Advanced Business Data Mining

MSIS 522 – Lesson 5

W | Foster
School of Business

Course Evaluation

https://uw.iasystem.org/survey/217316

Foster School of Business

Course Overview

- Lecture 1 - Fundamentals of Machine Learning
- Lecture 2 - Decision Tree
- Lecture 3 - Ensemble Learning
- Lecture 4 - Clustering
- **Lecture 5 - Recommendation Systems**

Outline

- Recap of Clustering

- Recommendation System
  - Problem statement
  - Approaches
    - Content-based model
    - Collaborative filtering
    - Latent factor model
  - Evaluation

- Lab

Recap of Clustering

Clustering

Goal: Given a set of data points, group them into clusters, so that:

- Points within each cluster are similar to each other
- Points from different clusters are dissimilar

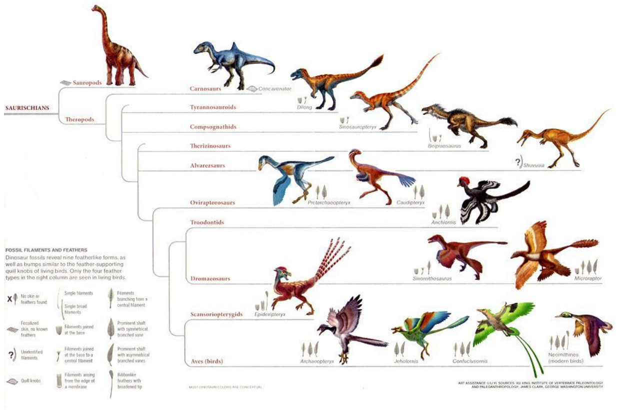
- Distance/similarity measures
  - Measures to determine how similar or different are instances to one another.
- Clustering algorithm
  - Algorithm to find the clusters based on the distance/similarity measures.
- Evaluation Criteria
  - Metrics to tell if one form of clustering is better than another form.

Distance/Similarity Measures

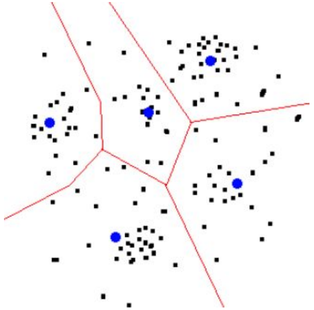
- Manhattan distance (for numerical data)
- Euclidean (for numerical data)
- Cosine similarity (for text data)
- Hamming Distance (nominal value)
- Jaccard Distance (set)

Clustering Algorithms

- **Hierarchical algorithms** build a tree-based hierarchical taxonomy.
  - Bottom up
  - Top down



- **Partition (Flat) algorithms** produce a single partition of the unlabeled data.
  - K-means
  - Mixture of Gaussian

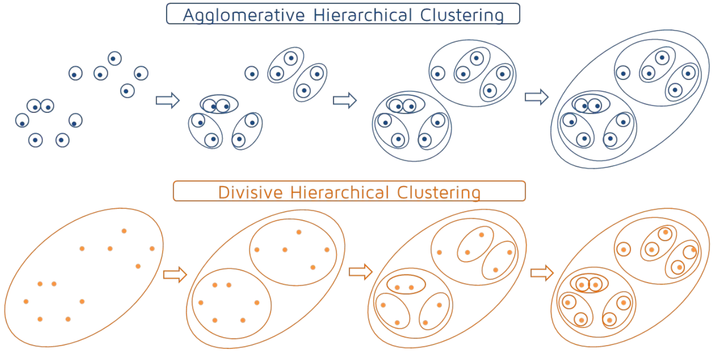


W Foster School of Business

Hierarchical Clustering

Goal: Build a hierarchy of clusters

- **Agglomerative Clustering** (Bottom up approach) :
  - Each instance is its own cluster
  - Pairs of clusters are merged as one moves up the hierarchy until there is only one cluster



- **Divisive Clustering** (Top Down approach) :
  - All observations start as one cluster
  - Recursively split the data as one moves down the hierarchy until each instance is a cluster

Hierarchical Agglomerative Clustering Algorithm

Basic agglomerative hierarchical clustering algorithm.	
1:	Compute the proximity matrix, if necessary.
2:	**repeat**
3:	Merge the closest two clusters.
4:	Update the proximity matrix to reflect the proximity between the new cluster and the original clusters.
5:	**until** Only one cluster remains.

Problem: How to measure the distance between two clusters?

Distance Between Clusters

- **Single Link:** the shortest distance between two points, *x* and *y*, that are in different clusters, *A* and *B*:

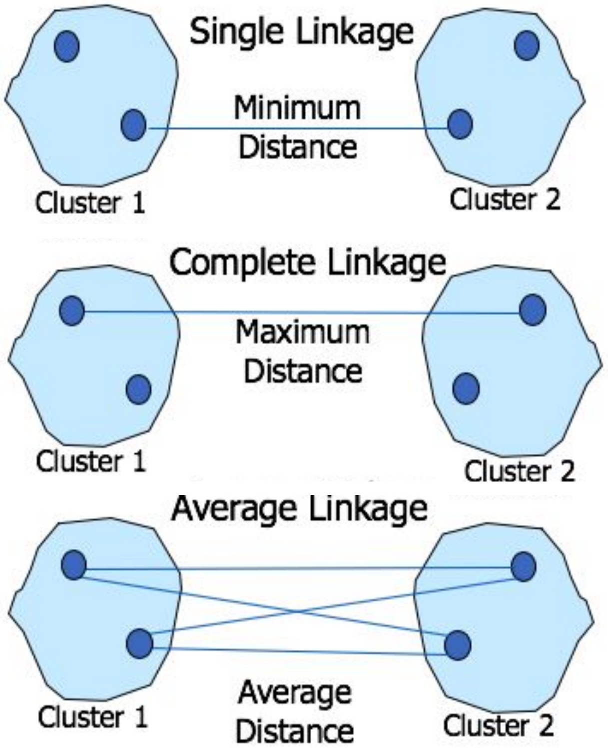
$$d(A,B) = \min_{x \in A, y \in B} d(x-y)$$

- **Complete Link:** the furthest distance between two points, *x* and *y*, that are in different clusters, *A* and *B*:

$$d(A,B) = \max_{x \in A, y \in B} d(x-y)$$

- **Average Link:** the average distance between two points, *x* and *y*, that are in different clusters, *A* and *B*:

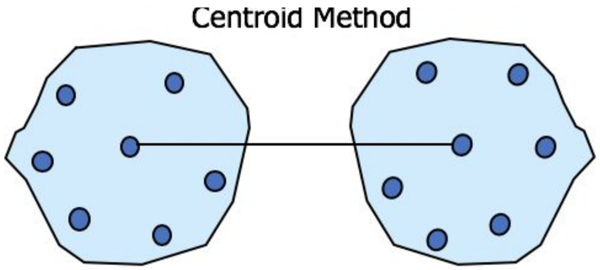
$$d(A,B) = \sum_{x \in A, y \in B} d(x-y)/n_A n_B$$



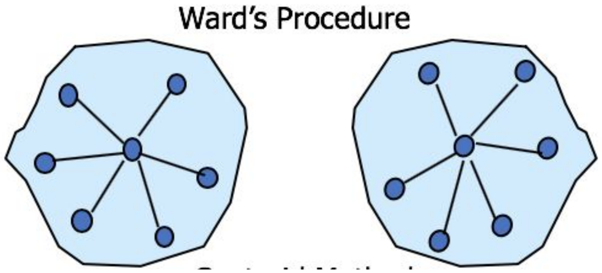
Foster School of Business

Distance Between Clusters

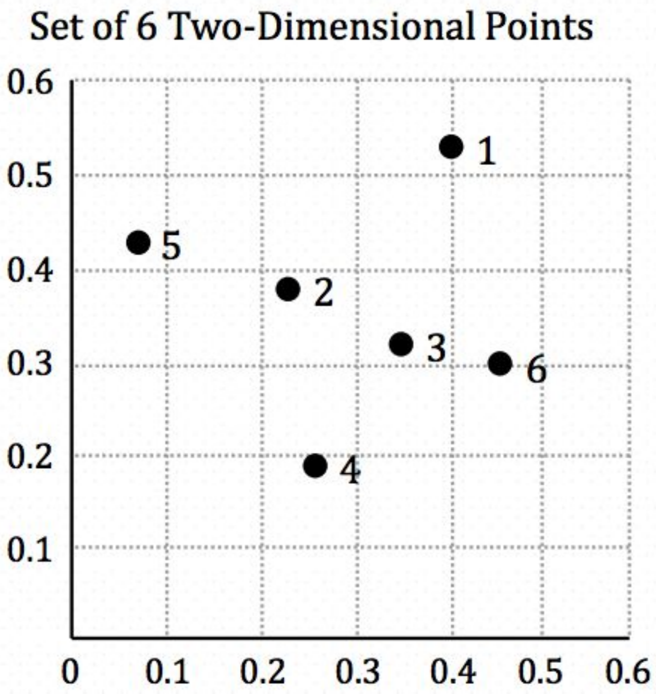
- **Centroids Method:** The distance between two clusters is the distance between their centroids.



