THE UNIVERSITY *of* EDINBURGH

informatics

Applied Machine Learning (AML)

Recommender Systems

Oisin Mac Aodha • Siddharth N.

Recommender Systems

Recommender Systems

- Recommender systems are systems which attempt to recommend **items** to **users**.
  - e.g. movies, books, online ads, restaurants, etc.

Recommender Systems

- Recommender systems are systems which attempt to recommend **items** to **users**.
  - e.g. movies, books, online ads, restaurants, etc.
- This is achieved by using information users provide about other items.
  - e.g. their past viewing/ purchasing behaviour, which movies they rated high or low, which ads they clicked on, etc.

Recommender Systems

- Recommender systems are systems which attempt to recommend **items** to **users**.
  - e.g. movies, books, online ads, restaurants, etc.
- This is achieved by using information users provide about other items.
  - e.g. their past viewing/ purchasing behaviour, which movies they rated high or low, which ads they clicked on, etc.
- They are commonly found on websites or services that attempt to personalise content for users.

The Recommendation Task

- The central task we would like to solve is how to predict a user's rating (i.e. score) for the items they have *not* yet seen.

The Recommendation Task

- The central task we would like to solve is how to predict a user's rating (i.e. score) for the items they have *not* yet seen.
- We assume we have access to rating data from other users, but this is likely to be very sparse, i.e. the average user only rates a very small number of items.

Sources of Rating Data

Explicit Feedback

- Ask users to rate items, e.g. rating from 1 to 5, or like (+1) versus dislike (-1), etc.
- Can be hard to get this information in practice.

Sources of Rating Data

Explicit Feedback

- Ask users to rate items, e.g. rating from 1 to 5, or like (+1) versus dislike (-1), etc.
- Can be hard to get this information in practice.

Implicit Feedback

- Extracted from user actions, e.g. click on a video, watch until the end, ...
- Weaker form of supervision.

Sources of Rating Data

Explicit Feedback

- Ask users to rate items, e.g. rating from 1 to 5, or like (+1) versus dislike (-1), etc.
- Can be hard to get this information in practice.

Implicit Feedback

- Extracted from user actions, e.g. click on a video, watch until the end, ...
- Weaker form of supervision.

Assumption

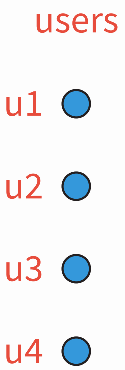
We will assume that we have explicit feedback, and that any missing ratings are **missing at random**.

User Rating Data - Example

- Suppose we observed data from **four** different users: $\mathcal{U} = \{u1, u2, u3, u4\}$.

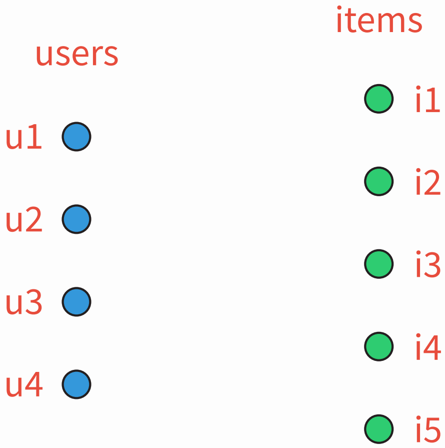
User Rating Data - Example

- Suppose we observed data from **four** different users: $\mathcal{U} = \{u1, u2, u3, u4\}$.



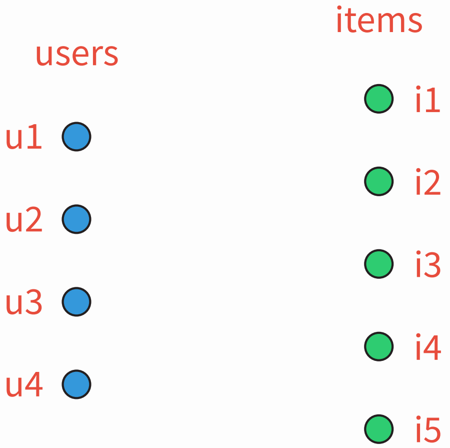
User Rating Data - Example

- Suppose we observed data from **four** different users: $\mathcal{U} = \{u1, u2, u3, u4\}$.
- Who provided ratings for **five** different items: $\mathcal{I} = \{i1, i2, i3, i4, i5\}$.



