IEOR E4650 Business Analytics

Session 16: Clustering and Feature Reduction

Spring 2018

Copyright © 2018

Prof. Adam Elmachtoub

Unsupervised Learning

- Unsupervised learning: There are no dependent variables.

  Data: n observations only including p features $X_1, X_2, \ldots, X_p$.

  Goal: not interested in prediction; want to discover interesting patterns and subgroups; can be used to pre-process and reduce dimension of data.

  - Clustering : Used to aggregate observations into meaningful subgroups (aggregate rows)
  - Principle Component Analysis (next time): Used to reduce dimensionality of feature space and for making visualizations, (aggregate columns)

Clustering Methods

- Clustering seeks to find homogeneous subgroups among the observations, so members in each subgroup are close to each other.
- Find homogeneity and heterogeneity among the data.
- K-mean clustering: partition observations into a pre-specified number of clusters
- Hierarchical clustering: no pre-specified number of clusters; end up with a tree-like visual representation

Applications of Clustering

- Market segmentation via clustering to identify similar consumers/users
  - Can use information about a consumer's cluster to try to understand their preferences, and what type of products or advertisements to display
- Product segmentation clusters store/site items into categories
  - Allows us to assign new products to categories, and manage each category separately (assign a manager, give coupon, improve quality)
- Cancer researchers cluster genes into subgroups to obtain better understanding of diseases
  - The similarities within clusters might lead to new hypothesis or discoveries

Overview of K-Means Clustering

- K-means clustering is a simple and elegant approach for partitioning a data set into K distinct, non-overlapping clustering
- To perform K-means clustering, we must first specify the desired number of clusters K
- Then, the K-mean algorithm assigns each observation to exactly one of the K clusters, relying on a distance metric between two observations (e.g., Euclidean distance or correlation distance)

Best K Clusters

Let $C_1, \ldots, C_K$ be the K clusters of observations we wish to create. The sets satisfy two properties:

- $C_1 \bigcup C_2 \bigcup \ldots \bigcup C_K = \{1, 2, \ldots, n\}$. In other words, each observation belongs to at least one of the K clusters.
- $C_k \bigcap C_{k'} = \emptyset$ for all $k \neq k'$. In other words,each observation belongs to at most one of the K clusters.

The idea behind K-means clustering is that a good clustering is one for which the within-cluster variation is as **small** as possible.

Within-Cluster Variation

- The within-cluster variation for cluster C_k, denotedy by $W(C_k)$, is the amount by which the observations within a cluster differ from each other.
- Using squared Euclidean distance, we define

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2,$$

  where $|C_k|$ denotes the number of observations in the k-th cluster.
- The best K-means clustering is the solution to

$$\min_{C_1, C_2, \ldots, C_K} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}$$

- This optimization problem is highly non-linear and too difficult to solve.

Some Intuition

- Let $\bar{x}_{kj}$ be the mean for feature j in cluster C_k. Then

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = 2 \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2$$

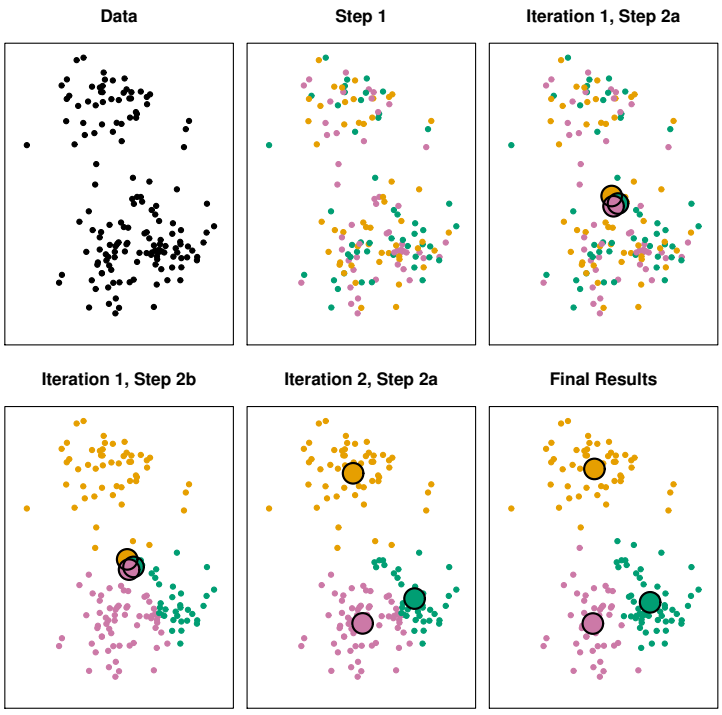
- Thus the within-cluster variation is equivalent to the total squared distance from the centroid of the cluster!

