Matrix Factorization for Movie Recommendation Systems

Nipun Batra

IIT Gandhinagar

September 3, 2025

Today's Learning Journey

- **The Problem**: Why do we need recommendation systems?

Today's Learning Journey

- **The Problem**: Why do we need recommendation systems?
- **Matrix View**: How ratings become a mathematical problem

Today's Learning Journey

- **The Problem**: Why do we need recommendation systems?
- **Matrix View**: How ratings become a mathematical problem
- **Key Insight**: Matrix factorization as the solution

Today's Learning Journey

- **The Problem**: Why do we need recommendation systems?
- **Matrix View**: How ratings become a mathematical problem
- **Key Insight**: Matrix factorization as the solution
- **Step-by-Step**: Building intuition with examples

Today's Learning Journey

- **The Problem**: Why do we need recommendation systems?
- **Matrix View**: How ratings become a mathematical problem
- **Key Insight**: Matrix factorization as the solution
- **Step-by-Step**: Building intuition with examples
- **Algorithms**: ALS vs Gradient Descent

Today's Learning Journey

- **The Problem**: Why do we need recommendation systems?
- **Matrix View**: How ratings become a mathematical problem
- **Key Insight**: Matrix factorization as the solution
- **Step-by-Step**: Building intuition with examples
- **Algorithms**: ALS vs Gradient Descent
- **Practice**: Hands-on understanding

Problem Setup

The Movie Recommendation Challenge

Real-World Scenario:

- Netflix: 200M+ users, 15K+ titles

The Movie Recommendation Challenge

Real-World Scenario:

- Netflix: 200M+ users, 15K+ titles
- Most ratings are missing!

The Movie Recommendation Challenge

Real-World Scenario:

- Netflix: 200M+ users, 15K+ titles
- Most ratings are missing!

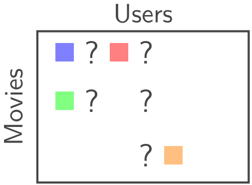
The Movie Recommendation Challenge

Real-World Scenario:

- Netflix: 200M+ users, 15K+ titles
- Most ratings are missing!

The Challenge:

- You've rated 100 movies out of 15,000



Sparse Rating Matrix

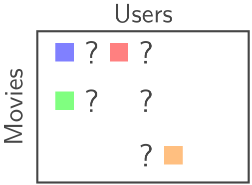
The Movie Recommendation Challenge

Real-World Scenario:

- Netflix: 200M+ users, 15K+ titles
- Most ratings are missing!

The Challenge:

- You've rated 100 movies out of 15,000
- How do we predict what you'll like?



Sparse Rating Matrix

Pop Quiz #1: Understanding the Scale

Answer this!

If Netflix has 200 million users and 15,000 movies, how many possible ratings exist?
Hint: Think about the total number of user-movie pairs

Pop Quiz #1: Understanding the Scale - Answer

Answer this!

Answer: $200 \times 10^6 \times 15 \times 10^3 = 3 \times 10^{12}$ possible ratings!
Follow-up: If typical users rate only 100 movies, what percentage of the matrix is filled?
Answer: $\frac{100}{15000} = 0.67\%$ - extremely sparse!

Pop Quiz #1: Understanding the Scale - Answer

Answer this!

Answer: $200 \times 10^6 \times 15 \times 10^3 = 3 \times 10^{12}$ possible ratings!
Follow-up: If typical users rate only 100 movies, what percentage of the matrix is filled?
Answer: $\frac{100}{15000} = 0.67\%$ - extremely sparse!

Important: The Sparsity Challenge

99.33% of the rating matrix is empty! This is why we need smart algorithms.

Mathematical Problem Setup

The Rating Matrix $\mathbf{A} \in \mathbb{R}^{N \times M}$ with proper labels:

Mathematical Problem Setup

The Rating Matrix A $\in \mathbb{R}^{N \times M}$ with proper labels:

	Sholay	**Swades**	**Batman**	**Interstellar**	**...**
Arjun	a_{11}	?	a_{13}	?	$\cdots$
Priya	?	a_{22}	?	a_{24}	$\cdots$
Ravi	a_{31}	?	?	a_{34}	$\cdots$
...	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Mathematical Problem Setup

The Rating Matrix A $\in \mathbb{R}^{N\times M}$ with proper labels:

	Sholay	**Swades**	**Batman**	**Interstellar**	**...**
Arjun	a_{11}	?	a_{13}	?	$\cdots$
Priya	?	a_{22}	?	a_{24}	$\cdots$
Ravi	a_{31}	?	?	a_{34}	$\cdots$
...	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

- **Rows**: Users $u_1 =$ Arjun, $u_2 =$ Priya, $u_3 =$ Ravi, $\ldots, u_N$

Mathematical Problem Setup

The Rating Matrix A $\in \mathbb{R}^{N \times M}$ with proper labels:

	Sholay	**Swades**	**Batman**	**Interstellar**	**...**
Arjun	a_{11}	?	a_{13}	?	$\cdots$
Priya	?	a_{22}	?	a_{24}	$\cdots$
Ravi	a_{31}	?	?	a_{34}	$\cdots$
...	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

