MSDS600 W6: Recommender systems, big data, and graph analysis

REGIS UNIVERSITY

Review from W5

- autoEDA, autoML, autoDS, automation with Python
- pandas-profiling, pycaret, H2O, TPOT, MLBox, others
- Python scripts for automation
- Data science GUIs for low- or no-code automation
- Cloud tools (AWS, GCP, Azure, IBM)

https://aws.amazon.com/rekognition/?blog-cards.sort-by=item.additionalFields.createdDate&blog-cards.sort-order=desc

This week's topics

- Distance calculations for similarity (can be used in recommenders, text analysis, graph analysis, and more)
- Recommender systems – main focus with FTE and assignment
- Big data – discuss methodology and tools
- Graph analysis – discuss methodology and tools

Types of recommender systems

Content based filtering

- Recommend content based on users' preferences and item details
  - Based only on users' previous actions
  - Community data is not used.
  - Difficult to predict new items.

Hybrid

- Combines content and collaborative systems

Collaborative filtering

- Recommend content based on similar users and/or items
- Similar users will like similar items
- Explicit or implicit
- Age, gender, geographical, occupation, etc.
  - Multiple accounts
  - People are bad at rating
  - Cold-start problem – impossible with no data on new users
  - Scalability – hard to scale with big data
  - Sparsity – lots of missing values (we saw this with movielens)

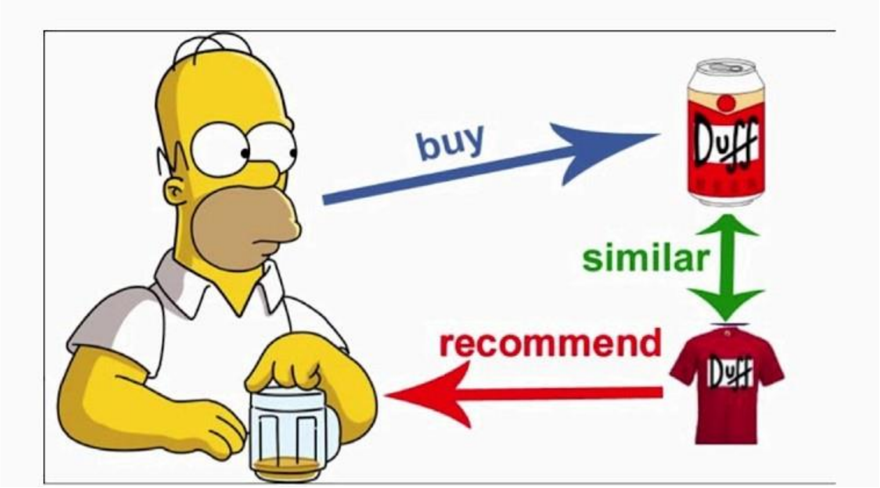
Naïve recommender systems

- Recommend highly-rated items from similar users
- Recommend most popular items
  - “Napoleon Dynamite” problem: Hard to predict ratings for the movie because it’s polarizing.
  - Users’ past preference is important.
- Recommend most similar items to a user’s top-rated items
- Netflix uses advanced models and even [had a competition](#)for recommender systems

Where we could use recommender systems

- E-commerce (e.g. Amazon)
- Movies/shows/audio (e.g. Netflix, music, podcasts)
- Jobs (e.g. consulting websites or hiring)
- Social media (e.g. Facebook friends)
- News articles
- Learning materials (tutorials, classes)
- Anywhere that there is a match between a user (person, organization, etc) and items (products, news, services, employees, etc)

