(MSc) Data Mining

Topic 10 : Anomaly Detection and Recommendation Systems

Part 01 : Overview

Dr Bernard Butler and Dr Kieran Murphy

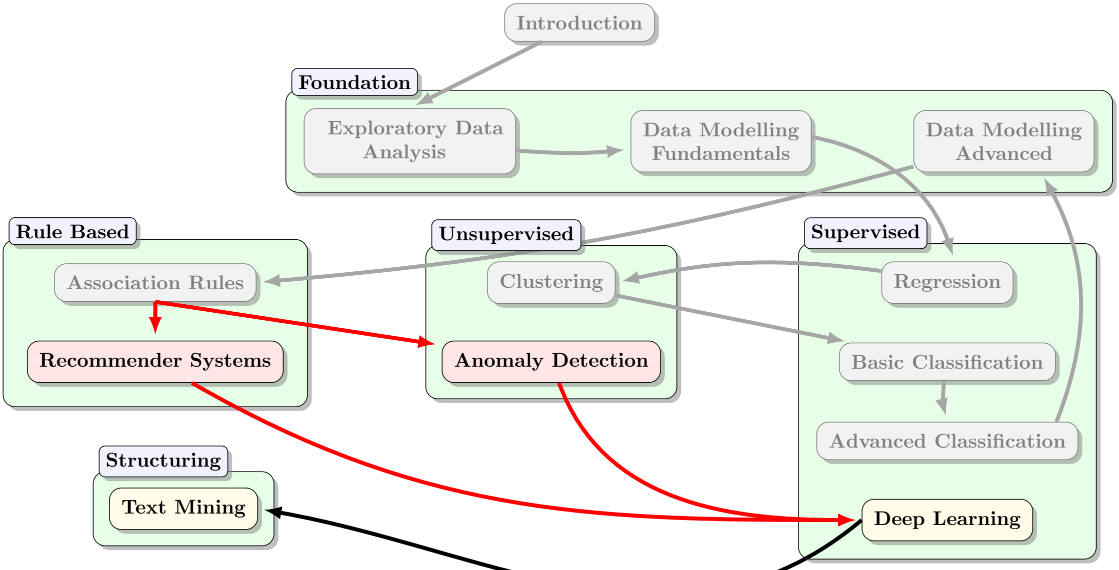
Department of Computing and Mathematics, WIT.
(bernard.butler@setu.ie; kmurphy@wit.ie)

Spring Semester, 2025

Outline

- How is anomaly detection used
- Why recommendation is big business
- Object view - Content-based filtering
- Object-user view - Collaborative filtering

Data Mining (Week 10)



Outline

This Week's Aim

This week's aim is to introduce the main concepts and representative algorithms used in two applications of data mining: anomaly detection and recommender systems

- Anomaly detection context and some algorithms
  - Unsupervised vs. Supervised vs Semi-Supervised techniques
  - Speicialised algorithms vs reuse of existing algorithms
- Uses of recommender systems
  - Item-item (Content-based) filtering
  - Item-User (Collaborative) filtering

These uses of data mining touch on many aspects of the module to date, especially clustering, dimensionality reduction, similarity measures and classification.

Outline

What is Anomaly Detection?

Definition 1 (Anomaly Detection)

Anomaly detection is a procedure that identifies data records (observations and/or events) that depart from their dataset's typical behavior, often for unexplained reasons.

Anomaly, Outlier, Novelty, Change Detection use many of the same analytical procedures but their goals are different.

What is Anomaly Detection?

Definition 1 (Anomaly Detection)

Anomaly detection is a procedure that identifies data records (observations and/or events) that depart from their dataset's typical behavior, often for unexplained reasons.

Anomaly, Outlier, Novelty, Change Detection use many of the same analytical procedures but their goals are different.

Anomaly

- interesting in their own right
- source is interesting
- don't change underlying data

What is Anomaly Detection?

Definition 1 (Anomaly Detection)

Anomaly detection is a procedure that identifies data records (observations and/or events) that depart from their dataset's typical behavior, often for unexplained reasons.

Anomaly, Outlier, Novelty, Change Detection use many of the same analytical procedures but their goals are different.

Anomaly

- interesting in their own right
- source is interesting
- don't change underlying data

Outlier

- not interesting in themselves
- source can be interesting
- don't change underlying data

What is Anomaly Detection?

Definition 1 (Anomaly Detection)

Anomaly detection is a procedure that identifies data records (observations and/or events) that depart from their dataset's typical behavior, often for unexplained reasons.

Anomaly, Outlier, Novelty, Change Detection use many of the same analytical procedures but their goals are different.

Anomaly

- interesting in their own right
- source is interesting
- don't change underlying data

Outlier

- not interesting in themselves
- source can be interesting
- don't change underlying data

Novelty

- looking for them!
- source is interesting
- don't change underlying data

What is Anomaly Detection?

Definition 1 (Anomaly Detection)

Anomaly detection is a procedure that identifies data records (observations and/or events) that depart from their dataset's typical behavior, often for unexplained reasons.

Anomaly, Outlier, Novelty, Change Detection use many of the same analytical procedures but their goals are different.

Anomaly

- interesting in their own right
- source is interesting
- don't change underlying data

Outlier

- not interesting in themselves
- source can be interesting
- don't change underlying data

Novelty

- looking for them!
- source is interesting
- don't change underlying data

Change

- not interesting in themselves
- source and effect is interesting
- changes underlying data

Example Applications

Identify 3 possible applications for anomaly detection

Anomaly Detection Techniques

Unsupervised

Dataset is unlabeled; assume most instances belong (have typical behaviour) but look for those that are *most different* to the remainder of the dataset.

Anomaly Detection Techniques

Unsupervised

Dataset is unlabeled; assume most instances belong (have typical behaviour) but look for those that are *most different* to the remainder of the dataset.

Supervised

Instances in the dataset are labeled as either “normal” (most instances) or “abnormal” (some instances; unbalanced distribution). Train a classifier to decide whether a test instance belongs or not.

Anomaly Detection Techniques

Unsupervised

Dataset is unlabeled; assume most instances belong (have typical behaviour) but look for those that are *most different* to the remainder of the dataset.

Supervised

Instances in the dataset are labeled as either “normal” (most instances) or “abnormal” (some instances; unbalanced distribution). Train a classifier to decide whether a test instance belongs or not.

Semi-supervised

Dataset is unlabeled but all instances are assumed to belong. Train a model to summarise the dataset’s “normal” behaviour based on this set. For each test instance, estimate the likelihood that it was generated by the same process that generated the training set.

Simple statistical techniques

Univariate, Numeric, Normally distributed

Derive the z-score: $z_i = \frac{x_i - \mu}{\sigma}$ where μ and σ are the *known* population mean (so semi-supervised at least).
Can then compare z_i to a threshold $z_{\text{thr}} = 3.5$, say. x_i is an outlier if $\|z_i\| > z_{\text{thr}}$.
Note normality assumption. If sample mean and standard deviation are used, adjustments are needed because they depend on the potential outlier!

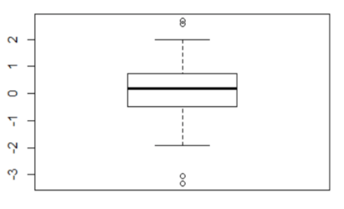
Simple statistical techniques

Univariate, Numeric, Normally distributed

Derive the z-score: $z_i = \frac{x_i - \mu}{\sigma}$ where μ and σ are the *known* population mean (so semi-supervised at least).
Can then compare z_i to a threshold $z_{\text{thr}} = 3.5$, say. x_i is an outlier if $\|z_i\| > z_{\text{thr}}$.
Note normality assumption. If sample mean and standard deviation are used, adjustments are needed because they depend on the potential outlier!

Univariate, Numeric, any distribution

Compute the *Inter-Quartile Range* $R = Q_3 - Q_1$. Then x_i is an outlier if $x_i < Q_1 - kR$ or $x_i > Q_3 + kR$, where $k = 1.5$, say. In the boxplot, circles outside the whiskers are considered outliers.