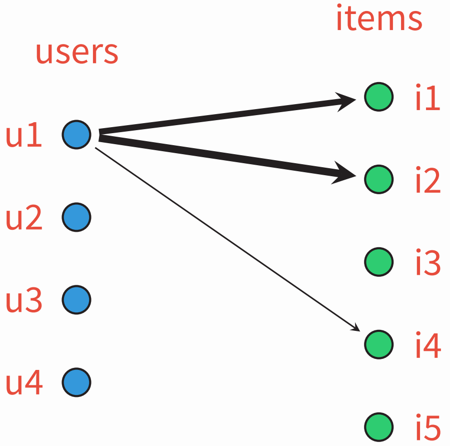
User Rating Data - Example

- Suppose we observed data from **four** different users: $\mathcal{U} = \{u1, u2, u3, u4\}$.
- Who provided ratings for **five** different items: $\mathcal{I} = \{i1, i2, i3, i4, i5\}$.
- Each rating can be one of five possible values: {1, 2, 3, 4, 5}.



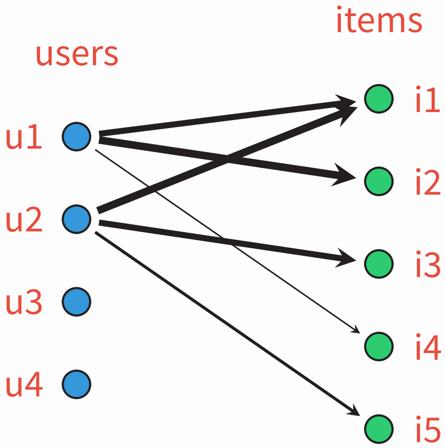
User Rating Data - Example

- Suppose we observed data from **four** different users: $\mathcal{U} = \{u1, u2, u3, u4\}$.
- Who provided ratings for **five** different items: $\mathcal{I} = \{i1, i2, i3, i4, i5\}$.
- Each rating can be one of five possible values: {1, 2, 3, 4, 5}.



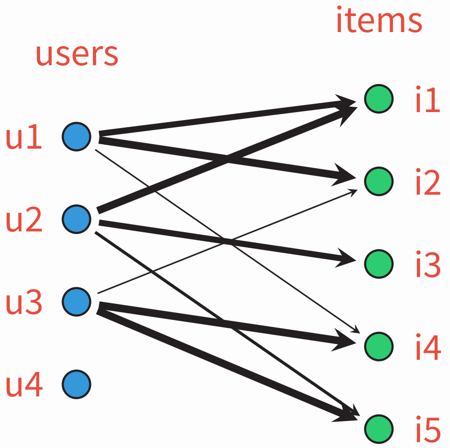
User Rating Data - Example

- Suppose we observed data from **four** different users: $\mathcal{U} = \{u1, u2, u3, u4\}$.
- Who provided ratings for **five** different items: $\mathcal{I} = \{i1, i2, i3, i4, i5\}$.
- Each rating can be one of five possible values: {1, 2, 3, 4, 5}.



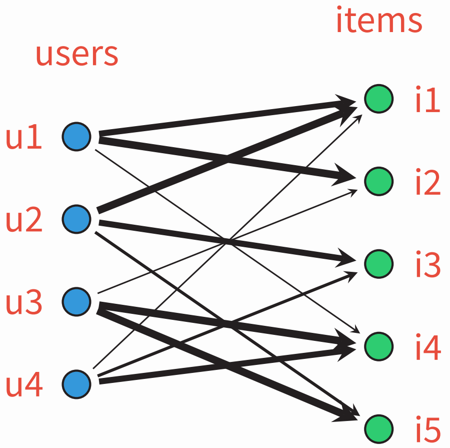
User Rating Data - Example

- Suppose we observed data from **four** different users: $\mathcal{U} = \{u1, u2, u3, u4\}$.
- Who provided ratings for **five** different items: $\mathcal{I} = \{i1, i2, i3, i4, i5\}$.
- Each rating can be one of five possible values: {1, 2, 3, 4, 5}.



User Rating Data - Example

- Suppose we observed data from **four** different users: $\mathcal{U} = \{u1, u2, u3, u4\}$.
- Who provided ratings for **five** different items: $\mathcal{I} = \{i1, i2, i3, i4, i5\}$.
- Each rating can be one of five possible values: {1, 2, 3, 4, 5}.



User Rating Data - Example

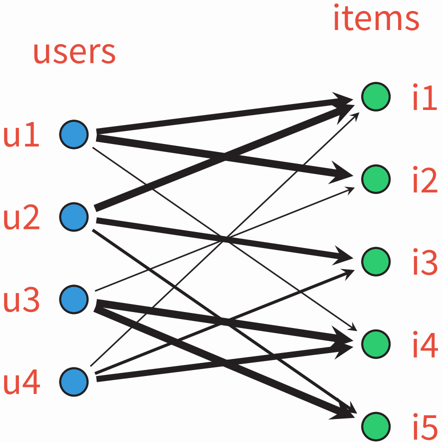
- We can represent user ratings this as a matrix Y of size $|\mathcal{U}| \times |\mathcal{I}|$, i.e. `num_users` $\times$ `num_items`.

items

users

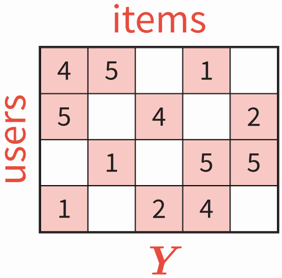
4	5		1	
5		4		2
	1		5	5
1		2	4	

Y



Missing Data

- Our ratings matrix can have many missing entries.