K-means Algorithm

- Instead, we use the following method which obtains a local optimal solution
- The K-means method is as follows:
  (1) Randomly assign a number, from 1 to K, to each of the observations. These serve as initial cluster assignments for the observations.
  (2) Repeat until the cluster assignments stop changing:
    (a) For each of the K clusters, compute the cluster centroid (mean). The kth cluster centroid is the vector of the p feature means for the observations in the k^{th} cluster.
    (b) Assign each observation to the cluster whose centroid is closest (where closest is defined using Euclidean distance).
- You can run the algorithm multiple times from different random initial configurations to obtain multiple local optima, then pick the best clustering.

Illustration of K-means Algorithm

Successive iterations of the K-means clustering algorithm.



Crime Data for 2014

- Arrests data on fifty states, per 100,000 people

```
> USAdata_0 =read.csv("CrimeOneYearofData2014.csv")
> USAdata=data.frame(USAdata_0[,-1],row.names=USAdata_0[,1])
> head(USAdata)
Population  Murder  Assault  Burglary
Alabama       4849377    5.7   283.4    819.0
Alaska         736732    5.6   440.2    427.6
Arizona       6731484    4.7   252.1    647.1
Arkansas      2966369    5.6   346.0    835.7
California   38802500    4.4   236.6    522.3
Colorado      5355866    2.8   192.8    438.2
```

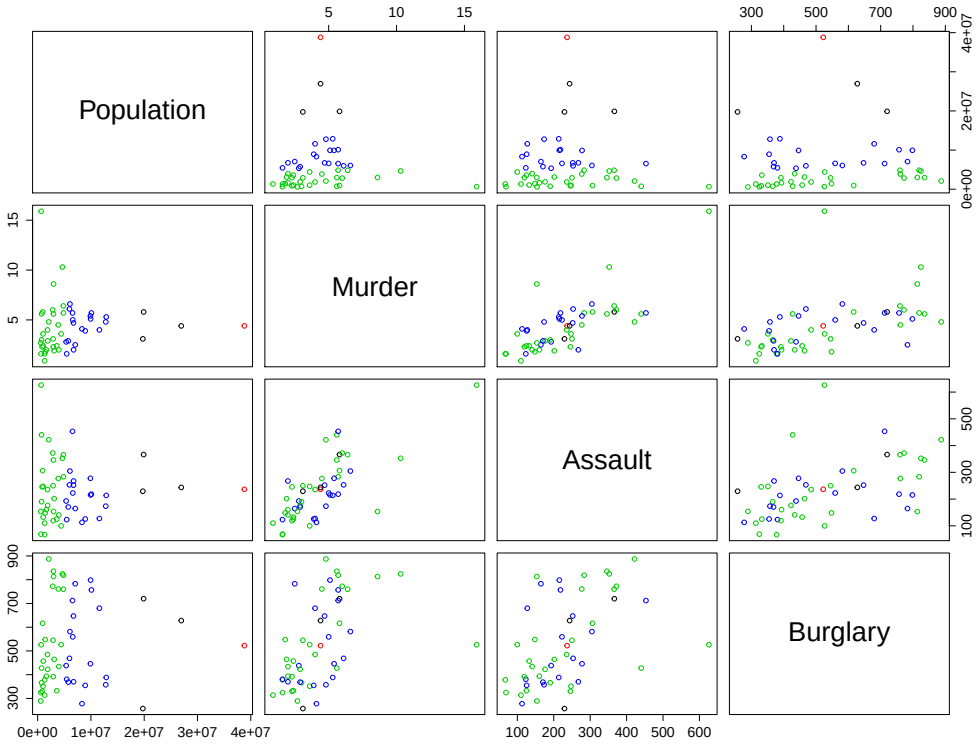
- Could rescale data using my.dat = scale(my.dat).
- scale() will make columns have mean 0 and standard deviation of 1

Data: USArrests

- Output of kmeans in R.

```
> km.fit=kmeans(USAdata, centers=4,nstart=20)
> names(km.fit)
[1] "cluster"      "centers"      "totss"
[4] "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
> head(km.fit$cluster)
Alabama     Alaska    Arizona   Arkansas California
2           3         2          3          4
Colorado
2
> km.fit$withinss
[1] 6.673602e+13 6.165444e+13 2.993154e+13 7.015843e+13
> km.fit$tot.withinss
[1] 2.284804e+14
> plot(USAdata,col=km.fit2$cluster)
```

