \epsilon_i$$

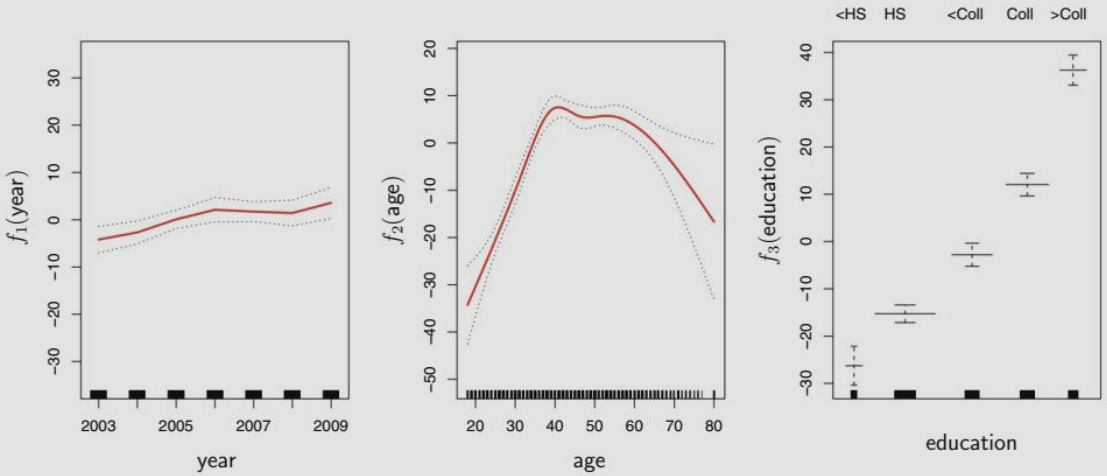
in order to allow for non-linear relationships between each feature and the response is to replace each linear component $\beta_j x_{ij}$ with a (smooth) nonlinear function $f_j(x_{ij})$. We would then write the model as

$$\begin{aligned} y_i &= \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i \\ &= \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon_i. \end{aligned}$$

GAM Example

It is called an additive model because we calculate a separate f_j for each Xj, and then add together all of their contributions.

$$\texttt{wage} = \beta_0 + f_1(\texttt{year}) + f_2(\texttt{age}) + f_3(\texttt{education}) + \epsilon$$



About GAM

GAMs allow us to fit a non-linear f_j to each X_j, so that we can automatically model non-linear relationships that standard linear regression will miss.

This means that we do not need to manually try out many different transformations on each variable individually. GAM libraries can do that automatically for you.

The non-linear fits can potentially make more accurate predictions for the response Y.

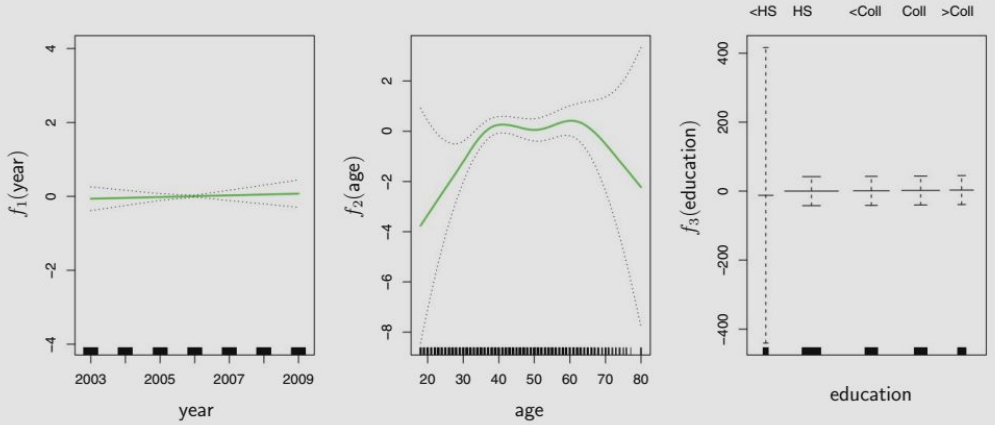
Because the model is additive, we can still examine the effect of each X_j on Y individually while holding all of the other variables fixed making inference easier.

GAMs for Classification

We fit a GAM to the Wage data in order to predict the probability that an individual's income exceeds $250,000 per year. The GAM that we fit takes the form

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 \times \texttt{year} + f_2(\texttt{age}) + f_3(\texttt{education})$$

f_1: year, linear
f_2 : age, smoothing spline
f_3: education, step function



Tree Based Methods

Trees

We can also use tree-based methods for regression and classification.

You stratify or segment the predictor space into a number of simple regions. In order to make a prediction for a given observation, we typically use the *mean* or the *mode* of the training observations in the region to which it belongs.

Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as *decision tree methods*.

Trees

Tree-based methods are simple and useful for interpretation. However, they typically are not competitive with the best supervised learning approaches in terms of *prediction accuracy*. (There are approaches to improve this, though)

Decision trees can be applied to both regression (Regression Trees) and classification problems (Classification Trees).

Tree Based Methods

Regression Trees

Regression Trees

When the outcome variable is numeric, these methods are said to generate Regression Trees.

Example: We use a data set to predict a baseball player's *Salary* based on *Years* (the number of years that he has played in the major leagues) and *Hits* (the number of hits that he made in the previous year).