Other univariate outlier tests include Grubbs, Dixon, Cochrane, …

Reusing existing algorithms

k-nearest neighbor for outlier detection

1. For each point, find its k nearest neighbors
2. Compute a metric, e.g., median, of the distances to its k nearest neighbors
3. Use univariate outlier techniques to find candidate outliers

One-class SVM for outlier detection

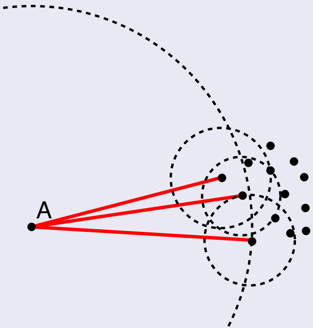
1. Ensure that the training set contains no outliers
2. Use OneClassSVM to decide the boundary of "normal" data
3. Use trained model to label test instance as 0 (normal) or 1 (outlier)

DBSCAN for outlier detection

1. DBSCAN searches for clusters as regions of high *density*
2. Some data is classed as *noise* (low density and far from other clusters)
3. Noise data can be interpreted as outliers for a given ϵ.

Local Outlier Factor algorithm

Overview of the algorithm

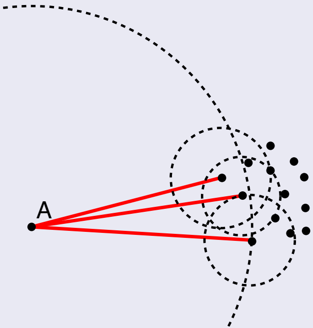


Source: Wikipedia

In the diagram, *A* is in a region with much lower point density than its nearest neighbours.

Local Outlier Factor algorithm

Overview of the algorithm

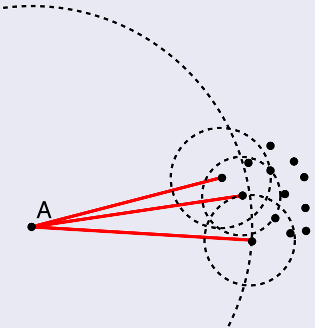


Source: Wikipedia

In the diagram, *A* is in a region with much lower point density than its nearest neighbours.
Its (average) *reachability distance* from other points is compared with the average reachability distance for each of its neighbours.

Local Outlier Factor algorithm

Overview of the algorithm

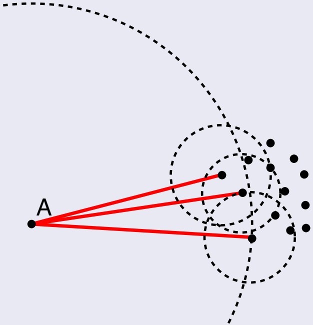


Source: Wikipedia

In the diagram, A is in a region with much lower point density than its nearest neighbours.
Its (average) *reachability distance* from other points is compared with the average reachability distance for each of its neighbours.
The LOF is a ratio of distances. If LOF $> k \geq 1$ it is an outlier, otherwise it is not.

Local Outlier Factor algorithm

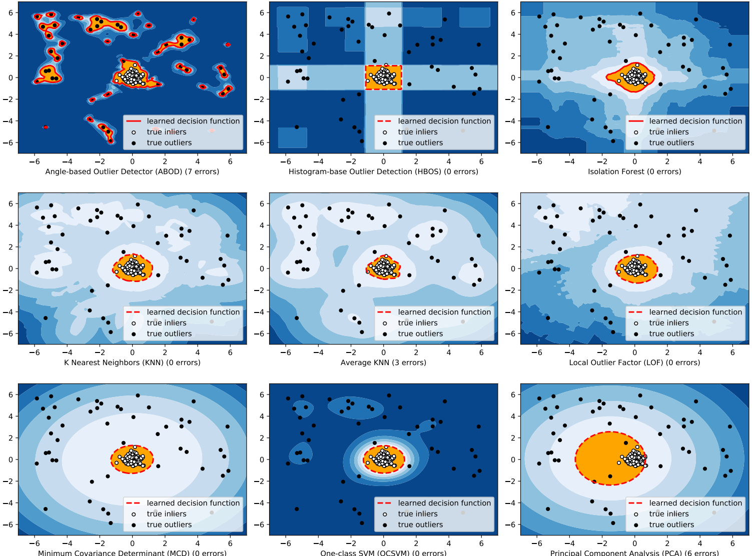
Overview of the algorithm



Source: Wikipedia

In the diagram, A is in a region with much lower point density than its nearest neighbours.
Its (average) *reachability distance* from other points is compared with the average reachability distance for each of its neighbours.
The LOF is a ratio of distances. If LOF > $k \geq 1$ it is an outlier, otherwise it is not.
Q: how to choose k? Depends on how reachability distances vary, but heuristics can help..

Selected algorithms in pyod library



- Each method generates vastly different boundaries between outliers and inliers
- Understanding the causes of outliers can suggest how they form, hence which outlier detection algorithm to use

Summary

- While there are some outlier-specific algorithms, many schemes are built upon existing classifiers and clustering algorithms

Summary

- While there are some outlier-specific algorithms, many schemes are built upon existing classifiers and clustering algorithms
- All algorithms have at least one “magic parameter” that needs to be chosen.

Summary

- While there are some outlier-specific algorithms, many schemes are built upon existing classifiers and clustering algorithms
- All algorithms have at least one “magic parameter” that needs to be chosen.
- Robust analyses are barely affected by the presence of outliers: they *accommodate* them (e.g., median as measure of central tendency)

Summary

- While there are some outlier-specific algorithms, many schemes are built upon existing classifiers and clustering algorithms
- All algorithms have at least one "magic parameter" that needs to be chosen.
- Robust analyses are barely affected by the presence of outliers: they *accommodate* them (e.g., median as measure of central tendency)
- For anomaly and especially novelty detection, finding anomalies is the main goal, e.g., fraudulent transactions.

Summary

- While there are some outlier-specific algorithms, many schemes are built upon existing classifiers and clustering algorithms
- All algorithms have at least one “magic parameter” that needs to be chosen.
- Robust analyses are barely affected by the presence of outliers: they *accommodate* them (e.g., median as measure of central tendency)
- For anomaly and especially novelty detection, finding anomalies is the main goal, e.g., fraudulent transactions.
- For supervised anomaly detection (with labeled training data), confusion matrix can be used to measure success.

Summary

- While there are some outlier-specific algorithms, many schemes are built upon existing classifiers and clustering algorithms
- All algorithms have at least one “magic parameter” that needs to be chosen.
- Robust analyses are barely affected by the presence of outliers: they *accommodate* them (e.g., median as measure of central tendency)
- For anomaly and especially novelty detection, finding anomalies is the main goal, e.g., fraudulent transactions.
- For supervised anomaly detection (with labeled training data), confusion matrix can be used to measure success.
- Otherwise anomaly detection success is difficult to measure. . .

Summary

- While there are some outlier-specific algorithms, many schemes are built upon existing classifiers and clustering algorithms
- All algorithms have at least one "magic parameter" that needs to be chosen.
- Robust analyses are barely affected by the presence of outliers: they *accommodate* them (e.g., median as measure of central tendency)
- For anomaly and especially novelty detection, finding anomalies is the main goal, e.g., fraudulent transactions.
- For supervised anomaly detection (with labeled training data), confusion matrix can be used to measure success.
- Otherwise anomaly detection success is difficult to measure…
- Anomaly detection is commonly applied to *time series* and event sequences but we do not cover either here

Outline

Background: Review of Online Business Models

Online businesses generally do at least one of the following:

- Sell a physical product
  - Manage your own sales or use 3rd party like Amazon
  - dropshipping: act as an intermediary
- Sell a digital information product
  - Downloadable material (digital content like ebooks, (offline) music, etc.)
  - Membership (recurring digital content, e.g., Spotify Premium)
- Sell a service
  - Marketing/promotion of offline service: generating leads, etc.
  - Marketing/promotion of affiliate online service, e.g., AdWords
  - Deliver the service itself, e.g., flight bookings, xAAS, . . .

Source: https://www.thebalance.com/most-common-online-business-models-2531863

Offline businesses are similar.

Business challenge

When somebody visits my website or uses my app. how do I encourage that person to buy/consume more (generating increased revenue), or to stay interested (fostering increased stickiness), in my business?

Enter. . . Recommender Systems!

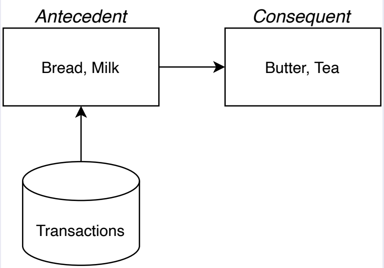
Definition 2 (Recommender Systems)

Recommender systems (also known as *recommendation engines*) seek to present options to users that are more likely to elicit a positive response, such as puchasing an item, consuming some content, retweeting, etc. To achieve this, these systems need to take account of information gleaned from the user, and any relevant context.

Recommender systems are typically generative and so can be contrasted with information filtering systems that select personalised content from a stream.