- **Ward's Procedure:** For each cluster calculate the sum of squares. The two clusters with the smallest increase in the overall sum of squares within cluster distances are combined.



Hierarchical Clustering Example



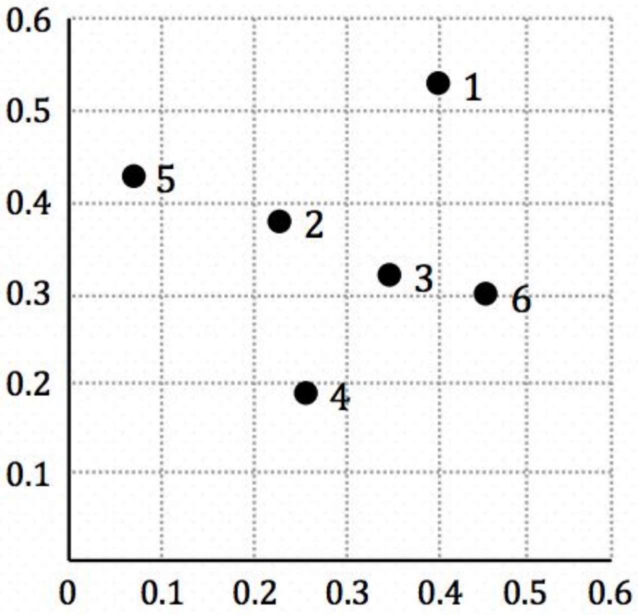
xy Coordinates of 6 Points

Point	*x* Coordinate	*y* Coordinate
p1	0.40	0.53
p2	0.22	0.38
p3	0.35	0.32
p4	0.26	0.19
p5	0.08	0.41
p6	0.45	0.30

W Foster School of Business

Hierarchical Clustering Example

Set of 6 Two-Dimensional Points



Euclidean Distance Matrix for 6 Points

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Hierarchical Clustering Example

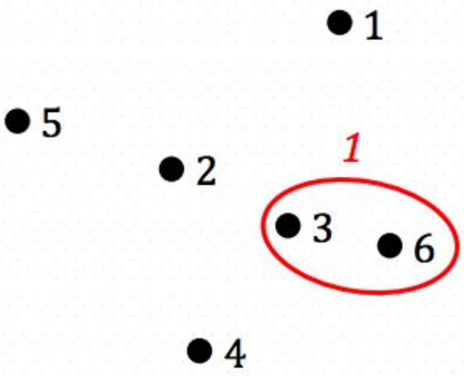
Nested Cluster Diagram

Single Link Distance Matrix

	1	2	3	4	5	6
1	0	0.24	0.22	0.37	0.34	0.23
2		0	0.15	0.20	0.14	0.25
3			0	0.15	0.28	0.11
4				0	0.29	0.22
5					0	0.39
6						0

Hierarchical Clustering Example

Nested Cluster Diagram

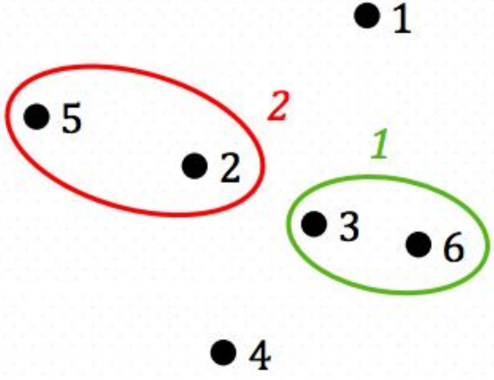


Single Link Distance Matrix

	1	2	3	4	5	6
1	0	0.24	*0.22*	0.37	0.34	0.23
2		0	*0.15*	0.20	0.14	0.25
3			0	*0.15*	*0.28*	0.11
4				0	0.29	0.22
5					0	0.39
6						0

Hierarchical Clustering Example

Nested Cluster Diagram

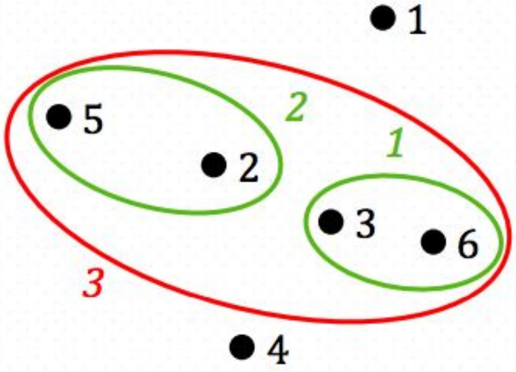


Single Link Distance Matrix

	1	2	4	5	3,6
1	0	*0.24*	0.37	0.34	0.22
2		0	*0.20*	0.14	0.15
4			0	0.29	0.15
5				0	0.28
3,6					0