(For this discussion, *Salary* is log-transformed to get a better distribution)

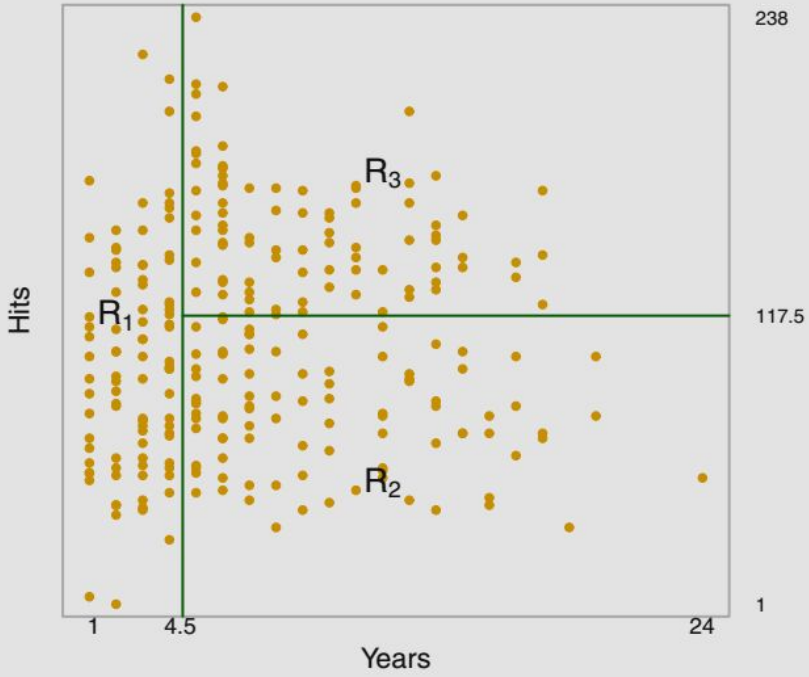
Building a Tree

Building a tree consists of a series of *splitting rules*, starting at the top of the tree.

Example: The top split assigns observations having *Years*<4.5 to the left branch

For these players, Salary is given by the mean response value for the players in the data set with Years<4.5.

For such players, the mean log salary is 5.107, and so we make a prediction of $e^{5.107}$ thousands of dollars, i.e. $165,174

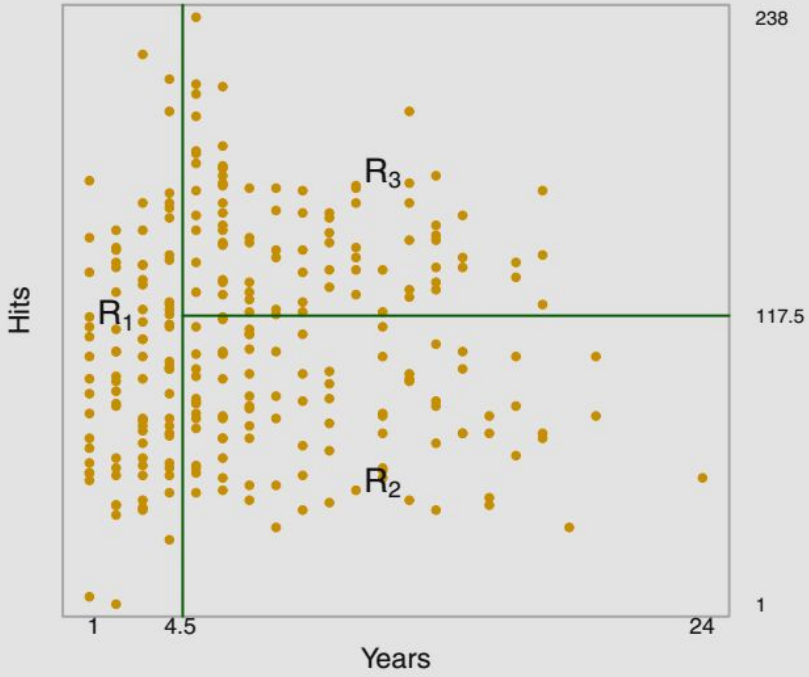


Continue Building (Splitting)

Players with Years>=4.5 are assigned to the right branch, and then that group is further subdivided by Hits.

Overall, the tree stratifies or segments the players into three regions of predictor space:

- players who have played for four or fewer years,
- players who have played for five or more years and who made fewer than 118 hits last year, and
- players who have played for five or more years and who made at least 118 hits last year.



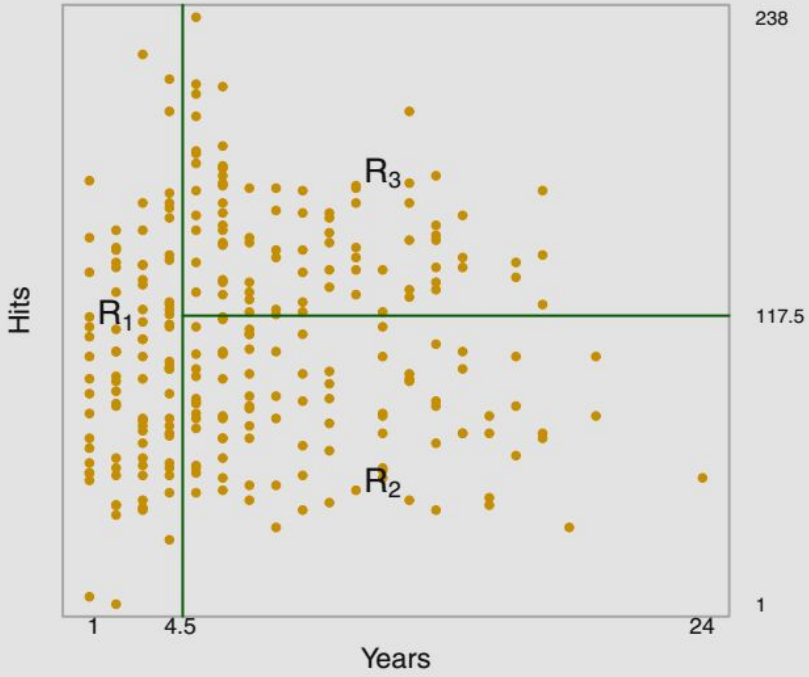
Meaning of the Tree

These three regions can be written as

R1 ={X | Years<4.5},
R2 ={X | Years>=4.5, Hits<117.5}, and
R3 ={X | Years>=4.5, Hits>=117.5}.