Association analysis vs Recommender Systems

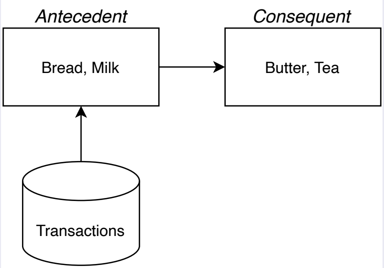
Association Analysis



- Rules are not specific to users
- Based on readily available data (transactions)
- Search and filter implicit rules
- Output is a *set* with length k (no sequence)
- "Frequently Bought Together"

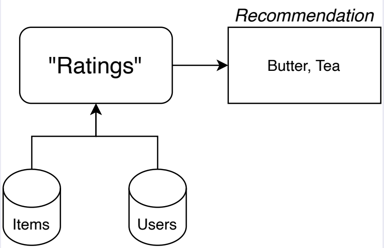
Association analysis vs Recommender Systems

Association Analysis



- Rules are not specific to users
- Based on readily available data (transactions)
- Search and filter implicit rules
- Output is a *set* with length k (no sequence)
- “Frequently Bought Together”

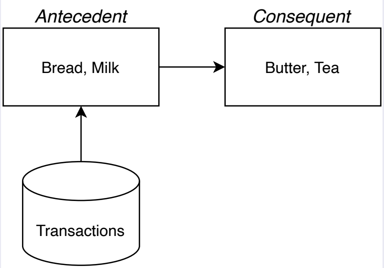
Recommender System



- Recommendations are personalised
- User and/or item attributes or rating-like data
- Prediction (with or without models)
- Output is an *ordered set* with length k
- “Customers who bought this item also bought”

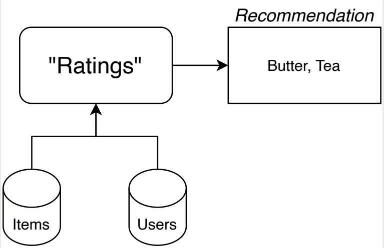
Association analysis vs Recommender Systems

Association Analysis



- Rules are not specific to users
- Based on readily available data (transactions)
- Search and filter implicit rules
- Output is a *set* with length k (no sequence)
- "Frequently Bought Together"

Recommender System



- Recommendations are personalised
- User and/or item attributes or rating-like data
- Prediction (with or without models)
- Output is an *ordered set* with length k
- "Customers who bought this item also bought"

Goal (cross-sell, up-sell) is often the same but techniques are very different

Example Applications

Identify 3 possible applications for recommender systems

General considerations

- Before online recommenders, there were *guides* (Which? etc.) and recommendations by friends and "authorities" (e.g., film reviewers)
  - Questions about trust/reputation, motives, personalisation, etc.
- the tradeoff between similarity and diversity(compare with the avoidance of overfitting in supervised learning)
  - Diversity is a function of novelty and unexpectedness, but too much can cause low accuracy
  - the Long Tailis based on the concepts of serendipity, availability, scale and choice—and getting the similarity-diversity balance right!
- the cold start problem: how to make recommendations to new users or others for which little data is available (e.g., due to security settings)
- privacy: recommendation systems can draw inferences based on past behaviour, c.f. Target recommender guessing that a teenage girl was pregnant that can harm users' privacy.
- with content-based filtering: the need for semantic alignment, not just keyword matching

Considerations for recommender systems

- The type of data available in its database (e.g., ratings, user registration information, item features, social relationships between users and context (especially location)
- The filtering algorithm used (e.g., demographic, content-based, collaborative, social-based, context-aware and hybrid)
- The model chosen (e.g., based on direct use of data: "memory-based" (neighbourhod methods), or a model generated from such data: "model-based").
- Techniques such as probabilistic approaches, Bayesian networks, nearest neighbors algorithm; bio-inspired such as neural networks and genetic algorithms; fuzzy models, computational linear algebra (SVD or NNMF) to reduce sparsity levels, etc.
- Sparsity level of the database and the desired scalability.
- Performance of the system (time and memory needed).
- The objective (e.g., specific predictions versus top N recommendations), and
- The desired quality of the results (e.g., *novelty*, *coverage* and *precision*.

Outline

Content-based filtering

Items have *explicit*, static, attributes with values

Users have *implicit*, dynamically changing, likes, dislikes and preferences

- The first recommender systems were based on content-based filtering
- Assume user has signalled his/her preferences in the past

  Active By rating items, e.g., giving a film 4 stars out of 5

  Passive By interacting with items, e.g., booking a holiday in Spain

- Wish to recommend other content based on the user's preferences

Recommended content should *be similar to* preferred content *for that user*.

Recommendations are tuned to a specific user, based on a) his/her previous activity and b) a rich model of item attributes.

Recap on distance measures

Definition 3 (Distance Measure)

A *distance measure* (c.f., its complement, a *similarity measure*) is a scalar number $d(x_1, x_2)$ that quantifies the degree of agreement between two (usually vector-valued) attribute vectors x_1 and x_2. When $x_1 = x_2$, $d(x_1, x_2) = 0$ and $d(x_1, x_2) > 0$ otherwise. It increases as the difference in the item attribute vectors increases.

Distance measures can be defined for numeric and non-numeric data (such as Strings).
By definition, content-based recommendation identifies content with high *similarity* (equivalently: low distance) to content previously rated highly by a user.

(Selected) Distance Measures for categorical data

Let $\boldsymbol{x}_1 = [e_{1,1}, e_{1,2}, \ldots e_{1,k}]^T$ and $\boldsymbol{x}_2 = [e_{2,1}, e_{2,2}, \ldots e_{2,k}]^T$. Furthermore let $e_{1,j}e_{2,j} = 1$ if $e_{1,j} = e_{2,j}$ and $e_{1,j}e_{2,j} = 0$ otherwise. To compute s, the number of matching attributes between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, we can just compute the dot product:

$$s = \boldsymbol{x}_1^T \boldsymbol{x}_2$$

and the number of mismatches is $d = k - s$, where k is the number of attributes in $\boldsymbol{x}$.

Definition 4 (Euclidean distance for categorical observations)

$\|\boldsymbol{x}_1 - \boldsymbol{x}_2\| = \sqrt{\boldsymbol{x}_1^T \boldsymbol{x}_1 - 2\boldsymbol{x}_1^T \boldsymbol{x}_2 + \boldsymbol{x}_2^T \boldsymbol{x}_2} = \sqrt{2(k - s)}$. So the maximum distance occurs when $s = 0$ ($\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ share no attribute values in common, as expected.

Definition 5 (Hamming Distance)

This is the number of mismatched values $k - s$.

(Selected) Distance Measures for categorical data - ratios

Definition 6 (Cosine similarity)

$\cos\theta_{1,2} = \frac{\boldsymbol{x}_1^T \boldsymbol{x}_1}{\|\boldsymbol{x}_1\| \|\boldsymbol{x}_2\|} = \frac{s}{k}$.

Definition 7 (Jaccard Coefficient)

This is the ratio of the number of matching values s to the number of distinct values that appear in $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, across the d attributes of both. It is $J(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{s}{2(k-s)+s} = \frac{s}{2k-s}$.

Note that all these distance measures are functions of s and k, where k is a constant and s is a count of the number of matching attribute values across the two observations in question.

Overview of Content-Based Filtering Algorithm (CBF)

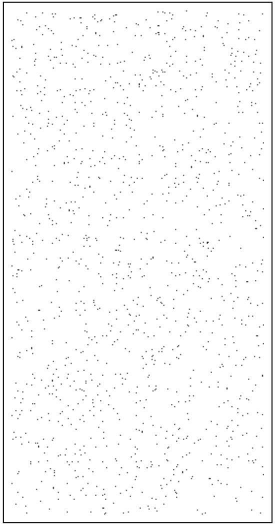
Method (Content-Based Filtering)

1. REGISTER EACH ITEM: Assign the item attributes used for CBF: *Each film has a genre*
2. ACCEPT RATING (BEHAVIOUR) EVENTS PER USER: Update the user's preferences, expressed as item attributes: *Joe likes fantasy films.*
3. GENERATE RECOMMENDATION FOR USER: Match the user's preferences to items with similar attributes: *Recommend the latest "Dune" film to Joe*

- Ratings could be explicit (say, 1 to 5 stars) or implicit (binary-valued: selected or not).
- The item attribute model should be highly expressive (many features...) to improve accuracy.
- Generating a recommendation is equivalent to evaluating a classifier, using algorithms like k-nearest-neighbours, or decision trees.
- Focus is always on item attributes, so there is limited use of either user attributes or context.