- Most users do not rate many items.
- As a result the data we observe can be very **sparse**.

Predicting Missing Ratings

- How can we predict missing ratings?

$$
\begin{array}{cc}
 & \text{items} \\
\text{users} & \begin{array}{|c|c|c|c|c|}
\hline
4 & 5 & & 1 & \\ \hline
5 & & 4 & & 2 \\ \hline
 & 1 & & 5 & 5 \\ \hline
1 & & 2 & 4 & \\ \hline
\end{array} \\
 & \boldsymbol{Y}
\end{array}
$$

Predicting Missing Ratings

- How can we predict missing ratings?

items

users

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

$\boldsymbol{Y}$

- Suppose we have a user u4 who has provided the following ratings:

  i1: 1 star

  i3: 2 stars

  i4: 4 stars

Predicting Missing Ratings

- How can we predict missing ratings?

items

users					
	4	5		1	
	5		4		2
		1		5	5
	1		2	4	

$\boldsymbol{Y}$

- Suppose we have a user u4 who has provided the following ratings:

  i1: 1 star

  i3: 2 stars

  i4: 4 stars

- We would like to predict how they would rate items 2 and 5.

Predicting Missing Ratings

- How can we predict missing ratings?

items

users

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

$\boldsymbol{Y}$

- Suppose we have a user u4 who has provided the following ratings:
  i1: 1 star
  i3: 2 stars
  i4: 4 stars
- We would like to predict how they would rate items 2 and 5.
- We refer to the estimated ratings as $\hat{Y}_{ui}$.

Predicting Missing Ratings - Average Rating

- One simple approach is to report the **average** rating for an item.

items

users

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

Y

Predicting Missing Ratings - Average Rating

- One simple approach is to report the **average** rating for an item.

items

users

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

$\boldsymbol{Y}$

- Average rating:

$$\hat{Y}_{ui} = k \sum_{\substack{u'=1 \\ Y_{u'i} \neq ?}}^{|\mathcal{U}|} Y_{u'i}$$

Predicting Missing Ratings - Average Rating

- One simple approach is to report the **average** rating for an item.

items

users

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

$\boldsymbol{Y}$

- Average rating:

$$\hat{Y}_{ui} = k \sum_{\substack{u'=1 \\ Y_{u'i} \neq ?}}^{|\mathcal{U}|} Y_{u'i}$$

- k is a normalisation factor:

$$k = 1 / \sum_{\substack{u'=1 \\ Y_{u'i} \neq ?}}^{|\mathcal{U}|} 1$$

Predicting Missing Ratings - Average Rating

- One simple approach is to report the **average** rating for an item.

items

users

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

$\boldsymbol{Y}$

- Average rating:

$$\hat{Y}_{ui} = k \sum_{\substack{u'=1 \\ Y_{u'i} \neq ?}}^{|\mathcal{U}|} Y_{u'i}$$

- k is a normalisation factor:

$$k = 1 / \sum_{\substack{u'=1 \\ Y_{u'i} \neq ?}}^{|\mathcal{U}|} 1$$

- This approach is simple, but it fails to capture differences between users.

Collaborative Filtering

- Another approach is to use rating information from **other** users.
- We can find **similar** users and then make predictions for our user of interest based their similarity to these existing users.

Collaborative Filtering

- Another approach is to use rating information from **other** users.
- We can find **similar** users and then make predictions for our user of interest based their similarity to these existing users.

Assumption

- The underlying assumption is that if user A and B rate items they have both seen similarly, then user A is more likely to rate items they have not seen similar to how user B would.

Predicting Missing Ratings - Weighted Average

- We can **weight** the ratings score based on the **similarity** between the users.

items

users

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

$\boldsymbol{Y}$

Predicting Missing Ratings - Weighted Average

- We can **weight** the ratings score based on the **similarity** between the users.

items

users

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

$\boldsymbol{Y}$

- Weighted average:

$$\hat{Y}_{ui} = k \sum_{\substack{u'=1 \\ Y_{u'i} \neq ?}}^{|\mathcal{U}|} \mathrm{sim}(u, u')\, Y_{u'i}$$

Predicting Missing Ratings - Weighted Average

- We can **weight** the ratings score based on the **similarity** between the users.

items

users

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

$\boldsymbol{Y}$

- Weighted average:

$$\hat{Y}_{ui} = k \sum_{\substack{u'=1 \\ Y_{u'i} \neq ?}}^{|\mathcal{U}|} \text{sim}(u, u')\, Y_{u'i}$$

- Here sim() is a similarity measure between a pair of users.

