Chapter 6
Unsupervised learning methods

Simon Fraser University
ECON 483
Summer 2023

SFU DEPARTMENT OF ECONOMICS

Disclaimer

These notes are based on the Book **Introduction to Statistical Learning with R**, by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani. Any error is my sole responsibility.

Unsupervised learning

- **Supervised learning** is well understood because we have a way to check the validity of our methods, thanks to the dependent variable: We are **supervised**
- **Unsupervised learning** is more tricky. How can we check if we are on the right path? We are **not supervised**
- Unsupervised learning is often used as an **observatory data analysis** tool, i.e. as a way to visualize the patterns in the data (correlations between variables, similarities among observations across different features)
- We are going over two different methods corresponding to two different objectives:
    - **Principal components analysis (PCA)**: Visualize high dimensional data using a low-dimensional representation containing as much of the variation of the initial data as possible
    - **Clustering**: Create groups of observations based on some similarity measure

Outline

- **Principal components**
- **Clustering**
    - **K-means clustering**
    - **Hierarchical clustering**
- Suggested reading: Chapter 10 in **ISLR**

Principal component analysis (PCA)

Principal component analysis

- Generally used to observe high-dimensional data before getting into further analysis
- It summarizes variance and correlation patterns of covariates in a more compact way, by reducing the dimension while keeping the important features
- With $\boldsymbol{p}$ variables at hand, **PCA** looks for $\boldsymbol{M} \ll \boldsymbol{p}$ variables $\boldsymbol{Z_m}$, the **principal components**, that capture the main features (in particular, variance and correlation) of the original variables:

$$\boldsymbol{Z_m} = \sum_{\boldsymbol{j=1}}^{\boldsymbol{p}} \boldsymbol{\phi_{j,m} X_j}$$

Principal component analysis

$$Z_m = \sum_{j=1}^{p} \phi_{j,m} X_j$$

- First, standardize each variable so they all have mean 0 and standard deviation 1 (as PCA is sensitive to scaling)
- This way, Z_m has a mean of 0 and a variance/standard deviation of 1
- PCA finds the coefficients $\phi_{j,m},\ j = 1, \ldots, p;\ \ m = 1, \ldots, M$ such that the **variance** of the new variable z_m is maximized:

$$\max_{\{\phi_{1,1},\ldots,\phi_{p,1}\}} \frac{1}{n} \sum_{i=1}^{n} z_{i,1}^2 \quad \text{subject to} \quad \sum_{j=1}^{p} \phi_{j,1}^2 = 1$$

- Result: $z_{1,1}, \ldots, z_{n,1}$ are the **scores** of the first principal component, and $(\phi_{1,1}, \ldots, \phi_{p,1})$ is the **loading vector** associated to the first principal component

Principal component analysis

- PCA finds the coefficients $\phi_{j,m},\ j = 1, \ldots, p,\ \ m = 1, \ldots, M$ such that the **variance** of the new variable z_m is maximized:

$$\max_{\{\phi_{1,1},\ldots,\phi_{p,1}\}} \frac{1}{n} \sum_{i=1}^{n} z_{i,1}^2 \quad \text{subject to} \quad \sum_{j=1}^{p} \phi_{j,1}^2 = 1$$

- Result: $z_{1,1}, \ldots, z_{n,1}$ are the **scores** of the first principal component, and $(\phi_{1,1}, \ldots, \phi_{p,1})$ is the **loading vector** associated to the first principal component
- Then, repeat the same maximization problem to find z_2. But an extra constraint is added: z_2 has maximal variance **and** is uncorrelated with z_1. Repeat until you get to z_M
- The coefficients ϕ in each loading vector represent the weight of each variable in that component