Hierarchical Clustering Example

Nested Cluster Diagram

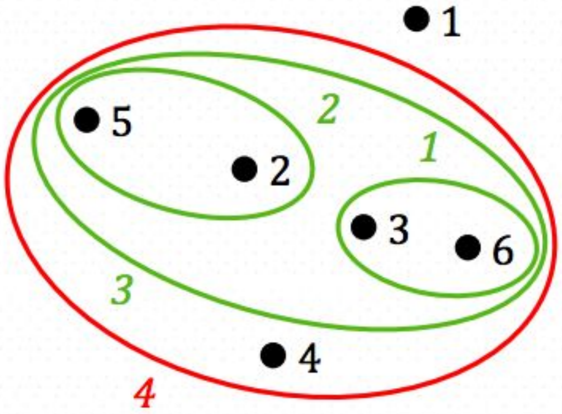


Single Link Distance Matrix

	1	4	2,5	3,6
1	0	0.37	0.24	*0.22*
4		0	0.20	*0.15*
2,5			0	0.15
3,6				0

Hierarchical Clustering Example

Nested Cluster Diagram

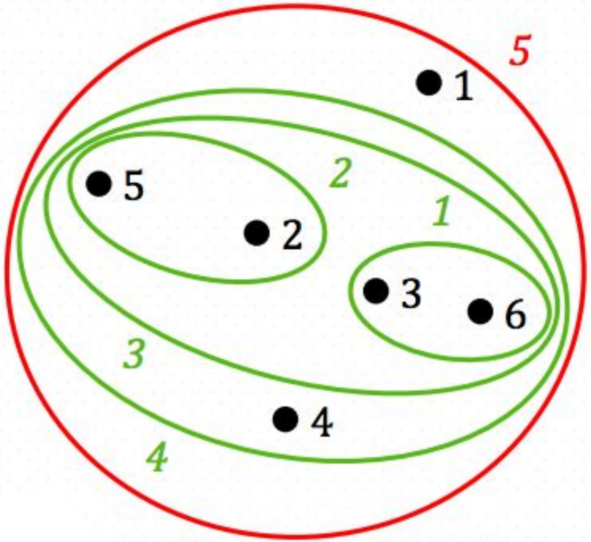


Single Link Distance Matrix

	1	4	2,5,3,6
1	0	0.37	*0.22*
4		0	0.15
2,5,3,6			0

Hierarchical Clustering Example

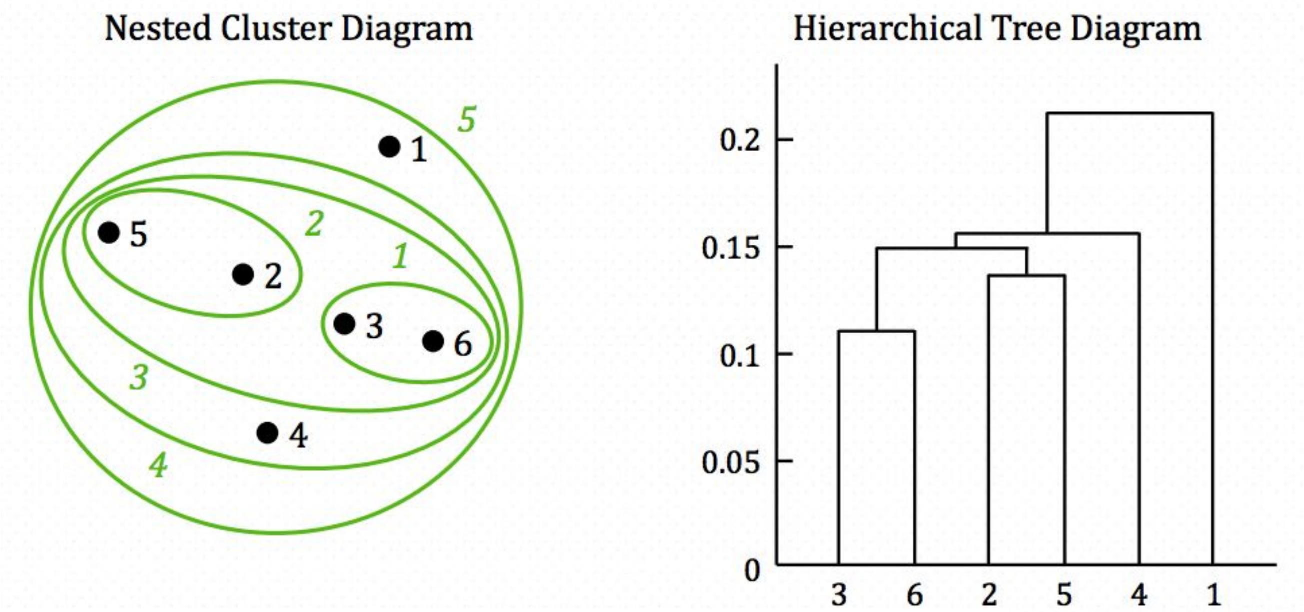
Nested Cluster Diagram



Single Link Distance Matrix

	1	4,2,5,3,6
1	0	0.22
2,5,3,6		0

Hierarchical Clustering Example



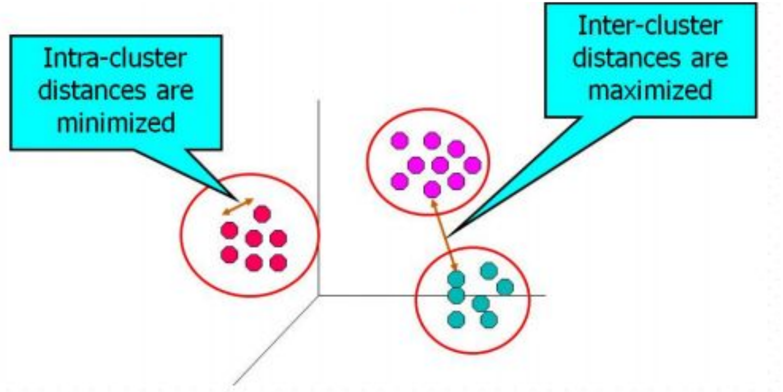
- Dendrogram can be used to identify
  - The number of clusters in data
  - Well-formed clusters
  - Outliers

Summary of Hierarchical Clustering

- Useful if the underlying application has a taxonomy.
- Ward’s procedure shows better results empirically.
- Agglomerative hierarchical clustering algorithms are expensive in terms of their computational and storage requirements.

Flat Clustering

- Given a data set $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n\}$. Partition D into k disjoint clusters, such that
  - Intra-cluster distances are minimized
  - Inter-cluster distances are maximized



Foster School of Business

The K-means algorithm

Input: $D = \{x_1\ x_2\ \dots x_n\}$ and desired number of clusters k

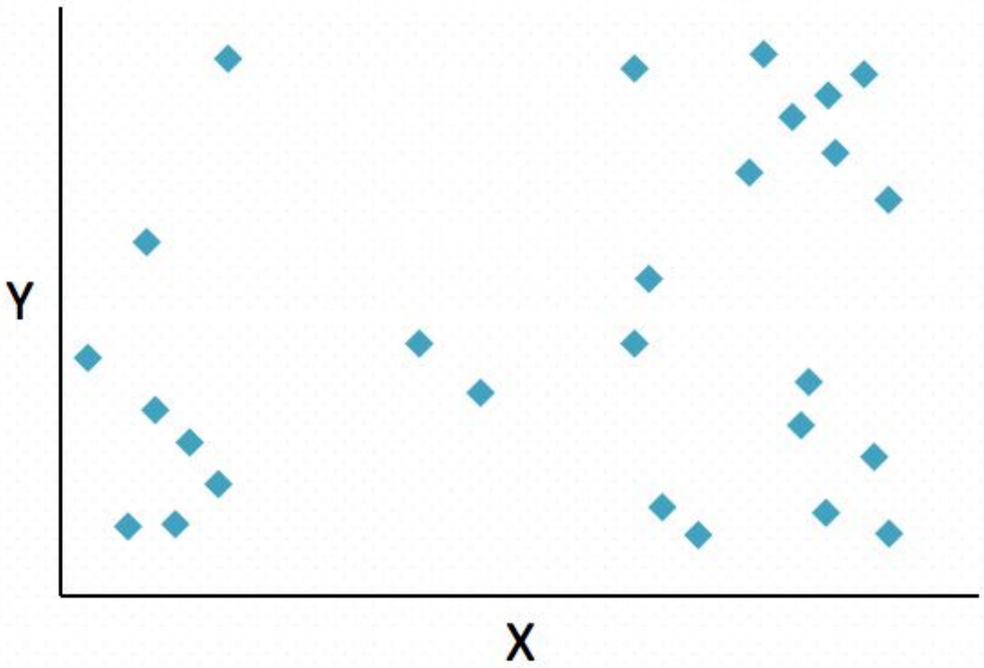
Output: a partition of D into k disjoint clusters $c_1\ \dots\ c_k$ (s.t. D = $c_1 \cup c_2 \cup \cdots \cup c_k$)

Let $\boldsymbol{d}$ be the distance function between examples

1. Select k random samples from D as centers $\{\mu_1\ \dots \mu_k\}$ **//Initialization**
2. Do
3. for each example $\boldsymbol{x}_i$,
4. assign $\boldsymbol{x}_i$ to $\boldsymbol{c}_j$ such that $\boldsymbol{d}(\boldsymbol{\mu}_j, \boldsymbol{x}_1)$ is minimized **// the Assignment step**
5. for each cluster j, update cluster center
6. $\mu_j = \frac{1}{|c_j|}\sum_{\mathbf{x}\in c_j}\mathbf{x}$ **// the update step**
7. Until convergence

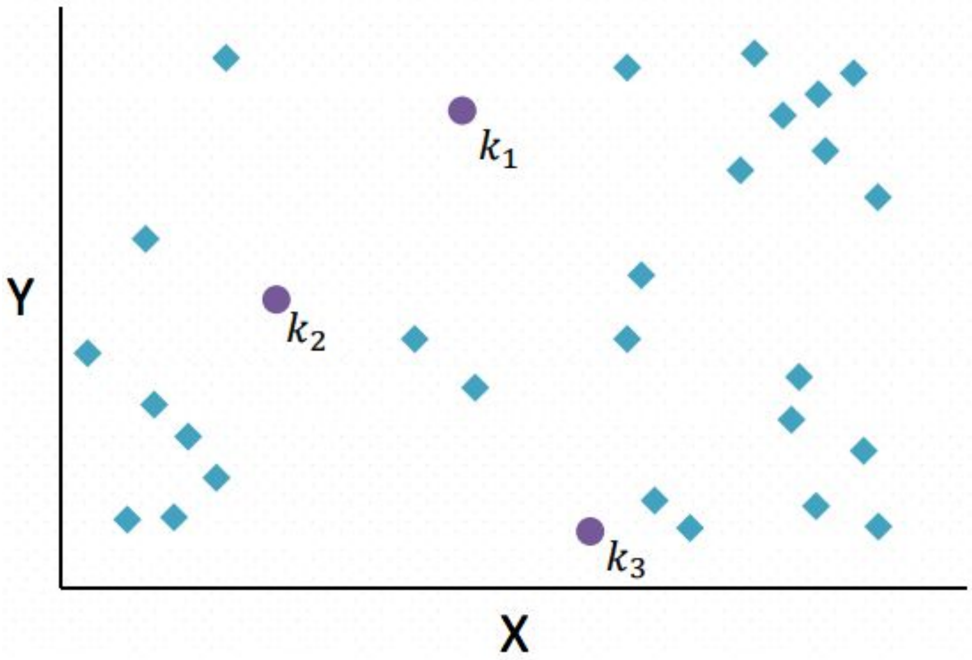
K-means Clustering Example

Suppose we want to cluster these data points.



