

Recommender Systems

Algorithms, Data and Security
A.Y. 2024/25

Valeria Cardellini

Global Governance, 3rd year
Science and Technology Major

Recommendations

Becoming
Michelle Obama (Author, Narrator), Random House Audio (Publisher)

What other items do customers buy after viewing this item?

- Educated: A Memoir** Audible Audiobook
Tara Westover
★★★★☆ 5,929
\$0.00 Free with Audible trial
- Girl, Stop Apologizing (Audiobook Exclusive Edition): A Shame-Free Plan for Embracing and Achieving Your Goals** Audible Audiobook
Rachel Hollis
★★★★☆ 689
\$0.00 Free with Audible trial
- Where the Crawdads Sing** Audible Audiobook
Delia Owens
★★★★☆ 7,377
\$0.00 Free with Audible trial
- Girl, Wash Your Face: Stop Believing the Lies About Who You Are So You Can Become Who You Were Meant to Be** Audible Audiobook
Rachel Hollis
★★★★☆ 9,792
\$0.00 Free with Audible trial

- how can i visit antarctica
- how can i visit the white house
- how can i visit north korea
- how can i visit canada
- how can i visit cuba

Recommender Systems: The Textbook 1st ed. 2016 Edition
by Charu C. Aggarwal -- (Author)
★★★★☆ 9 customer reviews

Look inside

ISBN-13: 978-3319296579
ISBN-10: 3319296574
Why is ISBN important?

Have one to sell? [Sell on Amazon](#)

[Add to List](#)

Share [Facebook](#) [Twitter](#) [LinkedIn](#) [Reddit](#)

Customers who bought this item also bought


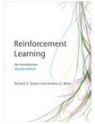
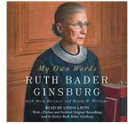
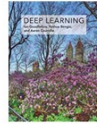




- CAUSALITY**
Judea Pearl
Causality: Models, Reasoning and Inference
- Judea Pearl
★★★★☆ 38
Hardcover \$52.99
- HADOOP**
The Definitive Guide: Storage and Analytics at Internet Scale
- Tom White
★★★★☆ 67
Paperback \$36.06

"We are leaving the age of information and entering the age of recommendation." Chris Anderson

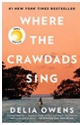
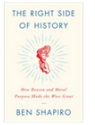



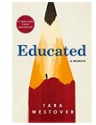

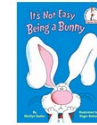
Recommendations

Your recently viewed items and featured recommendations

Inspired by your browsing history

 <p>Statistical Methods for Recommender Systems Deepak K. Agarwal ★★★★☆ 4 Hardcover \$59.29</p>	 <p>Reinforcement Learning: An Introduction... Richard S. Sutton ★★★★☆ 13 Hardcover \$50.71</p>	 <p>My Own Words Ruth Bader Ginsburg ★★★★☆ 234 Audiobook \$0.00 Free with Audible trial</p>	 <p>Deep Learning (Adaptive Computation and...) Ian Goodfellow ★★★★☆ 193 Hardcover \$28.80</p>	 <p>BECOMING (German edition): Meine Geschichte Michelle Obama ★★★★☆ 23 Audiobook \$0.00 Free with Audible trial</p>	 <p>Hands-On Machine Learning with Scikit-Learn & TensorFlow Aurélien Géron ★★★★☆ 265 Paperback \$56.99</p>	 <p>Born a Crime: Stories from a South African Childhood Trevor Noah ★★★★☆ 6,109 Audiobook \$0.00 Free with Audible trial</p>	 <p>Machine Learning for Text Charu C. Aggarwal ★★★★☆ 6 Hardcover \$48.61</p>
---	---	---	--	--	--	---	---

Best Sellers

 <p>Where the Crawdads Sing Delia Owens ★★★★☆ 7,377 Hardcover \$15.60</p>	 <p>The Right Side of History: How Reason and Moral... Ben Shapiro ★★★★☆ 202 Hardcover \$16.79</p>	 <p>Supermarket Bobby Hall Paperback \$13.51</p>	 <p>Girl, Wash Your Face: Stop Believing the Lies About... Rachel Hollis ★★★★☆ 9,792 Hardcover \$11.80</p>	 <p>Raise Your Hand Alice Paul Tapper Hardcover \$11.04</p>	 <p>Educated: A Memoir Tara Westover ★★★★☆ 5,929 Hardcover \$16.80</p>	 <p>Girl, Stop Apologizing: A Shame-Free Plan for... Rachel Hollis ★★★★☆ 689 Hardcover \$14.77</p>	 <p>It's Not Easy Being a Bunny (Beginner Books(R)) Marilyn Sadler ★★★★☆ 266 Hardcover \$6.00</p>
---	--	--	--	---	---	--	---

Valeria Cardellini - ADS 2024/25

2

Recommendations



A **recommender system** is a system able to provide or suggest items to end users

It helps to match users with items

Valeria Cardellini - ADS 2024/25

3

From scarcity to abundance

- Shelf space is a scarce commodity for traditional retailers
 - Also: TV networks, movie theaters,...
- Web enables **near-zero-cost dissemination of information** about products
 - From scarcity to abundance
- More choice necessitates **better filters**
 - Recommender systems

The power of recommender systems

- When does a recommender system do its job well?



The power of recommender systems

- How *Into Thin Air* made *Touching the Void* a bestseller <https://www.nytimes.com/2006/08/10/books/10manly.html>
- In 1988, a British mountain climber named Joe Simpson wrote a book called *Touching the Void*, a harrowing account of near death in the Peruvian Andes. It got good reviews but, only a modest success, it was soon forgotten. Then, a decade later, a strange thing happened. Jon Krakauer wrote *Into Thin Air*, another book about a mountain-climbing tragedy, which became a publishing sensation. Suddenly *Touching the Void* started to sell again.
- Random House rushed out a new edition to keep up with demand. Booksellers began to promote it next to their *Into Thin Air* displays, and sales rose further. A revised paperback edition, which came out in January, spent 14 weeks on the *New York Times* bestseller list. That same month, IFC Films released a docudrama of the story to critical acclaim. Now *Touching the Void* outsells *Into Thin Air* more than two to one.
- What happened? In short, [Amazon.com recommendations](#). The online bookseller's software noted patterns in buying behavior and suggested that readers who liked *Into Thin Air* would also like *Touching the Void*. People took the suggestion, agreed wholeheartedly, wrote rhapsodic reviews. More sales, more algorithm-fueled recommendations, and the positive feedback loop kicked in.

Why do we need recommender systems?

- Long tail and digital products/services
 - Reduce cognitive load on users
 - Enhance user experience
 - Increase loyalty and volume
 - Introduce quality
 - Help with inventory control
- ... and much more!

Netflix

NETFLIX

- Media services provider
- Netflix's initial business included DVD sales and rental by mail
- Current primary business: subscription-based streaming OTT service which offers online streaming of a library of films and television programs, including those produced in-house
 - More than 220 million subscribers
 - Netflix video streaming generated 9.3% of global Internet traffic... (1 out of 10 bits)

[Netflix revenue and usage statistics](#)

Netflix's prize (2006-2009)

- Goal: offer customers accurate movie recommendations so that they find content to watch and enjoy and Netflix maximizes customer satisfaction and retention
- On October 2006, Netflix offered a \$1,000,000 prize to the first developer of a movie-recommendation algorithm that could beat its existing algorithm Cinematch at predicting customer ratings for movies by more than 10% en.wikipedia.org/wiki/Netflix_Prize
- Data made available: 100 Million ratings (1999-2005)
 - Ratings came from > 480,000 users and 17,770 movies
- Really difficult: big, sparse and skewed data
 - Many users rated only few movies, and very few users rated huge number of movies (one user rated over 17,000 movies)

How to measure success?

- Which algorithm can find the predicted ratings most similar to the actual ones?
- From most relevant to most tractable metric:
 - Customer satisfaction
 - Prediction effectiveness: how well can we predict the ratings users give to movies they watched?
- **Prediction error**
 - Can be calculated for those (u,i) pairs (formed by user u and movie i) for which we have both prediction and actual rating (let's say C such pairs)
 - r_{ui} : actual rating given by user u to movie i
 - \hat{r}_{ui} : predicted rating for user u and movie i

Distance metrics

- How much are the prediction and actual rating distant? Use **distance metrics**
- Let's define two distance metrics
 - Hamming distance
 - Root Mean Square Error

Distance metrics

- **Hamming distance**

- Number of positions in which two binary strings (of equal length) differ, e.g.,
 - 100 and 011 differ in 3 positions: $\text{dist}_{\text{Hamming}} = 3$
 - 01011 and 11010 differ in 2 positions: $\text{dist}_{\text{Hamming}} = 2$
- To apply this distance, we need to transform real values into binary codes

Distance metrics

- **Root Mean Square Error (RMSE)**

- Square root of the second sample moment of the differences between observed values r_{ui} and predicted values \hat{r}_{ui} or the quadratic mean of these differences

$$RMSE = \sqrt{\sum_{(u,i) \in C} \frac{(r_{ui} - \hat{r}_{ui})^2}{|C|}}$$

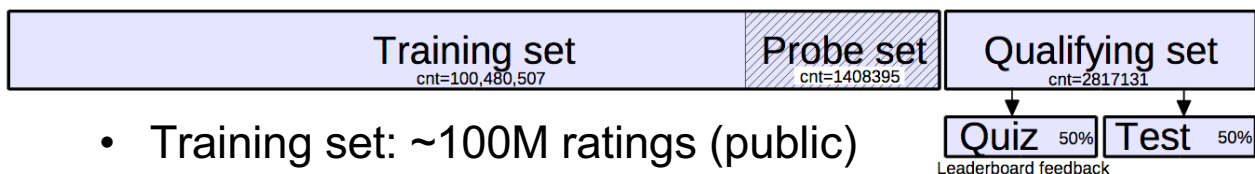
user u
movie i

- RMSE is always non-negative, and a value of 0 indicates a perfect fit to data
- The smaller the RMSE, the better the recommender system
- RMSE is the metric used to evaluate solutions for Netflix's prize

Which recommendation to users?

- Only top few predictions in rank will be recommended to each user
 - E.g., top 5 movies with the highest predicted rating
- Ultimate test is whether user decides to watch recommended movies, and whether she likes them or not

Netflix's prize dataset



- Training set: ~100M ratings (public)
- Probe set: ~1.4M ratings (public)
 - Similar statistical properties to Test and Quiz sets
 - Competitors could use Probe set to test their algorithms
- Quiz set: ~1.4M ratings (hidden)
 - Competitors could submit an algorithm that would run on quiz test, but not more than once a day
- RMSE scores updated on Netflix prize's website were based on performance on quiz set
- Final decision was based on comparison of RMSE on test set
 - Cinematch's RMSE is 0.9514

Netflix's prize: the contest

- Within a week, Cinematch was beaten
- By June 2007, over 20,000 teams had registered from over 150 countries
- By Sep. 2007, team BellKor made 8.26% improvement
- First place changed hands a few times, until after 1 year BellKor got 8.43%
- Teams ranked in the first positions merged and in June 2009, BellKor's Pragmatic Chaos were first to achieve $> 10\%$
- On test set both BellKor's Pragmatic Chaos and The Ensemble got RMSE equal to 0.8567!

Full story at <https://www.wired.com/2009/09/bellkors-pragmatic-chaos-wins-1-million-netflix-prize/>

Netflix: Predicting the best user ratings

- Netflix was willing to pay over \$1M for the best algorithm, which shows how critical the recommender system was to their business
 - Even though recommendation systems existed before this prize, new research began
- What data can be used to predict Netflix user ratings?
 - Every movie has the rating from users who have ranked it
 - We also know attributes about the movie itself: actors, director, genre classification, year released, etc.