Principal component analysis: Illustration

```r
# Data on crimes for each state of the US, along with other covariates.
# 4 covariates here: "Rape", "Assault", "Murder", and "UrbanPop"
data(USArrests)
states <- row.names(USArrests)
pr.out <- prcomp(USArrests, center = TRUE, scale = TRUE)
summary(pr.out)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4
## Standard deviation     1.5749 0.9949 0.59713 0.41645
## Proportion of Variance 0.6201 0.2474 0.08914 0.04336
## Cumulative Proportion  0.6201 0.8675 0.95664 1.00000
```

```r
pr.out$rotation
```

```
##                 PC1        PC2        PC3         PC4
## Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
## Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
## UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
## Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

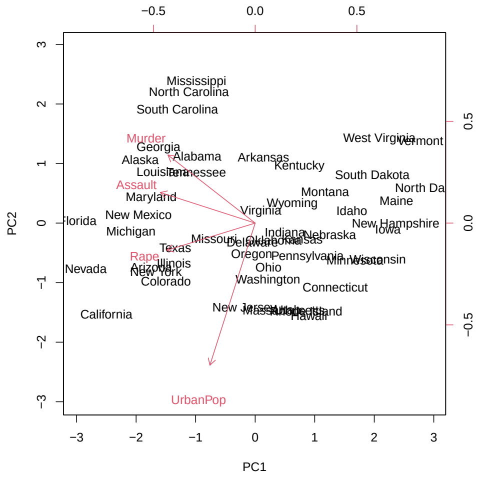
Principal component analysis: Illustration

- In the first principal component, ***Murder***, ***Assault*** and ***Rape*** have a similar magnitude **and** sign: There are positively correlated and contribute relatively equally to the first component: That component “summarizes” serious crime overall
- In the second principal component, ***UrbanPop*** has the highest loading in magnitude: That component “summarizes” the level of urbanization of a state
- Standardization of the data is important. When some features are measured in different units, we might end up in a feature having a very high variance, which will affect the results of the PCA

Principal component analysis: Biplot

- The main results of PCA can be represented by a biplot
- It shows vectors which are the original covariates. Their coordinates correspond to the loading components of the associated principal components on the right and top axes (in the next slide, the $\phi_{j,1}$ and $\phi_{j,2}$ as we look at PC1 and PC2. j corresponds to covariate j). For instance, $\phi_{UrbanPop,1} \simeq -0.28$ and $\phi_{UrbanPop,2} \simeq -0.87$. These vectors can be interpreted in three ways (Rossiter, 2014)
  - **Direction**: The more parallel to a principal component, the more the variable contributes to that PC
  - **Length**: The longer the vector, the more variability of the feature of this variable is represented by the PC. Short vectors are better represented by other PC
  - **Angles between vectors**: Indicates the correlation between the features. The closer they are (the smaller the angle), the more positively correlated they are. A right angle shows no correlation, and opposite angles show negative correlations
- The points are the observations plotted according to the principal components on the graph

Principal component analysis: Biplot



Principal component analysis: Illustration

- The closer the vectors, the more correlated the variables (here, ***murder***, ***Assault*** and ***Rape***). But ***Murder*** and ***UrbanPop*** are weakly negatively correlated (angle bigger than 90 degrees)
- Observations located far along a component will feature high values (in magnitude) in the variables most used for that component that go
- Ex: California, Nevada, Florida are low in terms of the first component, so they have high crime values as the first component puts more weight on the crime variables and the states follow the direction of the vectors. For North Dakota, these crime numbers are low as it is far along the first component, but in the opposite direction
- Ex: Mississippi features a low value of urban population (as it is far away from the ***UrbanPop*** vector)
- Ex: Vermont features a low urban population **and** low crime rates as it is located at the opposite of all the vectors

Principal component analysis: Takeaways

- PCA is a convenient tool to see what variables are correlated **and** feature the most variance when many covariates are available (particularly convenient for Big Data)
- How to choose the number of components?