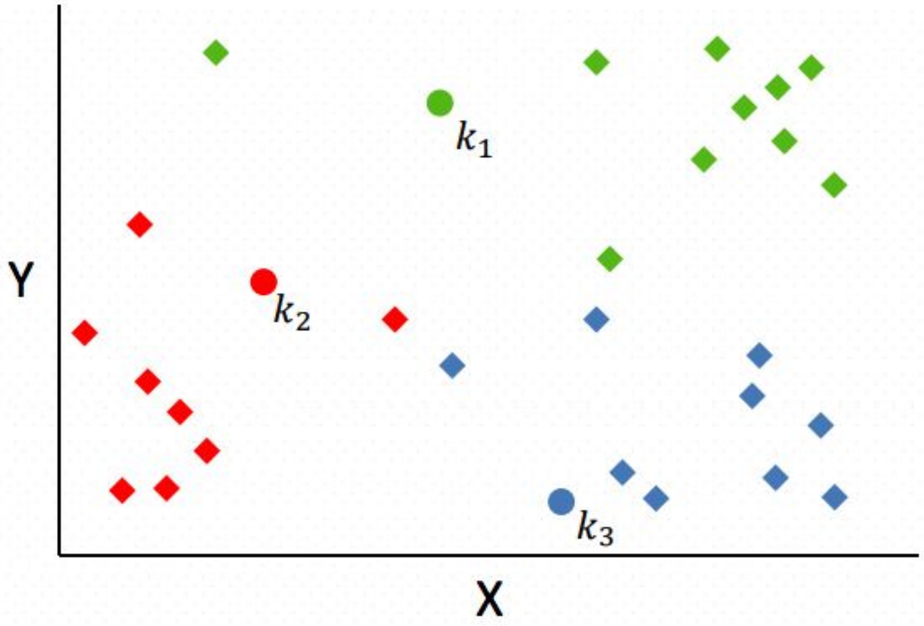
K-means Clustering Example

Pick 3 initial cluster centers at random



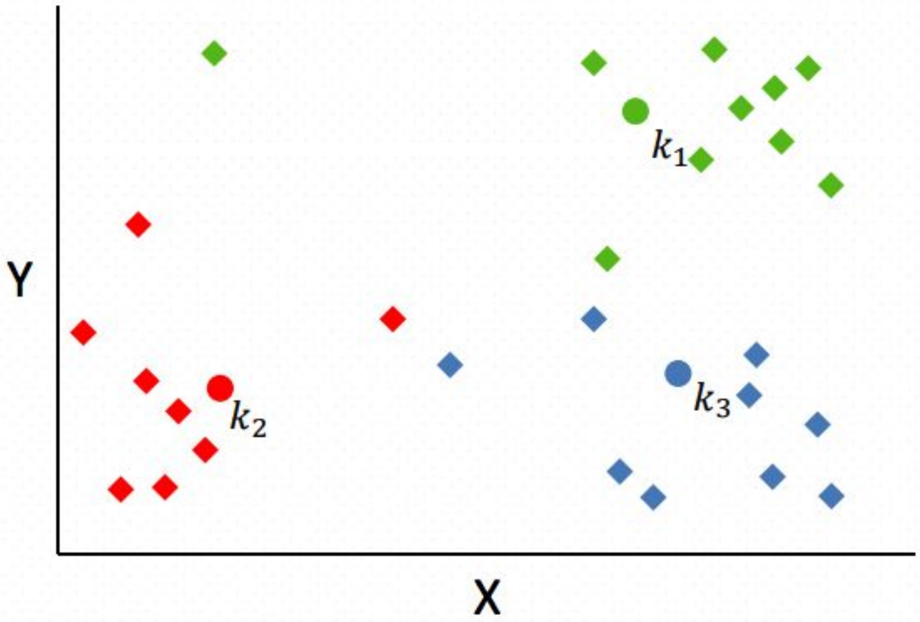
K-means Clustering Example

Assign each data point to the closest cluster center



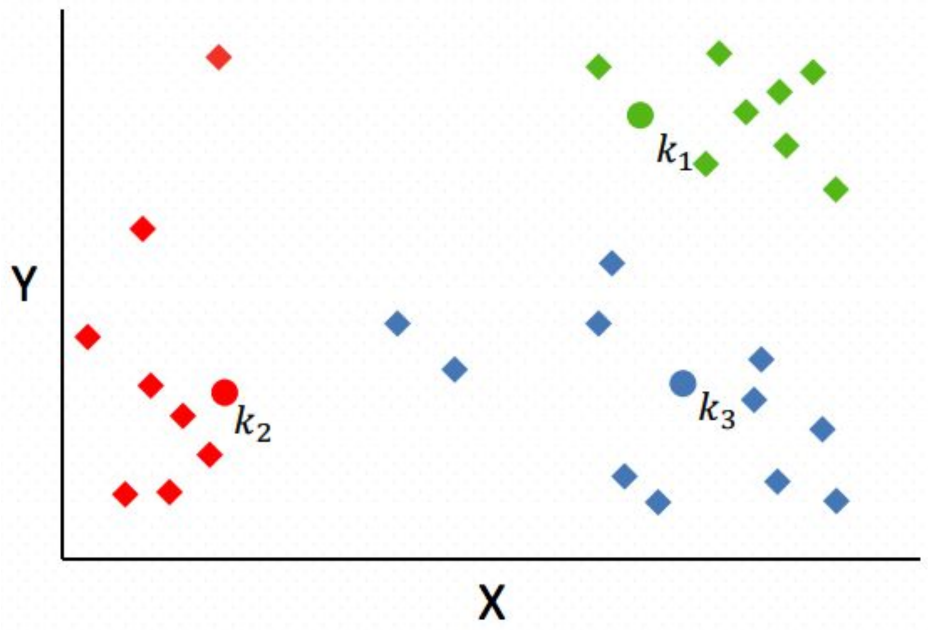
K-means Clustering Example

Move each cluster center to the mean of each cluster



K-means Clustering Example

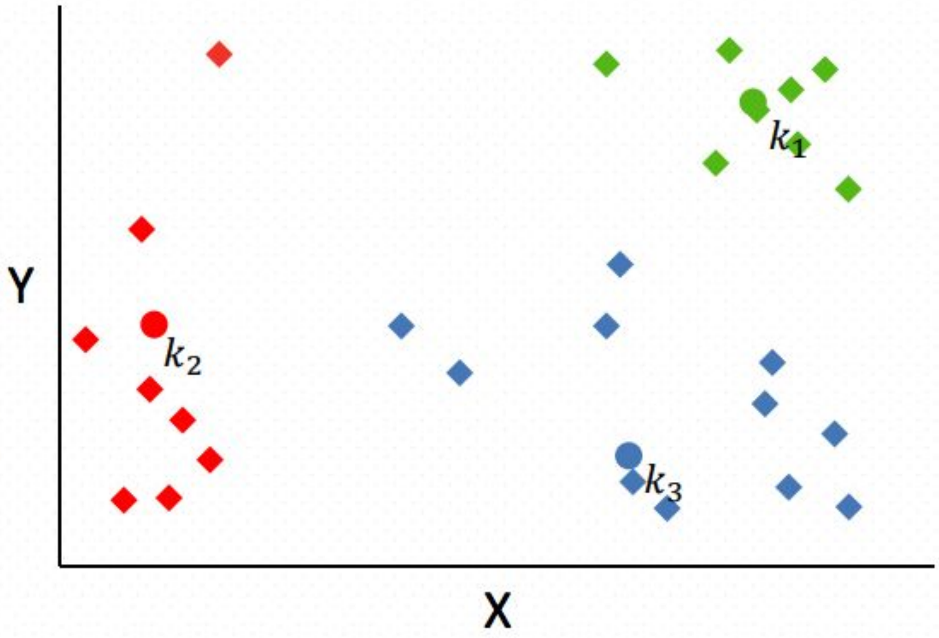
Assign each data point to the closest cluster center



K-means Clustering Example

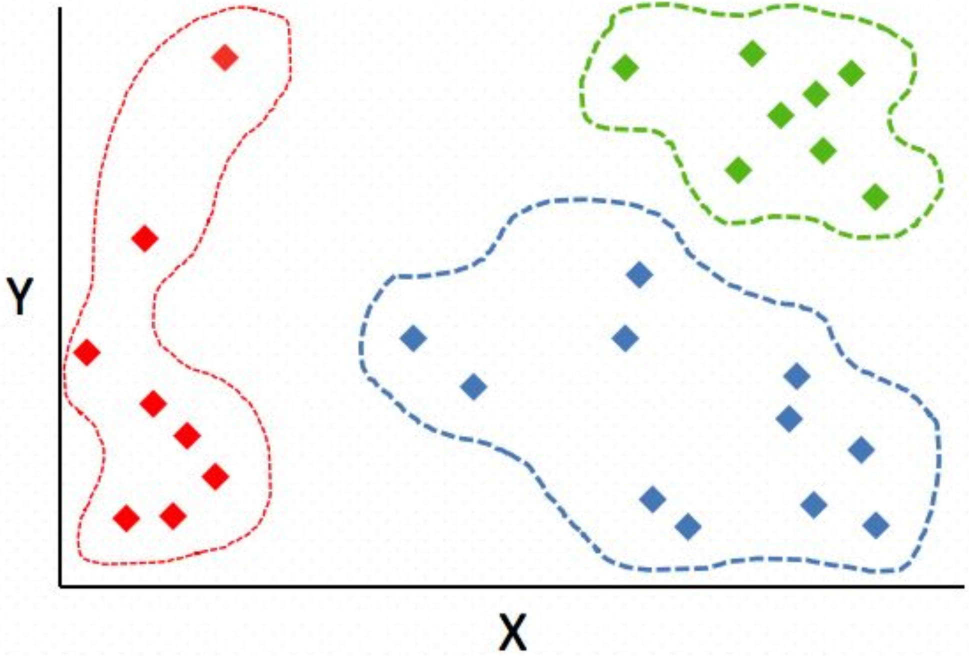
Reassign labels.