- **Rows**: Users $u_1 = \text{Arjun}, u_2 = \text{Priya}, u_3 = \text{Ravi}, \ldots, u_N$
- **Columns**: Movies $m_1 = \text{Sholay}, m_2 = \text{Swades}, \ldots, m_M$

Mathematical Problem Setup

The Rating Matrix A $\in \mathbb{R}^{N \times M}$ with proper labels:

	Sholay	**Swades**	**Batman**	**Interstellar**	**...**
Arjun	a_{11}	?	a_{13}	?	$\cdots$
Priya	?	a_{22}	?	a_{24}	$\cdots$
Ravi	a_{31}	?	?	a_{34}	$\cdots$
...	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

- **Rows**: Users $u_1 = \text{Arjun}, u_2 = \text{Priya}, u_3 = \text{Ravi}, \ldots, u_N$
- **Columns**: Movies $m_1 = \text{Sholay}, m_2 = \text{Swades}, \ldots, m_M$
- **Entries**: $a_{ij} \in \{1, 2, 3, 4, 5\}$ (when observed)

Mathematical Problem Setup

The Rating Matrix A $\in \mathbb{R}^{N \times M}$ with proper labels:

	Sholay	**Swades**	**Batman**	**Interstellar**	**...**
Arjun	a_{11}	?	a_{13}	?	$\cdots$
Priya	?	a_{22}	?	a_{24}	$\cdots$
Ravi	a_{31}	?	?	a_{34}	$\cdots$
...	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

- **Rows**: Users $u_1 = \text{Arjun}, u_2 = \text{Priya}, u_3 = \text{Ravi}, \ldots, u_N$
- **Columns**: Movies $m_1 = \text{Sholay}, m_2 = \text{Swades}, \ldots, m_M$
- **Entries**: $a_{ij} \in \{1, 2, 3, 4, 5\}$ (when observed)
- **Challenge**: Predict missing entries ?

Mathematical Problem Setup

The Rating Matrix A $\in \mathbb{R}^{N \times M}$ with proper labels:

	Sholay	**Swades**	**Batman**	**Interstellar**	**...**
Arjun	a_{11}	?	a_{13}	?	$\cdots$
Priya	?	a_{22}	?	a_{24}	$\cdots$
Ravi	a_{31}	?	?	a_{34}	$\cdots$
...	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

- **Rows**: Users $u_1 = \text{Arjun}, u_2 = \text{Priya}, u_3 = \text{Ravi}, \ldots, u_N$
- **Columns**: Movies $m_1 = \text{Sholay}, m_2 = \text{Swades}, \ldots, m_M$
- **Entries**: $a_{ij} \in \{1, 2, 3, 4, 5\}$ (when observed)
- **Challenge**: Predict missing entries ?
- **Notation**: $\Omega = \{(i, j) : a_{ij} \text{ is observed}\}$

Concrete Example: Our Movie Dataset

Let's work with a small, concrete example:

Concrete Example: Our Movie Dataset

Let's work with a small, concrete example:

User	Sholay	Swades	Batman	Interstellar	Shawshank
Arjun	5	4	2	3	2
Priya	?	5	1	4	?
Ravi	4	?	1	5	?

Concrete Example: Our Movie Dataset

Let's work with a small, concrete example:

User	Sholay	Swades	Batman	Interstellar	Shawshank
Arjun	5	4	2	3	2
Priya	?	5	1	4	?
Ravi	4	?	1	5	?

Observations:

- Arjun loves Bollywood films (Sholay, Swades)

Concrete Example: Our Movie Dataset

Let's work with a small, concrete example:

User	Sholay	Swades	Batman	Interstellar	Shawshank
Arjun	5	4	2	3	2
Priya	?	5	1	4	?
Ravi	4	?	1	5	?

Observations:

- Arjun loves Bollywood films (Sholay, Swades)
- Ravi enjoys Sci-Fi (Interstellar)

Concrete Example: Our Movie Dataset

Let's work with a small, concrete example:

User	Sholay	Swades	Batman	Interstellar	Shawshank
Arjun	5	4	2	3	2
Priya	?	5	1	4	?
Ravi	4	?	1	5	?

Observations:

- Arjun loves Bollywood films (Sholay, Swades)
- Ravi enjoys Sci-Fi (Interstellar)
- Can we predict Priya's rating for Sholay?

Concrete Example: Our Movie Dataset

Let's work with a small, concrete example:

User	Sholay	Swades	Batman	Interstellar	Shawshank
Arjun	5	4	2	3	2
Priya	?	5	1	4	?
Ravi	4	?	1	5	?

Observations:

- Arjun loves Bollywood films (Sholay, Swades)
- Ravi enjoys Sci-Fi (Interstellar)
- Can we predict Priya's rating for Sholay?
- Can we predict Ravi's rating for Swades?

Key Insight: Latent Features

Before We Dive In: A Simple Question

Why do you like the movies you like?

Before We Dive In: A Simple Question

Why do you like the movies you like?