The predicted salaries for these three groups are

- $\$1{,}000 \times e^{5.107} = \$165{,}174$,
- $\$1{,}000 \times e^{5.999} = \$402{,}834$,
- $\$1{,}000 \times e^{6.740} = \$845{,}346$

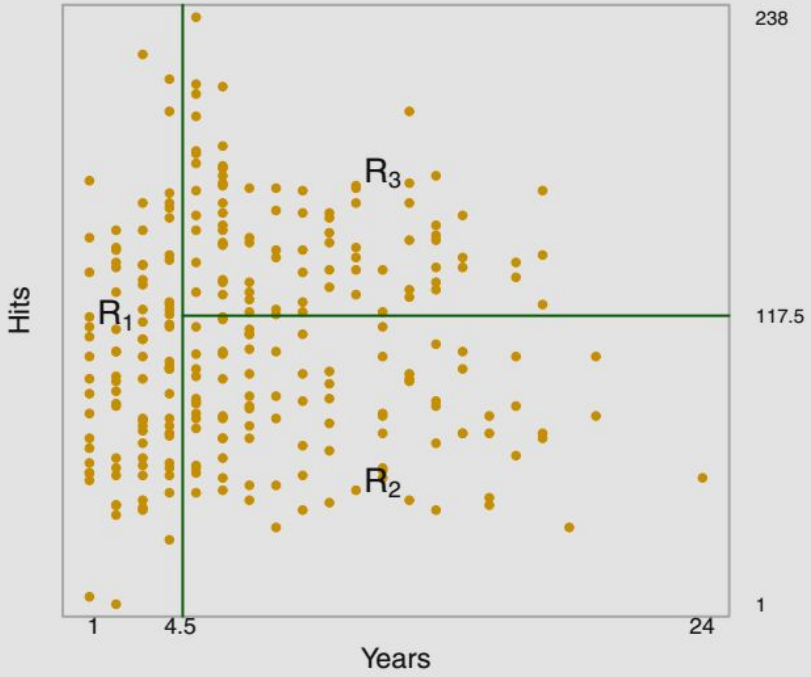


Interpreting the Tree

We might interpret the regression tree displayed as follows:

Years is the most important factor in determining Salary, and players with less experience earn lower salaries than more experienced players. Given that a player is less experienced, the number of hits that he made in the previous year seems to play little role in his salary.
But among players who have been in the major leagues for five or more years, the number of hits made in the previous year does affect salary, and players who made more hits last year tend to have higher salaries.

This is likely an *over-simplification* of the true relationship between Hits, Years, and Salary. It does have advantages over other types of regression models: it is easier to interpret, and has a nice graphical representation.



Tree Based Methods

Classification Trees

Classification Tree

A classification tree is very similar to a regression tree, except that it is 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, we might be 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.

Building a Classification Tree

Just as in the regression setting, we use recursive binary splitting to grow a classification tree. However, RSS cannot be used as a criterion for making the binary splits.

A natural alternative to is the classification error rate. Since we plan classification to assign an observation in a given region to the most commonly occurring training observations in that region, the classification error rate is simply the fraction of the training observations in that region that do not belong to the most common class

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

Here p_{mk} represents the proportion of training observations in the mth region that are from the kth class.

Better Error Sensitivity

Classification error is not sufficiently sensitive for tree-growing, and in practice other measures are preferable. A Gini index is defined by

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

This is a measure of total variance across the K classes. Gini index takes on a small value if all of the p_{mk}'s are close to zero or one.

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.

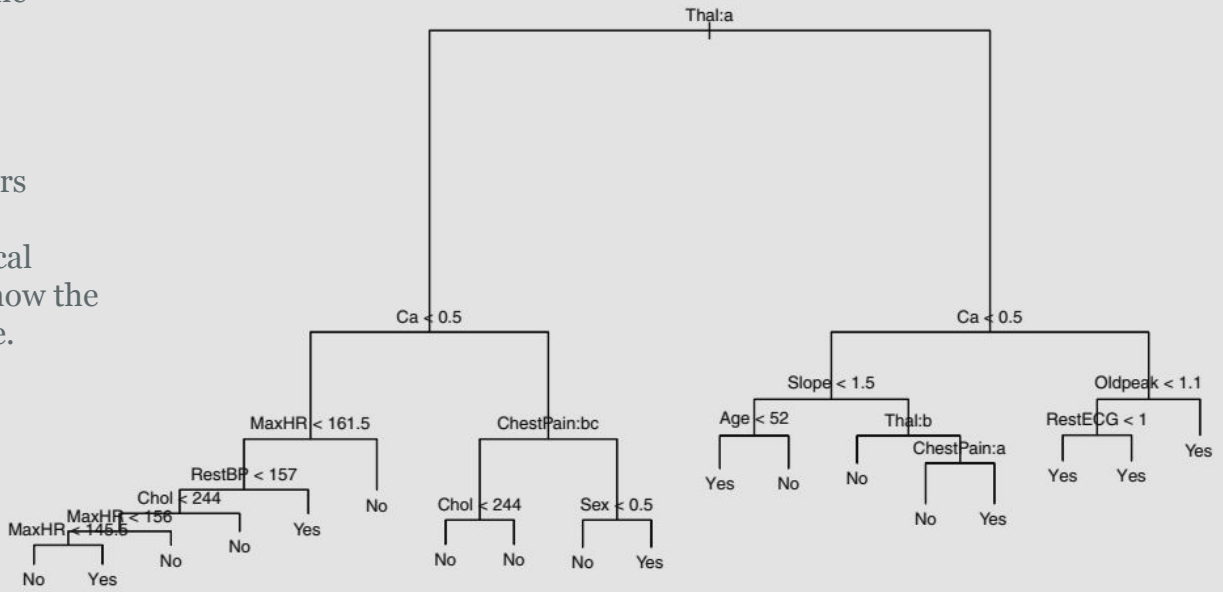
Classification Tree

Binary outcome Yes/No for the variable HD (Heart Disease)

Yes indicates presence of HD

Training data has 13 predictors

This tree also shows categorical variables, the letters *a, b, c* show the levels selected for the left tree.