No change. DONE!



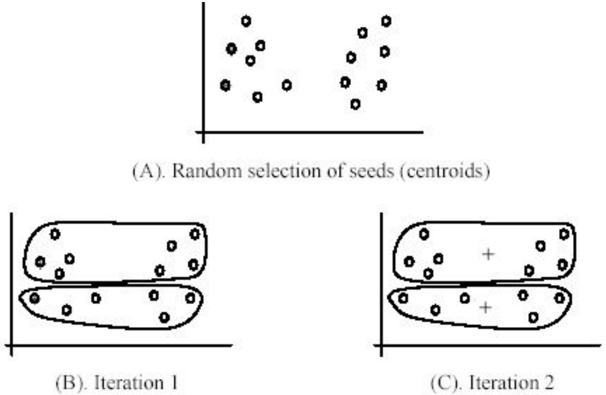
K-means Clustering Example

Final clusters



Weaknesses of K-Means: Local Minimal

- K-means is very sensitive to initial conditions.
- Different initialization can lead to very different clusters.



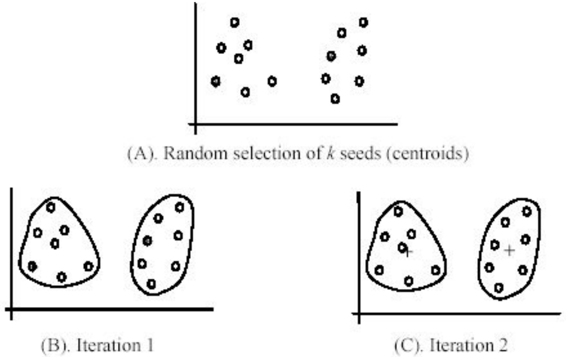
(A). Random selection of seeds (centroids)

(B). Iteration 1

(C). Iteration 2

Solutions

- Run multiple trials and choose the one with the best SSE.
- Heuristics. Try to choose initial centers to be far apart (e.g. K-means++).



(A). Random selection of k seeds (centroids)

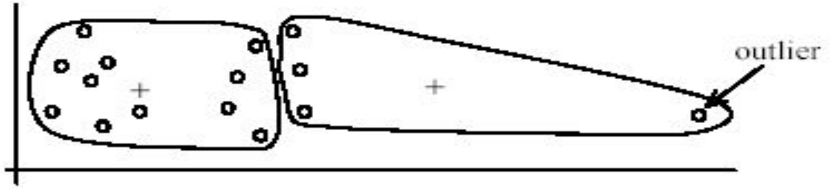
(B). Iteration 1

(C). Iteration 2

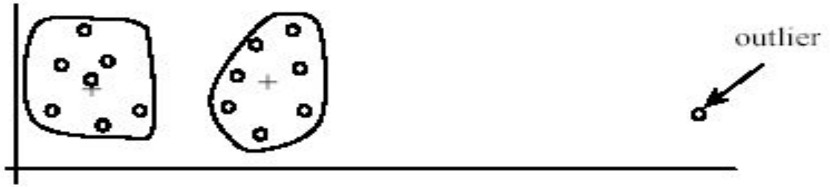
K-means++: http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf

Weaknesses of K-Means: Outliers

- K-means is very sensitive to outliers.
- The presence of outliers can lead to very *unreasonable* clusters.



(A): Undesirable clusters



(B): Ideal clusters

Solution

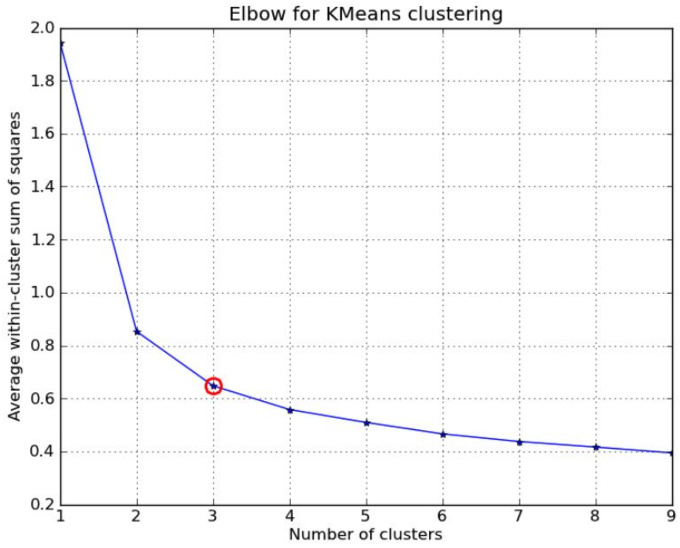
- Remove outliers before running K-means.
- K-medoids

$$\cancel{\mu = \frac{1}{|C|}\sum_{\mathbf{x}\in C} \mathbf{x}}$$

$$\mu = \underset{x\in C}{\operatorname{argmin}} \sum_{z\in C} \|x - z\|^2$$

Selection of K

- The choice of k is often dependent on scale and distribution of your dataset.
- Rule of thumb:
  - $k \approx \sqrt{n/2}$, where n is the number of data points.
  - A good starting point, but not very reliable.
- The Elbow Method:
  - Choose a number of clusters that covers most of the variance.



Summary of K-means

- **Pros**
  - very efficient (even if multiple runs are performed), can be used for a large variety of data types.

- **Cons**
  - Not suitable for all types of data, susceptible to initialization problems and outliers, restricted to data in which there is a notion of a center

Objective: Sum of Squared Errors

- Given a partition of the data into k clusters, we can compute the **center** (i.e., mean, center of mass) of each cluster.

$$\mu_i = \frac{1}{n_i} \sum_{x \in C_i} x$$

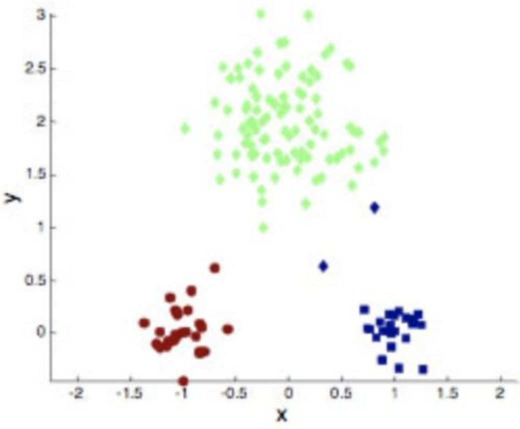
- For a well formed cluster, its points should be close to its center. We measure this with sum of squared error (SSE), and formulate our objective to find a partition $\mathbb{C}^*$ that minimizes sum of squared error:

$$\mathbb{C}^* = \underset{\mathbb{C}=\{C_1,\dots,C_k\}}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2$$

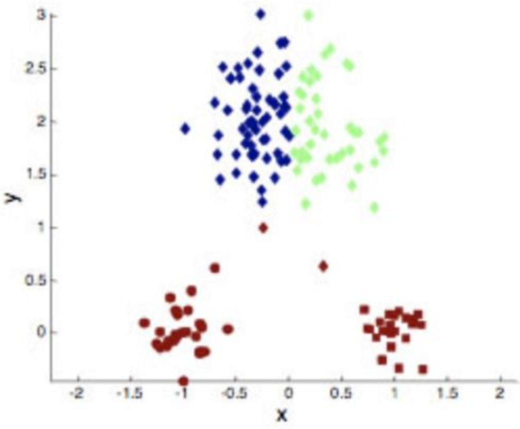
Internal Evaluation: Sum of Squared Errors

- The quality of clusters can be evaluated by the sum of squared error (SSE)

$$\sum_{i=1}^{k} \sum_{x \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2$$



Optimal Clustering



Sub-optimal Clustering

External Indexes: Rand Index

- If true class labels (ground truth) are known, the validity of a clustering can be verified by comparing the class labels and clustering labels.