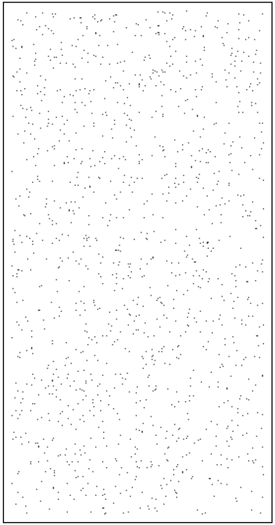
Outline

Ratings Matrices and their role



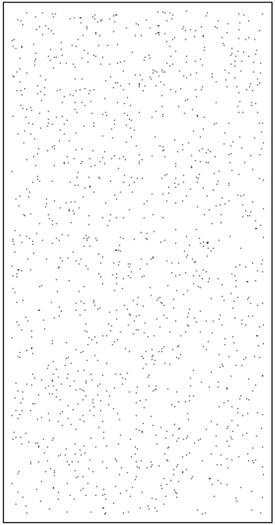
- A fragment of a ratings matrix, with a density of approximately 0.01, is illustrated

Ratings Matrices and their role



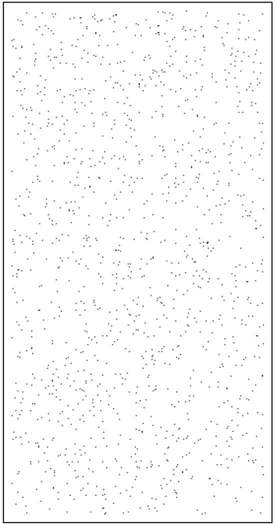
- A fragment of a ratings matrix, with a density of approximately 0.01, is illustrated
- By convention, a ratings matrix has a row per user and a column per item (but it can be transposed if $\#(\text{Users}) \ll \#(\text{Items})$).

Ratings Matrices and their role



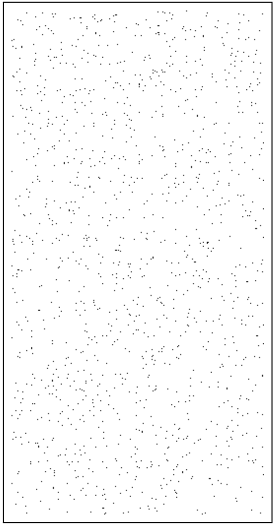
- A fragment of a ratings matrix, with a density of approximately 0.01, is illustrated
- By convention, a ratings matrix has a row per user and a column per item (but it can be transposed if $\#(\text{Users}) \ll \#(\text{Items})$).
- It is frequently sparse, because most users have rated just a few items. In this example, users have rated barely 1% of items, on average. Some will rate many items, others very few.

Ratings Matrices and their role



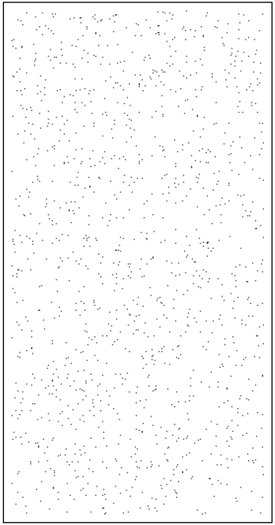
- A fragment of a ratings matrix, with a density of approximately 0.01, is illustrated
- By convention, a ratings matrix has a row per user and a column per item (but it can be transposed if $\#(\text{Users}) \ll \#(\text{Items})$).
- It is frequently sparse, because most users have rated just a few items. In this example, users have rated barely 1% of items, on average. Some will rate many items, others very few.
- Ratings can be *explicit* or *implicit*

Ratings Matrices and their role



- A fragment of a ratings matrix, with a density of approximately 0.01, is illustrated
- By convention, a ratings matrix has a row per user and a column per item (but it can be transposed if $\#(\text{Users}) \ll \#(\text{Items})$).
- It is frequently sparse, because most users have rated just a few items. In this example, users have rated barely 1% of items, on average. Some will rate many items, others very few.
- Ratings can be *explicit* or *implicit*
- In collaborative filtering, we predict the ratings that have not been assigned, using those that have.

Ratings Matrices and their role



- A fragment of a ratings matrix, with a density of approximately 0.01, is illustrated
- By convention, a ratings matrix has a row per user and a column per item (but it can be transposed if $\#(\text{Users}) \ll \#(\text{Items})$).
- It is frequently sparse, because most users have rated just a few items. In this example, users have rated barely 1% of items, on average. Some will rate many items, others very few.
- Ratings can be *explicit* or *implicit*
- In collaborative filtering, we predict the ratings that have not been assigned, using those that have.
- To recommend new item(s) to a user, pick the item(s) with the highest predicted rating.

User-User Collaborative Filtering: k-Nearest Neighbours

Motivation

Given a matrix of user-item ratings, we can view the ratings across items as attributes of each user. Example: Joe likes Action, dislikes Romance films. A direct search technique like k-nearest neighbours can identify the user neighbourhood of each user.

How it is used

Given the user neighbourhood (e.g., users similar to Joe), predict the rating (typically, the average rating where one is supplied) for all items *not rated* by the target user.

Consequences

The algorithm is relatively simple and often performs well, it can react to new data but does not scale well, has difficulties with ratings matrices that have low density, and suffers from the cold start problem.

User-user collaborative filtering

Predict a user rating by calculating the weighted sum of ratings by other users

Predicting the rating based on other user's ratings

$$\hat{r}_{u,i} = \bar{r}_u + \sum_{v \in V} \frac{(r_{v,i} - \bar{r}_v) S(u,v)}{\sum_{v \in V} S(u,v)}$$

where $\hat{r}_{u,i}$ is the *predicted* rating by user u of item i, $\bar{r}_u$ is the average rating of user u, V is the set of users (but not including u) in the neighbourhood of user u, $r_{v,i}$ is the rating made by user v on item i and $S(u,v)$ is a measure of the *similarity* between users u and v.

Important considerations are

- how to define the user-user similarity $S(u,v)$

For user u, recommend item $\tilde{\imath}$ where $\tilde{\imath} = \text{argmax}_i(\hat{r}_{u,i})$. In words: recommend the item with the highest *predicted rating* for that user.
This algorithm was the basis of *GroupLens* in 1994 and has been refined since.

User-user collaborative filtering

Predict a user rating by calculating the weighted sum of ratings by other users

Predicting the rating based on other user's ratings

$$\hat{r}_{u,i} = \bar{r}_u + \sum_{v \in V} \frac{(r_{v,i} - \bar{r}_v) S(u,v)}{\sum_{v \in V} S(u,v)}$$

where $\hat{r}_{u,i}$ is the *predicted* rating by user u of item i, $\bar{r}_u$ is the average rating of user u, V is the set of users (but not including u) in the neighbourhood of user u, $r_{v,i}$ is the rating made by user v on item i and $S(u,v)$ is a measure of the *similarity* between users u and v.

Important considerations are

- how to define the user-user similarity $S(u,v)$
- how to define the user neighbourhood V

For user u, recommend item $\tilde{\imath}$ where $\tilde{\imath} = \text{argmax}_i(\hat{r}_{u,i})$. In words: recommend the item with the highest *predicted rating* for that user.

This algorithm was the basis of *GroupLens* in 1994 and has been refined since.

Choice of user-user similarity measure $S(u, v)$

- For explicit ratings (typically on a scale $1 \ldots 5$), the Pearson correlation often works best.
- For implicit ratings (typically binary-valued: used/streamed/bought content or not), the cosine similarity is often used

Recall that the Pearson correlation coefficient, for two sets of random variables (ratings for users u and v in this instance: r_u and r_v), is

$$S(u, v) = \frac{\sum_k (r_{u,k} - \bar{r}_u)(r_{v,k} - \bar{r}_v)}{\sqrt{\sum_k (r_{u,k} - \bar{r}_u)^2 \sum_k (r_{v,k} - \bar{r}_v)^2}}$$

where k is the index of an item that was rated by user u and v and $\bar{r}_u$ is the mean rating (across all items) awarded by user u.

Choice of user neighbourhood V

Common criteria for choosing V include

- **size** needs to be large enough to remove bias of other users, small enough to make computation practical
- **similarity threshold** needs to be large enough for diversity, small enough for accuracy

In practice, a clustering algorithm can be used to group related users together. For a given user u, the other members of its cluster form V.
Each user u is an observation. Its features are the ratings that user u gave to an item.
The multiplicative inverse of the user-user similarity measure serves as the distance measure used in the clustering algorithm: $d(u,v) = \frac{1}{S(u,v)}$.