Predicting Missing Ratings - Weighted Average

- We can **weight** the ratings score based on the **similarity** between the users.

items

users					
	4	5		1	
	5		4		2
		1		5	5
	1		2	4	

$\boldsymbol{Y}$

- Weighted average:

$$\hat{Y}_{ui} = k \sum_{\substack{u'=1 \\ Y_{u'i} \neq ?}}^{|\mathcal{U}|} \mathsf{sim}(u, u')\, Y_{u'i}$$

- Here sim() is a similarity measure between a pair of users.
- k is a normalisation factor:

$$k = 1 / \sum_{u'=1}^{|\mathcal{U}|} \mathsf{sim}(u, u')$$

Limitations

- We need to define an appropriate similarity measure.
- Computing reliable similarity can be difficult with very sparse data.
- We need to store the entire dataset in memory at inference time.

Matrix Factorisation

Matrix Factorisation

- The recommendation task can be viewed as one of **matrix completion**.
- Here the goal is to predict all the missing entries of $\boldsymbol{Y}$ given the subset of user rating pairs $(u, i) \in \mathcal{S}$ we have observed (i.e. $Y_{ui} \neq ?$)

Matrix Factorisation

- The recommendation task can be viewed as one of **matrix completion**.
- Here the goal is to predict all the missing entries of $\boldsymbol{Y}$ given the subset of user rating pairs $(u, i) \in \mathcal{S}$ we have observed (i.e. $Y_{ui} \neq ?$)

$$\mathcal{L}(\hat{\boldsymbol{Y}}) = \sum_{(u,i)\in\mathcal{S}} (Y_{ui} - \hat{Y}_{ui})^2$$

$$= ||\boldsymbol{Y} - \hat{\boldsymbol{Y}}||^2$$

Low Rank Assumption

- The previous problem is under specified i.e. there are infinitely many ways of filing in the missing entries of Y.

Low Rank Assumption

- The previous problem is under specified i.e. there are infinitely many ways of filing in the missing entries of Y.
- We need to add some constraints.

Low Rank Assumption

- The previous problem is under specified i.e. there are infinitely many ways of filing in the missing entries of Y.
- We need to add some constraints.

Assumption

- Assume that Y is a low rank matrix.

Low Rank Assumption

- The previous problem is under specified i.e. there are infinitely many ways of filing in the missing entries of $\boldsymbol{Y}$.
- We need to add some constraints.

Assumption

- Assume that $\boldsymbol{Y}$ is a low rank matrix.
- We can rewrite it as $\hat{\boldsymbol{Y}} = \boldsymbol{U}\boldsymbol{V}^\intercal \approx \boldsymbol{Y}$.

Low Rank Assumption

- The previous problem is under specified i.e. there are infinitely many ways of filing in the missing entries of $\boldsymbol{Y}$.
- We need to add some constraints.

Assumption

- Assume that $\boldsymbol{Y}$ is a low rank matrix.
- We can rewrite it as $\hat{\boldsymbol{Y}} = \boldsymbol{U}\boldsymbol{V}^{\top} \approx \boldsymbol{Y}$.
- Here $\boldsymbol{U}$ is a $|\mathcal{U}| \times K$ matrix and $\boldsymbol{V}$ is a $|\mathcal{I}| \times K$ matrix.

Matrix Factorization

- Here we assume that our data can be represented as the product of two matrices $\hat{\boldsymbol{Y}} = \boldsymbol{U}\boldsymbol{V}^{\top} \approx \boldsymbol{Y}$.

items

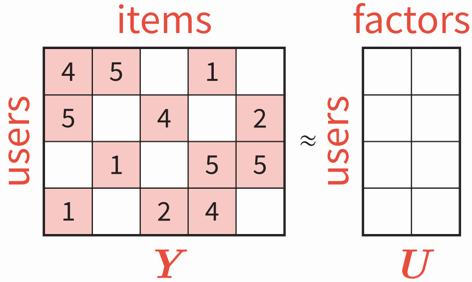
users

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

$\boldsymbol{Y}$

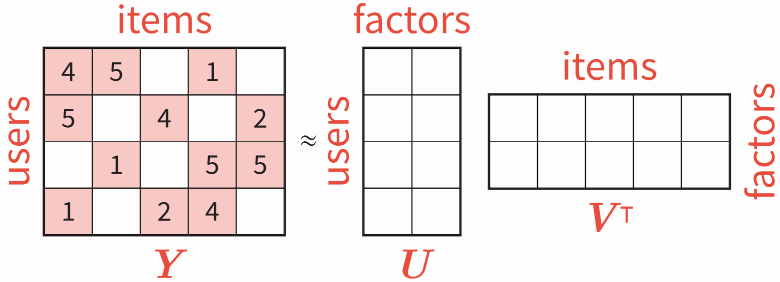
Matrix Factorization

- Here we assume that our data can be represented as the product of two matrices
  $\hat{Y} = UV^{\top} \approx Y$.