- No systematic way (if only we had cross validation... sigh), but one can use the percentage of variance explained by each component. If the first 3 components add up to, say, 90% of the explained variance, whereas adding a 4^{th} component increases that share by 1%, then 3 components summarize the main features pretty well and could be deemed enough
- It is a standard exploratory tool in statistics and Economics
- Can be used for regression if the number of covariates is very large (**Principal Components Regression**)

Clustering

Clustering

- **Clustering** refers to finding subgroups (clusters) in a data set
- We have access to a sample of size $\boldsymbol{n}$ for which we observe $\boldsymbol{p}$ **features** $\boldsymbol{x_j, j = 1, \ldots, p}$ (they are not called "covariates" anymore as there is no $\boldsymbol{Y}$)
- Idea: Create homogeneous groups among the observations (i.e. groups with **small within-cluster variation**)
- We can
  - Group observations that are similar according to the **features**
  - Group features that are similar according to the **observations**
- Applications: Marketing (market segmentation: find different types of consumers), medicine (identify groups of patients to understand a disease better), urban Economics (identify groups of houses according to type, value, location, etc)

Kmeans clustering

K-means clustering

- Assume we want to group observations that are similar according to the **features**
- Let C_k, $k = 1, \ldots, K$ be cluster k
- Objective: Obtain K clusters that do not overlap, and include all the observations
- Implementation: Minimize the amount of **dissimilarity** between observations of one cluster:
- We will use **Euclidean distance** for dissimilarity. For two observations i and i' belonging to a cluster C_k, the dissimilarity is defined as

$$\sum_{j=1}^{p} \left(x_{i,j} - x_{i',j} \right)^2$$

K-means clustering

$$\sum_{j=1}^{p} \left(x_{i,j} - x_{i',j}\right)^2$$

- It is the distance between observation i and j over all the features $j = 1, \ldots, p$
- Now, take the average of all the pairwise distances in the cluster C_k to define

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

K-means clustering

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

- We want to minimize dissimilarities for each cluster, so the end goal is to solve:

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

K-means clustering: Algorithm

- It is surprisingly easy to implement in a software

Algorithm: K-means clustering

- Randomly assign a number from $\mathbf{1}$ to $\boldsymbol{K}$ for each observation. Hence, each observation is randomly assigned a cluster
- For each cluster:
    - Compute the **centroid**, i.e. the vector of averages of the $\boldsymbol{p}$ features in the cluster
    - Assign observations to the cluster where their distance from the centroid is the smallest
    - Keep going until observations are not moved from clusters anymore

- Pretty straightforward, right?

K-means clustering in **R**

- To illustrate clustering methods, consider the ***Pokemon*** data set
- Each observation is a pokemon, along with their type, and some statistics: Health points, attack, defense, speed, etc. . .
- Clustering operates with numerical variables, so we are going to create groups of pokemon according to their specs
- In **R**, the ***kmeans*** function will achieve that

K-means clustering in R

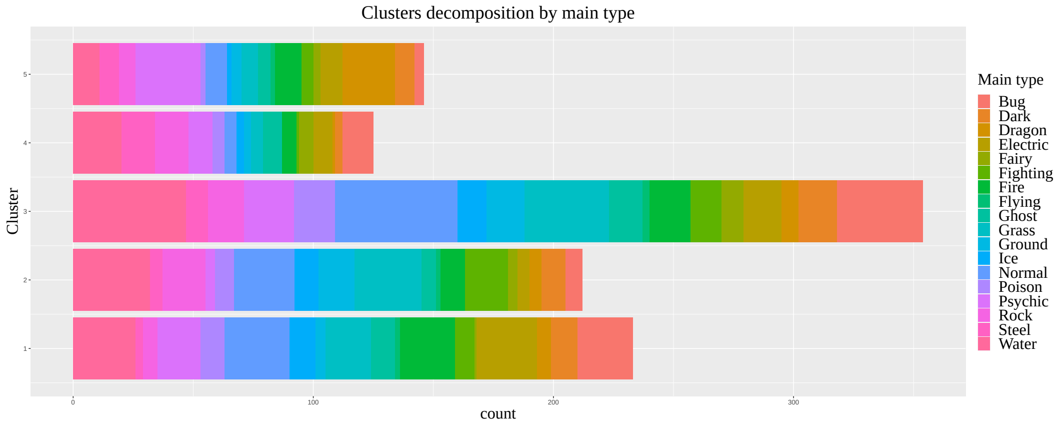
```
##                  name type1  type2 total hp attack defense sp_attack sp_defense
## 1          Bulbasaur Grass Poison   318 45     49      49        65         65
## 2            Ivysaur Grass Poison   405 60     62      63        80         80
## 3           Venusaur Grass Poison   525 80     82      83       100        100
## 4      Mega Venusaur Grass Poison   625 80    100     123       122        120
## 5 Gigantamax Venusaur Grass Poison   525 80     82      83       100        100
## 6          Charmander  Fire          309 39     52      43        60         50
##   speed generation legendary
## 1    45          1     FALSE
## 2    60          1     FALSE
## 3    80          1     FALSE
## 4    80          1     FALSE
## 5    80          1     FALSE
## 6    65          1     FALSE
```

K-means clustering in R