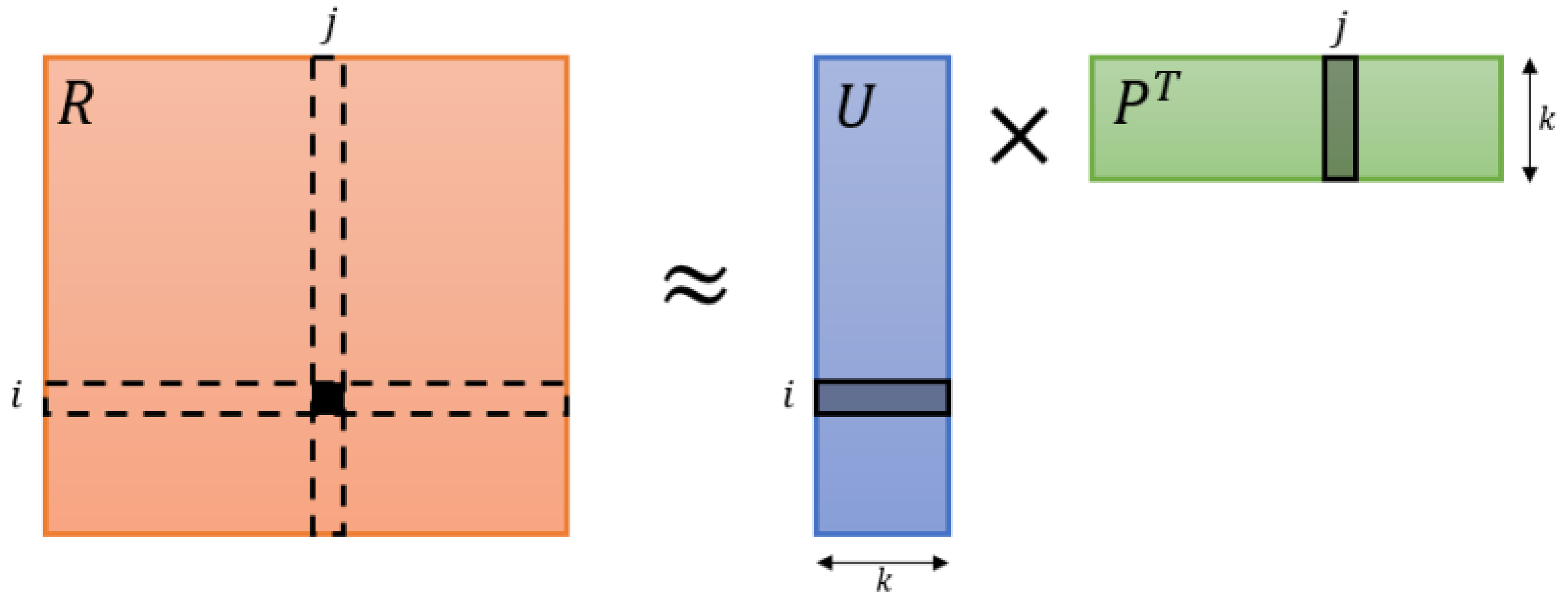
- Datasets: https://cseweb.ucsd.edu/~jmcauley/datasets.html



https://towardsdatascience.com/build-your-own-recommender-system-within-5-minutes-30dd40388fbf

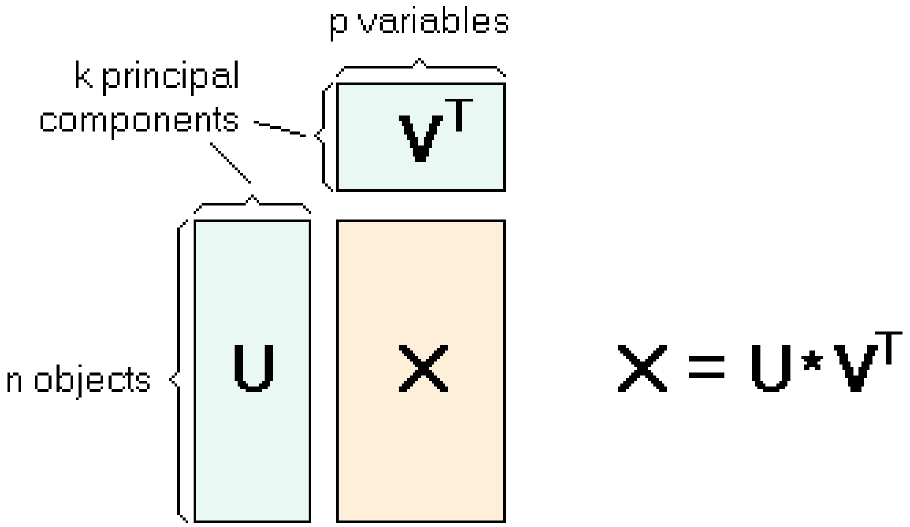
Matrix factorization

- Can be used for collaborative filtering
- R matrix of ratings is users (rows) and items (columns)
- Break up into 2 matrices with a new dimension k
- k is number of factors effecting each rating. Also called the 'rank'



Similar to PCA

- Principle component analysis
- Reduces dimensions of X from p to k
- Each row (datapoint) can be reduced in dimension from