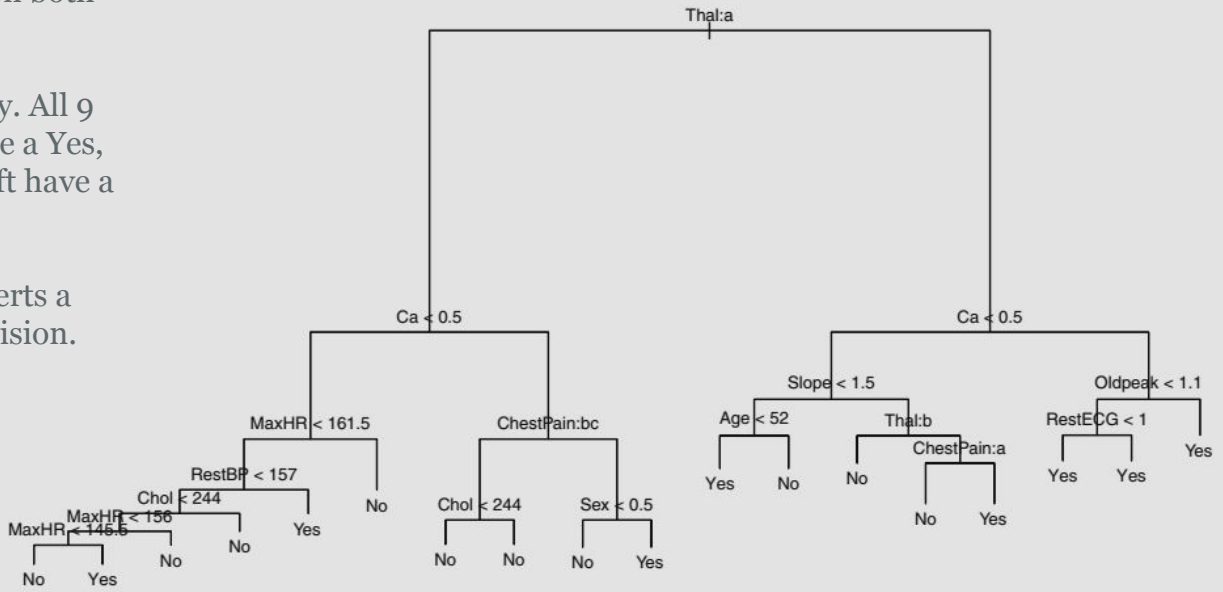


Classification Tree

Notice RestECG < 1 has Yes on both splits.

This is to increase node purity. All 9 observations on the right have a Yes, while the 7 out of 11 on the left have a Yes.

Why is this important? It asserts a confidence in the Yes/No decision.



Tree vs Linear Models

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

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

Where $R_1,\ldots,R_M$ represents a partition in feature space and c_m is the value of the outcome in that partition.

Linear Models or Trees for Classification?

Which model is better? It depends on the problem at hand.

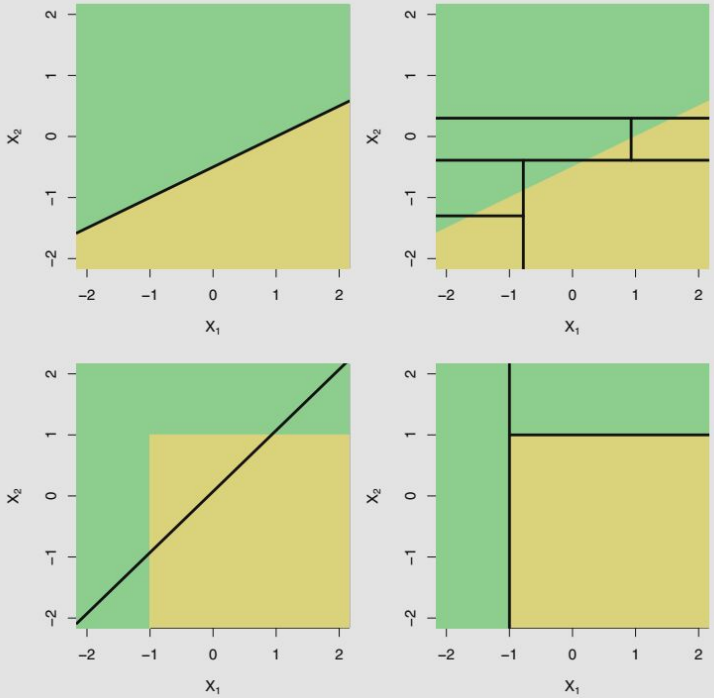
If the relationship between the features and the response is well approximated by a linear model, then an approach such as linear regression will likely work well, and will outperform a method such as a regression tree that does not exploit this linear structure.

If instead there is a highly non-linear and complex relationship between the features and the response, then decision trees may outperform classical approaches.

A Visual Comparison

A two-dimensional classification example in which the true decision boundary is linear, and is indicated by the shaded regions. A classical approach that assumes a linear boundary (left) will outperform a decision tree that performs splits parallel to the axes (right).