```r
pokemon[, c(4:10)] <- scale(pokemon[, c(4:10)]) # scale the data first
# K-means clustering
pokemon_cluster <- kmeans(x = pokemon[, -c(1, 2, 3, 11, 12)], centers = 5, nstart = 20)
# Code taken from Tyler Harris' post on towardsdatascience.com
kmeans_basic_table <- data.frame(pokemon_cluster$size, pokemon_cluster$centers)
kmeans_basic_df <- data.frame(Cluster = pokemon_cluster$cluster, pokemon)
head(kmeans_basic_df)
```

```
##   Cluster                    name type1  type2      total         hp      attack
## 1       3               Bulbasaur Grass Poison -1.0108971 -0.9478913 -0.98415453
## 2       1                 Ivysaur Grass Poison -0.2946151 -0.3900628 -0.58390924
## 3       1                Venusaur Grass Poison  0.6933601  0.3537084  0.03185273
## 4       5           Mega Venusaur Grass Poison  1.5166727  0.3537084  0.58603852
## 5       1 Gigantamax Venusaur Grass Poison  0.6933601  0.3537084  0.03185273
## 6       3              Charmander  Fire        -1.0849953 -1.1710227 -0.89179023
##      defense  sp_attack sp_defense      speed generation legendary
## 1 -0.8304659 -0.2527033 -0.2652549 -0.7903998          1     FALSE
## 2 -0.3820993  0.2064192  0.2722882 -0.2920921          1     FALSE
## 3  0.2584244  0.8185825  0.9890122  0.3723181          1     FALSE
## 4  1.5394718  1.4919622  1.7057363  0.3723181          1     FALSE
## 5  0.2584244  0.8185825  0.9890122  0.3723181          1     FALSE
## 6 -1.0226230 -0.4057441 -0.8027979 -0.1259896          1     FALSE
```

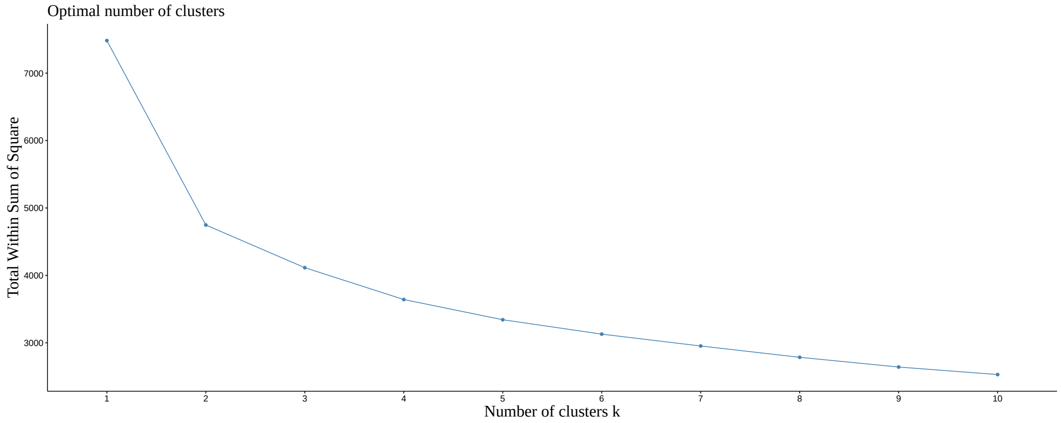
K-means clustering in R



Choosing the optimal amount of clusters

- There is no one best way, but we can use **scree plots** to see what the somewhat optimal number of clusters should be
- A scree plot relates the number of clusters to the sum of squares withing clusters overall
- The more clusters, the lower the sum of squares
- But we do not want to overfit, in which case one observation has its own cluster
- Where to draw the line depends on us. In general, we choose the number of clusters where the sum of squares does not decrease as fast anymore
- The ***fviz_nbclust*** function from the ***factoextra*** package makes such a plot

Choosing the optimal amount of clusters



Clustering features

- We can also cluster the features
- In order to achieve that, the data set needs to be transposed, i.e. rows become columns and columns become rows
- Clusters will be formed according to similarity between features
- In this example: **attack**, **speed**, **hp** etc

Clustering features