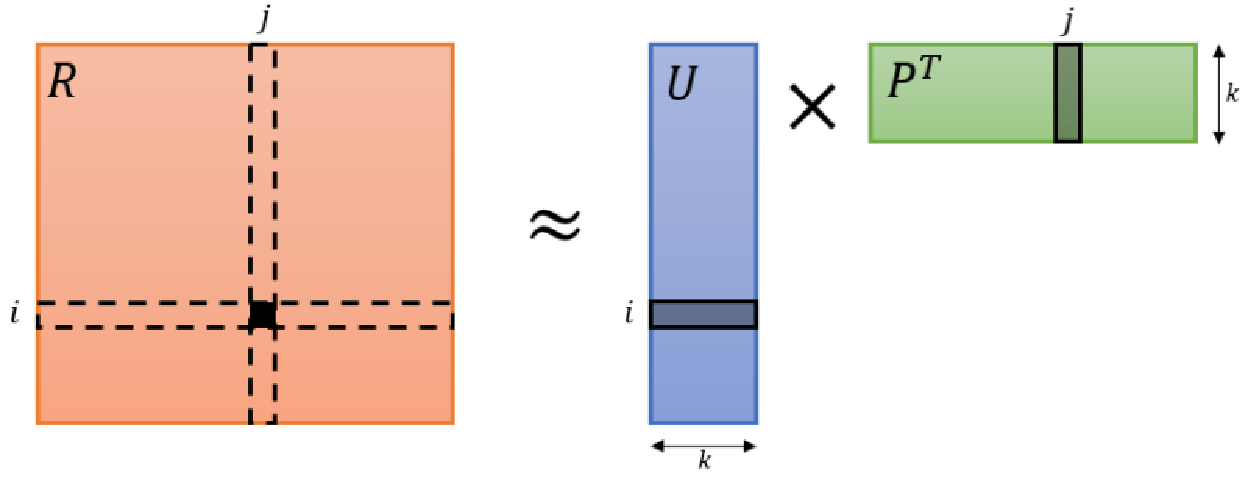


http://www.statistics4u.com/fundstat_eng/cc_pca_loadscore.html

ALS (alternating least squares)

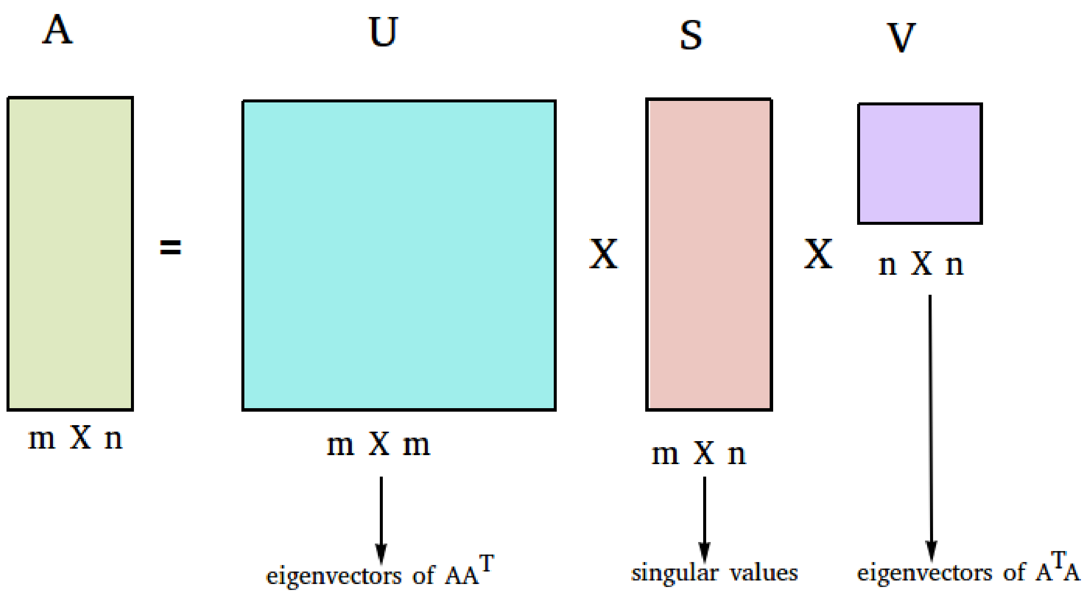
- Predicting ratings from two matrices, U and P^T
- Minimize cost function:

$$J = ||R - U \times P^T||_2 + \lambda \left(||U||_2 + ||P||_2 \right)$$



SVD – singular value decomposition

- Breaks up into 3 matrices
- Also used for topic modeling in NLP



https://blog.paperspace.com/dimension-reduction-with-principal-component-analysis/

Evaluation metrics

RMSE

- Root mean square error.