Here the true decision boundary is non-linear. Here a linear model is unable to capture the true decision boundary (left), whereas a decision tree is successful (right).



Benefits of Trees

Trees are very easy to explain to people - easier to explain than linear regression!

Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous chapters.

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 of Trees

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

Trees can be very non-robust. In other words, a small change in the data can cause a large change in the final estimated tree.

However, by aggregating many decision trees, using methods like bagging, random forests, and boosting, the predictive performance of trees can be substantially improved. We introduce some of these concepts in this lecture.

Tree Based Methods

Bagging

Challenges with Decision Trees

Decision trees 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.

Techniques to improve Trees

Bootstrap aggregation, or *bagging*, is a general-purpose procedure for reducing the variance of a statistical learning method; it is particularly useful and frequently used in the context of decision trees.

Key Premise:
Averaging a set of observations reduces variance. Hence a natural way to reduce the variance and hence increase the prediction accuracy of a statistical learning method is to *take many training sets from the population*, b*uild a separate prediction model using each training set, and average the resulting predictions*.

Building Multiple Trees

In other words, we could calculate $f_1(x), f_2(x), \ldots, f_B(x)$ using B separate training sets, and average them in order to obtain a single low-variance statistical learning model,

$$\hat{f}_{\text{avg}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^b(x).$$

This is not practical because we generally do not have access to multiple training sets. Instead, we can *bootstrap*, by taking repeated samples from the (single) training data set. *Bootstrapping is a sampling method.*

Bagging

Bagging = bootstrap aggregation

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

Bagging can improve predictions for many regression methods, but it is particularly useful for decision trees.

Construct B regression trees using B bootstrapped training sets, and average the resulting predictions.

Each individual tree has high variance. Averaging these B trees reduces the variance. Bagging has been demonstrated to give impressive improvements in accuracy by combining together hundreds or even thousands of trees into a single procedure.

Tree Based Methods

Random Forests

Random Forests

Random forests provide an improvement over bagged trees by way of a random small tweak that decorrelates the trees.

As in bagging, we build a number forest of decision trees on bootstrapped training samples. But 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. Typically we choose $m \approx \sqrt{p}$

Random Forests

At each split in the tree, the algorithm is not even allowed to consider a majority of the available predictors. *This may sound crazy, but it has a clever rationale*.

Eliminates the influence of one very strong predictor in the data set (even if there are other moderately strong predictors.)

In the collection of bagged trees, most or all of the trees will use a strong predictor in the top split, and all of the bagged trees will look quite similar to each other. Predictions from the bagged trees will be highly correlated.

This eliminates the fundamental purpose of bagging.

Random Forests

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.

The main difference between bagging and random forests is the choice of predictor subset size m. For instance, if a random forest is built using m = p, then this amounts simply to bagging.

Unsupervised Learning

Unsupervised Learning

Most of this course concerns *supervised learning methods* such as regression and classification. In the supervised learning setting, we typically have access to a set of p features $X_1, X_2,...,X_p$, measured on n observations, and a response Y also measured on those same n observations. The goal is then to predict Y using $X_1, X_2,...,X_p$.

In *unsupervised learning*, we have only a set of features $X_1, X_2,...,X_p$ measured on n observations. We are not interested in prediction, because we do not have an associated response variable Y. Rather, the goal is to discover interesting things about the measurements on $X_1, X_2,...,X_p$.

Goals of Unsupervised Learning

Unsupervised learning refers to a diverse set of techniques for answering questions such as

- Is there an informative way to visualize the data?
- Can we discover subgroups among the variables or among the observations?

Our focus: *principal components analysis*, for data visualization or data pre-processing before supervised techniques are applied, and *clustering*, methods for discovering unknown subgroups in data.

Supervised Learning is Easier

If you are asked to predict a binary outcome from a data set, we have very well developed set of tools (such as logistic regression) as well as a clear understanding of how to assess the quality of the results obtained (using cross-validation, validation on an independent test set).

Unsupervised learning is more subjective, and there is no simple goal for the analysis, such as prediction of a response. And there are no true mechanisms for doing cross-validation etc.

Applications of Unsupervised Learning

- A **cancer researcher** might assay gene expression levels in 100 patients with breast cancer.
  - He or she might then look for subgroups among the breast cancer samples, or among the genes, in order to obtain a better understanding of the disease.

- An **online shopping site** might try to identify groups of shoppers with similar browsing and purchase histories, as well as items that are of particular interest to the shoppers within each group.
  - Then an individual shopper can be preferentially shown the items in which he or she is particularly likely to be interested, based on the purchase histories of similar shoppers.

- A **search engine** might choose what search results to display to a particular individual based on the click histories of other individuals with similar search patterns.
  - These statistical learning tasks, and many more, can be performed via unsupervised learning techniques.

Unsupervised Learning

Principal Component Analysis

Principal Component 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.

Principal Components

Suppose that we wish to visualize n observations with measurements on a set of p features, *X1, X2,...,Xp*, as part of an exploratory data analysis.

We could do this by examining two-dimensional scatterplots of the data, each of which contains the n observations' measurements on two of the features.

There are p 2 = p(p−1)/2 such scatterplots; for example, with p = 10 there are 45 plots!

If p is large, then it will certainly not be possible to look at all of them; moreover, most likely none of them will be informative since they each contain just a small fraction of the total information present in the data set.

Principal Component Analysis

A better method is required to visualize the n observations when p is large. In particular, we would like to find a low-dimensional representation of the data that captures as much of the information as possible.

If we can obtain a two-dimensional representation of the data that captures most of the information, then we can plot the observations in this low-dimensional space.