Item-Item Collaborative Filtering: Slope One

Motivation

Given a matrix of user-item ratings, the similarity of *each pair of items* can be computed, given data from all users. Sometimes those ratings have a per-user bias (some users are more difficult to please than others!) so this should be considered.

How it is used

Given this setup, and a set of users having at least n of the same items rated as the target user, compare these shared ratings and predict the ratings for items that have not yet been rated by the target user, adjusting for per-user bias.

Consequences

This algorithm often gives good results, can react to new data and can also offer recommendations to new users, but it has scalability difficulties (e.g., item-item similarity matrix could be very large).

Item-Item Collaborative Filtering: Slope One Preprocessing

Method (Slope One: Preprocessing)

for all item I_i where $i \in \{1, \ldots, n_I\}$ **do**
  for all item I_j where $j \in \{1, \ldots, n_I\} \ \{i\}$ **do**
    $d_{ij} \leftarrow 0$
    $n_r \leftarrow 0$
    for all user U_k where $k \in \{1, \ldots, n_U\}$ and ratings r_{ik} and r_{jk} are both not null **do**
      $d_{ij} \leftarrow r_{jk} - r_{ik}$
      $n_r \leftarrow n_r + 1$
    end for
    $d_{ij} \leftarrow d_{ij}/n_r$
  end for
end for

This results in an upper- or lower-triangular matrix of average rating differences between items.

Item-Item Collaborative Filtering: Slope One Recommendation

Method (Slope One: Recommendation)

for all item I_i where $i \in \{1, \ldots, n_I\}$ and r_{ik} is null **do**
  $s_{ik} \leftarrow 0$
  $n_j \leftarrow 0$
  for all item I_j where $j \in \{1, \ldots, n_I\}$ and r_{jk} is not null **do**
    $s_{ik} \leftarrow s_{ik} + r_{jk} + d_{ij}$
    $n_j \leftarrow n_j + 1$
  end for
  $r_{ik} \leftarrow s_{ik}/n_j$
end for

We have predicted the ratings r_{ik} for all items for user k where we did not have such ratings before.
Sort these predicted ratings and recommended the item with largest predicted r_{ik} to user U_k.
If more than 1 recommendation per user is needed, say N=3, recommend the items associated with the top N predicted ratings $\{r_{ik}\}$.

Slope One example: Ratings Data and Initialisation

	Item1	Item2	Item3	Item4
UserX	5.0	3.5		
UserY	2.0	5.0	4.0	2.0
UserZ	4.5	3.5	1.0	4.0

Slope One: initialise

$$d_{21} = \frac{(3.5-5)+(5-2)+(3.5-4.5)}{3} = 0.17$$

$$d_{31} = \frac{(4-2)+(1-4.5)}{2} = -0.75$$

$$d_{41} = \frac{(2-2)+(4-4.5)}{2} = -0.25$$

$$d_{32} = \frac{(4-5)+(1-3.5)}{2} = -1.75$$

$$\ldots = \ldots$$

	Item1	Item2	Item3	Item4
Item1	-			
Item2	0.17	-		
Item3	-0.75	-1.75	-	
Item4	-0.25	-1.25	0.5	-

Slope One example: Main Step

- We use UserX's ratings for {Item1, Item2}, compared with those of {UserY, UserZ}, to predict UserX's ratings for {Item3, Item4}
- Recommend Item3 or Item4 (whichever has the highest predicted rating) to UserX

UserX's predicted rating for Item3

- Using {Item1, Item3}, we have $r_{1\to 3,X} = r_{1X} + d_{31} = 5 + (-0.75) = 4.25$.
- Using {Item2, Item3}, we have $r_{2\to 3,X} = r_{2X} + d_{32} = 3.5 + (-1.75) = 1.75$
- Then $\hat{r}_{3X} = \frac{r_{1\to 3,X} + r_{2\to 3,X}}{2} = \frac{4.25+1.75}{2} = 3$

UserX's predicted rating for Item4

- Using {Item1, Item4}, we have $r_{1\to 4,X} = r_{1X} + d_{41} = 5 + (-0.25) = 4.75$.
- Using {Item2, Item4}, we have $r_{2\to 4,X} = r_{2X} + d_{42} = 3.5 + (-1.25) = 2.25$
- Then $\hat{r}_{4X} = \frac{r_{1\to 4,X} + r_{2\to 4,X}}{2} = \frac{4.75+2.25}{2} = 3.5$

So recommend Item4 to UserX because the predicted rating for Item4 (3.5) is greater than the predicted rating for Item 3 (3.0)

Overview of model-based recommender systems

- memory-based: item-item (e.g., Slope One) and user-user (e.g., KNN and “GroupLens”) collaborative filtering
- model-based: methods based on *matrix factorisations*
  - Use Singular Value Decomposition (SVD) or Nonnegative Matrix Factorisation (NMF)
  - Take the matrix of user-item ratings and write it as a product of simpler matrices with special properties
  - Alternating Least Squares (ALS) algorithm: based on NMF, solve for items, then users, then items, . . .
- NMF-based algorithms: can make recommendations while respecting user privacy
- ALS used in Spotify to recommend streams

The python Surprise library offers both memory- and model-based RS algorithms.

Matrix factorisations - some intuition

- A matrix factorisation is a generalisation of scalar factorisation, such as $24 = 6 \times 4$.

Matrix factorisations - some intuition

- A matrix factorisation is a generalisation of scalar factorisation, such as $24 = 6 \times 4$.
- Two popular factorisations are Singular Value Decomposition (SVD) and Nonnegative Matrix Factorisation (NMF)

Matrix factorisations - some intuition

- A matrix factorisation is a generalisation of scalar factorisation, such as $24 = 6 \times 4$.
- Two popular factorisations are Singular Value Decomposition (SVD) and Nonnegative Matrix Factorisation (NMF)
- Principal Components Analysis (PCA) is based on SVD, so SVD was there in the background previously.

Matrix factorisations - some intuition

- A matrix factorisation is a generalisation of scalar factorisation, such as $24 = 6 \times 4$.
- Two popular factorisations are Singular Value Decomposition (SVD) and Nonnegative Matrix Factorisation (NMF)
- Principal Components Analysis (PCA) is based on SVD, so SVD was there in the background previously.
- For a general matrix A, SVD can be written $A = USV^{\mathsf{T}}$ where U, S, V have special properties and need to be computed

Matrix factorisations - some intuition

- A matrix factorisation is a generalisation of scalar factorisation, such as $24 = 6 \times 4$.
- Two popular factorisations are Singular Value Decomposition (SVD) and Nonnegative Matrix Factorisation (NMF)
- Principal Components Analysis (PCA) is based on SVD, so SVD was there in the background previously.
- For a general matrix A, SVD can be written $A = USV^{\mathsf{T}}$ where U, S, V have special properties and need to be computed
- If A is $(m \times n)$, then U is $(m \times m)$ orthonormal (so $UU^{\mathsf{T}} = I_m$); all off-diagonal elements of S are 0; V is $(n \times n)$ orthonormal (so $VV^{\mathsf{T}} = I_n$), where I_m and I_n are $(m \times m)$ and $(n \times n)$ identity matrices.

Matrix factorisations - some intuition

- A matrix factorisation is a generalisation of scalar factorisation, such as $24 = 6 \times 4$.
- Two popular factorisations are Singular Value Decomposition (SVD) and Nonnegative Matrix Factorisation (NMF)
- Principal Components Analysis (PCA) is based on SVD, so SVD was there in the background previously.
- For a general matrix A, SVD can be written $A = USV^{\mathsf{T}}$ where U, S, V have special properties and need to be computed
- If A is $(m \times n)$, then U is $(m \times m)$ orthonormal (so $UU^{\mathsf{T}} = I_m$); all off-diagonal elements of S are 0; V is $(n \times n)$ orthonormal (so $VV^{\mathsf{T}} = I_n$), where I_m and I_n are $(m \times m)$ and $(n \times n)$ identity matrices.
- For a nonnegative matrix A, NMF can be written $A = WH$ where W and H are both nonnegative

Matrix factorisations - some intuition

- A matrix factorisation is a generalisation of scalar factorisation, such as $24 = 6 \times 4$.
- Two popular factorisations are Singular Value Decomposition (SVD) and Nonnegative Matrix Factorisation (NMF)
- Principal Components Analysis (PCA) is based on SVD, so SVD was there in the background previously.
- For a general matrix A, SVD can be written $A = USV^{\mathsf{T}}$ where U, S, V have special properties and need to be computed
- If A is $(m \times n)$, then U is $(m \times m)$ orthonormal (so $UU^{\mathsf{T}} = I_m$); all off-diagonal elements of S are 0; V is $(n \times n)$ orthonormal (so $VV^{\mathsf{T}} = I_n$), where I_m and I_n are $(m \times m)$ and $(n \times n)$ identity matrices.
- For a nonnegative matrix A, NMF can be written $A = WH$ where W and H are both nonnegative
- An advantage of nonnegative W, H factors is that they tend to be sparse, because their product WH cannot rely on cancellation to match the sparsity of A.