$$RMSErrors = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$$

http://statweb.stanford.edu/~susan/courses/s60/split/node60.html

Confusion matrix

		True condition	
	Total population	Condition positive	Condition negative
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error
	Predicted condition negative	**False negative**, Type II error	**True negative**

https://en.wikipedia.org/wiki/Confusion_matrix

Similarity metrics in collaborative filtering

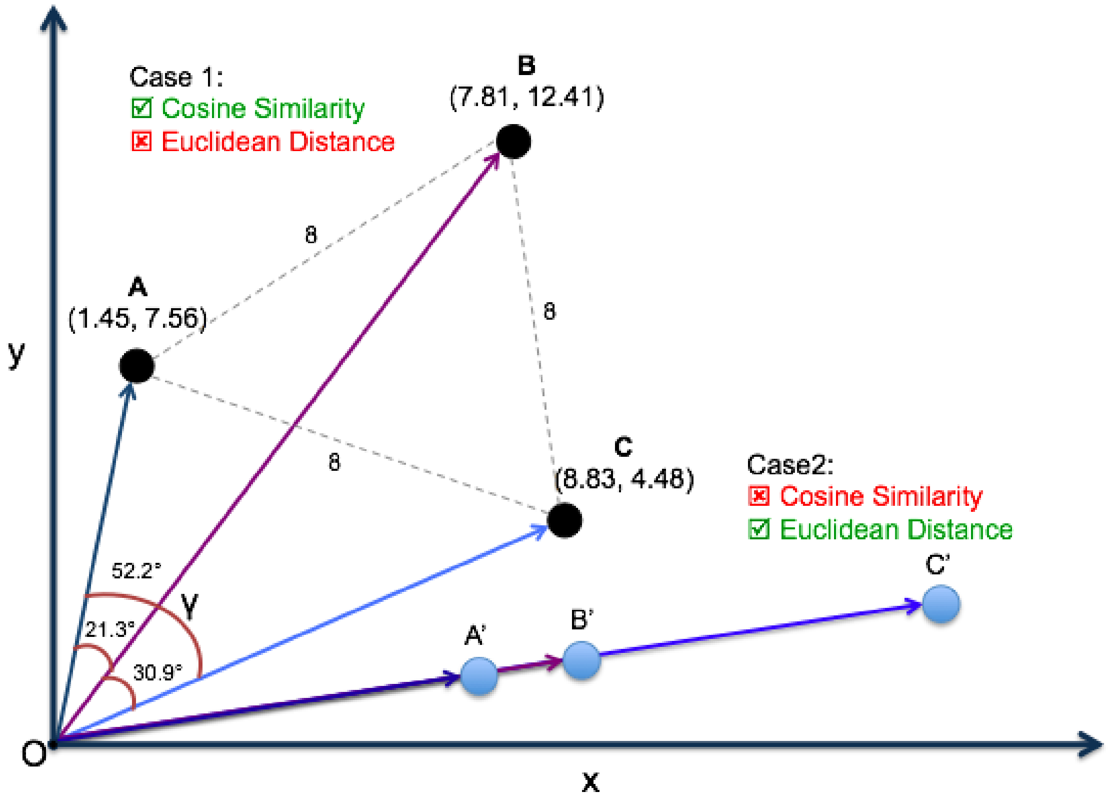
Euclidean distance $= \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$ https://en.wikipedia.org/wiki/Euclidean_distance

Cosine distance $\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}},$ https://en.wikipedia.org/wiki/Cosine_similarity

Pearson coefficient $\rho_{X,Y} = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y}$ (Eq.1) $=$ $r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$ (Eq.3)

Cosine and Euclidean distances

Cosine distance or similarity measures the angles between vectors, while Euclidean distance is the straight line distance between two points



https://medium.com/@sasi24/cosine-similarity-vs-euclidean-distance-e5d9a9375fc8

Python recommender packages

- Surprise
- Lightfm
- Turi create (https://github.com/apple/turicreate)
- pyspark can be used