Visualizing the Clusters

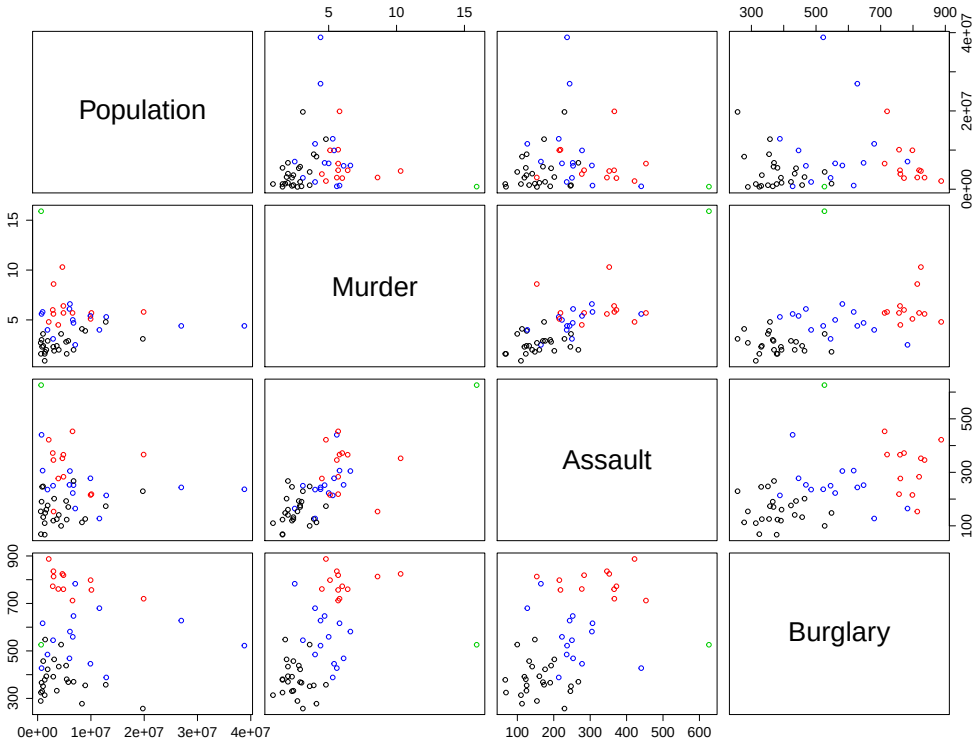


Better idea: Scale Data, Remove population

- Output of kmeans in R.

```
> USAdata2=data.frame(scale(USAdata[,-1]))
> head(USAdata2)
Murder     Assault    Burglary
Alabama     0.55667580  0.47832031  1.60552062
Alaska      0.51696885  1.89973681 -0.54863427
Arizona     0.15960635  0.19458091  0.65943165
Arkansas    0.51696885  1.04579909  1.69743270
California  0.04048551  0.05407099 -0.02743225
Colorado   -0.59482561 -0.34298285 -0.49029486
> km.fit2=kmeans(USAdata2, centers=4,nstart=20)
> plot(USAdata,col=km.fit2$cluster)
```

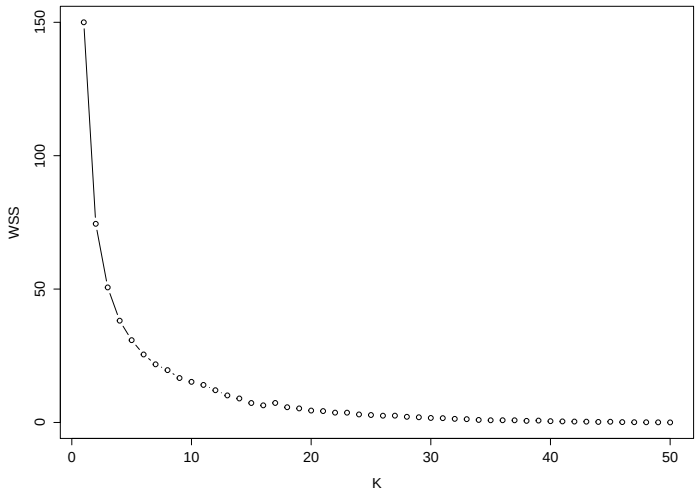
Visualizing the Clusters



How Many Clusters?

- Idea: Get small tot.withinss without using too many clusters

```
> wss=rep(0,nrow(USAdata2)-1)
> for (i in 1:length(wss)) wss[i]<-kmeans(USAdata2,centers=i,nstart=10)$tot.withinss
> plot(1:length(wss),wss,type="b",xlab="K",ylab= "WSS")
```



- It seems that $4 \leq K \leq 7$ should work well.