More in general: input data

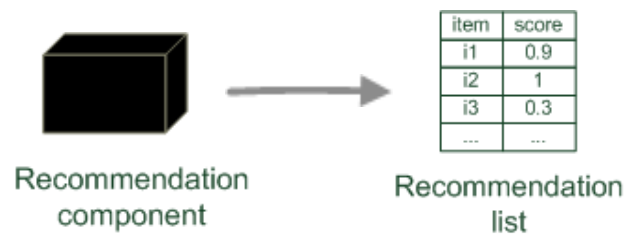
- What input data could be used to predict user ratings? How input data can be gathered?
- Explicit data: users intentionally provide indications about their preferences, e.g.,
 - Customer ratings
 - Feedback
 - Demographics
- Implicit data: stems from monitoring user behavior, e.g.,
 - Purchase history
 - Click or browse history
- Product information
 - Product taxonomy, attributes, description

Netflix's prize: Recommendation algorithm

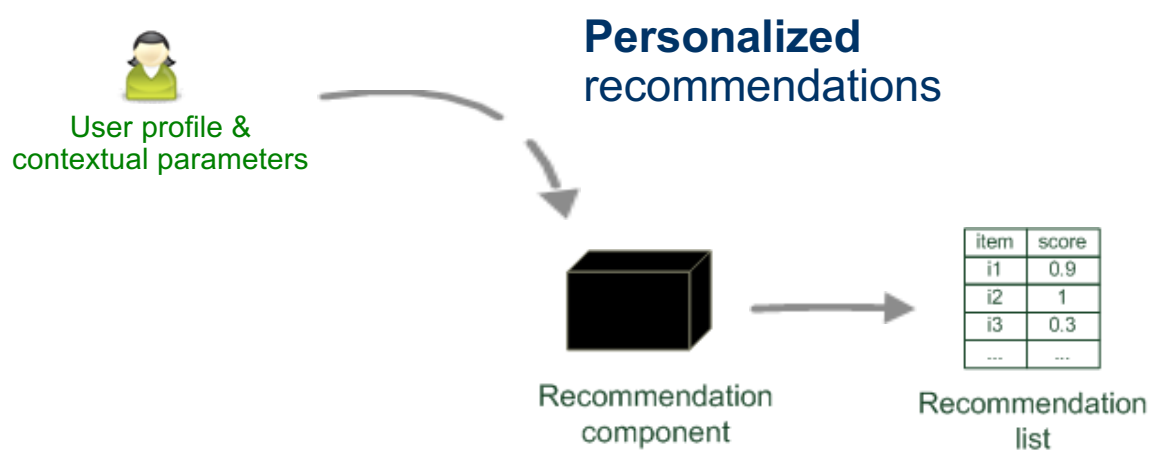
- What algorithm used in winning solution?
- Netflix's prize winner is a **cocktail of many methods combined**, with many different algorithms blended together
- This variety of approaches is a general principle of analytics
 - We will focus on main approaches, big ideas, and few key methodologies
- For an overview of techniques used by Netflix recommender system:
<https://dl.acm.org/doi/pdf/10.1145/2843948>

Approaches to recommender systems

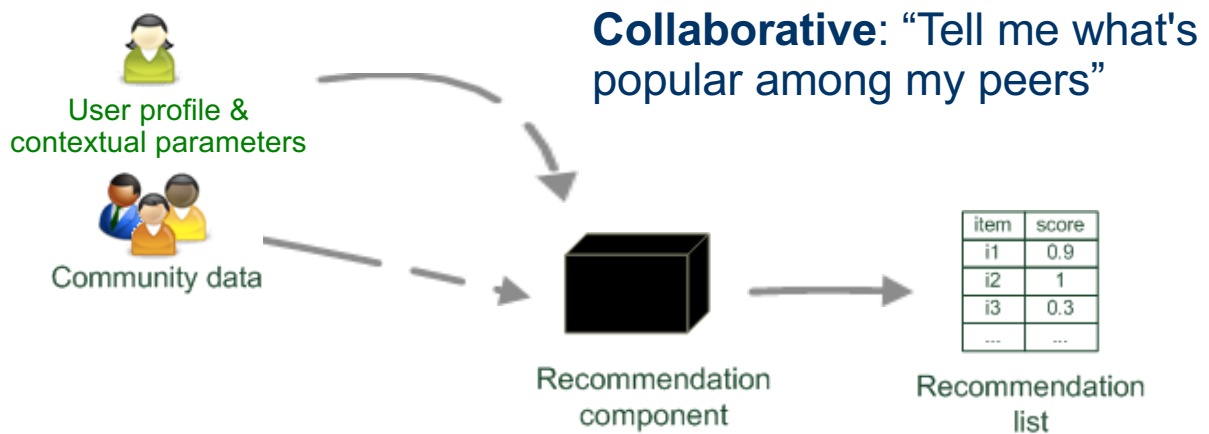
Recommender systems reduce information overload by estimating relevance



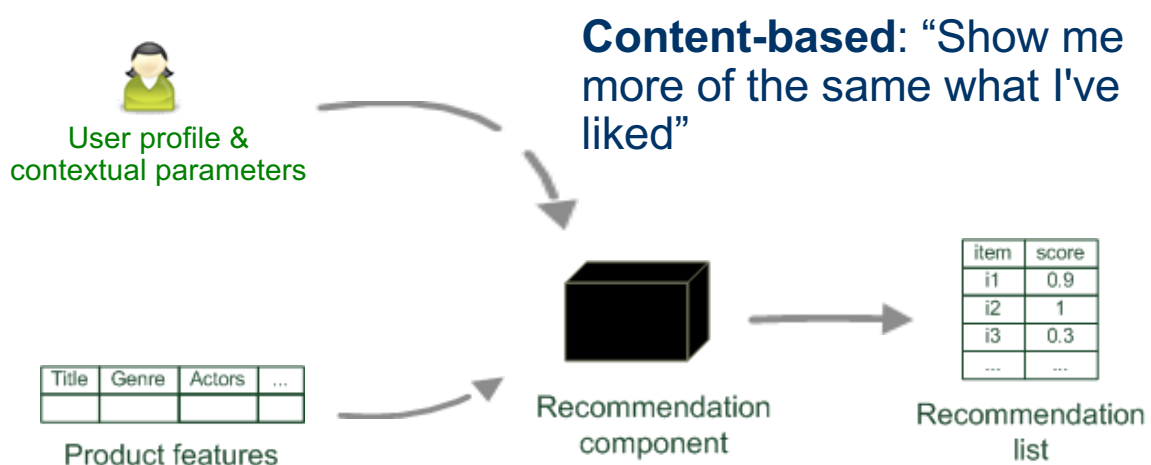
Approaches to recommender systems



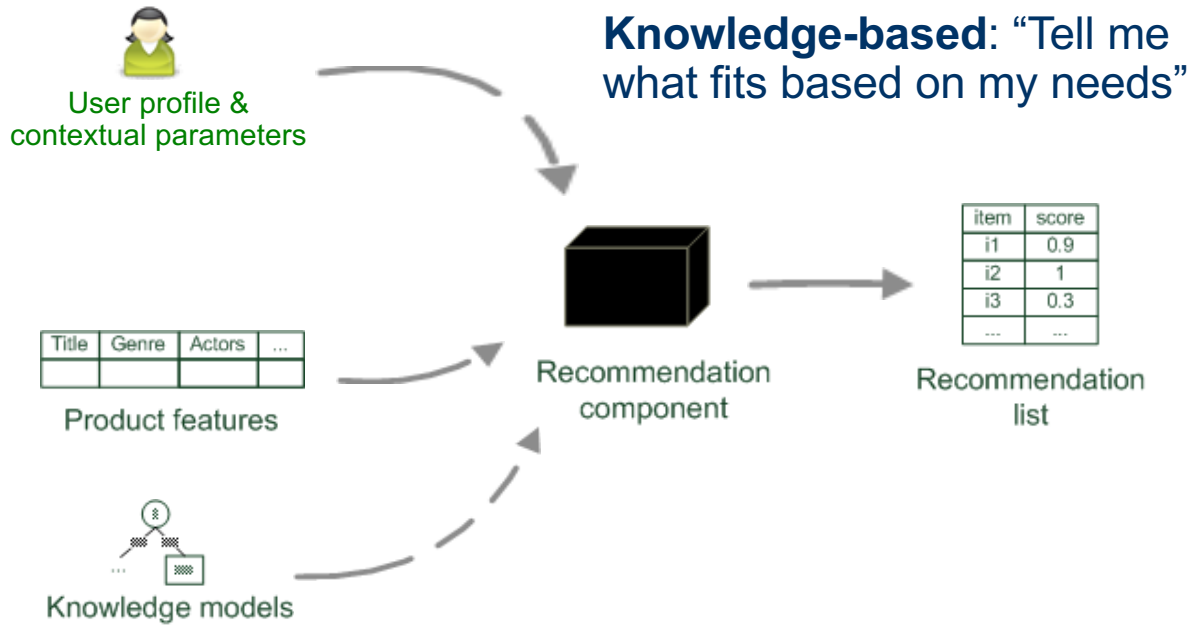
Approaches to recommender systems



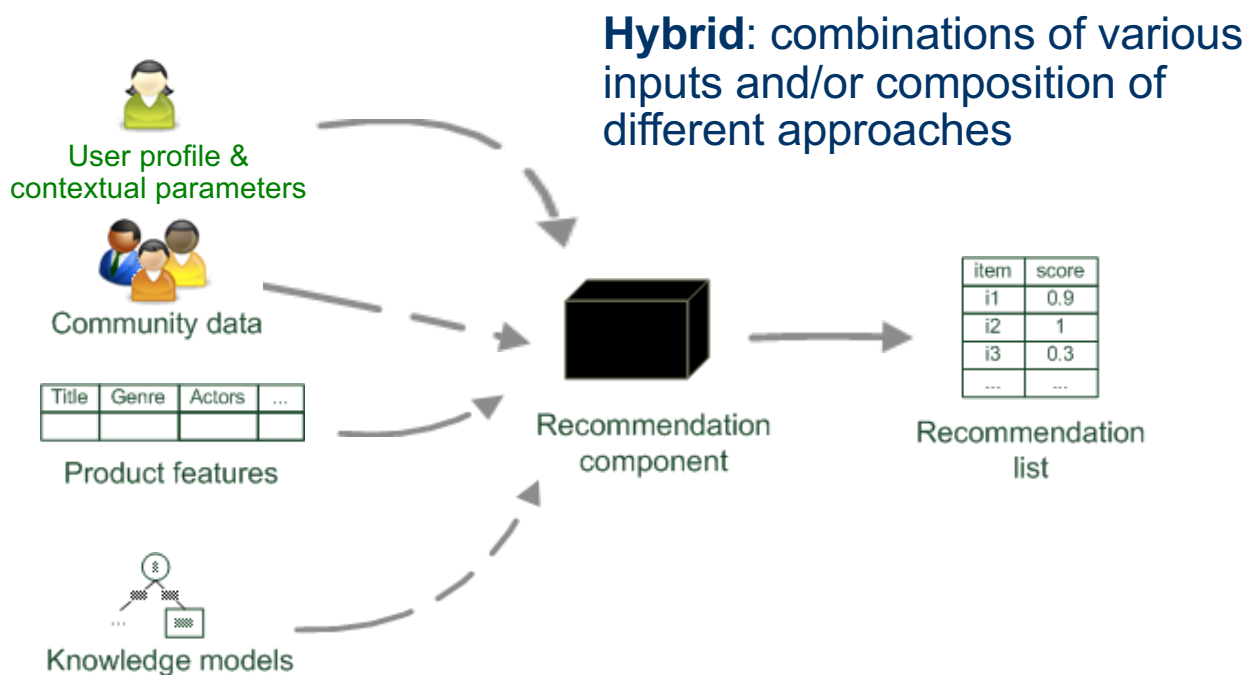
Approaches to recommender systems



Approaches to recommender systems



Approaches to recommender systems



Paradigms of recommender systems

- We analyze:
- Collaborative filtering
 - Make **automatic predictions** (filter) about the interests of a user by collecting **preferences information from many users** (collaborative)
 - Assumption: similar users will have similar ratings
- Content-based filtering
 - Recommend items based on a **comparison** between **content of items** and **user profile**

Collaborative filtering

- Use other users' rankings to make predictions about user's rating for an item, which the user hasn't rated yet
 - These predictions are built upon the known ratings of other users, who have similar ratings with the user
 - E.g., predict Eva's rating for Inception

Ratings matrix

	Forrest Gump	Godfather	Inception	Jaws
Amy	5		4	3
Bob	3	5	2	5
Carl		3	5	4
Dan	4	5	4	
Eva	4	4	?	3

Unknown rating we want to predict →

Collaborative filtering

- Let's calculate mean values for ratings

	Forrest Gump	Godfather	Inception	Jaws	Mean rating
Amy	5		4	3	4.00
Bob	3	5	2	5	3.75
Carl		3	5	4	4.00
Dan	4	5	4		4.33
Eva	4	4		3	3.67

User: u

Movie: i

Rating: 1-5

n_u : number of ratings by user u

Mean rating $\mu_u = \frac{1}{n_u} \sum_i r_{ui}$

Collaborative filtering

- Let's measure the similarity between each pair of users
 - We will see later how
 - For now: similarity values range between -1 and 1, where -1 is perfectly dissimilar and 1 is perfectly similar

	Amy	Bob	Carl	Dan	Eva
Amy	1	-0.54	0.00	-0.29	0.87
Bob	-0.54	1	-0.82	0.78	-0.32
Carl	0.00	-0.82	1	-0.87	-0.29
Dan	-0.29	0.78	-0.87	1	0.17
Eva	0.87	-0.32	-0.29	0.17	1

Collaborative filtering

	Forrest Gump	Godfather	Inception	Jaws	Mean rating
Amy	5		4	3	4.00
Bob	3	5	2	5	3.75
Carl		3	5	4	4.00
Dan	4	5	4		4.33
Eva	4	4		3	3.67