Matrix factorisations - some intuition

- A matrix factorisation is a generalisation of scalar factorisation, such as $24 = 6 \times 4$.
- Two popular factorisations are Singular Value Decomposition (SVD) and Nonnegative Matrix Factorisation (NMF)
- Principal Components Analysis (PCA) is based on SVD, so SVD was there in the background previously.
- For a general matrix A, SVD can be written $A = USV^\mathsf{T}$ where U, S, V have special properties and need to be computed
- If A is $(m \times n)$, then U is $(m \times m)$ orthonormal (so $UU^\mathsf{T} = I_m$); all off-diagonal elements of S are 0; V is $(n \times n)$ orthonormal (so $VV^\mathsf{T} = I_n$), where I_m and I_n are $(m \times m)$ and $(n \times n)$ identity matrices.
- For a nonnegative matrix A, NMF can be written $A = WH$ where W and H are both nonnegative
- An advantage of nonnegative W, H factors is that they tend to be sparse, because their product WH cannot rely on cancellation to match the sparsity of A.
- A ratings matrix is typically sparse so need to estimate the “missing” ratings to make a recommendation

Matrix factorisations - some intuition

- A matrix factorisation is a generalisation of scalar factorisation, such as $24 = 6 \times 4$.
- Two popular factorisations are Singular Value Decomposition (SVD) and Nonnegative Matrix Factorisation (NMF)
- Principal Components Analysis (PCA) is based on SVD, so SVD was there in the background previously.
- For a general matrix A, SVD can be written $A = USV^{\mathsf{T}}$ where U, S, V have special properties and need to be computed
- If A is $(m \times n)$, then U is $(m \times m)$ orthonormal (so $UU^{\mathsf{T}} = I_m$); all off-diagonal elements of S are 0; V is $(n \times n)$ orthonormal (so $VV^{\mathsf{T}} = I_n$), where I_m and I_n are $(m \times m)$ and $(n \times n)$ identity matrices.
- For a nonnegative matrix A, NMF can be written $A = WH$ where W and H are both nonnegative
- An advantage of nonnegative W, H factors is that they tend to be sparse, because their product WH cannot rely on cancellation to match the sparsity of A.
- A ratings matrix is typically sparse so need to estimate the "missing" ratings to make a recommendation
- Matrix factorisations "generalise" the ratings matrix A and so provide a way to predict the missing ratings.

Matrix factorisation Example

A_1	A_2	A_3	A_4
1	2	3	5
2	4	8	12
3	6	7	13

$m \times n$ matrix A ($m = 3$, $n = 4$)

All columns of A can be written as a linear combination of just two of the original columns.

$$A_1 = 1 \times A_1 + 0 \times A_3$$
$$A_2 = 2 \times A_1 + 0 \times A_3$$
$$A_3 = 0 \times A_1 + 1 \times A_3$$
$$A_4 = 2 \times A_1 + 1 \times A_3$$

Matrix factorisation Example

A_1	A_2	A_3	A_4
1	2	3	5
2	4	8	12
3	6	7	13

$m \times n$ matrix A $(m = 3,\, n = 4)$

All columns of A can be written as a linear combination of just two of the original columns.

$$A_1 = 1 \times A_1 + 0 \times A_3$$
$$A_2 = 2 \times A_1 + 0 \times A_3$$
$$A_3 = 0 \times A_1 + 1 \times A_3$$
$$A_4 = 2 \times A_1 + 1 \times A_3$$

A_1	A_3
1	3
2	8
3	7

$m \times k$ matrix P $(m = 3,\, k = 2)$

- P contains 2 “essential” columns (A_1, A_3) that *span* A, so A has rank $k = 2$.
- Since $m > n$, we combine columns; if $m < n$ we combine rows.

Matrix factorisation Example

$$\begin{array}{cccc} A_1 & A_2 & A_3 & A_4 \\ \end{array}$$

A_1	A_2	A_3	A_4
1	2	3	5
2	4	8	12
3	6	7	13

$m \times n$ matrix A ($m = 3$, $n = 4$)

All columns of A can be written as a linear combination of just two of the original columns.

$$A_1 = 1 \times A_1 + 0 \times A_3$$
$$A_2 = 2 \times A_1 + 0 \times A_3$$
$$A_3 = 0 \times A_1 + 1 \times A_3$$
$$A_4 = 2 \times A_1 + 1 \times A_3$$

A_1	A_3
1	3
2	8
3	7

$m \times k$ matrix P ($m = 3$, $k = 2$)

- P contains 2 "essential" columns (A_1, A_3) that *span* A, so A has rank $k = 2$.
- Since $m > n$, we combine columns; if $m < n$ we combine rows.

1	2	0	2
0	0	1	1

$k \times n$ matrix Q ($k = 2$, $n = 4$)

Q contains the two rows of loadings corresponding to P, so that

$$A = P \times Q.$$

This is exact because A's rank $k < \min(m, n)$. For a more general A, $P^{(k)} \times Q^{(k)} = A^{(k)} \approx A$.

Alternating Least Squares Example: Setup

- In recommender systems, k is unknown (it can be interpreted as a latent grouping, like *movie genre* or *music style*).
- The ratings matrix is typically sparse so we need to predict the missing ratings in a User row.

Alternating Least Squares Example: Setup

- In recommender systems, k is unknown (it can be interpreted as a latent grouping, like *movie genre* or *music style*).
- The ratings matrix is typically sparse so we need to predict the missing ratings in a User row.

Initial Sparse Matrix

5	3	0	1
4	0	0	1
1	1	0	5
1	0	0	4
0	1	5	4

$m \times n$ *sparse* Ratings matrix R $(m = 3, n = 4)$

Alternating Least Squares Example: Setup

- In recommender systems, k is unknown (it can be interpreted as a latent grouping, like *movie genre* or *music style*).
- The ratings matrix is typically sparse so we need to predict the missing ratings in a User row.

Initial Sparse Matrix

5	3	0	1
4	0	0	1
1	1	0	5
1	0	0	4
0	1	5	4

$m \times n$ *sparse* Ratings matrix R ($m = 3$, $n = 4$)

Overview of ALS

- There are $m = 5$ items and $n = 4$ users, unknown ratings are indicated by 0 values.
- We assume $k = 2$ and look for $P^{(k)}$ and $Q^{(k)}$ so that $P^{(k)} \times Q^{(k)} = R^{(k)} \approx R$.
- Starting with random estimates of $P^{(k)}$, we use least squares ($\sim$ regression) to find $Q^{(k)}$.
- Then holding $Q^{(k)}$ fixed, we use least squares to estimate $P^{(k)}$.
- We continue alternating as described until the RMS error on the known ratings is minimised.

Alternating Least Squares Example: Result

Estimated Full Matrix (rounded)

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

$m \times n$ *estimated* Ratings matrix R $(m = 3,\ n = 4)$

Alternating Least Squares Example: Result

Estimated Full Matrix (rounded)

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

$m \times n$ *estimated* Ratings matrix R $(m = 3,\ n = 4)$

Analysis of ALS

- Each iteration is very efficient, but many are needed if k is small and/or there are many ratings to estimate.
- The ratings alongside are rounded versions of those returned when ALS terminates.
- The prediction is user 3 would rate item 4 as 5.
- Here, all predicted ratings are on the required $1 \ldots 5$ star scale; this is not guaranteed.
- Nonnegative matrix factorisation is similar but ratings are constrained be non-negative.

Alternating Least Squares Example: Result

Estimated Full Matrix (rounded)

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

$m \times n$ *estimated* Ratings matrix R $(m = 3,\ n = 4)$

Analysis of ALS

- Each iteration is very efficient, but many are needed if k is small and/or there are many ratings to estimate.
- The ratings alongside are rounded versions of those returned when ALS terminates.
- The prediction is user 3 would rate item 4 as 5.
- Here, all predicted ratings are on the required $1 \ldots 5$ star scale; this is not guaranteed.
- Nonnegative matrix factorisation is similar but ratings are constrained be non-negative.