- Given partition (P) and ground truth (G), measure the number of vector pairs that are:
    - a: in the same class both in P and G.
    - b: in the same class in P, but in the different class in G.
    - c: in different classes in P, but in the same class in G.
    - d: in different classes both in P and G.

$$RI = \frac{a+d}{a+b+c+d}$$

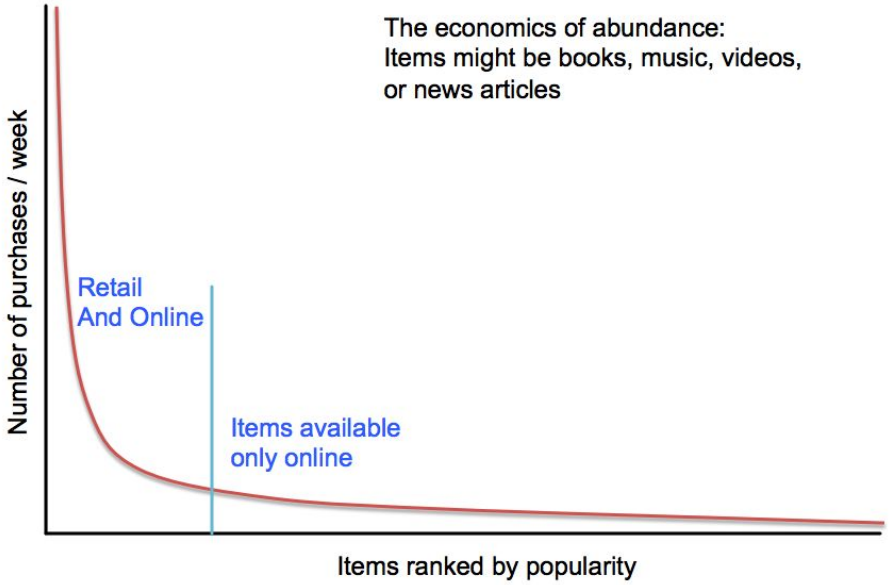
Conclusion

- There are a massive number of clustering algorithms.
- Clustering is hard to evaluate, but very useful in practice.
- Clustering is highly application dependent and to some extent subjective.
- There is no one universal recipe for choosing a clustering technique and its associated parameters.

Recommendation System

The Long Tail

- The web enables near-zero-cost dissemination of information about products
  - Give rise to the “Long Tail” phenomenon
  - Lead to information overload



W Foster
School of Business

Recommendation Systems

- Recommendation Systems recommend items (e.g. books, products, web pages) to users based on examples of their preferences.

- Recommendation systems help
    - Reduce the search space and mitigate information overload
    - Get exposed to new content that they may be interested in but may not be aware of their existence

Problem Definition

- **C** = Set of Customers
- **S** = Set of Items

- A utility function **u** measures the usefulness of item **s** to user **c.**

- For each user **c ϵ C**, we want to choose items **s ϵ S** that maximize **u**.

$$\forall c \in C, s'_c = argmax(u(c,s)) s \in S$$

Recommendation System Applications

- Movie/TV recommendation (Netflix, Hulu, iTunes)
- Product recommendation (Amazon)
- Social recommendation (Facebook)
- News content recommendation (Yahoo)
- Music recommendation (Spotify)
- Friend recommendation (Snapchat)
- Job recommendation (Linkedin)

The value of Recommendations

- Netflix: 2/3 of the movies watched are recommended
- Google News: recommendations generate 38% more click-through
- Amazon: 35% sales from recommendations
- Choicestream: 28% of the people would buy more music if they found what they liked.

The Netflix Prize (2006-2009)

The Netflix Prize

- In October, 2006 Netflix released a dataset containing 100 million anonymous movie ratings and challenged the machine learning communities to develop systems that could beat the accuracy of its recommendation system, **Cinematch**.
- Award $1MM to the team who can improve the RMSE of Cinematch by 10+%.

Movie Ratings

		Star Wars	Hoop Dreams	Contact	Titanic
	Joe	5	2	5	4
Users	John	2	5		3
	Al	2	2	4	2
	Everaldo	5	1	5	*?*

Goal: Predict *?* (a movie rating) for a user

Approaches to Recommendation System

- Content-based Model

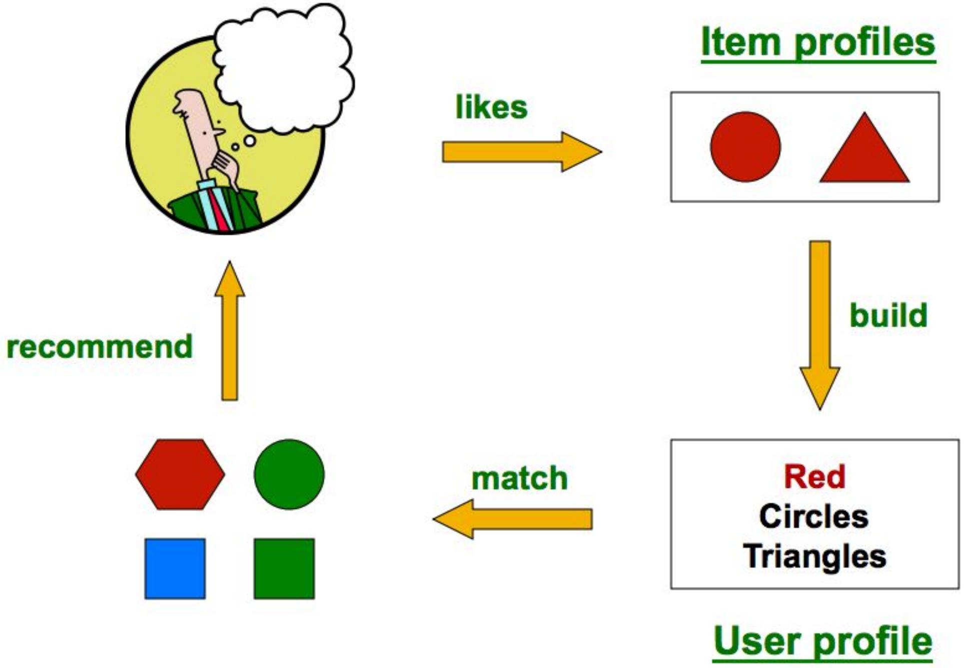
- Collaborative Filtering
  - User-User: Find similar users to me and recommend what they liked
  - Item-Item: Find similar items to those that I have previously liked

- Latent Factor Model

Content-based Systems

Content-based Recommendations

Recommend items to customer x similar to previous items rated highly by x.



Foster School of Business

Item Profile

- For each item, create an **item profile**.

- Item profile is a set of features:
  - *Movies*: <author, title, actor, director and etc.>
  - *Products*: <brand, price, category and etc.>
  - *Job*: <company, title, location and etc.>

- Product profile: <brand, price, category, ….>
  - iPad - <apple, 529.0, electronics, …>
  - iPhone - <apple, 999.0, electronics, …>

User Profile

- For each user x, create a **user profile** based on the items rated highly by x.

- Aggregated items profiles:
  - ***Simple Average***: average of rated item profiles.
  - ***Weighted Average***: weight each profile by the rating.
  - ***Normalized Weighted Average***: weight each profile by the normalized rating (i.e. adjust the rating by subtracting the mean rating of a user).

- User profile for movies: <romance, action, superhero, ….>
  - User A - <4.5, 1.5, 2.0, …>
  - User B - < 0.5, 4.7, 4.3, …>

Quiz: User Profile

- Items are movies with only feature being Actor.
    - Item profile: vector with 0 or 1 for each actor.

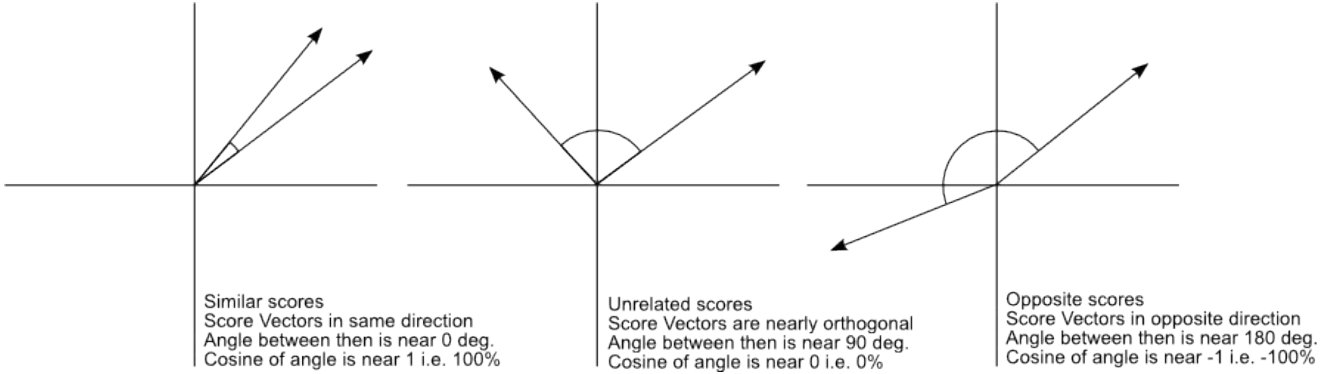
- The user x rated the following 5 movies
    - Movie 1: <Actor A: 1, Actor B: 0> - Rating 4
    - Movie 2: <Actor A: 1, Actor B: 0> - Rating 5
    - Movie 3: <Actor A: 0, Actor B: 1> - Rating 1
    - Movie 4: <Actor A: 0, Actor B: 1> - Rating 1
    - Movie 5: <Actor A: 1, Actor B: 1> - Rating 4