	Amy	Bob	Carl	Dan	Eva
Amy	1	-0.54	0.00	-0.29	0.87
Bob	-0.54	1	-0.82	0.78	-0.32
Carl	0.00	-0.82	1	-0.87	-0.29
Dan	-0.29	0.78	-0.87	1	0.17
Eva	0.87	-0.32	-0.29	0.17	1

- Can predict Eva's rating for Inception:

$$3.67 + [0.87(4-4) - 0.32(2-3.75) - 0.29(5-4) + 0.17(4-4.33)] / (0.87 - 0.32 - 0.29 + 0.17) = 4.17$$
- Eva will like Inception more than average

Exploit similarities between users

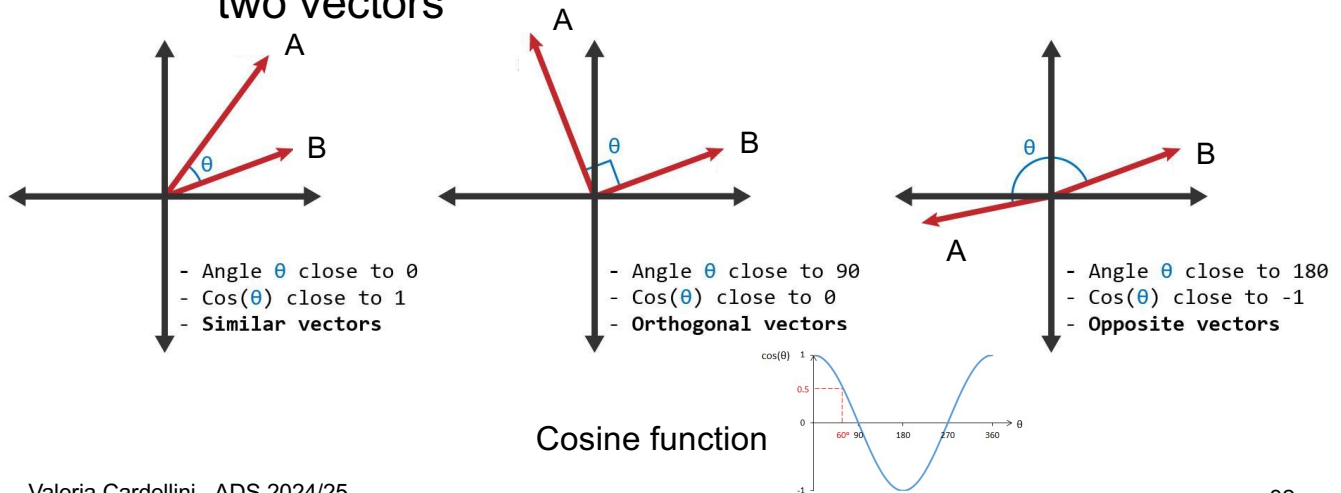
	Forrest Gump	Godfather	Inception	Jaws	Mean rating
Amy	5		4	3	4.00
Bob	3	5	2	5	3.75
Carl		3	5	4	4.00
Dan	4	5	4		4.33
Eva	4	4		3	3.67

	Amy	Bob	Carl	Dan	Eva
Amy	1	-0.54	0.00	-0.29	0.87
Bob	-0.54	1	-0.82	0.78	-0.32
Carl	0.00	-0.82	1	-0.87	-0.29
Dan	-0.29	0.78	-0.87	1	0.17
Eva	0.87	-0.32	-0.29	0.17	1

- Consider suggesting to Eva "Inception", since Amy rated it more than her average, and Eva and Amy seem to have similar preferences
- This technique is called **user-based collaborative filtering**

Collaborative filtering: measuring similarity

- How to measure similarity of users?
- We consider **cosine similarity**
 - Cosine of angle θ between two user vectors A and B in an n -dimensional space (figure: $n=2$)
 - It measures the similarity in the directions of the two vectors



Valeria Cardellini - ADS 2024/25

32

Cosine similarity

- We consider **cosine similarity**
 - Cosine of angle θ between two vectors A and B in an n -dimensional space
 - Similarity score ranges between -1 and 1
 - $\cos(0^\circ) = 1$: perfectly similar
 - $\cos(90^\circ) = 0$: orthogonal (or uncorrelated)
 - $\cos(180^\circ) = -1$: perfectly dissimilar
- For a pair of users: the larger the score, the closer their taste

$$\text{cosine similarity} = \cos(\theta) = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

- A_i and B_i are components of vector A and B, respectively

Valeria Cardellini - ADS 2024/25

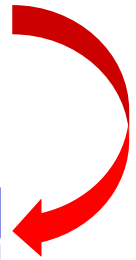
33

Cosine similarity: Example

- To deal with biases in the ratings (e.g., movie critic who always gives out low ratings), let's **normalize ratings** by subtracting from each rating the average rating of that user

	Forrest Gump	Godfather	Inception	Jaws	Mean rating
Amy	5		4	3	4.00
Bob	3	5	2	5	3.75
Carl		3	5	4	4.00
Dan	4	5	4		4.33
Eva	4	4		3	3.67

	Forrest Gump	Godfather	Inception	Jaws
Amy	5-4=1		4-4=0	3-4=-1
Bob	3-3.75=-0.75	5-3.75=1.25	2-3.75=-1.75	5-3.75=1.25
Carl		3-4=-1	5-4=1	4-4=0
Dan	4-4.33=-0.33	5-4.33=0.67	4-4.33=-0.33	
Eva	4-3.67=0.33	4-3.67=0.33		3-3.67=-0.67



Cosine similarity: Example

Normalized ratings matrix

	Forrest Gump	Godfather	Inception	Jaws
Amy	1		0	-1
Bob	-0.75	1.25	-1.75	1.25
Carl		-1	1	0
Dan	-0.33	0.67	-0.33	
Eva	0.33	0.33		-0.67

Note: after normalization, a negative number means below average rating and a positive number means above average ratings given by the same user

- Now compute the cosine similarity for each pair (A, B) of users

$$\text{cosine similarity} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

- E.g., cosine similarity between Amy and Bob:

$$\frac{1 * (-0.75) + 0 * (-1.75) + (-1) * 1.25}{\sqrt{1^2 + 0^2 + (-1)^2} \sqrt{(-0.75)^2 + 1.25^2 + (-1.75)^2 + 1.25^2}} = \frac{-2}{3.6742} = -0.5443$$

- Note: lack of ratings is considered as 0 ratings

Cosine similarity: Example

- Now compute the cosine similarity for each pair (A, B) of user

$$\text{cosine similarity} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

- E.g., cosine similarity between Amy and Eva:

$$\frac{1 * (0.33) + (-1) * (-0.67)}{\sqrt{1^2 + 0^2 + (-1)^2} \sqrt{0.33^2 + 0.33^2 + (-0.67)^2}} = \frac{1}{1.1547} = 0.8660$$

- We obtain the following similarity values:

	Amy	Bob	Carl	Dan	Eva
Amy	1	-0.54	0.00	-0.29	0.87
Bob	-0.54	1	-0.82	0.78	-0.32
Carl	0.00	-0.82	1	-0.87	-0.29
Dan	-0.29	0.78	-0.87	1	0.17
Eva	0.87	-0.32	-0.29	0.17	1

How to predict the rating?

- Given the similarity table, the predicted rating \hat{r}_{ui} of item i for user u is given by:

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \neq u} s_{uv} (r_{vi} - \mu_v)}{\sum_{v \neq u} s_{uv}}$$

s_{uv} : similarity between u and v
 μ_u : mean rating for u

	Forrest Gump	Godfather	Inception	Jaws	Mean rating
Amy	5		4	3	4.00
Bob	3	5	2	5	3.75
Carl		3	5	4	4.00
Dan	4	5	4		4.33
Eva	4	4		3	3.67

	Amy	Bob	Carl	Dan	Eva
Amy	1	-0.54	0.00	-0.29	0.87
Bob	-0.54	1	-0.82	0.78	-0.32
Carl	0.00	-0.82	1	-0.87	-0.29
Dan	-0.29	0.78	-0.87	1	0.17
Eva	0.87	-0.32	-0.29	0.17	1

- Eva's rating for Inception:

$$\hat{r}_{Eva, Inception} = 3.67 + \frac{[0.87(4-4) - 0.32(2-3.75) - 0.29(5-4) + 0.17(4-4.33)]}{(0.87 - 0.32 - 0.29 + 0.17)} = 4.1674$$

More on the predicted rating formula

- Given the similarity table, the predicted rating \hat{r}_{ui} of item i for user u is given by:

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \neq u} s_{uv}(r_{vi} - \mu_v)}{\sum_{v \neq u} s_{uv}} \quad \begin{array}{l} s_{uv}: \text{similarity between } u \text{ and } v \\ \mu_u: \text{mean rating for } u \end{array}$$

- What does this formula mean?
 - μ_u is a baseline predictor for \hat{r}_{ui}
 - Weighted average approach: we multiply each normalized rating by a similarity factor (which tells how similar the users are)
 - We add weights to the ratings: the heavier the weight, the more the rating would matter

Collaborative filtering: all the steps so far

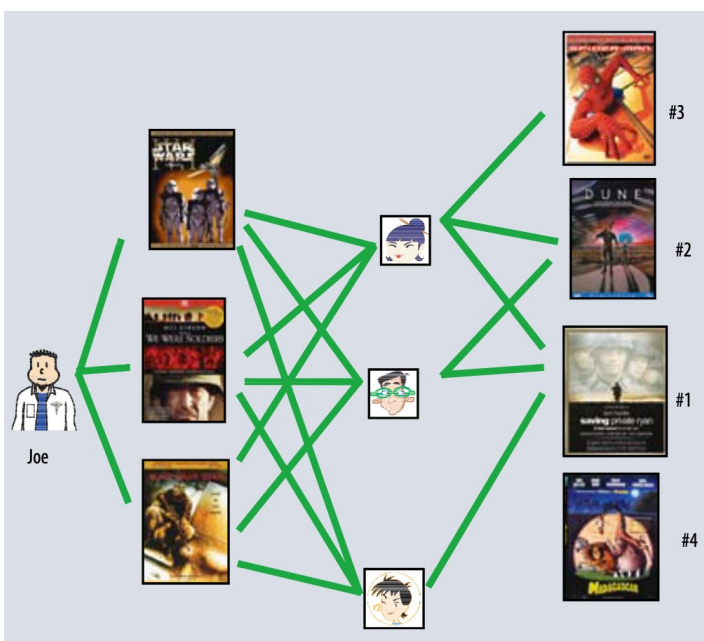
1. Calculate the mean rating for each user
2. Normalize the ratings matrix
3. Compute the cosine similarity for each pair of users
4. Compute the predicted rating(s)

Collaborative filtering: neighborhood-based

- Real problems are much larger and sparser than our example
- When dealing with large dataset, we do not consider all the users (too expensive!) but only like-minded users that is, users with high similarity (**neighbors** set K) to the user in question
 - E.g., select top-K similar users and use their ratings to recommend movies: weighted average of the ratings given only by **similar users**

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in K} s_{uv}(r_{vi} - \mu_v)}{\sum_{v \in K} s_{uv}}$$

Collaborative filtering: neighborhood-based



Joe likes the 3 movies on the left.

To make a prediction for him, the recommender system finds similar users who also liked those movies, and then determines which other movies they liked.

In this case, all three liked Saving Private Ryan, so that is the first recommendation. Two of them liked Dune, so that is next, and so on.