Maybe because of:

- Genre (Action, Romance, Comedy)

Before We Dive In: A Simple Question

Why do you like the movies you like?

Maybe because of:

- Genre (Action, Romance, Comedy)
- Star cast (Shah Rukh Khan, Tom Cruise)

Before We Dive In: A Simple Question

Why do you like the movies you like?

Maybe because of:

- Genre (Action, Romance, Comedy)
- Star cast (Shah Rukh Khan, Tom Cruise)
- Director (Christopher Nolan, Rajkumar Hirani)

Why do you like the movies you like?

Maybe because of:

- Genre (Action, Romance, Comedy)
- Star cast (Shah Rukh Khan, Tom Cruise)
- Director (Christopher Nolan, Rajkumar Hirani)
- Language (Hindi, English, Tamil)

Before We Dive In: A Simple Question

Why do you like the movies you like?

Maybe because of:

- Genre (Action, Romance, Comedy)
- Star cast (Shah Rukh Khan, Tom Cruise)
- Director (Christopher Nolan, Rajkumar Hirani)
- Language (Hindi, English, Tamil)
- Era (90s classics, modern CGI)

Key Insight:

- Your taste = combination of preferences
- Movie appeal = combination of features
- But we don't know these explicitly!

The Core Insight: Hidden Patterns

Hypothesis: User preferences and movie characteristics can be captured by a small number of **latent features**.

The Core Insight: Hidden Patterns

Hypothesis: User preferences and movie characteristics can be captured by a small number of **latent features**.

Intuition: Think of latent features as "hidden DNA" of movies and users!

The Core Insight: Hidden Patterns

Hypothesis: User preferences and movie characteristics can be captured by a small number of **latent features**.
Intuition: Think of latent features as "hidden DNA" of movies and users!

For Movies:

- Bollywood vs Hollywood

The Core Insight: Hidden Patterns

Hypothesis: User preferences and movie characteristics can be captured by a small number of **latent features**.
Intuition: Think of latent features as "hidden DNA" of movies and users!

For Movies:

- Bollywood vs Hollywood
- Action vs Drama

The Core Insight: Hidden Patterns

Hypothesis: User preferences and movie characteristics can be captured by a small number of **latent features**.
Intuition: Think of latent features as "hidden DNA" of movies and users!

For Movies:

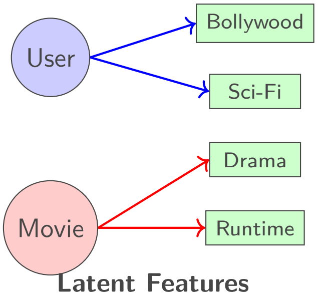
- Bollywood vs Hollywood
- Action vs Drama
- Comedy vs Serious

The Core Insight: Hidden Patterns

Hypothesis: User preferences and movie characteristics can be captured by a small number of **latent features**.
Intuition: Think of latent features as "hidden DNA" of movies and users!

For Movies:

- Bollywood vs Hollywood
- Action vs Drama
- Comedy vs Serious
- Runtime (Short vs Long)

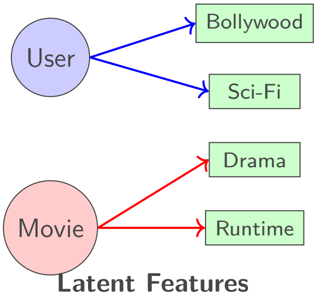


The Core Insight: Hidden Patterns

Hypothesis: User preferences and movie characteristics can be captured by a small number of **latent features**.
Intuition: Think of latent features as "hidden DNA" of movies and users!

For Movies:

- Bollywood vs Hollywood
- Action vs Drama
- Comedy vs Serious
- Runtime (Short vs Long)
- Year (Classic vs Modern)



Step 1: Define Movie Features Explicitly

Let's manually define features for our 5 movies:

Step 1: Define Movie Features Explicitly

Let's manually define features for our 5 movies:

Movie	**Bollywood**	**Sci-Fi**	**Drama**
Sholay	0.95	0.10	0.85
Swades	1.00	0.20	0.90
Batman	0.05	0.80	0.30
Interstellar	0.05	0.95	0.70
Shawshank	0.05	0.15	0.95

Step 1: Define Movie Features Explicitly

Let's manually define features for our 5 movies:

Movie	Bollywood	Sci-Fi	Drama
Sholay	0.95	0.10	0.85
Swades	1.00	0.20	0.90
Batman	0.05	0.80	0.30
Interstellar	0.05	0.95	0.70
Shawshank	0.05	0.15	0.95

Movie Feature Matrix H $\in \mathbb{R}^{3\times 5}$:

$$\mathbf{H} = \begin{bmatrix} 0.95 & 1.00 & 0.05 & 0.05 & 0.05 \\ 0.10 & 0.20 & 0.80 & 0.95 & 0.15 \\ 0.85 & 0.90 & 0.30 & 0.70 & 0.95 \end{bmatrix}$$

Step 2: What About User Preferences?