Matrix Factorization

- Here we assume that our data can be represented as the product of two matrices $\hat{\boldsymbol{Y}} = \boldsymbol{U}\boldsymbol{V}^{\top} \approx \boldsymbol{Y}$.



Matrix Factorisation - Prediction

- We assume that our data can be represented as the product of a user matrix $\boldsymbol{U}$ and an item matrix $\boldsymbol{V}$, i.e. $\hat{\boldsymbol{Y}} = \boldsymbol{U}\boldsymbol{V}^{\intercal} \approx \boldsymbol{Y}$.

Matrix Factorisation - Prediction

- We assume that our data can be represented as the product of a user matrix $\boldsymbol{U}$ and an item matrix $\boldsymbol{V}$, i.e. $\hat{\boldsymbol{Y}} = \boldsymbol{U}\boldsymbol{V}^\top \approx \boldsymbol{Y}$.
- Each row $\boldsymbol{u}_u$ of $\boldsymbol{U}$ represents a different **user**.

Matrix Factorisation - Prediction

- We assume that our data can be represented as the product of a user matrix $\boldsymbol{U}$ and an item matrix $\boldsymbol{V}$, i.e. $\hat{\boldsymbol{Y}} = \boldsymbol{U}\boldsymbol{V}^\top \approx \boldsymbol{Y}$.
- Each row $\boldsymbol{u}_u$ of $\boldsymbol{U}$ represents a different **user**.
- Similarly, each row $\boldsymbol{v}_i$ of $\boldsymbol{V}$ represents a different **item**.

Matrix Factorisation - Prediction

- We assume that our data can be represented as the product of a user matrix $\boldsymbol{U}$ and an item matrix $\boldsymbol{V}$, i.e. $\hat{\boldsymbol{Y}} = \boldsymbol{U}\boldsymbol{V}^\intercal \approx \boldsymbol{Y}$.
- Each row $\boldsymbol{u}_u$ of $\boldsymbol{U}$ represents a different **user**.
- Similarly, each row $\boldsymbol{v}_i$ of $\boldsymbol{V}$ represents a different **item**.
- To predict a missing entry we simply evaluate $\hat{Y}_{ui} = \boldsymbol{u}_u^\intercal \boldsymbol{v}_i$.

Model Training With SGD

- We can estimate the model weights $\boldsymbol{\theta} = \{\boldsymbol{U}, \boldsymbol{V}\}$ using stochastic gradient descent (SGD).

Model Training With SGD

- We can estimate the model weights $\boldsymbol{\theta} = \{\boldsymbol{U}, \boldsymbol{V}\}$ using stochastic gradient descent (SGD).
- As there is missing data, we only have a subset of user rating pairs $(u, i) \in \mathcal{S}$, where $Y_{ui} \neq ?$.

Model Training With SGD

- We can estimate the model weights $\boldsymbol{\theta} = \{\boldsymbol{U}, \boldsymbol{V}\}$ using stochastic gradient descent (SGD).
- As there is missing data, we only have a subset of user rating pairs $(u, i) \in \mathcal{S}$, where $Y_{ui} \neq ?$.
- Our loss function is

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{(u,i)\in\mathcal{S}} (Y_{ui} - \boldsymbol{u}_u^\top \boldsymbol{v}_i)^2.$$

Model Training With SGD

- We can estimate the model weights $\boldsymbol{\theta} = \{\boldsymbol{U}, \boldsymbol{V}\}$ using stochastic gradient descent (SGD).
- As there is missing data, we only have a subset of user rating pairs $(u, i) \in \mathcal{S}$, where $Y_{ui} \neq ?$.
- Our loss function is

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{(u,i) \in \mathcal{S}} (Y_{ui} - \boldsymbol{u}_u^\mathsf{T} \boldsymbol{v}_i)^2.$$

Require: step size η, number steps N, valid indices $\mathcal{S}$

$\boldsymbol{U} \leftarrow$ initialisation

$\boldsymbol{V} \leftarrow$ initialisation

for $n \leftarrow 1$ to N **do**

$\quad (u, i) \in \mathcal{S}$

$\quad e_{ui} = (Y_{ui} - \boldsymbol{u}_u^\mathsf{T} \boldsymbol{v}_i)$

$\quad \boldsymbol{u}_u \leftarrow \boldsymbol{u}_u + 2\eta(e_{ui} \boldsymbol{v}_i)$

$\quad \boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + 2\eta(e_{ui} \boldsymbol{u}_u)$

return $\boldsymbol{U}, \boldsymbol{V}$

Regularisation

- We can also regularise our weights so that they do not become too large.