- In each step, we need to add regularisation, by also minimising the size of each column of P and Q.
- It is also advisable to include *bias* terms for each user and each item, by analogy with Slope-One earlier.
- Generally accurate, scales quite well, can offer cold-start recommendations (at lower accuracy).

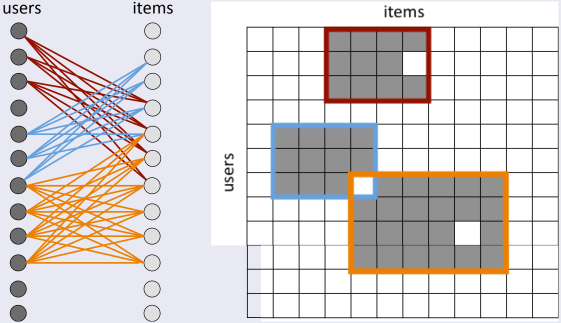
Other Techniques

SVD

- Given rating $A = USV^{\mathsf{T}}$ where rating rows are users and columns are items
- Rows of U represent Users and columns of U represent User loadings
- Similarly for Items and V^{T}
- Models relationships between Users (U) and between Items (V^{T}).
- Singular values capture strength of relationships between Users and Items
- $AA^{\mathsf{T}} \equiv US^2U^{\mathsf{T}}$ match User ratings to each other; $A^{\mathsf{T}}A = VS^2V^{\mathsf{T}}$ does same for Items
- However, calculating SVD is slow and both U and V tend to be dense

Co-clustering

- Both Users and Items are grouped together
- Sparse, overlapping groupings are formed
- Can deal with cold start problem



Summary of recommender systems

Type	Parameters	Advantages	Disadvantages
Item-based	Item similarity metric	Fast with few items; can specialise metric	More setup effort
Item-item (Slope One)		Fast at runtime, good with few items	Slow to precompute
User-User	User similarity; Neighbourhood	Fast with few users	Cold start problems
Model-based (ALS)	number of target features	Exploits sparsity	Slow to precompute

These are indicative: most algorithms have variants with different tradeoffs.
As an unsupervised technique, validation is tricky and needs feedback from users.

Outline

Performance metrics overview

- Each recommendation that is acted upon is a successful prediction (not an error)
- Need follow-up (monitoring) to measure success

- Prediction (set-valued)
  - Accuracy - Mean Absolute Error, RMS Error, etc.
  - Coverage - percentage of items that the system can recommend to other users
- Set recommendation: precision, recall, etc.
- diversity and novelty - look beyond accuracy, prevent overfitting and/or suggesting the obvious
- stability and reliability—minimise off-beat or unwelcome suggestions

Use of the metrics

Given these metrics, it is possible to compare different recommendation algorithms, or the same algorithm with different parameter settings.
You can use cross-validation, possibly within a grid-search loop, to fine-tune the parameter settings.

Performance metrics—Some definitions

Remember: recommender systems predict an ordered set of K items ($k \geq 1$) to each user

Definition 8 (Jaccard index)

Jaccard is used to measure agreement between sets (where element order is ignored). Let A_K be the *actual* top-K preferences of user i, and $\hat{A_K}$ be the *predicted* top-K preferences, then Jaccard index

$$J_K^{(i)}(A_K, \hat{A_K}) = \frac{|A_K \cap \hat{A_K}|}{|A_K \cup \hat{A_K}|}, \text{ where } 0 \leq J_K^{(i)}(A_K, \hat{A_K}) \leq 1.$$

The corresponding metric for *all* users is computed as the arithmetic mean of the metrics for *each* user i.

Definition 9 (MAP-K measure)

If order is important, we need to weight recommendation matches. Let $P_k^{(i)}, 1 \leq k \leq K$ be the *precision* of the match between Top-k predicted and Top-K actual preferences for user i. Then the *Average Precision* AP-K is the mean of $P_1^{(i)}, P_2^{(i)}, \ldots, P_K^{(i)}$ and the *Mean Average Precision for K items* is computed as the mean of the Average Precisions for each i.

MAP-K cn be used as the basis for other confusion matrix-oriented metrics like TP-K, Recall-K, etc.

Performance metrics—applied to fruit preferences

							
Order of preference	**Alice likes**	**We recommend**	**Bob likes**	**We recommend**	**Carol likes**	**We recommend**	
1		 ✓		 ✗		 ✗	
2		 ✓		 ✓		 ✓	Mean score:
3		 ✓		 ✗		 ✓	
MAP@3 score	1		0.33		0.39		0.574
Jaccard score	1		0.2		0.5		0.566

Source: https://tinyurl.com/52trnucr

Performance metrics—example calculations

Carol

- Two fruit match: pineapples and apples. So $|A \cap B| = 2$.

Jaccard score ignores recommendation order, MAP-k penalises “early” mismatches more than later ones.

Performance metrics—example calculations

Carol

- Two fruit match: pineapples and apples. So $|A \cap B| = 2$.
- Between likes and recommendations, there are 4 fruit in all, $|A \cup B| = 4$.

Jaccard score ignores recommendation order, MAP-k penalises “early” mismatches more than later ones.

Performance metrics—example calculations

Carol

- Two fruit match: pineapples and apples. So $|A \cap B| = 2$.
- Between likes and recommendations, there are 4 fruit in all, $|A \cup B| = 4$.
- So Jaccard score is $2/4 = 0.5$.

Jaccard score ignores recommendation order, MAP-k penalises “early” mismatches more than later ones.

Performance metrics—example calculations

Carol

- Two fruit match: pineapples and apples. So $|A \cap B| = 2$.
- Between likes and recommendations, there are 4 fruit in all, $|A \cup B| = 4$.
- So Jaccard score is $2/4 = 0.5$.
- Recommendation 1 (Lemons) does not match, so precision $x_1 = \frac{0}{1} = 0$.

Jaccard score ignores recommendation order, MAP-k penalises “early” mismatches more than later ones.

Performance metrics—example calculations

Carol

- Two fruit match: pineapples and apples. So $|A \cap B| = 2$.
- Between likes and recommendations, there are 4 fruit in all, $|A \cup B| = 4$.
- So Jaccard score is $2/4 = 0.5$.
- Recommendation 1 (Lemons) does not match, so precision $x_1 = \frac{0}{1} = 0$.
- Recommendations 1,2 (Lemons, Pineapples) has 1 match from 2, so $x_2 = \frac{1}{2} = 0.5$.

Jaccard score ignores recommendation order, MAP-k penalises "early" mismatches more than later ones.

Performance metrics—example calculations

Carol

- Two fruit match: pineapples and apples. So $|A \cap B| = 2$.
- Between likes and recommendations, there are 4 fruit in all, $|A \cup B| = 4$.
- So Jaccard score is $2/4 = 0.5$.
- Recommendation 1 (Lemons) does not match, so precision $x_1 = \frac{0}{1} = 0$.
- Recommendations 1,2 (Lemons, Pineapples) has 1 match from 2, so $x_2 = \frac{1}{2} = 0.5$.
- Recommendations 1,2,3 (Lemons, Pineapples, Apples) has 2 matches from 3, so $x_3 = \frac{2}{3} = 0.66$.

Jaccard score ignores recommendation order, MAP-k penalises "early" mismatches more than later ones.

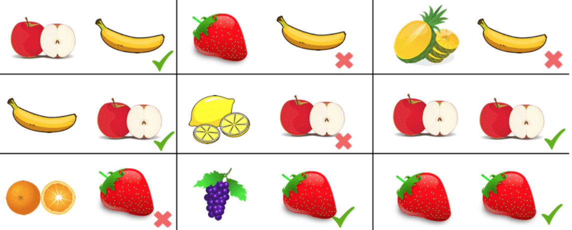
Performance metrics—example calculations

Carol

- Two fruit match: pineapples and apples. So $|A \cap B| = 2$.
- Between likes and recommendations, there are 4 fruit in all, $|A \cup B| = 4$.
- So Jaccard score is $2/4 = 0.5$.
- Recommendation 1 (Lemons) does not match, so precision $x_1 = \frac{0}{1} = 0$.
- Recommendations 1,2 (Lemons, Pineapples) has 1 match from 2, so $x_2 = \frac{1}{2} = 0.5$.
- Recommendations 1,2,3 (Lemons, Pineapples, Apples) has 2 matches from 3, so $x_3 = \frac{2}{3} = 0.66$.
- MAP-k ($k = 3$) is the average of the $\{x_i\}$, which is $0, 39$ (approximately).