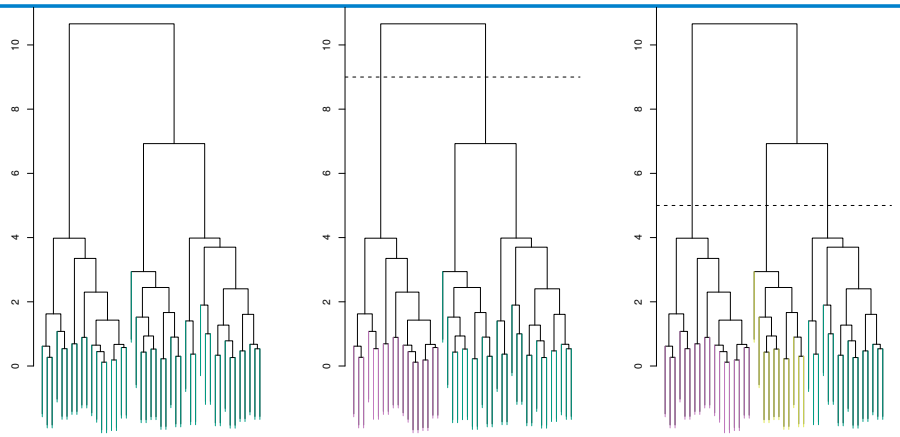
Hierarchical Clustering

- Disadvantage of K-mean clustering: it requires to pre-specify the number of clusters K
- Hierarchical Clustering is an alternative: Does not require K
- Advantage of Hierarchical Clustering: it results in an attractive tree-based representation of the observations, called a dendrogram

Hierarchical Clustering

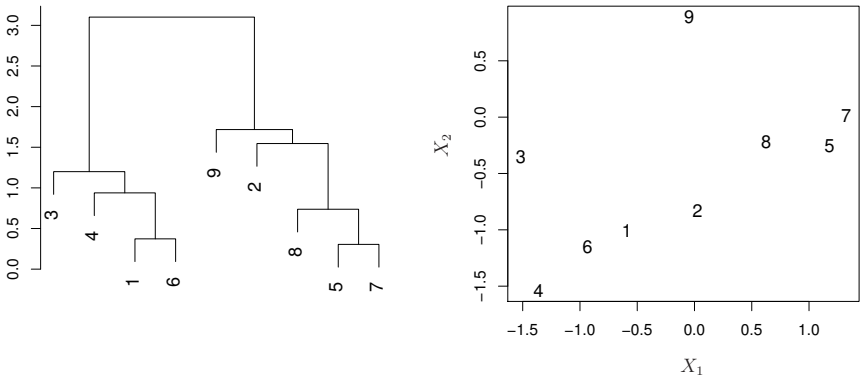
- An agglomerative approach
  - Find closest two things
  - Put them together
  - Find next closest
- Requires
  - A defined distance
  - A merging approach
- Produces
  - A tree showing how close things are to each other

Interpreting Dendrograms



- The leaves at the bottom of the dendrogram represent the individual observations
- Leaves are combined to form branches
- Small branches are combined into larger branches, until one reaches the trunk or the root
- Draw any horizontal (dashed) line - this corresponds to a clustering
- Thus we can visualize all possible clusterings in one dendrogram

Interpreting Dendrograms



- Distance between two observations corresponds to where their branches are fused
- Observations 1 and 6 are .4 apart, 5 and 7 are also about .4 apart
- Observation 3 and 4 are 1.2 apart, NOT .5

Hierarchical Clustering Algorithm

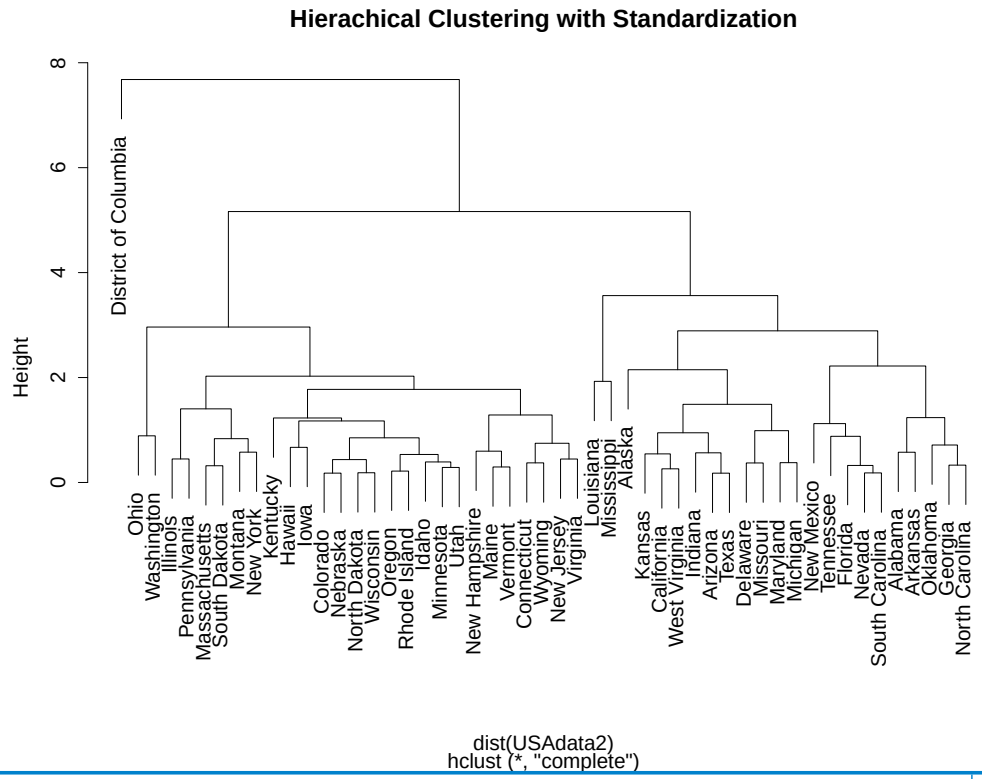
- Algorithm
  (1) Begin with n observations and a measure (such as Euclidean distance) of all the pairwise dissimilarities. Treat each observation as its own cluster.
  (2) For $i = n, n-1, \ldots, 2$
    (a) Examine all pairwise inter-cluster dissimilarities among the i clusters and identify the pair of clusters that are least dissimilar (that is, most similar). Fuse these two clusters.
    (b) Compute the new pairwise inter-cluster dissimilarities among the $i-1$ remaining clusters.
- Four common types of dissimilarity (linkages): complete, average, centroid, and single.