User Feature Matrix W $\in \mathbb{R}^{3\times 3}$ represents user preferences:

Step 2: What About User Preferences?

User Feature Matrix W $\in \mathbb{R}^{3\times 3}$ represents user preferences:

	Bollywood	**Sci-Fi**	**Drama**
Arjun	3.8	0.8	1.5
Priya	1.8	4.1	3.2
Ravi	2.4	4.8	2.1

Step 2: What About User Preferences?

User Feature Matrix W $\in \mathbb{R}^{3\times 3}$ represents user preferences:

	Bollywood	**Sci-Fi**	**Drama**
Arjun	3.8	0.8	1.5
Priya	1.8	4.1	3.2
Ravi	2.4	4.8	2.1

Each row = one user's preference profile

Step 2: What About User Preferences?

User Feature Matrix W $\in \mathbb{R}^{3\times 3}$ represents user preferences:

	Bollywood	**Sci-Fi**	**Drama**
Arjun	3.8	0.8	1.5
Priya	1.8	4.1	3.2
Ravi	2.4	4.8	2.1

Each row = one user's preference profile

Next: Let's understand what each number means!

User Preference Interpretation

What do these numbers tell us?

User Preference Interpretation

What do these numbers tell us?

	Bollywood	**Sci-Fi**	**Drama**
Arjun	3.8	0.8	1.5
Priya	1.8	4.1	3.2
Ravi	2.4	4.8	2.1

User Preference Interpretation

What do these numbers tell us?

	Bollywood	Sci-Fi	Drama
Arjun	3.8	0.8	1.5
Priya	1.8	4.1	3.2
Ravi	2.4	4.8	2.1

- **Arjun:** Strong Bollywood fan, weak Sci-Fi

User Preference Interpretation

What do these numbers tell us?

	Bollywood	**Sci-Fi**	**Drama**
Arjun	3.8	0.8	1.5
Priya	1.8	4.1	3.2
Ravi	2.4	4.8	2.1

- **Arjun:** Strong Bollywood fan, weak Sci-Fi
- **Priya:** Strong Sci-Fi fan, moderate Drama

User Preference Interpretation

What do these numbers tell us?

	Bollywood	**Sci-Fi**	**Drama**
Arjun	3.8	0.8	1.5
Priya	1.8	4.1	3.2
Ravi	2.4	4.8	2.1

- **Arjun:** Strong Bollywood fan, weak Sci-Fi
- **Priya:** Strong Sci-Fi fan, moderate Drama
- **Ravi:** Very strong Sci-Fi fan

User Preference Interpretation

What do these numbers tell us?

	Bollywood	**Sci-Fi**	**Drama**
Arjun	3.8	0.8	1.5
Priya	1.8	4.1	3.2
Ravi	2.4	4.8	2.1

- **Arjun:** Strong Bollywood fan, weak Sci-Fi
- **Priya:** Strong Sci-Fi fan, moderate Drama
- **Ravi:** Very strong Sci-Fi fan

User Preference Interpretation

What do these numbers tell us?

	Bollywood	Sci-Fi	Drama
Arjun	3.8	0.8	1.5
Priya	1.8	4.1	3.2
Ravi	2.4	4.8	2.1

- **Arjun:** Strong Bollywood fan, weak Sci-Fi
- **Priya:** Strong Sci-Fi fan, moderate Drama
- **Ravi:** Very strong Sci-Fi fan

The Magic: These values $\times$ movie features should recreate observed ratings!

Focus: Arjun's Bollywood Preference

Understanding w_{11} - how much Arjun likes Bollywood:

Focus: Arjun's Bollywood Preference

Understanding w_{11} - how much Arjun likes Bollywood:

	Bollywood	**Sci-Fi**	**Drama**
Arjun	3.8	0.8	1.5
Priya	1.8	4.1	3.2
Ravi	2.4	4.8	2.1

Focus: Arjun's Bollywood Preference

Understanding w_{11} - how much Arjun likes Bollywood:

	Bollywood	Sci-Fi	Drama
Arjun	3.8	0.8	1.5
Priya	1.8	4.1	3.2
Ravi	2.4	4.8	2.1

$w_{11} = 3.8$: Arjun's strong Bollywood preference!

Focus: Arjun's Bollywood Preference

Understanding w_{11} - how much Arjun likes Bollywood:

	Bollywood	**Sci-Fi**	**Drama**
Arjun	3.8	0.8	1.5
Priya	1.8	4.1	3.2
Ravi	2.4	4.8	2.1

$w_{11} = 3.8$: Arjun's strong Bollywood preference!
This explains why he gave Sholay (high Bollywood content) a rating of 5.