- What is the user profile for x?
    - Simple Average = ? < Actor A: 3/5, Actor B: 3/5>
    - Weighted Average = ? < Actor A: 13/5, Actor B: 6/5>
    - Normalized Weighted Average = ? < Actor A: 4/5, Actor B: -3/5>

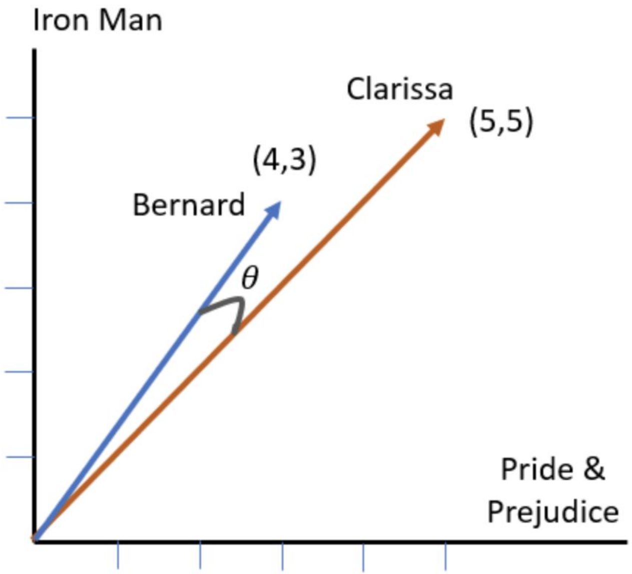
Make Recommendations

Given user profile **A** and item profile **B,** we calculate the similarity between the two using **cosine** similarity.

$$cos(\theta) = \frac{A \cdot B}{||A \; || \; B||} = \frac{\sum_i A_i B_i}{\sqrt{\sum_i A_i^2}\sqrt{\sum_i B_i^2}}$$



Quiz: Cosine Similarity



$Calculating:$

$$b.c = \sum_{i=1}^{n} b_i c_i = (4 \times 5) + (3 \times 5) = 35$$

$$\|b\| = \sqrt{4^2 + 3^2} = 5$$

$$\|c\| = \sqrt{5^2 + 5^2} = 5\sqrt{2}$$

$$similarity = \frac{35}{5 \times 5\sqrt{2}} \sim 0.989$$

Summary of Content-based Approach

- Used by Pandora.com where trained music analyst scores each song based on hundreds of distinct musical characteristics.
  - These attributes, or genes, capture not only a song's musical identity, but also qualities that are relevant to understanding listener's musical preferences.
- Pros
  - No need for data on other users.
  - Able to recommend new & unpopular items (No first-rater problem).
  - Interpretability: Explanations for recommended items.
- Cons
  - Finding the appropriate features can be hard.
  - Overspecialization: never recommends items outside user's current interest.
  - Cold-start problem for new users (How to build a user profile).

Collaborative Filtering

Key to Collaborative Filtering

Common insight: personal tastes are correlated

If Alice and Bob both like X and Alice likes Y, then Bob is more likely to like Y, especially (perhaps) if Bob knows Alice.

Collaborative Filtering

Collaborative filtering (CF) systems work by collecting user feedback in the form of ratings for items in a given domain and exploiting similarities in rating behavior amongst several users in determining how to recommend an item.

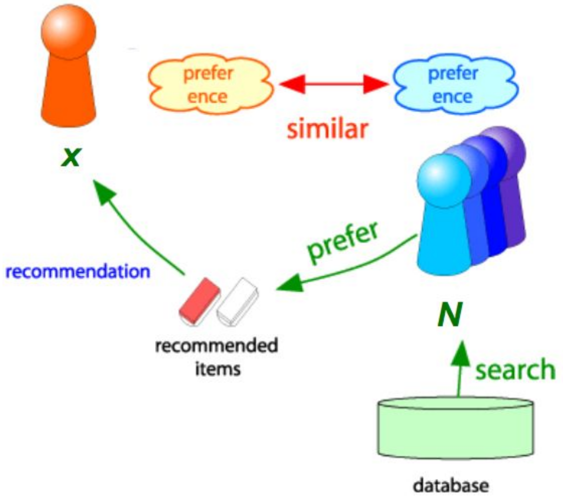
		Items			
		1	2	..	m
	1	5	2	5	4
Users	2	2	5		3
	⋮	2	2	4	2
	n	5	1	5	?

Goal: Predict *?* (an item) for n (a user)

User-User Collaborative Filtering

A subset of users are chosen based on their similarity to the active user, and a weighted combination of their ratings is used to produce predictions for this user.

1. Assign a weight to all users with respect to similarity with the active user.
2. Select k users that have the highest similarity with the active user - commonly called the *neighborhood*.
3. Compute a prediction from a weighted combination of the selected neighbors' ratings.



Foster School of Business

User-User Collaborative Filtering

Step 1: the weight $w_{a,u}$ is a measure of similarity between the user u and the active user a. The most commonly used measure of similarity is the correlation coefficient between the ratings of the two users:

$$w_{a,u} = \frac{\sum_{i \in I}(r_{a,i} - \bar{r}_a)(r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i \in I}(r_{a,i} - \bar{r}_a)^2 \sum_{i \in I}(r_{u,i} - \bar{r}_u)^2}}$$

where I is the set of items rated by both users, $r_{u,i}$ is the rating given to item i by user u, and $\bar{r}_u$ is the mean rating given by user u.

Quiz: Similar Users

$$w_{a,u} = \frac{\sum_{i \in I}(r_{a,i} - \bar{r}_a)(r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i \in I}(r_{a,i} - \bar{r}_a)^2 \sum_{i \in I}(r_{u,i} - \bar{r}_u)^2}}$$

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

sim(Alice, User1) = ?

sim(Alice, User2) = ?

sim(Alice, User3) = ?

sim(Alice, User4) = ?

Hint: use pearsonr API from package scipy.stats.stats

Quiz: Similar Users

$$w_{a,u} = \frac{\sum_{i \in I}(r_{a,i} - \bar{r}_a)(r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i \in I}(r_{a,i} - \bar{r}_a)^2 \sum_{i \in I}(r_{u,i} - \bar{r}_u)^2}}$$

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

sim(Alice, User1) = 0.85

sim(Alice, User2) = 0.70

sim(Alice, User3) = 0.0

sim(Alice, User4) = -0.79

Missing ratings in Similarity

- Most of ratings are missing in the user-item matrix
  - It’s often difficult to find similar users who have rated same items in the past.

- Instead of calculating similarity between two users based on ONLY common items, we use the imputed ratings of all items.
  - Fill in the missing ratings with the average rating of that user.
  - Calculate Pearson correlation coefficient based on all items.

Example

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	$2/3$			$5/3$	$-7/3$		
B	$1/3$	$1/3$	$-2/3$				
C				$-5/3$	$1/3$	$4/3$	
D		0					0

sim(A,B) = cos(r_A, r_B) = 0.09; sim(A,C) = -0.56

User-User Collaborative Filtering

Step 2: some sort of threshold is used on the similarity score to determine the K most similar users as the "neighborhood."

User-User Collaborative Filtering

Step 3: predictions are generally computed as the weighted average of deviations from the neighbor's mean, as in:

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u \in K} (r_{u,i} - \bar{r}_u) \times w_{a,u}}{\sum_{u \in K} w_{a,u}}$$

where $p_{a,i}$ is the prediction for the active user a for item i, $w_{a,u}$ is the similarity between users a and u, and K is the neighborhood or set of most similar users.

Problems of User-User Collaborative Filtering

- The search for similar users has high computational complexity, causing conventional neighborhood-based CF algorithms to not scale well.
- It is common for the active user to have highly correlated neighbors that are based on very few co-rated (overlapping) items, which often result in bad predictors.
- When measuring the similarity between users, items that have been rated by all (and universally liked or disliked) are not as useful as less common items.

Item-Item Collaborative Filtering

- An extension to User-User CF.

- Addresses the problem of high computational complexity of searching for similar users.

- **The idea: Rather than matching similar users, match a user's rated items to similar items.**

Item-Item Collaborative Filtering

> In this approach, similarities between pairs of items i and j are computed off-line using Pearson correlation, given by:

$$w_{i,j} = \frac{\sum_{u \in U}(r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U}(r_{u,i} - \bar{r}_i)^2 \sum_{u \in U}(r_{u,j} - \bar{r}_j)^2}}$$

where U is the set of all users who have rated both items i and j, $r_{u,i}$ is the rating of user u on item i, and $\bar{r}_i$ is the average rating of the ith item across users.