Jaccard score ignores recommendation order, MAP-k penalises “early” mismatches more than later ones.

Performance metrics—base (not personalised) model for comparison



Order of preference	Alice likes	We recommend	Bob likes	We recommend	Carol likes	We recommend	Mean score:
1							
2							
3							
MAP@3 score	0.67		0.11		0.39		0.39
Jaccard score	0.5		0.2		0.5		0.40

> Base model generally did not perform as well as CF model, but can even do better sometimes. . .

The `scikit-surprise` library

Spark and AWS each offer libraries of recommendation engines, but we use `scikit-surprise` here

imports.py
```python
from surprise import SVD
from surprise import Dataset
from surprise.model_selection import cross_validate
```

loadData.py
```python
# Load the movielens-100k dataset (download it if needed).
data = Dataset.load_builtin('ml-100k')
```

selectSVDalgorithm.py
```python
# Use the SVD algorithm to estimate missing ratings
algo = SVD()
```

applyCrossValidation.py
```python
# Run 5-fold cross-validation and print results.
cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

Source: http://surpriselib.com/

Outline

Approaches used by recommender systems

Method	Advantages	Disadvantages
Most Popular / Latest Items	Simple; unaffected by cold start	Poor accuracy; not personalised

Approaches used by recommender systems

Method	Advantages	Disadvantages
Most Popular / Latest Items	Simple; unaffected by cold start	Poor accuracy; not personalised
ARM: high affinity item pairs (Amazon)	Unaffected by cold start	Unsuited to large itemsets: no user preferences

Approaches used by recommender systems

Method	Advantages	Disadvantages
Most Popular / Latest Items	Simple; unaffected by cold start	Poor accuracy; not personalised
ARM: high affinity item pairs (Amazon)	Unaffected by cold start	Unsuited to large itemsets: no user preferences
Content Filtering: item attributes $\rightarrow$ user classifier	Very flexible; multi-purpose user model	Needs rich item attribute model for accuracy

Approaches used by recommender systems

Method	Advantages	Disadvantages
Most Popular / Latest Items	Simple; unaffected by cold start	Poor accuracy; not personalised
ARM: high affinity item pairs (Amazon)	Unaffected by cold start	Unsuited to large itemsets: no user preferences
Content Filtering: item attributes $\rightarrow$ user classifier	Very flexible; multi-purpose user model	Needs rich item attribute model for accuracy
Collaborative Filtering: user-user similarity; vector factorisation	Item attributes not needed; can have high accuracy	Needs many ratings, not good for long tail

Approaches used by recommender systems

Method	Advantages	Disadvantages
Most Popular / Latest Items	Simple; unaffected by cold start	Poor accuracy; not personalised
ARM: high affinity item pairs (Amazon)	Unaffected by cold start	Unsuited to large itemsets: no user preferences
Content Filtering: item attributes $\rightarrow$ user classifier	Very flexible; multi-purpose user model	Needs rich item attribute model for accuracy
Collaborative Filtering: user-user similarity; vector factorisation	Item attributes not needed; can have high accuracy	Needs many ratings, not good for long tail
Hybrid Models: Expert knowledge or Combined	Lots of potential for tuning	Complex, need good way to combine suggestions

Approaches used by recommender systems

Method	Advantages	Disadvantages
Most Popular / Latest Items	Simple; unaffected by cold start	Poor accuracy; not personalised
ARM: high affinity item pairs (Amazon)	Unaffected by cold start	Unsuited to large itemsets: no user preferences
Content Filtering: item attributes $\rightarrow$ user classifier	Very flexible; multi-purpose user model	Needs rich item attribute model for accuracy
Collaborative Filtering: user-user similarity; vector factorisation	Item attributes not needed; can have high accuracy	Needs many ratings, not good for long tail
Hybrid Models: Expert knowledge or Combined	Lots of potential for tuning	Complex, need good way to combine suggestions

Among the machine learning techniques used are classification (particularly for Content Filtering)

Summary

- Recommender systems are widely used and seen as directly benefiting business

Summary

- Recommender systems are widely used and seen as directly benefiting business
- Lively research topic, mostly relating to improving accuracy-diversity tradeoff, hybrid techniques, moving beyond ratings, etc.

Summary

- Recommender systems are widely used and seen as directly benefiting business
- Lively research topic, mostly relating to improving accuracy-diversity tradeoff, hybrid techniques, moving beyond ratings, etc.
- Collaborative filtering (item-item (Slope One) and user-user (weighted ratings)) models are used everywhere from Amazon to Netflix to Facebook

Summary

- Recommender systems are widely used and seen as directly benefiting business
- Lively research topic, mostly relating to improving accuracy-diversity tradeoff, hybrid techniques, moving beyond ratings, etc.
- Collaborative filtering (item-item (Slope One) and user-user (weighted ratings)) models are used everywhere from Amazon to Netflix to Facebook
- For a long time, RS were seen as company secrets, but the rise of open source libraries like Surprise library means that anybody can join in!

Summary

- Recommender systems are widely used and seen as directly benefiting business
- Lively research topic, mostly relating to improving accuracy-diversity tradeoff, hybrid techniques, moving beyond ratings, etc.
- Collaborative filtering (item-item (Slope One) and user-user (weighted ratings)) models are used everywhere from Amazon to Netflix to Facebook
- For a long time, RS were seen as company secrets, but the rise of open source libraries like Surprise library means that anybody can join in!
- RS are examples of online (real time) machine learning so generally algorithms are deployed on Big Data platforms (e.g., hadoop, mahout, Spark)

Summary

- Recommender systems are widely used and seen as directly benefiting business
- Lively research topic, mostly relating to improving accuracy-diversity tradeoff, hybrid techniques, moving beyond ratings, etc.
- Collaborative filtering (item-item (Slope One) and user-user (weighted ratings)) models are used everywhere from Amazon to Netflix to Facebook
- For a long time, RS were seen as company secrets, but the rise of open source libraries like Surprise library means that anybody can join in!
- RS are examples of online (real time) machine learning so generally algorithms are deployed on Big Data platforms (e.g., hadoop, mahout, Spark)
- Recommender systems have become so effective that there are social/ethical concerns about their use:

Summary

- Recommender systems are widely used and seen as directly benefiting business
- Lively research topic, mostly relating to improving accuracy-diversity tradeoff, hybrid techniques, moving beyond ratings, etc.
- Collaborative filtering (item-item (Slope One) and user-user (weighted ratings)) models are used everywhere from Amazon to Netflix to Facebook
- For a long time, RS were seen as company secrets, but the rise of open source libraries like Surprise library means that anybody can join in!
- RS are examples of online (real time) machine learning so generally algorithms are deployed on Big Data platforms (e.g., hadoop, mahout, Spark)
- Recommender systems have become so effective that there are social/ethical concerns about their use:
    - social media offers hyper-personalised echo chambers, increasing intolerance and reducing social cohesion

Summary

- Recommender systems are widely used and seen as directly benefiting business
- Lively research topic, mostly relating to improving accuracy-diversity tradeoff, hybrid techniques, moving beyond ratings, etc.
- Collaborative filtering (item-item (Slope One) and user-user (weighted ratings)) models are used everywhere from Amazon to Netflix to Facebook
- For a long time, RS were seen as company secrets, but the rise of open source libraries like Surprise library means that anybody can join in!
- RS are examples of online (real time) machine learning so generally algorithms are deployed on Big Data platforms (e.g., hadoop, mahout, Spark)
- Recommender systems have become so effective that there are social/ethical concerns about their use:
    - social media offers hyper-personalised echo chambers, increasing intolerance and reducing social cohesion
    - recommendations can lead to unsuitable content, harmful to User and society more generally

Summary

- Recommender systems are widely used and seen as directly benefiting business
- Lively research topic, mostly relating to improving accuracy-diversity tradeoff, hybrid techniques, moving beyond ratings, etc.
- Collaborative filtering (item-item (Slope One) and user-user (weighted ratings)) models are used everywhere from Amazon to Netflix to Facebook
- For a long time, RS were seen as company secrets, but the rise of open source libraries like Surprise library means that anybody can join in!
- RS are examples of online (real time) machine learning so generally algorithms are deployed on Big Data platforms (e.g., hadoop, mahout, Spark)
- Recommender systems have become so effective that there are social/ethical concerns about their use:
  - social media offers hyper-personalised echo chambers, increasing intolerance and reducing social cohesion
  - recommendations can lead to unsuitable content, harmful to User and society more generally
  - passive consumption vs active search

References