```
##   Cluster      specs    Bulbasaur      Ivysaur     Venusaur Mega.Venusaur
## 4       3         hp  -0.94789129  -0.39006284   0.35370842    0.35370842
## 5       3     attack -0.984154531 -0.583909245  0.031852734   0.586038515
## 6       1    defense -0.830465932 -0.382099333  0.258424380   1.539471807
## 7       2  sp_attack -0.252703307  0.206419183  0.818582502   1.491962153
## 8       1 sp_defense  -0.26525488   0.27228816   0.98901221    1.70573626
## 9       2      speed  -0.79039981  -0.29209213   0.37231811    0.37231811
```

Hierarchical clustering

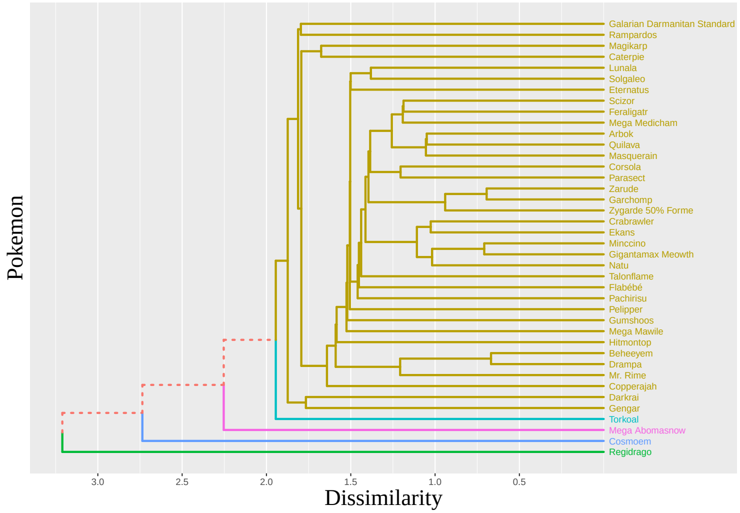
Hierarchical clustering

- K-means clustering requires to choose $\boldsymbol{K}$, which is no easy task (CV would do if it was a supervised problem...)
- **Hierarchical clustering** produces **dendrograms**, but unlike regression trees in supervised learning, they start at the bottom and go up
- At the bottom, each node has a single observation, i.e. each observation is its own cluster
- Observations are “fused” in order of smallest distance (i.e. smallest **dissimilarity**), one by one
- At the top, there is only one cluster where everybody is, so we need to cut the process before then
- In practice, look at the clusters corresponding to different cuts before choosing where to cut, which is not easy task either (CV would do if it was a supervised problem...)

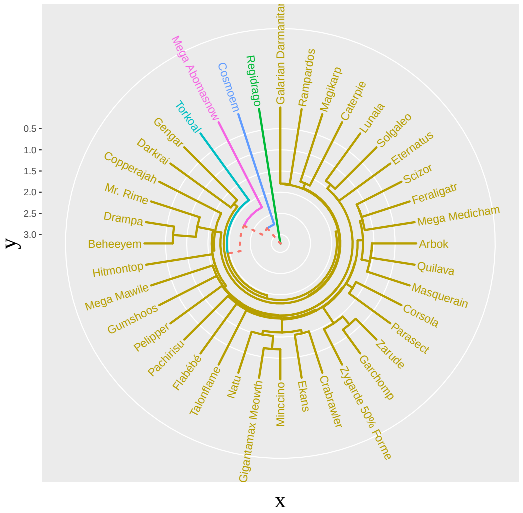
Hierarchical clustering: Height of a dendrogram

- Clusters are fused at different heights in the dendrogram
- The height corresponds to the dissimilarity between the corresponding fused clusters
- The higher the clusters are fused, the less similar they are
- We know how to define dissimilarity between two observations (Euclidean distance). But how to compute the dissimilarity between two groups?
- We need to define the notion of **linkage**. It is based on comparing all pairwise dissimilarities between two clusters. There are four different types:
  - **Complete**: Keep the **largest** of the pairwise dissimilarities
  - **Single**: Keep the **smallest** of the pairwise dissimilarities
  - **Average**: Keep the **average** of the pairwise dissimilarities
  - **Centroid**: Compute the dissimilarity between the centroids of the 2 clusters

Hierarchical clustering in R



Hierarchical clustering in R



Practical issues with clustering

- As with supervised methods, clustering requires fine tuning of some parameters
- But being unsupervised, it is more difficult to choose an optimal cluster structure, as we do not have access to a criterion function to assess the performance of the structure
- In K-means clustering, how many clusters should we choose?
- In hierarchical clustering, what type of linkage should we use? What measure of dissimilarity? Where should we cut the dendrograms?
- Clustering methods assign **each** observation to a cluster, including outliers, which will make some clusters less meaningful
- More generally, clustering methods are very sensitive to a change in the data, i.e. they organize clusters very differently after removing some subsets of the data. But they can also help detect anomalies (outliers with regards to one particular feature)

Conclusion

- Unsupervised problems are more tricky due to the absence of supervision
- Many different configurations are acceptable, so make sure you try many to catch all the relevant patterns
- They are usually a **pre-training** analysis (i.e. before estimation/prediction problems)
- Many other methods exist: fuzzy/soft clustering (assigns probabilities of being in each cluster), distribution-based (clusters data points by probability of belonging to the same distribution), density-based (clusters data points based on their concentration)
- They can save a whole analysis as they can help detect strange patterns or outliers, so do not neglect it!