Focus: Arjun's Sci-Fi Preference

Understanding w_{12} - how much Arjun likes Sci-Fi:

Focus: Arjun's Sci-Fi Preference

Understanding w_{12} - how much Arjun likes Sci-Fi:

	Bollywood	Sci-Fi	Drama
Arjun	3.8	0.8	1.5
Priya	1.8	4.1	3.2
Ravi	2.4	4.8	2.1

Focus: Arjun's Sci-Fi Preference

Understanding w_{12} - how much Arjun likes Sci-Fi:

	Bollywood	**Sci-Fi**	**Drama**
Arjun	3.8	0.8	1.5
Priya	1.8	4.1	3.2
Ravi	2.4	4.8	2.1

$w_{12} = 0.8$: Arjun's weak Sci-Fi preference

Focus: Arjun's Sci-Fi Preference

Understanding w_{12} - how much Arjun likes Sci-Fi:

	Bollywood	Sci-Fi	Drama
Arjun	3.8	0.8	1.5
Priya	1.8	4.1	3.2
Ravi	2.4	4.8	2.1

$w_{12} = 0.8$: Arjun's weak Sci-Fi preference
This explains why he gave Interstellar (high Sci-Fi) only a rating of 3.

Focus: Arjun's Drama Preference

Understanding w_{13} **- how much Arjun likes Drama:**

Focus: Arjun's Drama Preference

Understanding w_{13} - how much Arjun likes Drama:

	Bollywood	**Sci-Fi**	**Drama**
Arjun	3.8	0.8	1.5
Priya	1.8	4.1	3.2
Ravi	2.4	4.8	2.1

Focus: Arjun's Drama Preference

Understanding w_{13} - how much Arjun likes Drama:

	Bollywood	Sci-Fi	Drama
Arjun	3.8	0.8	1.5
Priya	1.8	4.1	3.2
Ravi	2.4	4.8	2.1

$w_{13} = 1.5$: Arjun's low Drama preference

Focus: Arjun's Drama Preference

Understanding w_{13} - how much Arjun likes Drama:

	Bollywood	Sci-Fi	Drama
Arjun	3.8	0.8	1.5
Priya	1.8	4.1	3.2
Ravi	2.4	4.8	2.1

$w_{13} = 1.5$: Arjun's low Drama preference

This explains why he gave Shawshank (high Drama) only a rating of 2.

Arjun's Complete Taste Profile

Putting it all together:

Arjun's Complete Taste Profile

Putting it all together:

	Bollywood	**Sci-Fi**	**Drama**
Arjun	3.8	0.8	1.5
Priya	1.8	4.1	3.2
Ravi	2.4	4.8	2.1

Arjun's Complete Taste Profile

Putting it all together:

	Bollywood	**Sci-Fi**	**Drama**
Arjun	3.8	0.8	1.5
Priya	1.8	4.1	3.2
Ravi	2.4	4.8	2.1

Key Points:

Arjun's taste profile: [3.8 Bollywood, 0.8 Sci-Fi, 1.5 Drama]

Step 3: The Matrix Factorization Idea

Core Hypothesis: Rating $=$ User preferences $\cdot$ Movie features

Step 3: The Matrix Factorization Idea

Core Hypothesis: Rating = User preferences · Movie features

$$\boxed{a_{ij} \approx \mathbf{w}_i^T \mathbf{h}_j = \sum_{k=1}^{r} w_{ik} h_{kj}}$$

Step 3: The Matrix Factorization Idea

Core Hypothesis: Rating = User preferences · Movie features

$$\boxed{a_{ij} \approx \mathbf{w}_i^T \mathbf{h}_j = \sum_{k=1}^{r} w_{ik} h_{kj}}$$

In Matrix Form:

$$\boxed{\mathbf{A} \approx \mathbf{W}\mathbf{H}}$$

Step 3: The Matrix Factorization Idea

Core Hypothesis: Rating = User preferences · Movie features

$$\boxed{a_{ij} \approx \mathbf{w}_i^T \mathbf{h}_j = \sum_{k=1}^{r} w_{ik} h_{kj}}$$

In Matrix Form:

$$\boxed{\mathbf{A} \approx \mathbf{WH}}$$

$$\mathbf{A}_{3\times 5} = \begin{bmatrix} 5 & 4 & 2 & 3 & 2 \\ ? & 5 & 1 & 4 & ? \\ 4 & ? & 1 & 5 & ? \end{bmatrix} \approx$$

$$\begin{bmatrix} 3.8 & 0.8 & 1.5 \\ 1.8 & 4.1 & 3.2 \\ 2.4 & 4.8 & 2.1 \end{bmatrix} \begin{bmatrix} 0.95 & 1.00 & 0.05 & 0.05 & 0.05 \\ 0.10 & 0.20 & 0.80 & 0.95 & 0.15 \\ 0.85 & 0.90 & 0.30 & 0.70 & 0.95 \end{bmatrix} = \mathbf{W}_{3\times 3}\mathbf{H}_{3\times 5}$$

Step 4: Understanding the Calculation

Let's predict Arjun's rating for Sholay step by step...

Step 4: Understanding the Calculation

Let's predict Arjun's rating for Sholay step by step...

Arjun's Profile:

- How much does he like Bollywood? $w_{11} = 3.8$

Step 4: Understanding the Calculation

Let's predict Arjun's rating for Sholay step by step...