How PCA Works

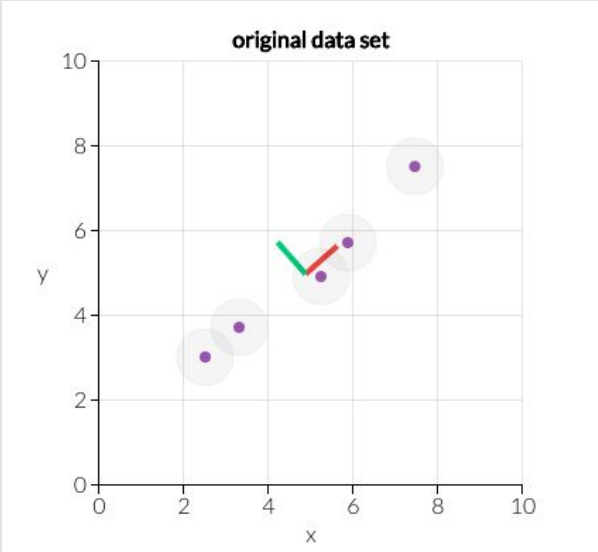
It finds a low-dimensional representation of a data set that contains as much as possible of the variation.

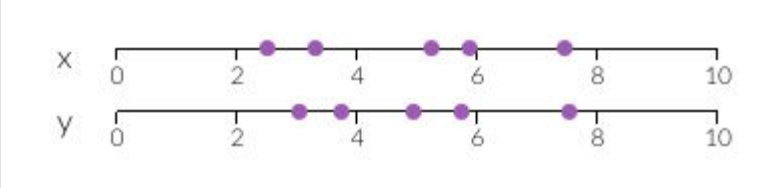
The idea is that each of the n observations lives in p-dimensional space, but not all of these dimensions are equally interesting.

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.

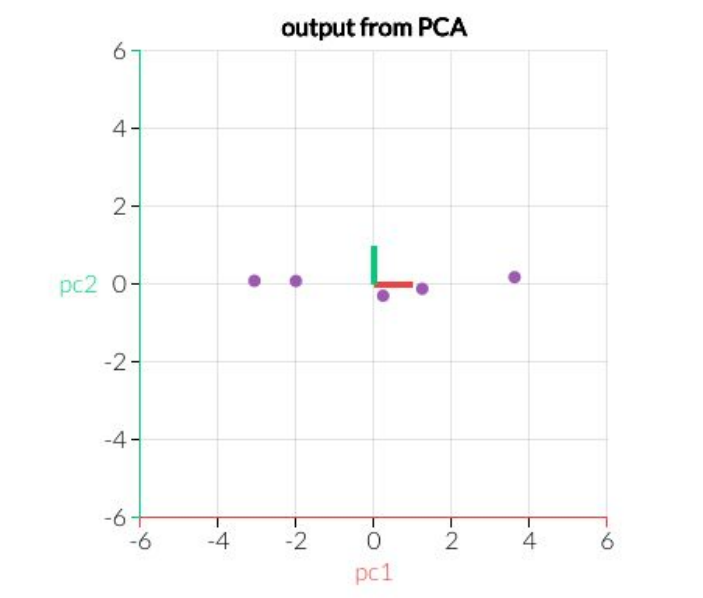
Each of the dimensions found by PCA is a linear combination of the p features.

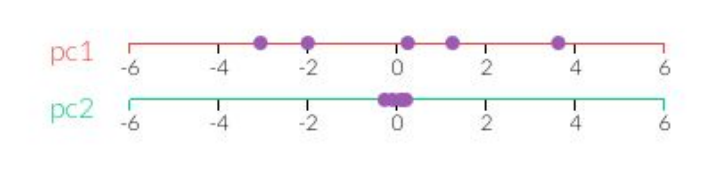
PCA Example





PCA Example - Plotting Principal Components





Principal Component Formula

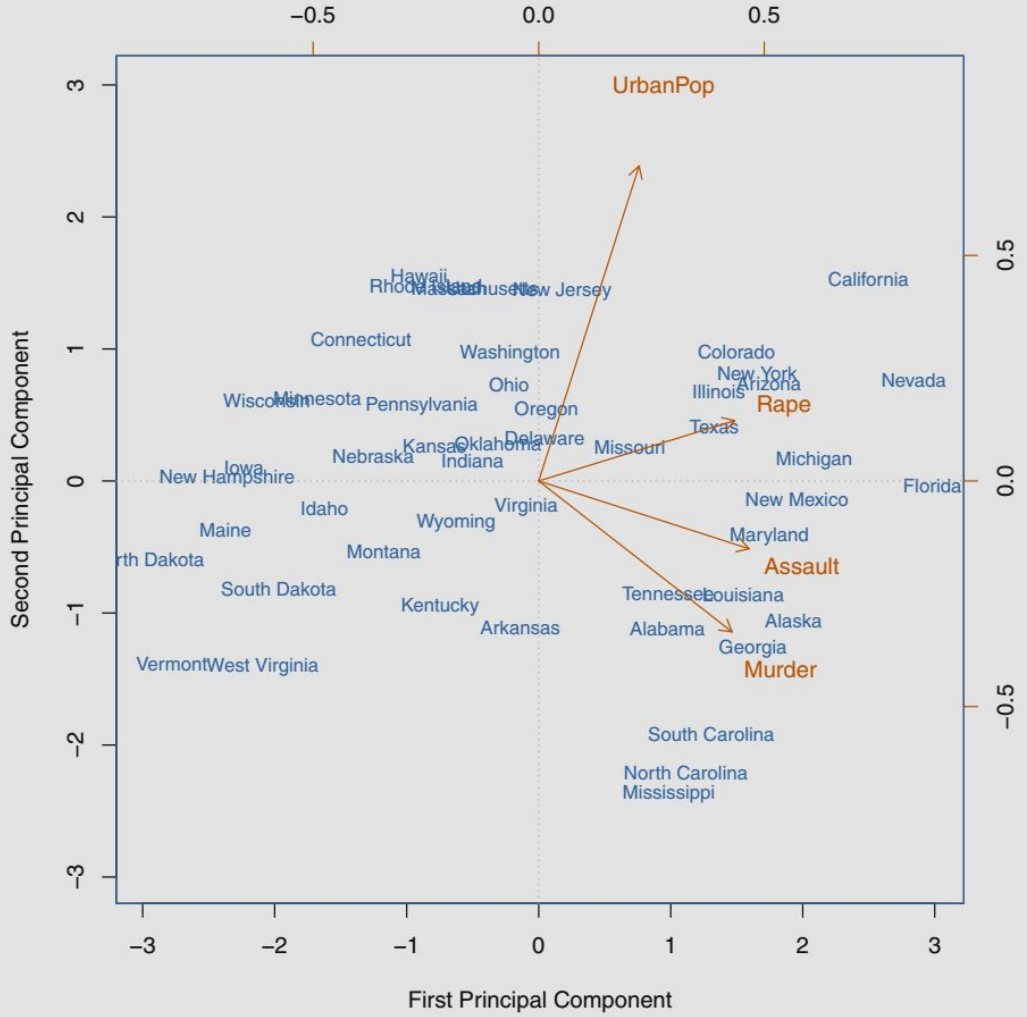
The first principal component of a set of features *X1, X2,…,Xp* is the normalized linear combination of the features