Linkages

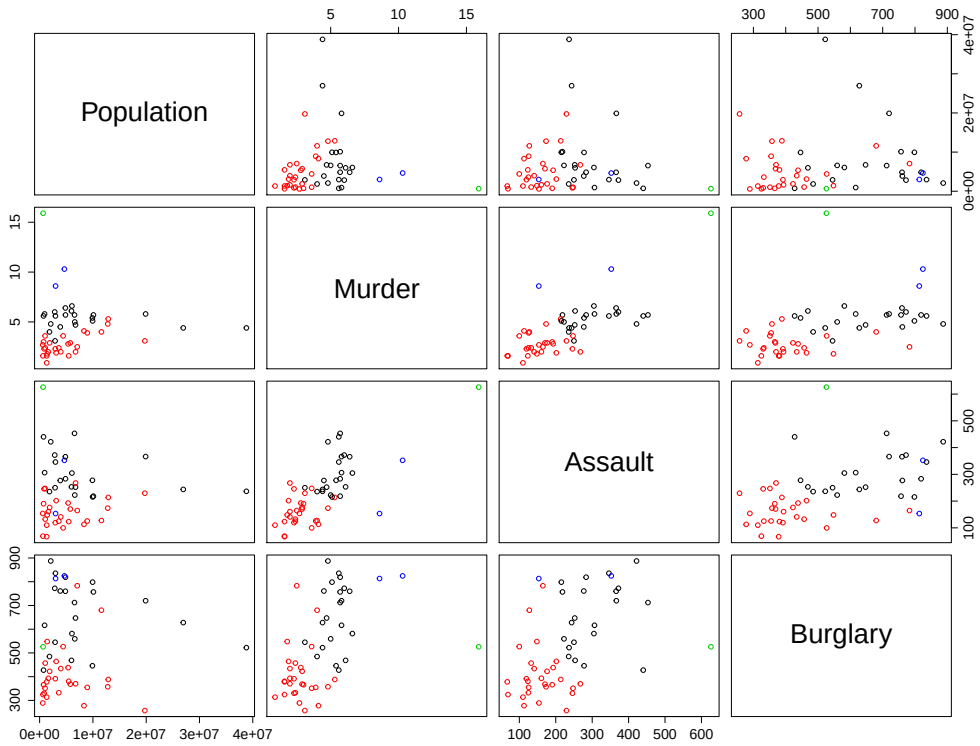
Linkage is known as the dissimilarity between two clusters. There are three types of linkages. For each, begin by computing Euclidean distance between all pairs of points in both clusters.

- Complete: Maximum distance is dissimilarity between the two clusters.
- Single: Smallest distance is dissimilarity between the two clusters.
- Average: Average of distances is dissimilarity between the two clusters.
- Centroid: Distance between centroids is dissimilarity between the two clusters.

Hierarchical Clustering: Dendrogram



Visualizing Clusters



Dimension Reduction

- Suppose we have a large number of features/covariates p, with possibly many correlated
- Let $\mathbf{X}$ be the n by p matrix representing the data
- We would like to reduce the number of features from p to M, where M is much smaller than p
- In other words, we build a smaller dataset Z in a lower dimensional space, represented by an n by M matrix
- Using these $M < p$ new features allows us to keep track of less data, build simpler predictive models, and visualize the data

Principal Components Analysis (PCA)

- PCA seeks a M features that are as informative as possible, capturing as much of the variability in $\mathbf{X}$ as possible
- Each new component/feature Z_j found by PCA is a linear combination of the p original features
- The first principal component is the 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 subject to $\sum_{j=1}^{p} \phi_{j1}^2 = 1$.