Arjun's Profile:

- How much does he like Bollywood? $w_{11} = 3.8$
- How much does he like Sci-Fi? $w_{12} = 0.8$

Step 4: Understanding the Calculation

Let's predict Arjun's rating for Sholay step by step...

Arjun's Profile:

- How much does he like Bollywood? $w_{11} = 3.8$
- How much does he like Sci-Fi? $w_{12} = 0.8$
- How much does he like Drama? $w_{13} = 1.5$

Sholay's DNA:

- Bollywood-ness: 0.95 (very high!)
- Sci-Fi-ness: 0.10 (low)
- Drama-ness: 0.85 (high)

Step 4: Understanding the Calculation

Let's predict Arjun's rating for Sholay step by step...

Arjun's Profile:

- How much does he like Bollywood? $w_{11} = 3.8$
- How much does he like Sci-Fi? $w_{12} = 0.8$
- How much does he like Drama? $w_{13} = 1.5$

Sholay's DNA:

- Bollywood-ness: 0.95 (very high!)
- Sci-Fi-ness: 0.10 (low)
- Drama-ness: 0.85 (high)

Step 4: Understanding the Calculation

Let's predict Arjun's rating for Sholay step by step...

Arjun's Profile:

- How much does he like Bollywood? $w_{11} = 3.8$
- How much does he like Sci-Fi? $w_{12} = 0.8$
- How much does he like Drama? $w_{13} = 1.5$

Sholay's DNA:

- Bollywood-ness: 0.95 (very high!)
- Sci-Fi-ness: 0.10 (low)
- Drama-ness: 0.85 (high)

The Magic Formula:

$$\text{Arjun's rating} = \text{Arjun's preferences} \cdot \text{Sholay's features}$$

Step 4: Detailed Calculation Example

Let's compute Arjun's predicted rating for Sholay:

Step 4: Detailed Calculation Example

Let's compute Arjun's predicted rating for Sholay:
Arjun's preferences: $\mathbf{w}_1 = [3.8, 0.8, 1.5]$
Sholay's features: $\mathbf{h}_1 = [0.95, 0.10, 0.85]^T$

Step 4: Detailed Calculation Example

Let's compute Arjun's predicted rating for Sholay:
Arjun's preferences: $\mathbf{w}_1 = [3.8, 0.8, 1.5]$
Sholay's features: $\mathbf{h}_1 = [0.95, 0.10, 0.85]^T$

$$\hat{a}_{11} = \mathbf{w}_1^T \mathbf{h}_1 \tag{1}$$
$$= 3.8 \cdot 0.95 + 0.8 \cdot 0.10 + 1.5 \cdot 0.85 \tag{2}$$
$$= 3.61 + 0.08 + 1.28 = \mathbf{4.97} \tag{3}$$

Step 4: Detailed Calculation Example

Let's compute Arjun's predicted rating for Sholay:
Arjun's preferences: $\mathbf{w}_1 = [3.8, 0.8, 1.5]$
Sholay's features: $\mathbf{h}_1 = [0.95, 0.10, 0.85]^T$

$$\hat{a}_{11} = \mathbf{w}_1^T \mathbf{h}_1 \tag{1}$$
$$= 3.8 \cdot 0.95 + 0.8 \cdot 0.10 + 1.5 \cdot 0.85 \tag{2}$$
$$= 3.61 + 0.08 + 1.28 = \mathbf{4.97} \tag{3}$$

$\hat{a}_{11} = 4.97 \approx 5$ ✓
This shows how matrix factorization works: find user preferences that recreate observed ratings!

Pop Quiz #2: Matrix Dimensions

Answer this!

If we have N users, M movies, and r latent features:

Pop Quiz #2: Matrix Dimensions

Answer this!

If we have N users, M movies, and r latent features:

1. What are the dimensions of $\mathbf{A}$?

Pop Quiz #2: Matrix Dimensions

Answer this!

If we have N users, M movies, and r latent features:

1. What are the dimensions of $\mathbf{A}$?
2. What are the dimensions of $\mathbf{W}$?

Pop Quiz #2: Matrix Dimensions

Answer this!

If we have N users, M movies, and r latent features:

1. What are the dimensions of $\mathbf{A}$?
2. What are the dimensions of $\mathbf{W}$?
3. What are the dimensions of $\mathbf{H}$?

Pop Quiz #2: Matrix Dimensions

Answer this!

If we have N users, M movies, and r latent features:

1. What are the dimensions of $\mathbf{A}$?
2. What are the dimensions of $\mathbf{W}$?
3. What are the dimensions of $\mathbf{H}$?
4. How many parameters do we need to learn?

Pop Quiz #2: Matrix Dimensions - Answers

Answer this!

Answers:

1. $\mathbf{A} \in \mathbb{R}^{N \times M}$ (rating matrix)
2. $\mathbf{W} \in \mathbb{R}^{N \times r}$ (user preferences)
3. $\mathbf{H} \in \mathbb{R}^{r \times M}$ (movie features)
4. Total parameters: $Nr + rM = r(N + M)$

Pop Quiz #2: Matrix Dimensions - Answers

Answer this!