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

that has the largest variance. By normalized, we mean that $\sum_{j=1}^{p} \phi_{j1}^2 = 1$.

We refer to the elements $\varphi_{11},\ldots,\varphi_{p1}$ as the loadings of the first principal component; together, the loadings make up the principal component loading vector, $\varphi_1 = (\varphi_{11} \ \varphi_{21} \ldots \ \varphi_{p1})^T$.

PCA Example


UrbanPop
Rape
Assault
Murder
California
Nevada
Florida
New Jersey
Connecticut
Washington
Colorado
New York
Arizona
Illinois
Ohio
Oregon
Pennsylvania
Wisconsin
Minnesota
Texas
Delaware
Missouri
Kansas
Oklahoma
Indiana
Nebraska
Michigan
Iowa
New Hampshire
New Mexico
Virginia
Idaho
Wyoming
Maine
Maryland
Montana
South Dakota
Tennessee
Louisiana
Kentucky
Arkansas
Alabama
Alaska
Georgia
Vermont
West Virginia
South Carolina
North Carolina
Mississippi
Second Principal Component
First Principal Component
−0.5
0.0
0.5
−3
−2
−1
0
1
2
3

Unsupervised Learning

Clustering

Clustering

Clustering refers to techniques for finding subgroups, or clusters, in a data set.

When we cluster the observations of a data set, we seek to partition them into distinct groups so that the observations within each group are quite similar to each other, and observations in different groups are quite different from each other.

We must define what it means for two or more observations to be similar or different (this is often a domain-specific consideration that must be made based on knowledge of the data.)

Example of an Unsupervised Clustering Problem

- n observations correspond to tissue samples for patients with cancer,
- p features correspond to measurements collected for each tissue sample; these could be clinical measurements, such as tumor stage or grade, or they could be gene expression measurements.
- We may have a reason to believe that there is some heterogeneity among the n tissue samples; for instance, perhaps there are a few different unknown subtypes of cancer.
- Clustering could be used to find these subgroups.

This is an unsupervised problem because we are trying to discover structure—in this case, distinct clusters—on the basis of a data set. The goal in supervised problems, is to try to predict some outcome such as survival time or response to drug treatment.

Clustering vs PCA

- 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.

Marketing Example: We may have access to a large number of measurements (e.g. median household income, occupation, distance from nearest urban area, and so forth) for a large number of people. *Our goal is to perform market segmentation by identifying subgroups of people who might be more receptive to a particular form of advertising, or more likely to purchase a particular product*.

The task of performing market segmentation amounts to clustering the people in the data set.

Two Clustering Techniques

K-means clustering: Here we seek to partition the observations into a pre-specified number of clusters.

Hierarchical clustering: Here we do not know in advance how many clusters we want; we end up with a tree-like visual representation of the observations, called a dendrogram, dendrogram that allows us to view at once the clusterings choices for each possible number of clusters, from 1 to n.

Unsupervised Learning

K-Means Clustering

K-means Clustering

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

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

How K-means Works

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.

Hence we want to solve the problem

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

Within-Cluster Variation

The *within-cluster variation* for the kth cluster is the sum of all of the pairwise squared Euclidean distances between the observations in the kth cluster, divided by the total number of observations in the kth cluster.

K-means Algorithm

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:
    a. For each of the K clusters, compute the cluster centroid. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster.
    b. Assign each observation to the cluster whose centroid is closest (where closest is defined using Euclidean distance).

Issue with Local Optimum

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

It is important to run the algorithm multiple times from different random initial configurations. Then one selects the best solution, i.e. that for which the objective is smallest.

Unsupervised Learning

Hierarchical Clustering

Hierarchical Clustering

K-means Clustering requires us to pre-specify the number of clusters K.

Hierarchical clustering 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 useful tree-based representation of all observations, called a dendrogram.