- In general, we have

$$Z_j = \phi_{1j}X_1 + \phi_{2j}X_2 + \ldots + \phi_{pj}X_p$$

and

$$Z = X\Phi = X[\ \phi_1 \ldots \phi_M]$$

- The vectors ϕ_j are called loading vectors

Principal Components Analysis (PCA), Cont.

- We look for linear combination of the sample feature values of the form

$$z_{i1} = \sum_{k=1}^{p} x_{ik}\phi_{k1}$$

that has largest sample variance.
  - We refer $z_{11}, z_{21}, \ldots, z_{n1}$ as the scores of the first principal component.
  - **Assume the data is normalized so each feature has mean 0**

- The PCA loading vector solves the optimization problem

$$\max_{\phi_{11},\ldots,\phi_{p1}} \quad \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{k=1}^{p} x_{ik}\phi_{k1} \right)^2 \right\}$$

$$s.t., \quad \sum_{k=1}^{p} \phi_{k1}^2 = 1.$$

Principal Components Analysis (PCA), Cont.

- We look for the second principal component Z_2,

$$z_{i2} = \sum_{k=1}^{p} x_{ik}\phi_{k2} \quad i = 1, \ldots, n$$

that has largest sample variance under the constraint that Z_2 is uncorrelated with Z_1.

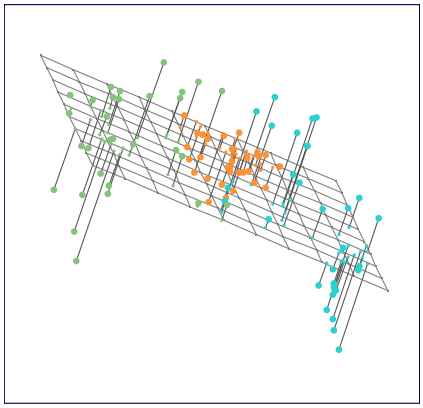
- The uncorrelated constraint is equivalent to constraining the direction ϕ_2 to the orthogonal to the direction ϕ_1.
- We look for the third principal component Z_3,

$$z_{i3} = \sum_{k=1}^{p} x_{ik}\phi_{k3} \quad i = 1, \ldots, n$$

that has largest sample variance under the constraint that Z_3 (ϕ_3) is uncorrelated with Z_2 (ϕ_2) and Z_1 (ϕ_1).....

Principal Components

- We can interpret the first M principal components as the best M-dimensional approximation to the data X
  - For $M = 1$ we get a line,
  - For $M = 2$ we get a plane, etc.



- Here we project 3-dimensional points on to the best 2 dimensional plane

Proportion of Variance Explained

- Question: how much of the information is explained by the principal components?
- We consider the proportion of variance explained (PVE) by each principal component
- The total variance present in a data set (assuming variables have been centered to have mean zero)

$$\sum_{j=1}^{p} Var(X_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2$$

Proportion of Variance Explained, Cont.

- The variance explained by the m-th principal component is

$$\frac{1}{n}\sum_{i=1}^{n} z_{im}^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{k=1}^{p} x_{ik}\phi_{km}\right)^2$$

- Therefore, the PVE of the m-th principal component is given by

$$\frac{\sum_{i=1}^{n}\left(\sum_{k=1}^{p} x_{ik}\phi_{km}\right)^2}{\sum_{j=1}^{p}\sum_{i=1}^{n} x_{ij}^2}$$

Proportion of Variance Explained, Cont.

- We want to use the smallest number of principal components to get a good understanding of the data.
- How many principal component are needed?
- No single answer when used in unsupervised learning.
- Produce a plot, choose the number of principal components in order to explain a sizeable amount of variation in the data.

Examples in R

- Data set `USArrests`

```
USAdata_0 =read.csv("CrimeOneYearofData2014.csv")
USAdata=data.frame(USAdata_0[,-1],row.names=USAdata_0[,1])
head(USAdata)

states=row.names(USAdata)
tail(USAdata)
apply(USAdata,2,mean)
apply(USAdata,2,var)
```

- Function `apply()`: arguments (dataset, row or column, function)
- Finding: variables are in different scales.