Answers:

1. $\mathbf{A} \in \mathbb{R}^{N \times M}$ (rating matrix)
2. $\mathbf{W} \in \mathbb{R}^{N \times r}$ (user preferences)
3. $\mathbf{H} \in \mathbb{R}^{r \times M}$ (movie features)
4. Total parameters: $Nr + rM = r(N + M)$

Key Points:

If $r \ll \min(N, M)$, we have huge parameter reduction!

Learning the Factorization

The Optimization Problem

Objective: Minimize prediction error on observed ratings only

The Optimization Problem

Objective: Minimize prediction error on observed ratings only

$$\boxed{\text{minimize}_{\mathbf{W},\mathbf{H}} \sum_{(i,j)\in\Omega} (a_{ij} - \mathbf{w}_i^T \mathbf{h}_j)^2}$$

The Optimization Problem

Objective: Minimize prediction error on observed ratings only

$$\boxed{\text{minimize}_{\mathbf{W},\mathbf{H}} \sum_{(i,j)\in\Omega} (a_{ij} - \mathbf{w}_i^T \mathbf{h}_j)^2}$$

In Matrix Notation:

$$\boxed{\text{minimize}_{\mathbf{W},\mathbf{H}} \, \|P_\Omega(\mathbf{A} - \mathbf{WH})\|_F^2}$$

The Optimization Problem

Objective: Minimize prediction error on observed ratings only

$$\boxed{\text{minimize}_{\mathbf{W},\mathbf{H}} \sum_{(i,j)\in\Omega} (a_{ij} - \mathbf{w}_i^T \mathbf{h}_j)^2}$$

In Matrix Notation:

$$\boxed{\text{minimize}_{\mathbf{W},\mathbf{H}} \|P_\Omega(\mathbf{A} - \mathbf{WH})\|_F^2}$$

Where:

- $P_\Omega(\cdot)$: only consider entries where we have ratings

The Optimization Problem

Objective: Minimize prediction error on observed ratings only

$$\boxed{\text{minimize}_{\mathbf{W},\mathbf{H}} \sum_{(i,j)\in\Omega} (a_{ij} - \mathbf{w}_i^T \mathbf{h}_j)^2}$$

In Matrix Notation:

$$\boxed{\text{minimize}_{\mathbf{W},\mathbf{H}} \|P_\Omega(\mathbf{A} - \mathbf{WH})\|_F^2}$$

Where:

- $P_\Omega(\cdot)$: only consider entries where we have ratings
- $\|\cdot\|_F$: Frobenius norm

The Optimization Problem

Objective: Minimize prediction error on observed ratings only

$$\boxed{\text{minimize}_{\mathbf{W},\mathbf{H}} \sum_{(i,j)\in\Omega} (a_{ij} - \mathbf{w}_i^T \mathbf{h}_j)^2}$$

In Matrix Notation:

$$\boxed{\text{minimize}_{\mathbf{W},\mathbf{H}} \|P_\Omega(\mathbf{A} - \mathbf{WH})\|_F^2}$$

Where:

- $P_\Omega(\cdot)$: only consider entries where we have ratings
- $\|\cdot\|_F$: Frobenius norm
- Ω: set of observed (i, j) pairs

Why This is Challenging

Problem Characteristics:

- **Non-convex:** Multiple local minima exist

Why This is Challenging

Problem Characteristics:

- **Non-convex:** Multiple local minima exist
- **Bilinear:** Linear in $\mathbf{W}$ when $\mathbf{H}$ fixed, and vice versa

Why This is Challenging

Problem Characteristics:

- **Non-convex:** Multiple local minima exist
- **Bilinear:** Linear in $\mathbf{W}$ when $\mathbf{H}$ fixed, and vice versa
- **Large-scale:** Millions of users and items

Why This is Challenging

Problem Characteristics:

- **Non-convex:** Multiple local minima exist
- **Bilinear:** Linear in $\mathbf{W}$ when $\mathbf{H}$ fixed, and vice versa
- **Large-scale:** Millions of users and items
- **Sparse:** Only 0.1-1% of entries observed

Why This is Challenging

Problem Characteristics:

- **Non-convex:** Multiple local minima exist
- **Bilinear:** Linear in $\mathbf{W}$ when $\mathbf{H}$ fixed, and vice versa
- **Large-scale:** Millions of users and items
- **Sparse:** Only 0.1-1% of entries observed

Why This is Challenging

Problem Characteristics:

- **Non-convex:** Multiple local minima exist
- **Bilinear:** Linear in $\mathbf{W}$ when $\mathbf{H}$ fixed, and vice versa
- **Large-scale:** Millions of users and items
- **Sparse:** Only 0.1-1% of entries observed

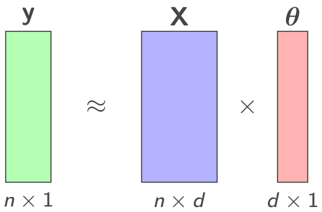
Key Insight: While non-convex jointly, it's convex in each matrix individually!