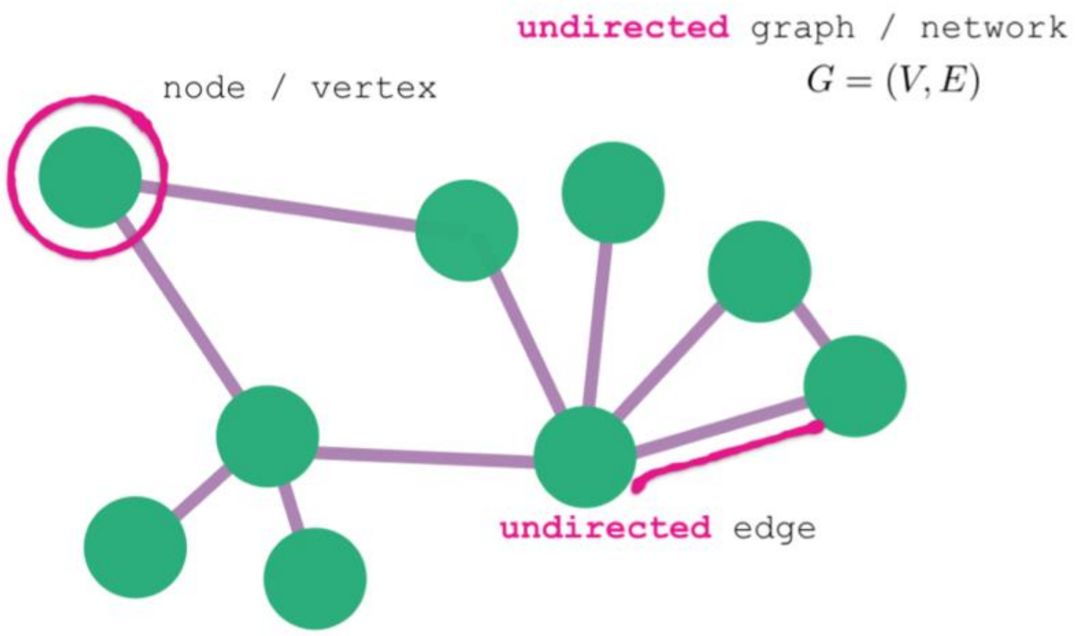
Big Data

- Solutions: Scale up or scale out
- Up: bigger single machine
- Out: parallelize (use a cluster)
- Easy to use clusters on cloud services (AWS, GCP)
- Spark/pyspark can be used for data analysis and EDA, ML, recommender systems, data pipelines, etc

- Big data packages in Python:
- pyspark
- Dask
- Cloud libraries (GCP BigQuery, AWS Redshift, SDKs for cloud)
- Mrjobs (Hadoop MapReduce, now mostly done with spark/pyspark)

Graph theory and modeling

- Represents connected networks
- The networkx package is one for analysis in Python (there are at least a few books on this)
  - Other packages [here]
- We can visualize networks with plots
- We can use ML with graph theory to predict nodes, connections, etc
  - E.g. suggest friends on a social network



https://www.kdnuggets.com/2019/08/neighbours-machine-learning-graphs.html

REGIS UNIVERSITY

Appendix: More detail on Alternating Least Squares (ALS)

ALS

- $||R - UxP^T||_2$ is the mean squared error between actual and predicted values
- Second term with gamma is a regularization term

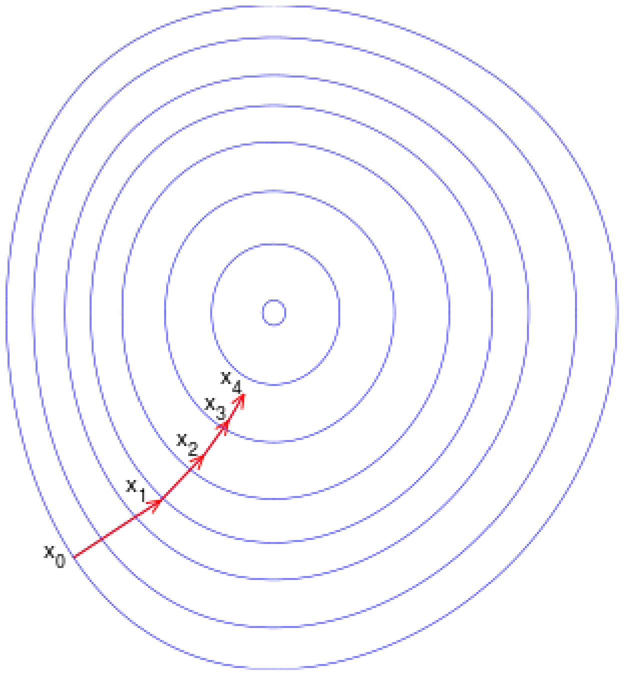
$$J = ||R - U \times P^T||_2 + \lambda \left(||U||_2 + ||P||_2\right)$$

ALS

- Minimizes loss function with gradient descent
- Take the derivative of the loss function w.r.t. variables we are changing (predicted values in the U/P matrices), move these variables in the direction that decreases the loss function (negative value of the gradient or derivative)

$$J = ||R - U \times P^T||_2 + \lambda \left(||U||_2 + ||P||_2 \right)$$

https://en.wikipedia.org/wiki/Gradient_descent



ALS on big data

- If you want to see how this can be parallelized with spark, see here: http://stanford.edu/~rezab/classes/cme323/S15/notes/lec14.pdf