Regularisation

- We can also regularise our weights so that they do not become too large.
- Our loss function then becomes

$$\sum_{(u,i)\in\mathcal{S}} (Y_{ui} - \boldsymbol{u}_u^\mathsf{T} \boldsymbol{v}_i)^2 + \lambda \left(\sum_u ||\boldsymbol{u}_u||^2 + \sum_i ||\boldsymbol{v}_i||^2\right).$$

Regularisation

- We can also regularise our weights so that they do not become too large.
- Our loss function then becomes

$$\sum_{(u,i)\in\mathcal{S}}(Y_{ui}-\boldsymbol{u}_u^\top\boldsymbol{v}_i)^2+\lambda(\sum_u||\boldsymbol{u}_u||^2+\sum_i||\boldsymbol{v}_i||^2).$$

Require: step size η, number steps N, regularisation strength λ, valid indices $\mathcal{S}$

1: $\boldsymbol{U} \leftarrow$ initialisation
2: $\boldsymbol{V} \leftarrow$ initialisation
3: **for** $n \leftarrow 1$ to N **do**
4: $\quad (u, i) \in \mathcal{S}$
5: $\quad e_{ui} = (Y_{ui} - \boldsymbol{u}_u^\top \boldsymbol{v}_i)$
6: $\quad \boldsymbol{u}_u \leftarrow \boldsymbol{u}_u + 2\eta(e_{ui}\boldsymbol{v}_i - \lambda\boldsymbol{u}_u)$
7: $\quad \boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + 2\eta(e_{ui}\boldsymbol{u}_u - \lambda\boldsymbol{v}_i)$
8: **return** $\boldsymbol{U}, \boldsymbol{V}$

Matrix Factorization Example

Example Overview

- Here we will walk through an example of applying matrix factorisation to predict unobserved user ratings for a small toy problem.

Example Overview

- Here we will walk through an example of applying matrix factorisation to predict unobserved user ratings for a small toy problem.
- We will use SGD to estimate the model weights $\boldsymbol{\theta} = \{\boldsymbol{U}, \boldsymbol{V}\}$.

Example Overview

- Here we will walk through an example of applying matrix factorisation to predict unobserved user ratings for a small toy problem.
- We will use SGD to estimate the model weights $\boldsymbol{\theta} = \{\boldsymbol{U}, \boldsymbol{V}\}$.
- Our training loss is the squared error with additional weight regularisation

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{(u,i)\in\mathcal{S}} (Y_{ui} - \boldsymbol{u}_u^\intercal \boldsymbol{v}_i)^2 + \lambda(\sum_u ||\boldsymbol{u}_u||^2 + \sum_i ||\boldsymbol{v}_i||^2).$$

Matrix Factorisation - Example

- In this example we have 6 users and 5 items, where 13 of the ratings are missing.

items

users

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

$\boldsymbol{Y}$

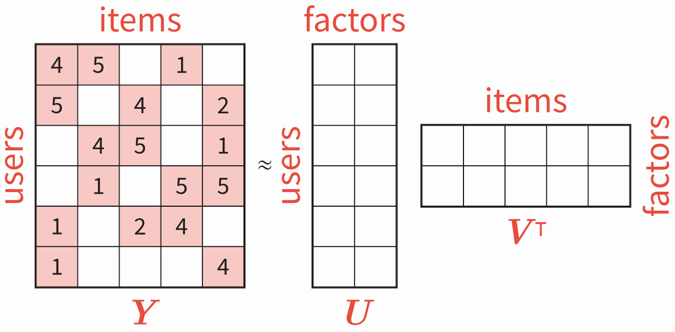
Matrix Factorisation - Example

- In this example we have 6 users and 5 items, where 13 of the ratings are missing.
- Our model will use 2 factors i.e. $K = 2$.

$$
\begin{array}{cc}
 & \text{items} \\
\text{users} & \begin{bmatrix} 4 & 5 & & 1 & \\ 5 & & 4 & & 2 \\ & 4 & 5 & & 1 \\ & 1 & & 5 & 5 \\ 1 & & 2 & 4 & \\ 1 & & & & 4 \end{bmatrix} \\
 & \boldsymbol{Y}
\end{array}
\approx
\begin{array}{cc}
 & \text{factors} \\
\text{users} & \begin{bmatrix} & \\ & \\ & \\ & \\ & \\ & \end{bmatrix} \\
 & \boldsymbol{U}
\end{array}
$$