Principal Component Analysis in R

- Function `prcomp()`

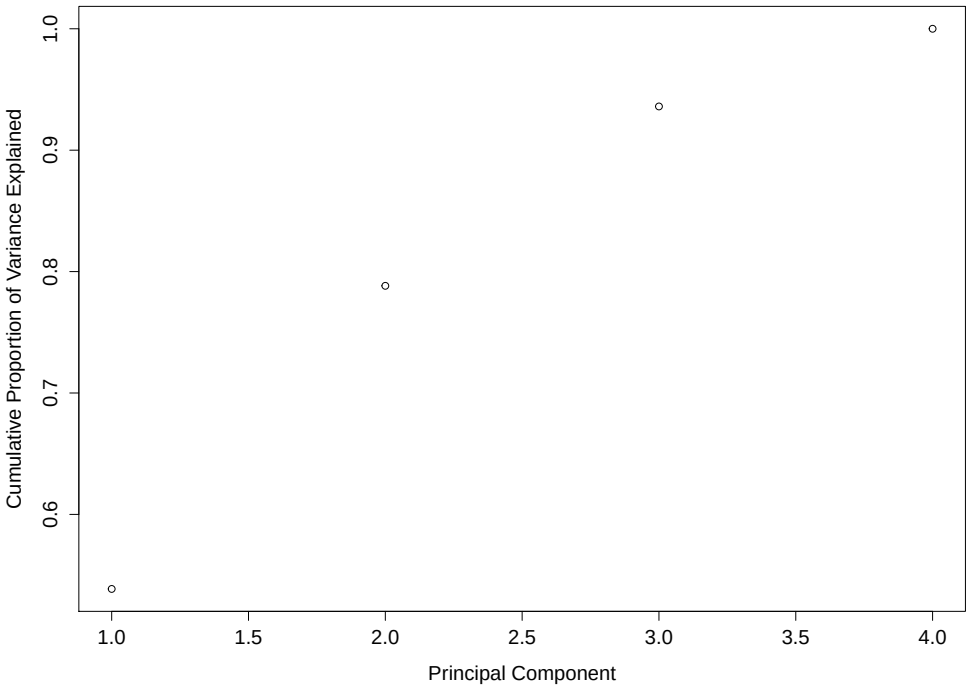
```
pr.out=prcomp(USAdata,scale=T)
names(pr.out)
pr.out$center
pr.out$scale
pr.out$rotation
summary(pr.out)
names(summary(pr.out))
plot(summary(pr.out)$importance[3,],ylab="Cumulative Proportion o
biplot(pr.out, scale=0)
```

- First two principal variables explain 79% variability. We can write

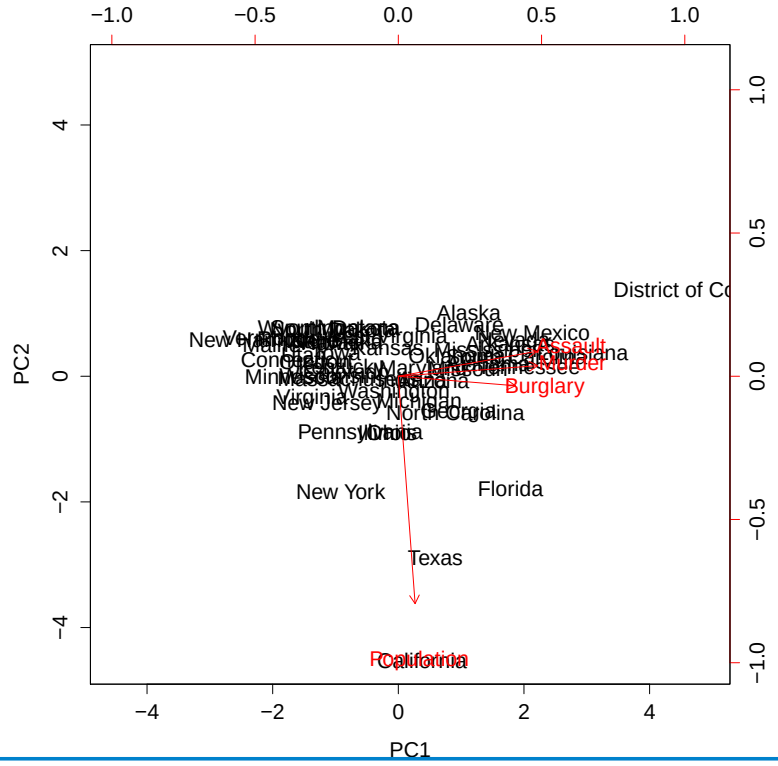
$$x_{ij} = \sum_{m=1}^{p} z_{im}\phi_{mj} \simeq \sum_{m=1}^{M} z_{im}\phi_{mj}$$

by using the first $M < p$ principal components.