Collaborative filtering: neighborhood-based

- Summing up the general approach for neighborhood-based collaborative filtering:
 1. Find the cohort of other similar users who have rated the item in question
 2. Predict the rating from the ratings of similar users

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in K} s_{uv}(r_{vi} - \mu_v)}{\sum_{v \in K} s_{uv}}$$

Exercise: Collaborative filtering

- Applying user-based collaborative filtering, predict Eric's rating for Titanic

	The Matrix	Titanic	Die Hard	Forrest Gump	Wall•E
John	5	1		2	2
Lucy	1	5	2	5	5
Eric	2	?	3	5	4
Diane	4	3	5	3	

Netflix's prize solution

- Netflix's prize solution uses **matrix factorization**, a **more complex collaborative filtering algorithm** than the user-based one
- Idea: represent users and items in a lower dimensional latent space, exploiting relationships among users and items (that could also be hidden that is, *latent*) to predict ratings
 - For a gentle introduction: watch <https://www.youtube.com/watch?v=ZspR5PZemcs> and read "Recommendation Systems: Collaborative Filtering using Matrix Factorization - Simplified" <https://medium.com/sfu-csmp/recommendation-systems-collaborative-filtering-using-matrix-factorization-simplified-2118f4ef2cd3>
 - For more details: read "Matrix factorization techniques for recommender systems" [https://datajobs.com/data-science-repo/Recommender-Systems-\[Netflix\].pdf](https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf)

Valeria Cardellini - ADS 2024/25

44

Matrix factorization: dependencies

- Let's analyze some example of row and column dependencies
 - Relationship is easy: two very similar movies

	M1	M2	M3	M4	M5
U1	3			3	
U2	1			1	
U3	3			3	
U4	4			4	



M1 = M4



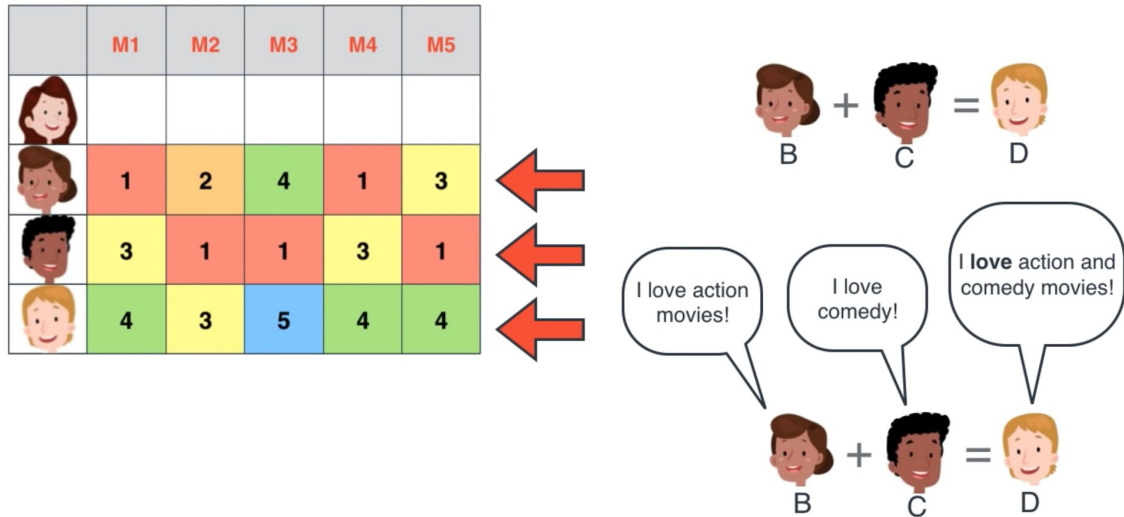
Mall Cop



Observe and Report

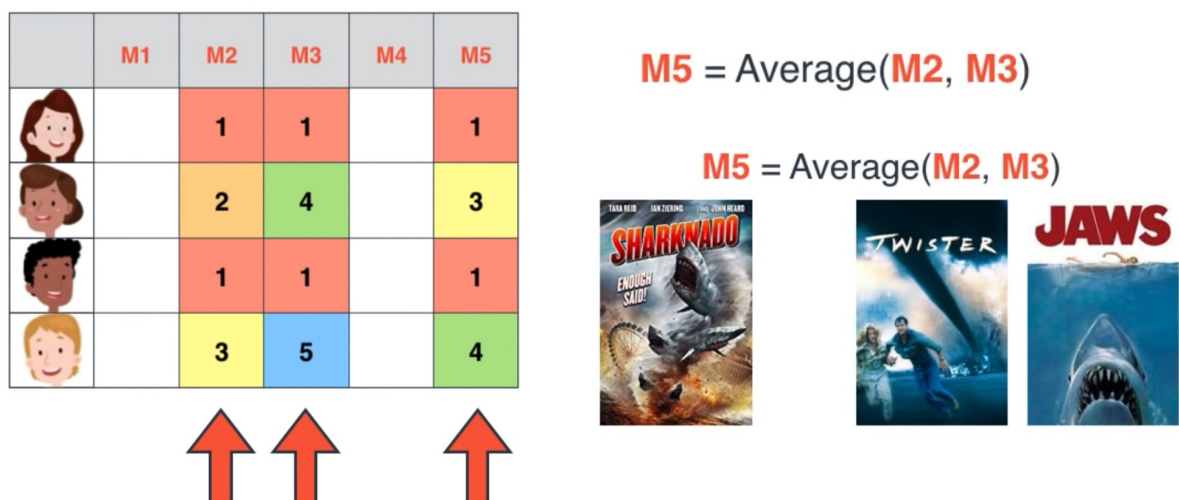
Matrix factorization: dependencies

- Let's analyze some example of row and column dependencies
 - Relationship is less easy



Matrix factorization: dependencies


- Let's analyze some example of row and column dependencies
 - Another example of hidden relationship



Matrix factorization: the idea

- We need a mathematical tool to figure out all the dependencies
- Specifically, we conjecture that the rating matrix R is actually the **product of two matrices**
- Why “matrix factorization”? We decompose R into the products of two matrices P and Q

this \times that =



	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

- Formally, $R \approx P \times Q = \hat{R}$







Matrix factorization





- We need a mathematical tool to figure out all the dependencies

Movie features Q

	M1	M2	M3	M4	M5
 Comedy	3	1	1	3	1
 Action	1	2	4	1	3

User features P

	 Comedy	 Action
 A	✓	✗
 B	✗	✓
 C	✓	✗
 D	✓	✓

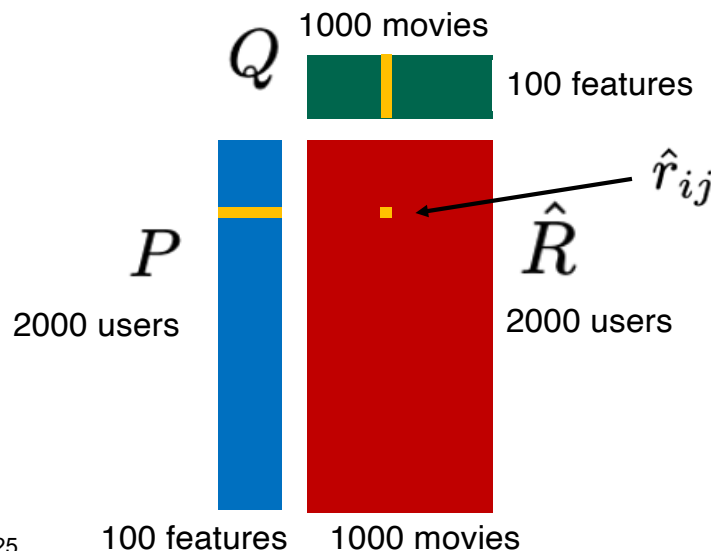
	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

Ratings R

Matrix factorization: dot product

- How to get each value of matrix \hat{R} ? We use the **dot product**

predicted rating $\rightarrow \hat{r}_{ij} = \mathbf{p}_i \cdot \mathbf{q}_j = \sum_{k=1}^d p_{ik} q_{kj}$



Matrix factorization: dot product

- We can use the entries of the product $P \times Q$ to estimate the corresponding unknown entries (i.e., the predicted ratings) in the matrix \hat{R}

Movie features Q

	M1	M2	M3	M4	M5
F1	3	1	1	3	1
F2	1	2	4	1	3

User features P

	F1	F2
A	1	0
B	0	1
C	1	0
D	1	1

	M1	M2	M3	M4	M5
A	3	?	1		1
B	1		4	1	
C	3	1		3	1
D		3		4	4

Matrix factorization: dot product

- Let's use the dot product to predict the unknown ratings
- Predict user A rating for movie M2 using the dot product

Movie features Q

F1	3	1	1	3	1
F2	1	2	4	1	3

$$\hat{r}_{12} = (1, 0) \cdot (1, 2) = 1 \times 1 + 0 \times 2 = 1$$

User features P

	F1	F2
A	1	0
B	0	1
C	1	0
D	1	1

	M1	M2	M3	M4	M5
A	3	1	1	3	1
B	1	2	4	1	3
C	3	1	1	3	1
D	4	3	5	4	4

Matrix factorization

- How do we find P and Q? Let's use Machine Learning
- Idea: the system learns the model to find latent factors by fitting the known ratings
- First initialize the two matrices P and Q with some values, calculate how different their product is to the known ratings and then try to minimize this difference (i.e., error) iteratively
- Such a method is called **gradient descent**, aiming at finding a local minimum of the error

Matrix factorization

- First initialize the two matrices P and Q with some values, calculate how different their product is to the known ratings

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = \left(r_{ij} - \sum_{k=1}^d p_{ik} q_{kj} \right)^2$$

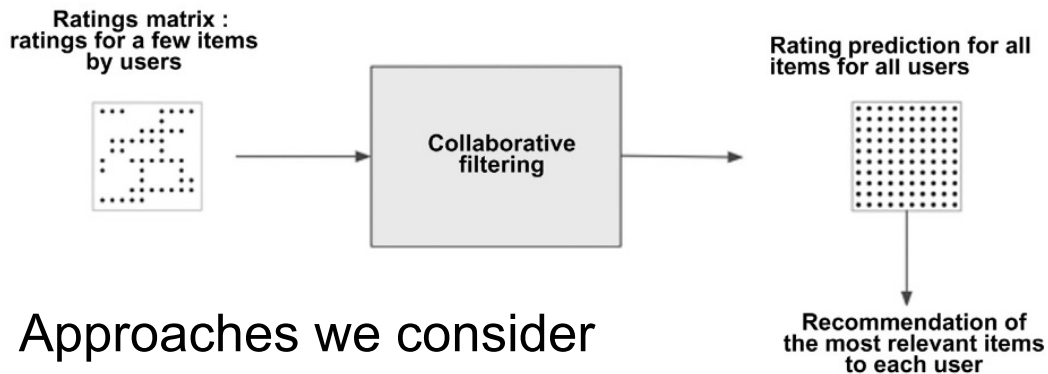
- A good measure of how close the product PQ is to the given rating matrix is the RMSE
- Then try to minimize this difference iteratively by adjusting some element of P or Q

User-based vs. item-based collaborative filtering

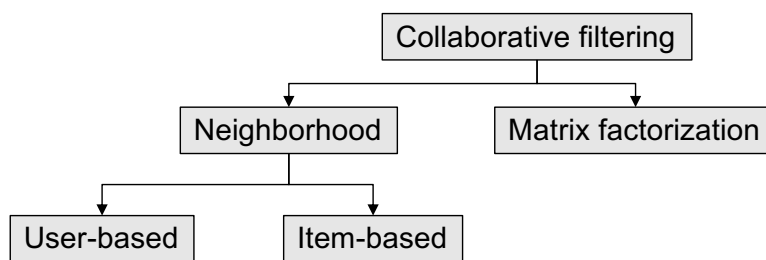
- So far, we have considered **user-based collaborative filtering**
- Collaborative filtering can also be **item-based**
 - Based on the assumption that users prefer items that are similar to previously preferred items
 - Compute item similarities based on user ratings
 - How to? Transpose the normalized ratings matrix (swapping rows and columns) and then compute the cosine similarity between each pair of items

Collaborative filtering: summing up

- Overall picture



- Approaches we consider



Issues with ratings

- Explicit information of user taste is great
- But in reality ratings are
 - Sparse
 - Noisy
 - Biased
- Netflix complains that ratings quality has decreased over time
- Recommendation is not just ratings

Content-based recommender systems

- Leverage features (or attributes) about items
- Similarity of items is determined by measuring the similarity in their features
- E.g., consider relevant movie features
 1. Cast: some users prefer movies with their favorite actors
 2. Director: some users have a preference for the work of certain directors
 3. Released year: some users prefer old movies; others watch only the latest releases
 4. Genre: some users like only comedies, others dramas or romances

Content filtering

- If we know that Amy likes “Men In Black”
 - Directed by Barry Sonnenfeld
 - Classified in the genres of action, adventure, sci-fi and comedy
 - Stars actor Will Smith
 - Consider recommending to Amy:
 - Barry Sonnenfeld’s movie “Get Shorty”
 - “Jurassic Park”, which is in the genres of action, adventure, and sci-fi
 - Will Smith’s movie “Hitch”
-
- This technique is called **content filtering**

<https://www.ibm.com/think/topics/content-based-filtering>

Strengths and weaknesses

- Collaborative filtering
 - ✓ Item domain independent: can be applied to predicting ratings of different items without having any knowledge about the items except the ratings given by the users
 - ✗ Requires a lot of data about users to make accurate recommendations
 - ✗ Millions of items and users: time-consuming and a lot of computing power
 - ✗ Suffers from cold-start: difficulty of making recommendations when users (or items) are new
 - How to mitigate cold-start? Ask new users to select or enter their preferred items once they have signed up

Strengths and weaknesses

- Content filtering
 - ✗ Item domain dependent
 - ✓ Requires little data to get started
 - ✗ Can be limited in scope (recommend only similar items)
 - ✓ Does not suffer from cold-start

Hybrid recommendation systems

- As an example, consider a collaborative filtering approach where we determine that Amy and Eva have similar preferences
- We could then do content filtering, where we would find that “Forrest Gump”, which both Amy and Eva liked, is classified in almost the same set of genres as “Rain Man”
- As result, we recommend “Rain Man” to both Amy and Eva, even though neither of them have seen it before

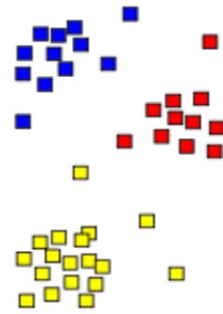
How to find groups of movies with similar sets of genres?