Dendrogram

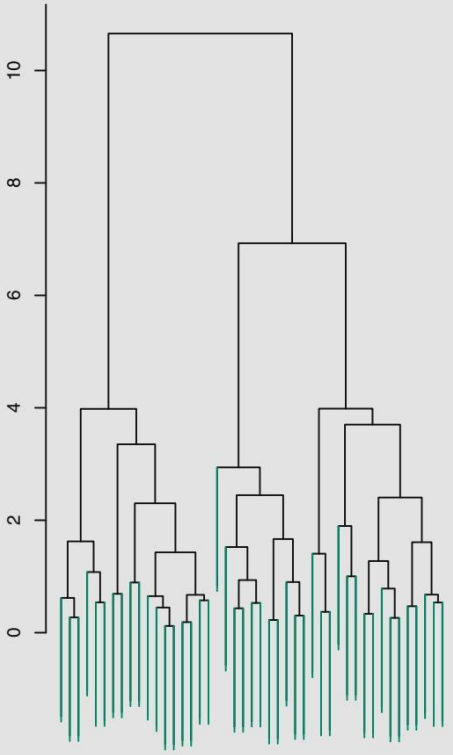
Each leaf of the dendrogram represents one of the 45 observations

As we move up the tree, some leaves begin to fuse into branches. These correspond to observations that are similar to each other.

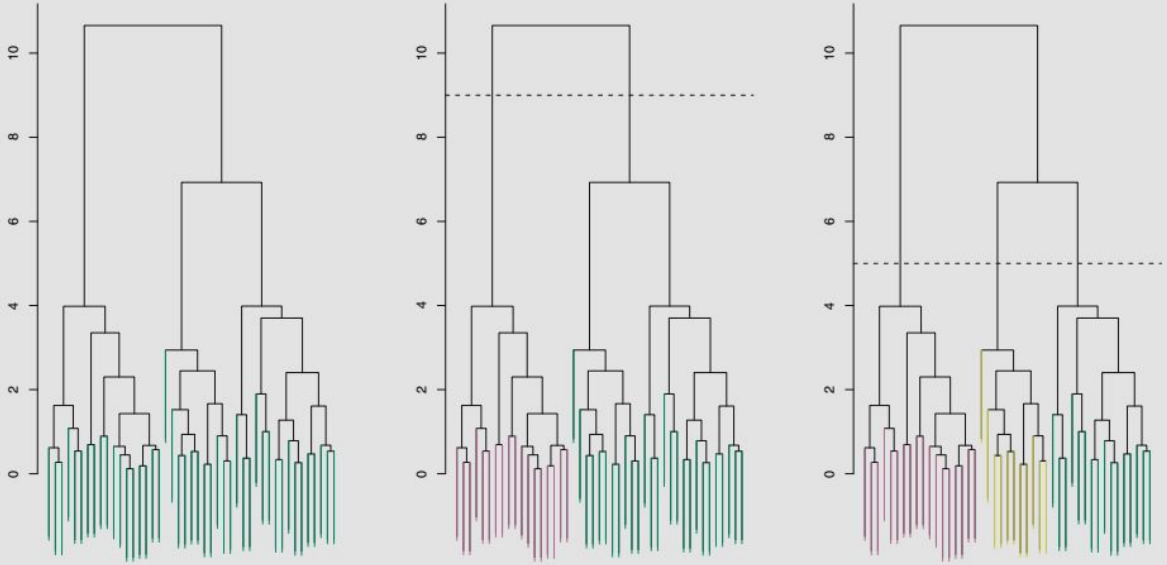
As we move higher up the tree, branches themselves fuse, either with leaves or other branches.

The earlier (lower in the tree) fusions occur, the more similar the groups of observations are to each other.

Observations that fuse later (near the top of the tree) can be quite different.

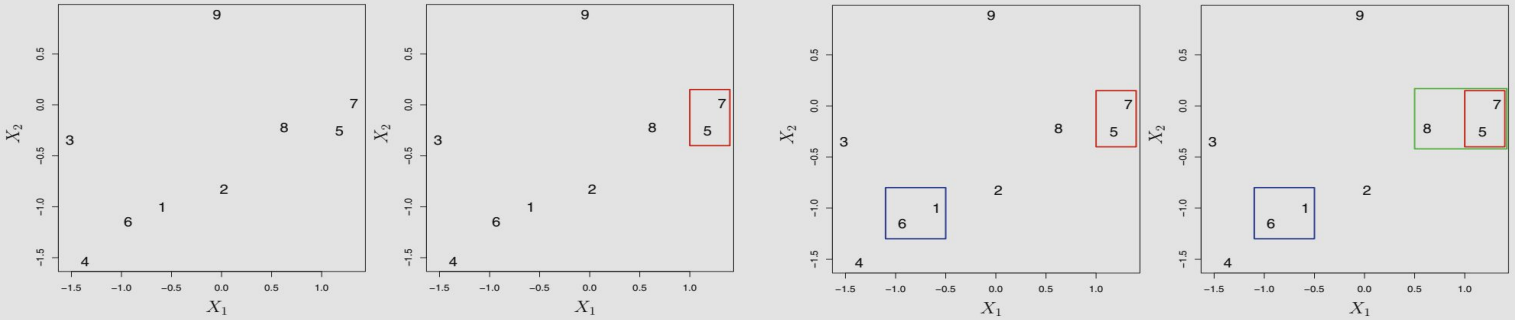


Deciding on Clusters



Algorithm for Hierarchical Clustering

1. Begin with n observations and a measure (such as Euclidean distance) of all the n 2 = n(n – 1)/2 pairwise dissimilarities. Treat each observation as its own cluster.
2. For i = n, n – 1,..., 2:
   a. 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.
   b. Compute the new pairwise inter-cluster dissimilarities among the i – 1 remaining clusters.



Measuring Inter-cluster Similarities

Linkage	*Description*
Complete	Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities.
Single	Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time.
Average	Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable *inversions*.

Impact of Linkage Choice