Principal Components



Principal Components biplot



Principal Components biplot

- In the previous figure we are looking at a representation of the data using 2 components (originally 4 features)
- The black names correspond to where each observation is in regards to the two components... use left and bottom axes
- The red lines describe ϕ_{j1} and ϕ_{j2} for each feature j. In other words, how much feature contributed to each of the two components

Principal Components Regression

- A dimension reduction technique for regression
- The Principal Components Regression (PCR) approach involves constructing the first M principal components $Z_1, \ldots, Z_M$, and then use these components as the predictors in a linear regression
- The key idea: a small number of principal components suffice to explain most of the variability in the data, as well as the relationship with the response.
    - PCA reduces dimension, which is always good.
    - Higher variance covariates are good in regression, and we choose the top PCs to have highest variance.
    - The PCs are independent: no multicollinearity
- We can determine the number of components to use by k-fold cross-validation

Principal Components Regression in R

- Principal Components Regression can be performed using function `pcr()` in library `pls`
- For the hitters data, we will use 7 principal components.
- In function `pcr()`, specify number of principal components by using `ncomp=`

```
> library(pls)
> library(ISLR)
> pcr.fit=pcr(Salary~.,data=Hitters,scale=T,validation="CV")
> summary(pcr.fit)
```

- Argument `validation="CV"` causes `pcr()` to compute the ten-fold cross-validation error for each possible M, the number of principal components used.

Principal Components Regression

Note that the smallest error occurs when $M = 16$, but it is roughly the same when only one component is included.

Principal Components Regression in R, cont

```
> pcr.fit2=pcr(Salary~.,data=Hitters,scale=T,validation="CV",ncomp=7)
> summary(pcr.fit2)
Data:  X dimension: 263 19
Y dimension: 263 1
Fit method: svdpc
Number of components considered: 7

VALIDATION: RMSEP
Cross-validated using 10 random segments.
         (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
CV             452    352.0    351.4    351.3    348.6    343.4    340.3    341.6
adjCV          452    351.6    351.1    350.9    348.1    342.9    339.6    340.9

TRAINING: % variance explained
           1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
X         38.31    60.16    70.84    79.03    84.29    88.63    92.26
Salary    40.63    41.58    42.17    43.22    44.90    46.48    46.69
```


Principal Components Regression in R, cont.

- Perform `pcr()` on a training subset
- Make predictions on test subset
- Calculate prediction errors

```
set.seed(1)
Hitters=na.omit(Hitters)
x = model.matrix(Salary~., Hitters)[,-1]
y=Hitters$Salary
train = sample(1:nrow(x), nrow(x)/2)
test=(-train)
y.test=y[test]

pcr.fit = pcr(Salary~., data= Hitters, subset=train, scale=T,
validation = "CV")
pcr.pred = predict(pcr.fit, x[test,], ncomp=7)

mean((pcr.pred - y.test)^2)
[1] 96556.22
```

Partial Least Squares

- The PCR approach that we just described involves identifying linear combinations that best represent the predictors
- The response does not supervise the identification of the principal components.
- PCR suffers from a drawback: there is no guarantee that the directions that best explain the predictors will also be the best directions to use for predicting the response.
- Partial Least Squares (PLS), a supervised alternative to PCR.
  - Unlike PCR, PLS identifies new directions in a supervised way – that is, it makes use of the response Y in order to identify new features.
  - Roughly speaking, the PLS approach attempts to find directions that help explain both the response and the predictors.

Partial Least Squares

- Partial Least Squares (PLS)
  - PLS computes the first direction Z_1 by setting each ϕ_{j1} equal to the coefficient from the simple linear regression of Y onto X_j.
  - To identify the second PLS direction we first adjust each of the variables for Z_1, by regressing each variable on Z_1 and taking residuals. These residuals can be interpreted as the remaining information
  - We then compute Z_2 using this orthogonalized data in exactly the same fashion as Z_1
  - Finally, at the end of this procedure, we use least squares to fit a linear model to predict Y using $Z_1, \ldots, Z_M$ in exactly the same fashion as for PCR.

Partial Least Squares in R

- Implement partial least squares using `plsr()` in library `pls`
- Determine number of principal components using eyes
- In this data set, 2 may be a good size for principal components.
- In function `plsr()`, specify number of principal components by using `ncomp=`

```
set.seed(1)
Hitters=na.omit(Hitters)
x = model.matrix(Salary~., Hitters)[,-1]
y=Hitters$Salary
train = sample(1:nrow(x), nrow(x)/2)
test=(-train)
y.test=y[test]
pls.fit = plsr(Salary ~., data=Hitters, subset=train,
scale=TRUE, validation ="CV")
summary(pls.fit)

validationplot(pls.fit,val.type="MSEP")
```


Partial Least Squares

Partial Least Squares in R

- Notice that the percentage of variance in Salary that the two-component PLS fit explains, 46.40%, is almost as much as that explained using the final seven-component model PCR fit, 46.69%.
- This is because PCR only attempts to maximize the amount of variance explained in the predictors, while PLS searches for directions that explain variance in both the predictors and the response.

```
> pls.fit = plsr(Salary ~., data=Hitters, scale=TRUE, ncomp=2)
> summary(pls.fit)
Data:  X dimension: 263 19
Y dimension: 263 1
Fit method: kernelpls
Number of components considered: 2
TRAINING: % variance explained
          1 comps  2 comps
X         38.08    51.03
Salary    43.05    46.40
```