- Consider MovieLens dataset
 - A movie recommendation website <https://movielens.org/>
- Movies in the dataset are categorized as belonging to different genres

Unknown	Action	Adventure	Animation	Children's
Comedy	Crime	Documentary	Drama	Fantasy
Film Noir	Horror	Musical	Mystery	Romance
Sci-Fi	Thriller	War	Western	
- Each movie may belong to many genres
- How can we systematically find groups of movies with similar sets of genres?

Why clustering?

- Clustering is the process of examining a collection of “points,” and grouping points into “clusters” according to some distance measure
- Cluster analysis is part of **unsupervised learning**
 - Goal is to segment data into similar groups instead of prediction
 - Can also cluster data into “similar” groups and then build a predictive model for each group
 - Be careful not to overfit your model!
 - This works best with large datasets

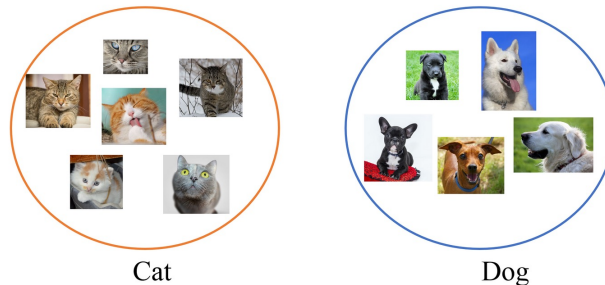


Break: Machine learning (ML)

- ML is the study of computer algorithms that improve automatically through **experience** and by use of **data**
 - «Machine learning is a field of study that gives computers the ability to learn without being explicitly programmed», Arthur Samuel (1959)
- ML approaches
 - Supervised learning
 - Unsupervised learning
 - Reinforcement learning

ML approaches: supervised learning

- **Supervised learning**: task of learning a general rule that maps an input to an output based on example input-output pairs
 - It infers a function from *human-labelled training data* consisting of a set of *training examples*
 - Example: **image classification** defines a set of target classes (objects to identify in images, e.g., cats and dogs) and trains a model to recognize them using labelled example photos

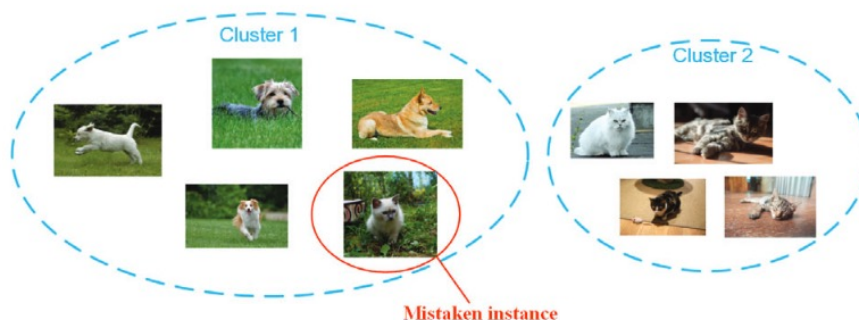


Valeria Cardellini - ADS 2024/25

66

ML approaches: unsupervised learning

- **Unsupervised learning**: no labelled training data are given to the ML algorithm, leaving it on its own to find structure in its input
 - The ML algorithm looks for previously undetected patterns in a data set with no pre-existing labels and with a minimum of human supervision
 - Example: **clustering**



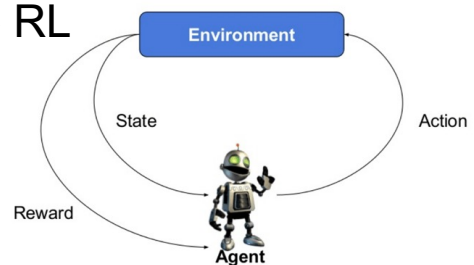
Valeria Cardellini - ADS 2024/25

67

ML approaches: reinforcement learning

- **Reinforcement learning**: an intelligent agent ought to take actions in an environment in order to maximize the notion of cumulative reward
 - Differs from supervised learning in not needing labelled input/output pairs
 - Focus is on finding a balance between *exploration* (of uncharted territory) and *exploitation* (of current knowledge)
 - Example: AlphaGo use deep RL

Typical RL scenario

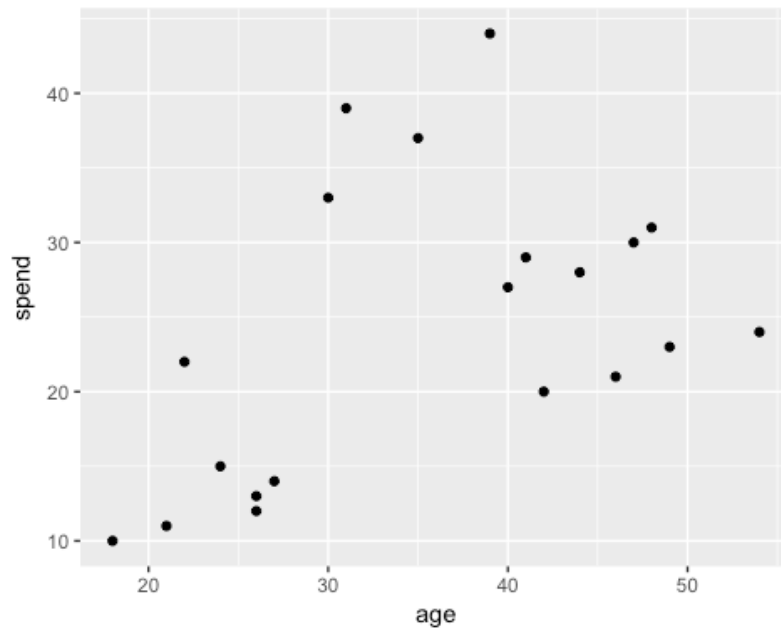


Examples of cluster analysis

- Customer segmentation: look for similarity between groups of customers
- Stock market clustering: group stocks based on performances
- Reduce dimensionality of a dataset by grouping observations with similar values

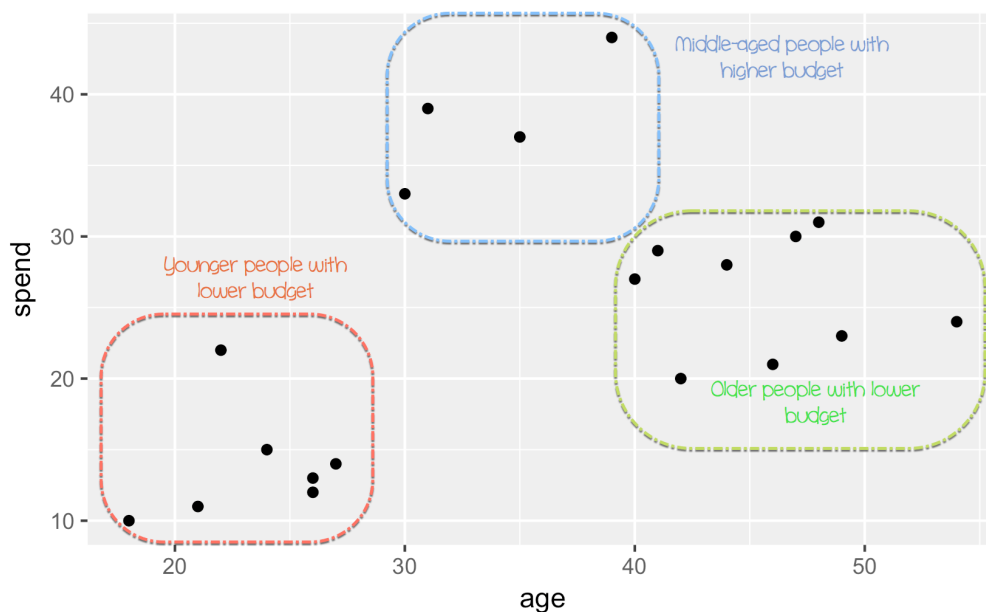
Example of cluster analysis

- Total spend and age of customers



Example of cluster analysis

- Total spend and age of customers



Example of cluster analysis

- Place pizza delivery on the map



Source: <https://www.youtube.com/watch?v=lpGxLWOIZy4>

Clustering algorithms

- There are many different algorithms for clustering
- Differ in what makes clusters and how to find them
- We will study:
 - **Hierarchical clustering**: belongs to **hierarchical or agglomerative** class of clustering algorithms
 - **K-means**: belongs to **point assignment** class of clustering algorithms

Distance between points

- We first need to define distance between two data points
- Most popular is **Euclidean distance**
 - Distance between points p and q is given by:

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2}$$

where n is the number of independent variables in the space

Distance between points: Example

- “Toy Story” movie is categorized as Animation, Comedy, and Children (see slide 63 for categories): we can encode this information
 - Toy Story: (0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0)
- “Batman Forever” movie is categorized as Action, Adventure, Comedy, and Crime
 - Batman Forever: (0,1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0)

$$\begin{aligned} d &= \sqrt{(0-0)^2 + (0-1)^2 + (0-1)^2 + (1-0)^2 + (1-0)^2 + (1-1)^2 + (0-1)^2 + (0-0)^2 + \dots + (0-0)^2} \\ &= \sqrt{0+1+1+1+1+0+1+0\dots+0} = \sqrt{5} \end{aligned}$$

Distance between points

- Other popular distance metrics
- **Manhattan distance**
 - Sum of absolute values instead of squares
 - Based on grid-like street geography of New York borough of Manhattan

$$d_{Manhattan}(p, q) = \sum_{i=1}^n |p_i - q_i|$$

- **Chebyshev distance**
 - Only consider measurement for which data points deviate the most

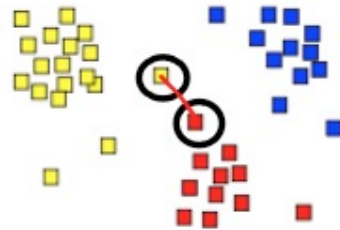
$$d_{Chebyshev}(p, q) = \max_i |p_i - q_i|$$

Distance between cluster

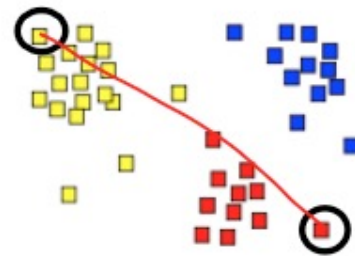
- How to define distance between two clusters?
- Which are the most representative data points for the cluster between which we can calculate the distance?

Distance between clusters

- Minimum distance
 - Distance between clusters is the distance between points that are the closest

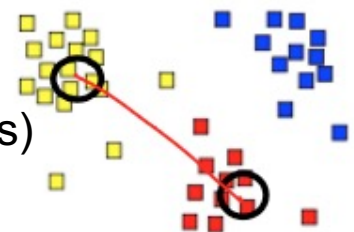


- Maximum distance
 - Distance between clusters is the distance between points that are the farthest



Distance between clusters

- Centroid distance
 - Distance between centroids (or centers) of clusters
- The cluster centroid is the point that has the mean position of all data points in each component
 - Example: for points $(-1, 10, 3)$, $(0, 5, 2)$, and $(1, 20, 10)$, the centroid is located at
$$\left(\frac{-1+0+1}{3}, \frac{10+5+20}{3}, \frac{3+2+10}{3}\right) = (0, 35/3, 5)$$
- Note: the centroid does not have to be - and rarely is - one of the original data points



Normalize data

- Distance is highly influenced by scale of data, so customary to **normalize** first
- In MovieLens dataset, genre data is on the same scale (being equal to 0 or 1) and normalization is not necessary
- However, if we included “Box office revenue”, we would need to normalize

Hierarchical clustering

- The algorithm works in a bottom-up manner
- Start with every point in its own cluster

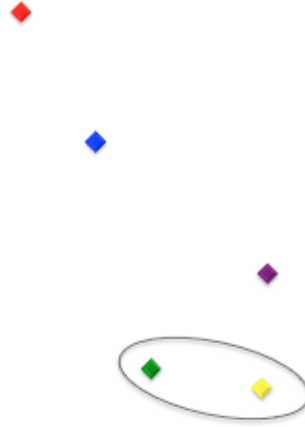


5 clusters

Hierarchical clustering

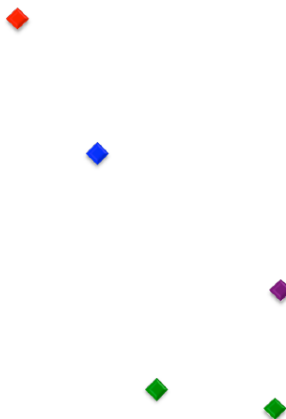
- Combine two nearest clusters (Euclidean distance, centroids)