Item-Item Collaborative Filtering

> Now, the rating for item i for user a can be predicted using a simple weighted average, as in:

$$p_{a,i} = \frac{\sum_{j \in K} r_{u,i} w_{i,j}}{\sum_{j \in K} |w_{i,j}|}$$

where K is the neighborhood set of the k items rated by a that are most similar to i.

Item-Item CF Example

movies \ users	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3			5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

(white) - unknown rating (yellow) - rating between 1 to 5

Item-Item CF Example

users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

- estimate rating of movie **1** by user **5**

Item-Item CF Example

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		**0.41**
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		**0.59**

Neighbor selection:
Identify movies similar to movie **1**, **rated by user 5**

Here we use Pearson correlation as similarity:

1) Subtract mean rating m_i from each movie i

$m_1 = (1+3+5+5+4)/5 = 3.6$

row 1: [-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]

2) Compute cosine similarities between rows

Foster School of Business

Item-Item CF Example

users

movies	1	2	3	4	5	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		0.41
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		0.59

Compute similarity weights:

$s_{13}=0.41$, $s_{16}=0.59$

Item-Item CF Example

users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		2.6	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Predict by taking weighted average:

$r_{15} = (0.41*2 + 0.59*3) / (0.41+0.59) = 2.6$

Item-Item vs. User-user

- In theory, user-user and item-item are dual approaches.
- In practice, item-item outperforms user-user in many use cases.
- Items are “simpler” than users
  - User similarity is dynamic, pre-computing user neighborhood can lead to poor predictions.
  - Item similarity is static, allowing pre-computing.

The Sparsity Problem

> Typically: large product sets, user ratings for a small percentage of them

> Example Amazon: millions of books and a user may have bought hundreds of books
  – The probability that two users that have bought 100 books have a common book (in a catalogue of 1 million books) is 0.01 (with 50 and 10 millions is 0.0002).

> If you represent the Netflix Prize rating data in a User/Movie matrix you get.
  – 500,000 x 17,000 = 8,500 M positions
  – Out of which only 100M are not 0's!

> Standard CF must have a number of users comparable to one tenth of the size of the product catalogue

The Cold-start Problem

> Tough to use without any ratings information to start with

- **New User Problem**: New users should rate some initial items to have personalized recommendations.

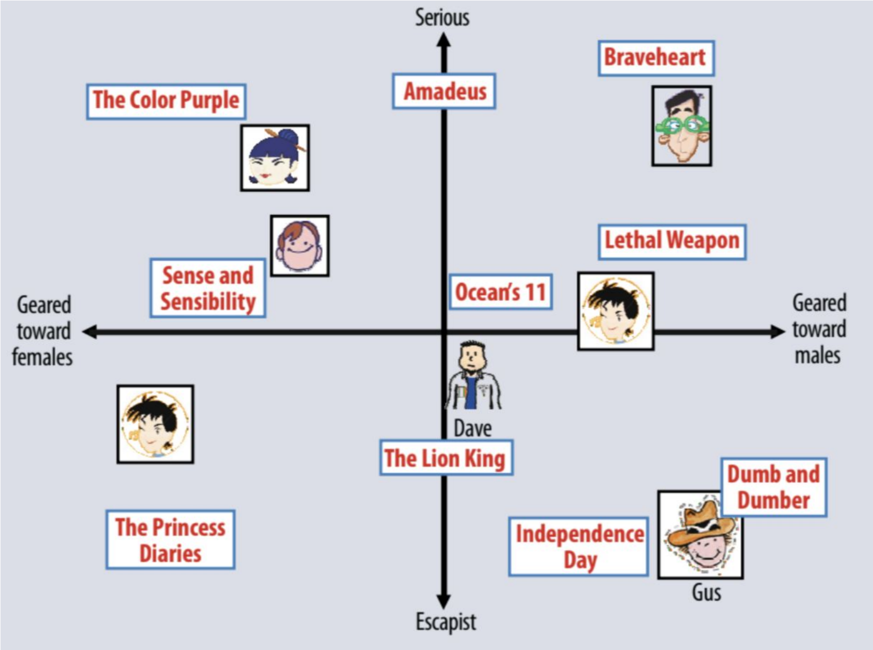
- **New Item Problem**: New items are added regularly to recommender systems. Until the new item is rated by a substantial number of users, the recommender system is not able to recommend it.

Latent Factor Model

Foster
School of Business

Latent Factor Model

- There could be a number of latent factors that affect the recommendation
  - For movies, these latent factors might measure genre such as comedy, drama, action, and etc.
- Decompose user ratings on movies into separate item and movie matrices to capture latent factors.

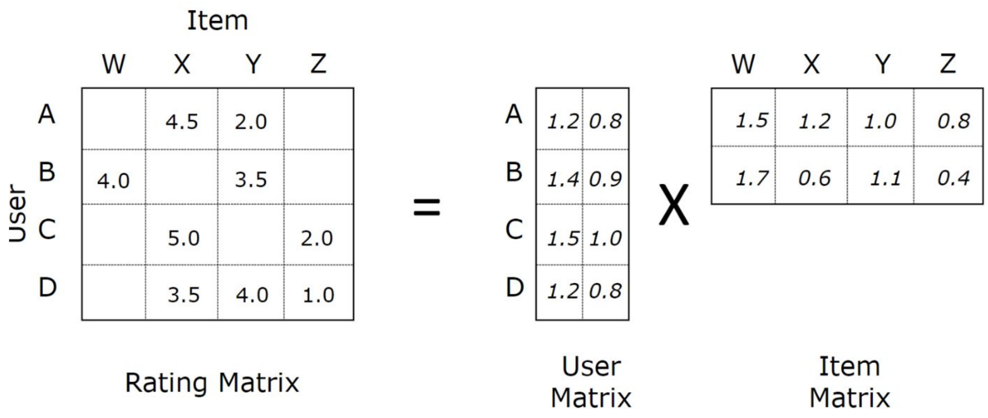


Foster School of Business

Matrix Factorization

- Express a matrix **M** approximately as a product of two matrices **A** and **B**.

- Similarly, approximate the user-items matrix **M** as a product of user latent matrix **A** and item latent matrix **B**.
  - Explain the ratings by projecting items and users to the same latent space.
  - Estimate unknown ratings as inner-products of factors.



Foster School of Business

Matrix Factorization

We can express finding the "closest" matrix as an optimization problem

$$\min_{A,B} \sum_{(u,i)\ observed} \left(M_{u,i} - \langle A_{u,:}, B_{:,i} \rangle\right)^2 + \lambda\left(\|A\|_F^2 + \|B\|_F^2\right)$$

Matrix Factorization

We can express finding the "closest" matrix as an optimization problem

$$\min_{A,B} \sum_{(u,i)\ observed} \left(M_{u,i} - \langle A_{u,:}, B_{:,i} \rangle\right)^2 + \lambda(\|A\|_F^2 + \|B\|_F^2)$$

Computes the error
in the approximation
of the observed
matrix entries

Matrix Factorization

We can express finding the "closest" matrix as an optimization problem

$$\min_{A,B} \sum_{(u,i)\ observed} \left(M_{u,i} - \langle A_{u,:}, B_{:,i} \rangle\right)^2 + \lambda\left(\|A\|_F^2 + \|B\|_F^2\right)$$

Regularization preferences matrices with small Frobenius norm

Matrix Factorization Example

users

items

1		3			5			5		4	
		5	4			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

~

items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

•

users

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

Matrix Factorization Example

users

items

1		3			5			5		4	
		5		?		4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

~

items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

•

users

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

Matrix Factorization Example

users

items

1		3			5			5		4	
		5		?		4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

~

items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

•

users

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

Matrix Factorization Example

users (columns) × items (rows)

1		3			5			5		4	
		5	?			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

~

items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

•

users

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

$$[-0.5, 0.6, 0.5] \cdot [-2, 0.3, 2.4] = 2.38$$

W | Foster School of Business

Evaluation

Evaluation Split

movies

users

1	3	4			
	3	5			5
		4	5		5
		3			
		3			
2			2		2
				5	
	2	1			1
	3			3	
1					

movies

users

1	3	4			
	3	5			5
		4	5		5
		3			
		3			
2			?		?
				?	
	2	1			?
	3			?	
1					

Test Data Set

Evaluation Metric: RMSE

- Compare predictions against withheld known ratings (test set *T*)
- Root Mean Squared Error (RMSE)

$$\sqrt{\frac{\sum_{(x,i)\in T}(r_{xi}-r^*_{xi})^2}{N}}$$

where $N = |T|$

r_{xi} is the predicted rating

r_{xi}^* is the actual rating

Issues with RMSE

- **In practice, we care only to predict high ratings**
  - RMSE might penalize a method that does well for high ratings and badly for others
- **Narrow focus on accuracy sometimes misses the point**
  - Prediction Diversity
  - Order of predictions
- **Alternative: precision at top k**
  - Percentage of predictions in the user's top k withheld ratings

Lab