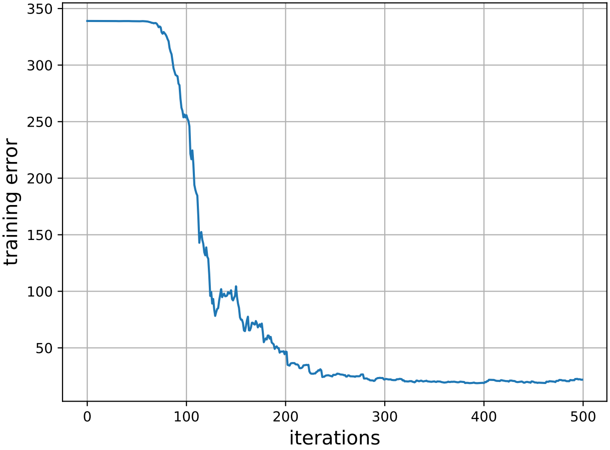
Matrix Factorisation - Example

- In this example we have 6 users and 5 items, where 13 of the ratings are missing.
- Our model will use 2 factors i.e. $K = 2$.



Training Error

- Here we plot the error $||Y - UV^\top||^2$ obtained during training.



Matrix Factorisation - Example

- Below we see the learned factors after running SGD.

$$
\underbrace{\text{users}\overset{\text{items}}{\begin{bmatrix} 4 & 5 & & 1 & \\ 5 & & 4 & & 2 \\ & 4 & 5 & & 1 \\ & 1 & & 5 & 5 \\ 1 & & 2 & 4 & \\ 1 & & & & 4 \end{bmatrix}}}_{\boldsymbol{Y}}
\approx
\underbrace{\text{users}\overset{\text{factors}}{\begin{bmatrix} 2.0 & -0.4 \\ 1.7 & -1.3 \\ 1.6 & -1.0 \\ -0.1 & -2.2 \\ -0.4 & -1.7 \\ -0.5 & -1.6 \end{bmatrix}}}_{\boldsymbol{U}}
\underbrace{\overset{\text{items}}{\begin{bmatrix} 1.9 & 2.3 & 1.9 & 0.0 & -0.6 \\ -1.2 & -0.5 & -1.7 & -2.3 & -2.2 \end{bmatrix}}\text{factors}}_{\boldsymbol{V}^\mathsf{T}}
$$

Matrix Factorisation - Example Results

- On the left we see the observed input data Y, and on the right we see the model predictions $\hat{Y}$.

items (columns), users (rows)

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

Y

items (columns), users (rows)

4.4	4.8	4.6	1.0	-0.3
4.9	4.5	5.5	3.1	1.9
4.3	4.1	4.7	2.3	1.1
2.4	0.8	3.5	5.1	4.9
1.3	0.0	2.1	3.9	3.9
1.1	-0.3	1.9	3.8	3.9