At each step of the algorithm, the two clusters that are the most similar are merged into a new bigger cluster



Hierarchical clustering

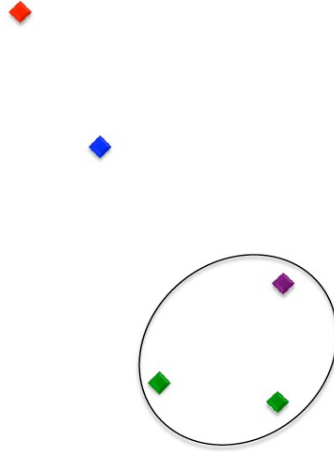
- Combine two nearest clusters (Euclidean distance, centroids)



4 clusters

Hierarchical clustering

- Combine two nearest clusters (Euclidean distance, centroids)



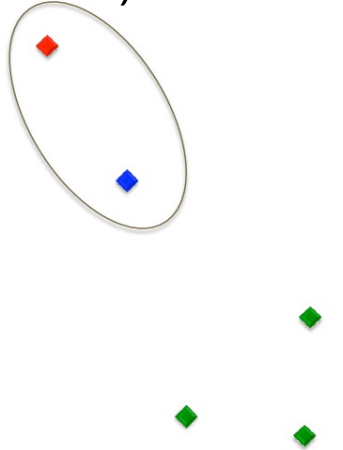
Hierarchical clustering

- Combine two nearest clusters (Euclidean distance, centroids)



Hierarchical clustering

- Combine two nearest clusters (Euclidean distance, centroids)



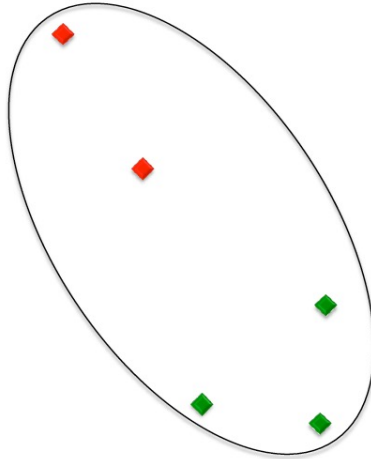
Hierarchical clustering

- Combine two nearest clusters (Euclidean distance, centroids)



Hierarchical clustering

- Combine two nearest clusters (Euclidean distance, centroids)



Hierarchical clustering

- Combine two nearest clusters (Euclidean, Centroid)

The procedure is iterated until all points are members of just one single big cluster



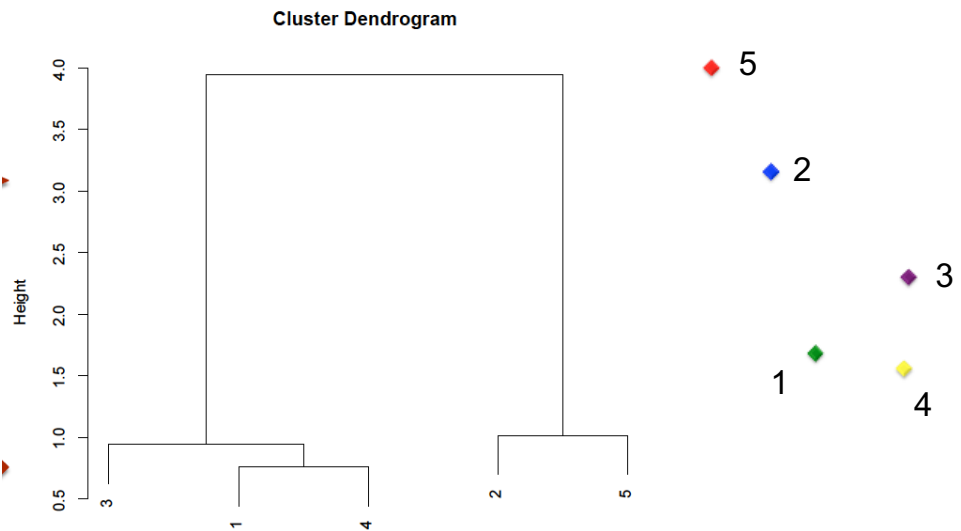
1 cluster

Hierarchical clustering: dendrogram

- The result of hierarchical clustering is a tree that can be plotted as a cluster **dendrogram**
 - Diagram showing hierarchical relationship between clusters

Height of **dendrogram**: order in which clusters were joined

Height of vertical lines: distance between points or clusters

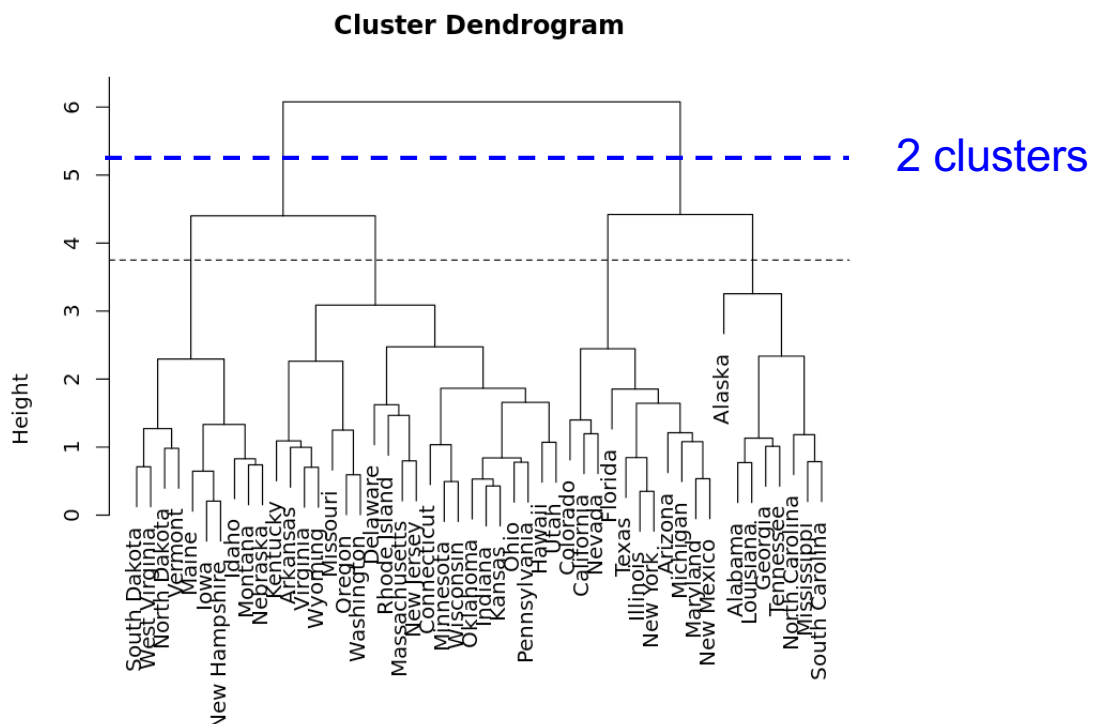


Valeria Cardellini - ADS 2024/25

90

Hierarchical clustering: select clusters

- Example: cluster analysis of US states according to number of arrests made

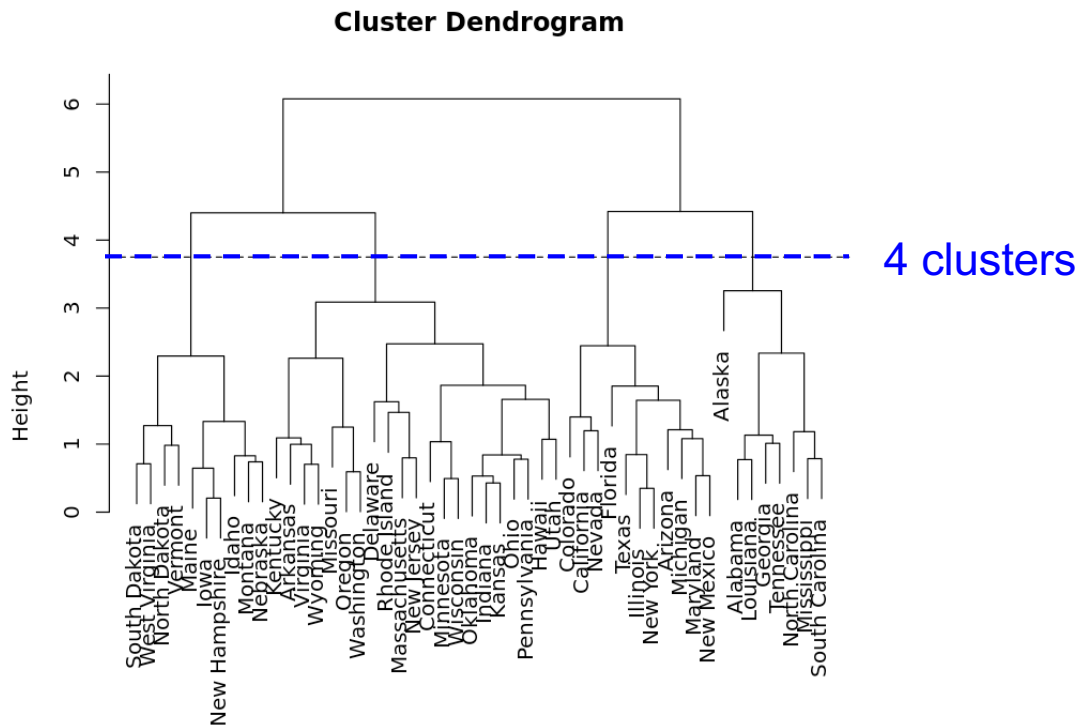


Valeria Cardellini - ADS 2024/25

91

Hierarchical clustering: select clusters

- Example: cluster analysis of US states according to the number of arrests made



Valeria Cardellini - ADS 2024/25

92

Hierarchical clustering: pros and cons

- ✓ Simple to understand and implement
- ✗ But once two data points are clustered together, they can never be separated
“Once the damage is done, it can never be repaired.” (Kaufman, 1990)

Valeria Cardellini - ADS 2024/25

93

k -means clustering

- k -means clustering at a glance:
 - Initialize by setting the desired number of clusters k
 - *Iterative* algorithm
 - Each iteration is composed by 2 steps:
 1. **Cluster assignment**
 2. **Centroid update**

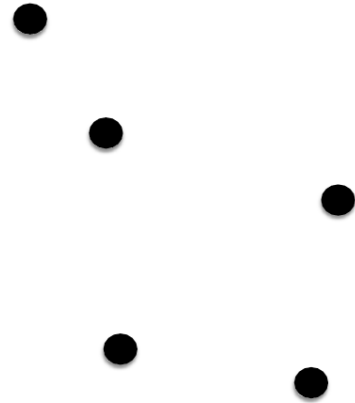
k -means clustering

- k -means clustering algorithm
 1. Specify desired number of clusters k
 2. Randomly assign each data point to a cluster
 3. Compute cluster centroids
 4. Re-assign each point to the closest cluster centroid (e.g., using Euclidian distance)
 5. Re-compute cluster centroids
 6. Repeat 4 and 5 until no improvement is made
- See animation https://en.wikipedia.org/wiki/File:K-means_convergence.gif

k -means clustering: Example

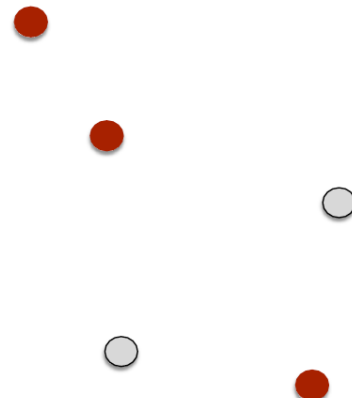
1. Specify desired number of clusters k

Let's start with $k=2$



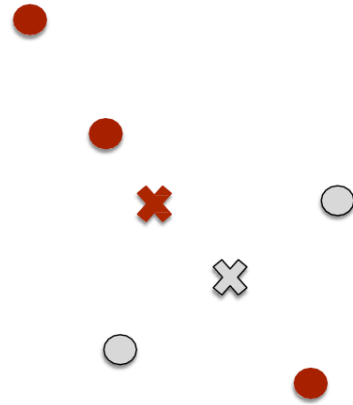
k -means clustering: Example

1. Specify desired number of clusters k
2. Randomly assign each data point to a cluster



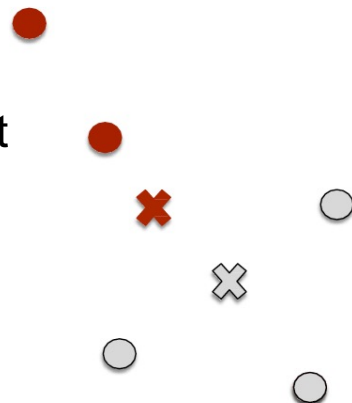
k -means clustering: Example

1. Specify desired number of clusters k
2. Randomly assign each data point to a cluster
3. Compute cluster centroids



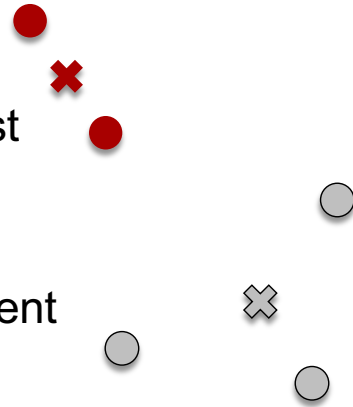
k -means clustering: Example

1. Specify desired number of clusters k
2. Randomly assign each data point to a cluster
3. Compute cluster centroids
4. Re-assign each point to the closest cluster centroid



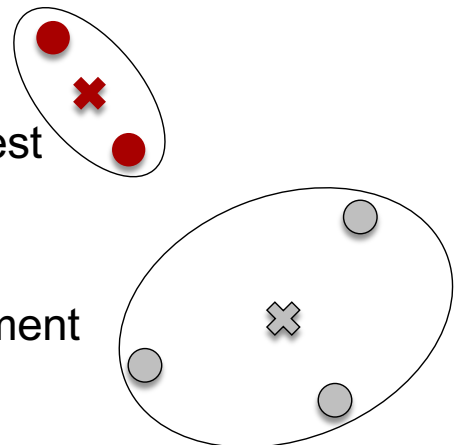
k -means clustering: Example

1. Specify desired number of clusters k
2. Randomly assign each data point to a cluster
3. Compute cluster centroids
4. Re-assign each point to the closest cluster centroid
5. Re-compute cluster centroids
6. Repeat 4 and 5 until no improvement is made



k -means clustering: Example

1. Specify desired number of clusters k
2. Randomly assign each data point to a cluster
3. Compute cluster centroids
4. Re-assign each point to the closest cluster centroid
5. Re-compute cluster centroids
6. Repeat 4 and 5 until no improvement is made



k -means clustering: when do we stop?

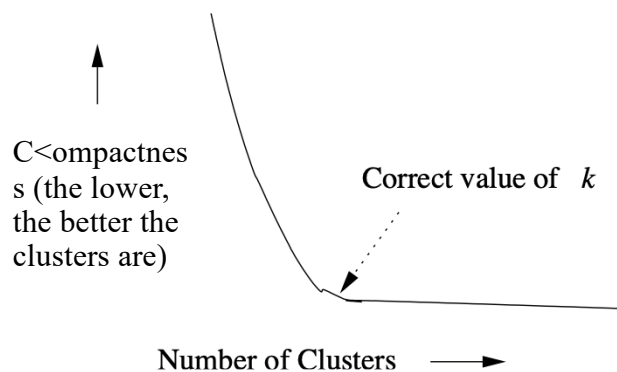
- Can set a maximum number of iterations
- Can check if centroids do not change (or change very little) between successive iterations
 - See slide 106, line 11
- In software, we typically terminate when one of the above criteria is met

k -means clustering: Observations

- 1) **How to pick the right value for k ?** Can exploit some knowledge on dataset or experiments with different values of k
 - Alternatives: elbow method, silhouette score
- 2) **How to find the solution faster?** Can strategically select initial partition of points into clusters if you have some knowledge on dataset
- 3) **How to improve the solution quality?** Can run k -means algorithm several times with different initial assignments

How to pick the right k ? Elbow method

- Idea: let's measure a score of clustering quality (e.g., *compactness*) for various values of k
- If the line chart resembles an arm, the “elbow” (inflection point on curve) is considered as an indicator of the appropriate number of clusters



k -means clustering: Alternative for initialization

- How to choose the initial k centroids?
 - Our solution so far: assign randomly each data point to a cluster and then compute the centroids
- Alternatively, we can initialize randomly the cluster centroids
- Then iterate on cluster assignment and centroid update
 - That is, repeat steps 4 and 5 on slide 95
- Let's now consider this alternative (aka *Lloyd's algorithm*)

k-means clustering (Lloyd): Pseudocode

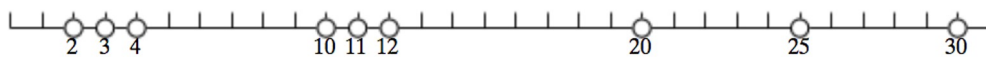
K-MEANS (\mathbf{D}, k, ϵ): \mathbf{D} is the dataset, k the number of clusters, ϵ the threshold to stop

- 1 $t = 0$
- 2 Randomly initialize k centroids: $\mu_1^t, \mu_2^t, \dots, \mu_k^t \in \mathbb{R}^d$
- 3 **repeat**
 - 4 $t \leftarrow t + 1$
 - 5 $C_i \leftarrow \emptyset$ for all $i = 1, \dots, k$
// Cluster Assignment Step
 - 6 **foreach** $\mathbf{x}_j \in \mathbf{D}$ **do**
 - 7 $i^* \leftarrow \arg \min_i \{ \|\mathbf{x}_j - \mu_i^{t-1}\|^2 \}$ Find the cluster centroid closest to point \mathbf{x}_j
 - 8 $C_{i^*} \leftarrow C_{i^*} \cup \{\mathbf{x}_j\}$ // Assign \mathbf{x}_j to closest centroid
 - 9 // Centroid Update Step
foreach $i = 1, \dots, k$ **do**
 - 10 $\mu_i^t \leftarrow \frac{1}{|C_i|} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j$ Assign point \mathbf{x}_j to the closest cluster C_{i^*}
Update the cluster centroids
 - 11 **until** $\sum_{i=1}^k \|\mu_i^t - \mu_i^{t-1}\|^2 \leq \epsilon$ Iterate over and over again until the centroids of last and second-to-last iterations are quite similar

Valeria Cardellini - ADS 2024/25

106

k-means clustering (Lloyd): Example

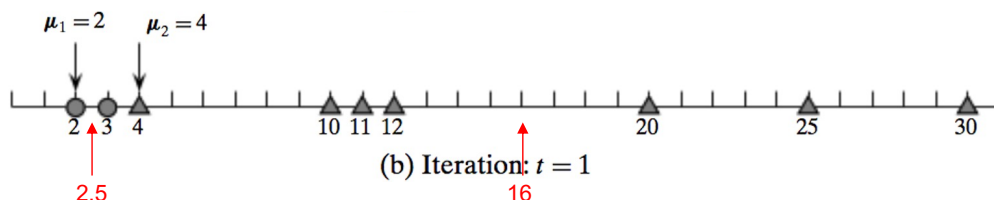


(a) Initial dataset

- Consider 1-dimension data set and set $k=2$ clusters
- Let's randomly initialize the centroids to $\mu_1^0 = 2$ and $\mu_2^0 = 4$
- 1st iteration: assign each point to the closest cluster; get $C_1 = \{2, 3\}$ and $C_2 = \{4, 10, 11, 12, 20, 25, 30\}$
- Recalculate the centroids:

$$\mu_1^1 = (2+3)/2 = 5/2 = 2.5$$

$$\mu_2^1 = (4+10+11+12+20+25+30)/7 = 112/7 = 16$$



(b) Iteration: $t = 1$

Valeria Cardellini - ADS 2024/25

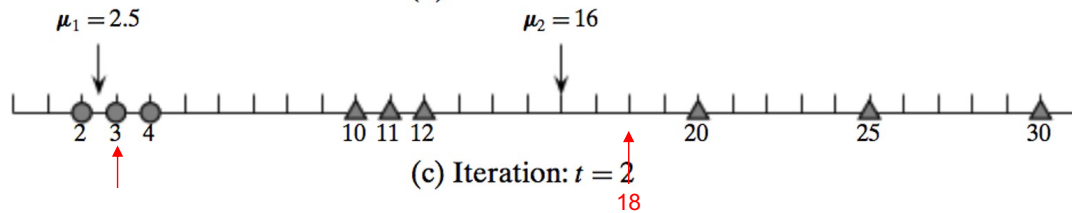
107

k-means clustering (Lloyd): Example

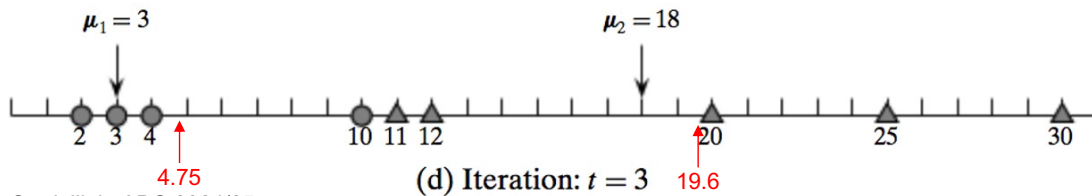
- 2nd iteration: repeat the cluster assignment to get $C_1 = \{2, 3, 4\}$ and $C_2 = \{10, 11, 12, 20, 25, 30\}$ and recalculate the new centroids

$$\mu_1^2 = (2+3+4)/3 = 9/3 = 3$$

$$\mu_2^2 = (10+11+12+20+25+30)/6 = 108/6 = 18$$

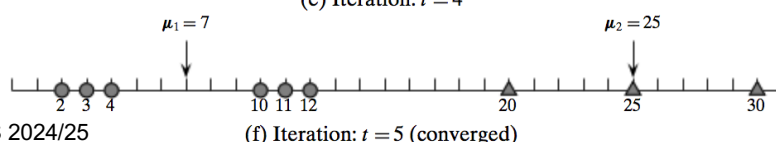
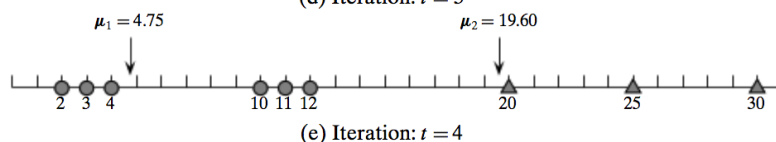
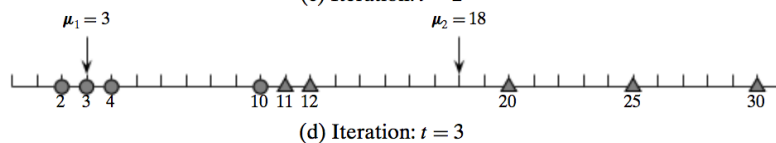
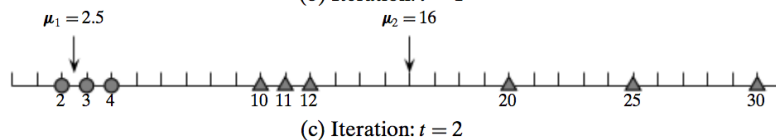
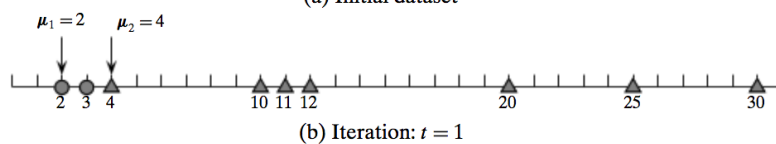
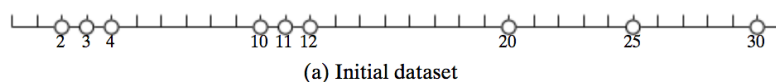


- 3rd iteration: obtain $C_1 = \{2, 3, 4, 10\}$ and $C_2 = \{11, 12, 20, 25, 30\}$ and the new centroids $\mu_1^3 = (2+3+4+10)/4 = 17/4 = 4.75$ and $\mu_2^3 = (11+12+20+25+30)/5 = 19.6$



k-means clustering (Lloyd): Example

- We continue until convergence: the final clusters are $C_1 = \{2, 3, 4, 10, 11, 12\}$ and $C_2 = \{20, 25, 30\}$



k-means in R

- How to use *k*-means in R programming?
- R uses by default an efficient algorithm by [Hartigan and Wong \(1979\)](#) that partitions the data points into *k* clusters such that the total within-cluster sum of squares is minimized
 - [Total within-cluster sum of squares \(WSS\)](#): sum of distance between each cluster data point and its cluster centroid summed over all clusters

$$WSS = \sum_{i=1}^k \sum_{j \in C_i} \sum_{l=1}^n (x_{jl} - \mu_{jl})^2$$

C_i : set of data points in cluster *i*

μ_{jl} : coordinate *l* of the centroid for cluster *j*

k-means in R

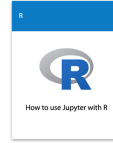
- How to use *k*-means in R?
 - Type `help(kmeans)`
- ```
kmeans(x, centers, iter.max = 10, nstart = 1,
algorithm = c("Hartigan-Wong", "Lloyd", "Forgy",
"MacQueen"), trace=FALSE)
```
- `x`: numeric data
  - `centers`: number of clusters (*k*)
  - `iter.max`: maximum number of iterations allowed
  - `nstart`: how many random starts (hint: choose a high value, e.g., 20)
  - `algorithm`: default is Hartigan-Wong