$\hat{Y}$

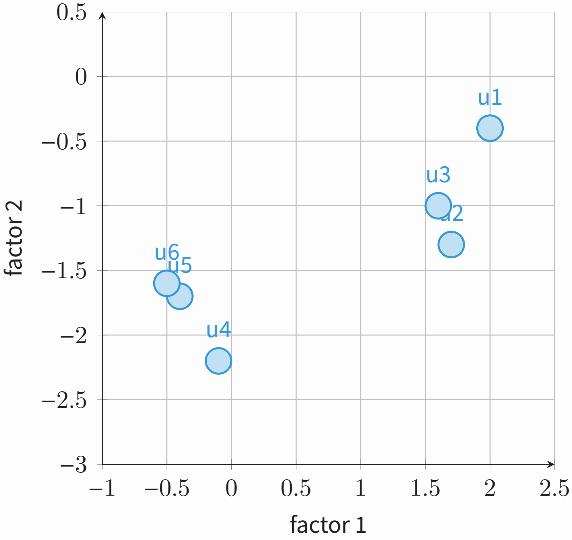
Visualising the Learned User Factors

factors

users

2.0	-0.4
1.7	-1.3
1.6	-1.0
-0.1	-2.2
-0.4	-1.7
-0.5	-1.6

U



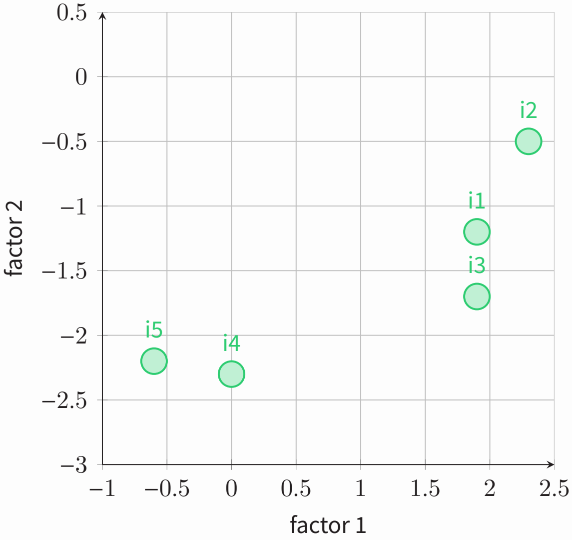
Visualising the Learned Item Factors

factors

items

1.9	-1.2
2.3	-0.5
1.9	-1.7
0.0	-2.3
-0.6	-2.2

V



The Netflix Prize

- In 2006 Netflix released a dataset containing 100,480,507 movie ratings that 480,189 users gave to 17,770 movies.

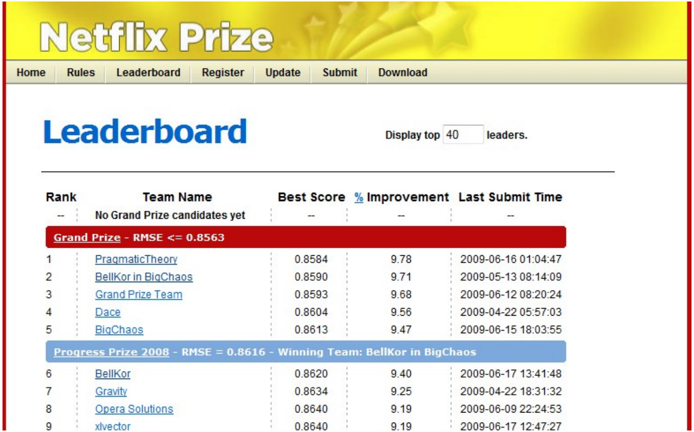
The Netflix Prize

- In 2006 Netflix released a dataset containing 100,480,507 movie ratings that 480,189 users gave to 17,770 movies.
- Each rating was an integer between 1 and 5, where a higher number indicated that a user liked a movie more.

The Netflix Prize

- In 2006 Netflix released a dataset containing 100,480,507 movie ratings that 480,189 users gave to 17,770 movies.
- Each rating was an integer between 1 and 5, where a higher number indicated that a user liked a movie more.
- They offered a prize of $1,000,000 for the team that could improve prediction performance compared to the algorithm Netflix used at the time on a held out test set.

The Netflix Prize - Leaderboard



Netflix Prize

Home | Rules | Leaderboard | Register | Update | Submit | Download

Leaderboard

Display top 40 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
--	No Grand Prize candidates yet	--	--	--
Grand Prize - RMSE <= 0.8563				
1	PragmaticTheory	0.8584	9.78	2009-06-16 01:04:47
2	BellKor in BigChaos	0.8590	9.71	2009-05-13 08:14:09
3	Grand Prize Team	0.8593	9.68	2009-06-12 08:20:24
4	Dace	0.8604	9.56	2009-04-22 05:57:03
5	BigChaos	0.8613	9.47	2009-06-15 18:03:55
Progress Prize 2008 - RMSE = 0.8616 - Winning Team: BellKor in BigChaos				
6	BellKor	0.8620	9.40	2009-06-17 13:41:48
7	Gravity	0.8634	9.25	2009-04-22 18:31:32
8	Opera Solutions	0.8640	9.19	2009-06-09 22:24:53
9	xlvector	0.8640	9.19	2009-06-17 12:47:27

Image credit: https://www.wired.com/2012/04/netflix-prize-costs

The Netflix Prize - Data Privacy

- The original dataset did **not** provide any user identifying information, i.e. user names and IDs were anonymised.

The Netflix Prize - Data Privacy

- The original dataset did **not** provide any user identifying information, i.e. user names and IDs were anonymised.
- However, researchers were able to identify users in the dataset by matching their ratings against other publicly available sources of data (e.g. reviews they provided on the website IMDB).
- As a result, the Netflix Prize dataset is no longer publicly available.

Summary

- We discussed the recommendation problem.

Summary

- We discussed the recommendation problem.
- Recommendation systems are commonly deployed in services that aim to recommend items to users, e.g. movies, books, ads, ...

Summary

- We discussed the recommendation problem.
- Recommendation systems are commonly deployed in services that aim to recommend items to users, e.g. movies, books, ads, ...
- We outlined a memory-based collaborative filtering method and a model-based matrix factorisation approach for solving for unobserved ratings.