## *k*-means in R: Example

---

- Let's use Jupiter Notebook in web browser

<https://jupyter.org/try> and click on



- Let's consider a simple dataset with two well-separated clusters
  - Let's generate the data points using normal distribution

## *k*-means in R: Example ( $k=2$ )

---

#Generate a sample data set with two well-separated clusters

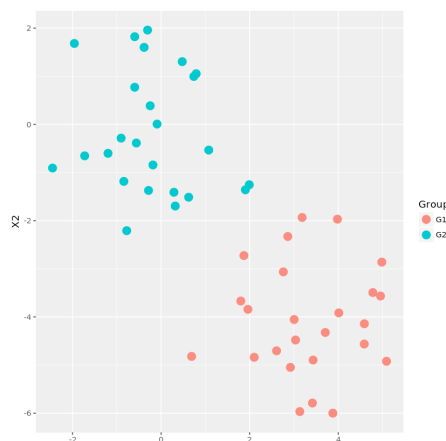
```
set.seed(2)
```

```
f.data=data.frame("Group"=c(rep("G1",25),rep("G2",25)),
"X1"=c(rnorm(25)+3,rnorm(25)), "X2"=c(rnorm(25)-4,rnorm(25)))
```

#Plot data set

```
library(ggplot2)
```

```
gg1=ggplot(f.data,aes(x=X1,y=X2,color=Group))+geom_point(size=4)
gg1
```



## *k*-means in R: Example (*k*=2)

---

```
#Run k-means with k=2
km.out=kmeans(f.data[,2:3],centers=2,nstart=20)
km.out

#Print the two resulting clusters
#Vector indicating the cluster to which each point is assigned
km.out$cluster

#Print the centroids
km.out$centers[
```

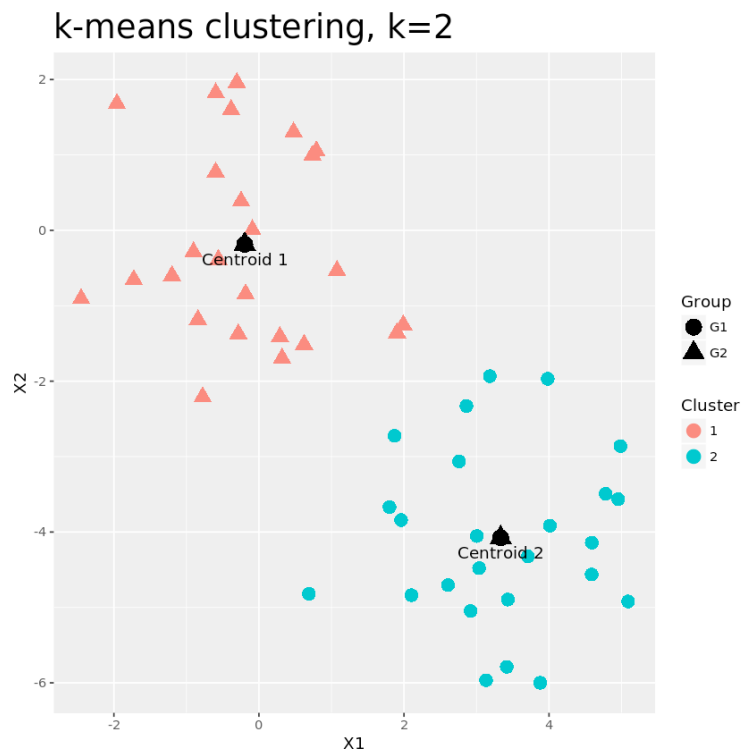
## *k*-means in R: Example (*k*=2)

---

```
#Prepare the output for plotting
f.data$Cluster=as.factor(km.out$cluster)
#Plot the clustering result
gg2=ggplot(f.data,aes(x=X1,y=X2,color=Cluster,shape=Group,))+g
geom_point(size=4)
gg2+geom_point(aes(x=km.out$center[1,1],y=km.out$center[1,2]),s
ize=5,color="black")+
geom_point(aes(x=km.out$center[2,1],y=km.out$center[2,2]),size=
5,color="black")+
annotate("text", x=km.out$center[1,1],y=km.out$center[1,2]-0.2,
label = "Centroid 1")+
annotate("text", x=km.out$center[2,1],y=km.out$center[2,2]-0.2,
label = "Centroid 2")+
ggtitle("k-means clustering, k=2")+ theme(plot.title =
element_text(size = rel(2)))
```

## *k*-means in R: Example (*k*=2)

---



## *k*-means in R: Example (*k*=3)

---

```
#Run k-means algorithm with k=3
km.out.3=kmeans(f.data[,2:3],centers=3,nstart=20)
km.out.3
```

```
#Prepare the output for plotting
f.data$Cluster.k3=as.factor(km.out.3$cluster)
```

## k-means in R: Example ( $k=3$ )

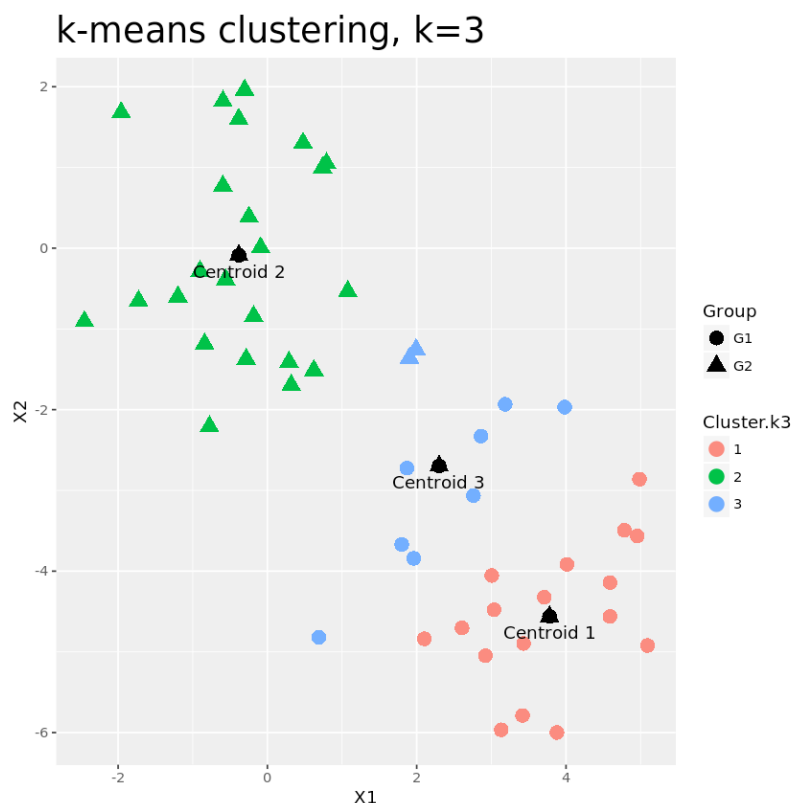
#Plot the clustering result

```
gg2=ggplot(f.data,aes(x=X1,y=X2,color=Cluster.k3,shape=Group,))
+geom_point(size=4)
gg2+geom_point(aes(x=km.out.3$center[1,1],y=km.out.3$center[1,
2]),size=4,color="black")+
geom_point(aes(x=km.out.3$center[2,1],y=km.out.3$center[2,2]),si
ze=4,color="black")+
geom_point(aes(x=km.out.3$center[3,1],y=km.out.3$center[3,2]),si
ze=4,color="black")+
annotate("text", x=km.out.3$center[1,1],y=km.out.3$center[1,2]-0.2,
label = "Centroid 1")+
annotate("text", x=km.out.3$center[2,1],y=km.out.3$center[2,2]-0.2,
label = "Centroid 2")+
annotate("text", x=km.out.3$center[3,1],y=km.out.3$center[3,2]-0.2,
label = "Centroid 3")+
ggtitle("k-means clustering, k=3")+ theme(plot.title =
element_text(size = rel(2)))
```

Valeria Cardellini - ADS 2024/25

118

## k-means in R: Example ( $k=3$ )



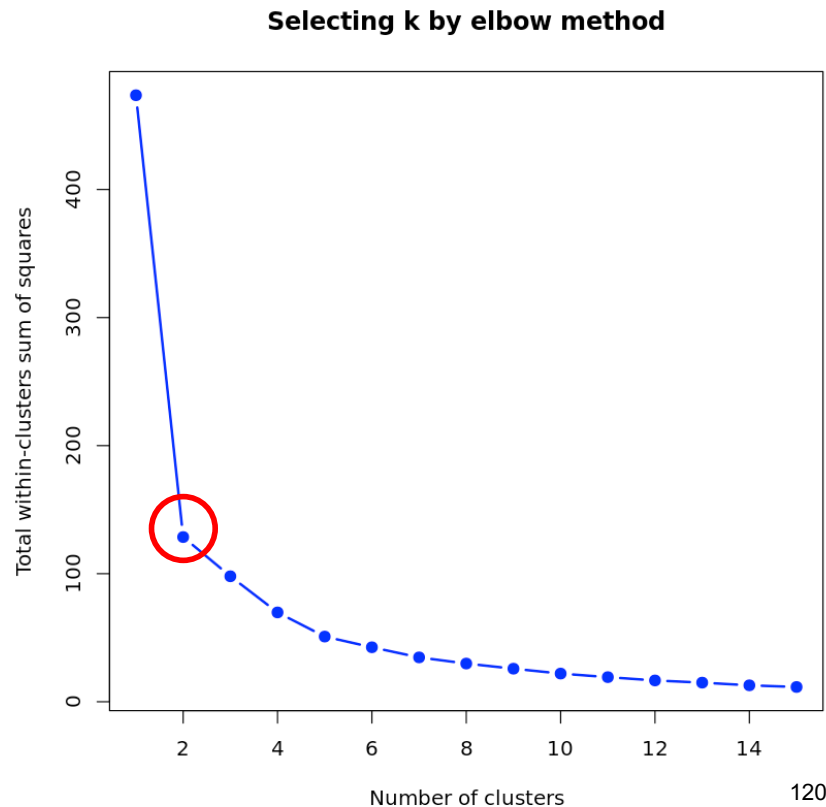
Valeria Cardellini - ADS 2024/25

119



## Elbow method: Example

- Let's consider our example and plot total within-cluster sum of squares (slide 110)
- WSS measures the **cluster compactness** and we want it to be as small as possible (but not 0!)
  - As  $k$  increases, WSS tends to 0



Valeria Cardellini - ADS 2024/25

## Elbow method in R: Example

- Let's consider our example in R and plot the total within-cluster sum of squares

```
#Generate a sample data set with two well-separated clusters
set.seed(2)
f.data=data.frame("Group"=c(rep("G1",25),rep("G2",25)),
 "X1"=c(rnorm(25)+3,rnorm(25)), "X2"=c(rnorm(25)-
 4,rnorm(25)))
```

# Elbow method: Example

---

```
#Execute k-means from k=1 to k=15
set.seed(123)
k.max = 15
wss = numeric(k.max)
for (i in 1:k.max) {
 KM = kmeans(f.data[,2:3], centers=i, nstart=20, iter.max = 15)
 wss[i] = KM$tot.withinss
}

#Plot total within-cluster sum of squares vs. number of clusters
plot(wss~seq(1:k.max),
 type="b", pch = 19, col=4,
 xlab="Number of clusters",
 ylab="Total within-clusters sum of squares", lwd=2,
 main="Selecting k by Elbow method")
```

Valeria Cardellini - ADS 2024/25

122

## References

---

- J. Leskovec, A. Rajaraman and J. Ullman, Recommendation Systems, ch. 9 of “Mining of Massive Datasets” book <http://infolab.stanford.edu/~ullman/mmds/ch9.pdf>
- J. Leskovec, A. Rajaraman and J. Ullman, Clustering, ch. 7 of “Mining of Massive Datasets” book <http://infolab.stanford.edu/~ullman/mmds/ch7.pdf>
- Some videos:
  - Overview of Recommender Systems, Stanford University <https://www.youtube.com/watch?v=1JRrCEgiyHM>
  - L. Serrano, Clustering: K-means and Hierarchical <https://www.youtube.com/watch?v=QXOkPvFM6NU>
  - V. Lavrenko, Agglomerative Clustering: How It Works <https://www.youtube.com/watch?v=XJ3194AmH40>
  - V. Lavrenko, K-means Clustering: How It Works [https://www.youtube.com/watch?v=\\_aWzGGNrcic](https://www.youtube.com/watch?v=_aWzGGNrcic)

Valeria Cardellini - ADS 2024/25

123