Alternating Least Squares (ALS): A Visual Derivation

Step 0: Recap of Standard Least Squares

Objective: Estimate $\boldsymbol{\theta} \in \mathbb{R}^{d \times 1}$ from data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ and label vector $\mathbf{y} \in \mathbb{R}^{n \times 1}$.

$$\mathbf{y} \approx \mathbf{X}\boldsymbol{\theta} \quad \Rightarrow \quad \hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2$$



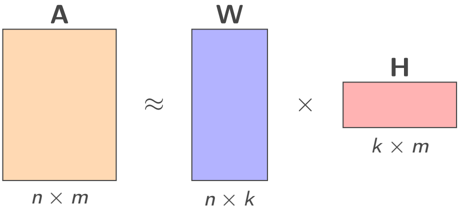
Solution: $\hat{\boldsymbol{\theta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$

ALS Problem Setup

Goal: Decompose $\mathbf{A} \in \mathbb{R}^{n \times m}$ as $\mathbf{WH}$ with:

$$\mathbf{W} \in \mathbb{R}^{n \times k}$$
$$\mathbf{H} \in \mathbb{R}^{k \times m}$$

Objective: $\mathbf{A} \approx \mathbf{WH}$ with $k \ll \min(n, m)$



Challenge: Both $\mathbf{W}$ and $\mathbf{H}$ are unknown.

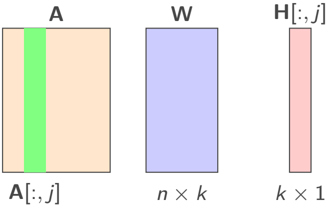
Step 1: Fix **W**, Learn **H** One Column at a Time

Idea: With **W** fixed, each column of **H** can be estimated via LS.

$$\mathbf{A}[:,j] \approx \mathbf{W} \cdot \mathbf{H}[:,j]$$

$$\mathbf{W} : n \times k \qquad \mathbf{H}[:,j] : k \times 1 \qquad \mathbf{A}[:,j] : n \times 1$$

This becomes: $\mathbf{y} \approx \mathbf{X}\theta$ (just like standard LS)



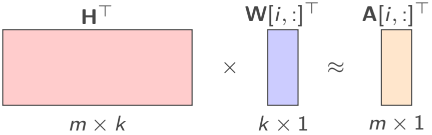
Repeat: for $j = 0$ to $m - 1$
$\mathbf{H}[:,j] \leftarrow \mathrm{LS}(\mathbf{W}, \mathbf{A}[:,j])$

Step 2: Fix $\mathbf{H}$, Learn $\mathbf{W}$ One Row at a Time

Goal: Learn $\mathbf{W}[i,:] \in \mathbb{R}^{1\times k}$ row by row.
Start from: $\mathbf{A}[i,:] \approx \mathbf{W}[i,:] \cdot \mathbf{H}$
Transpose both sides:

$$\mathbf{A}[i,:]^\top \approx \mathbf{H}^\top \cdot \mathbf{W}[i,:]^\top$$

Looks like LS again: $\mathbf{y} \approx \mathbf{X}\boldsymbol{\theta}$



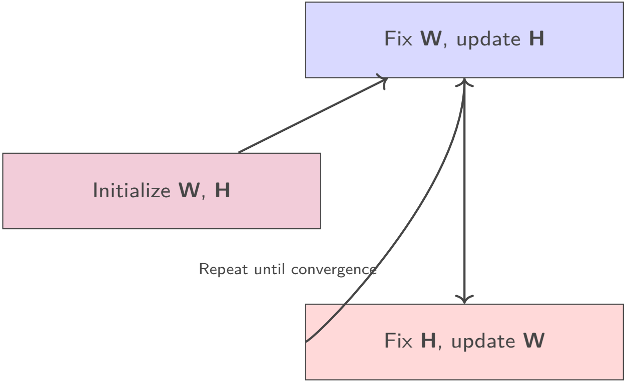
Solve: $\mathbf{W}[i,:]^\top \leftarrow \mathrm{LS}(\mathbf{H}^\top, \mathbf{A}^\top[:,i])$ for all i

Final ALS Loop Summary

Repeat for T iterations:

- Fix $\mathbf{W}$, then for each j, compute $\mathbf{H}[:,j] \leftarrow \text{LS}(\mathbf{W}, \mathbf{A}[:,j])$
- Fix $\mathbf{H}$, then for each i, compute $\mathbf{W}[i,:]^\top \leftarrow \text{LS}(\mathbf{H}^\top, \mathbf{A}^\top[:,i])$

All updates use standard least squares!



Algorithm 2: Gradient Descent

Gradient Descent: Matrix Factorization Setup

Goal: Find $\mathbf{W} \in \mathbb{R}^{n \times k}$ and $\mathbf{H} \in \mathbb{R}^{k \times m}$ such that:

$$\mathbf{A} \approx \mathbf{W}\mathbf{H}$$

Objective Function:

$$L(\mathbf{W}, \mathbf{H}) = \sum_{(i,j) \in \Omega} (a_{ij} - \mathbf{w}_i^\top \mathbf{h}_j)^2$$

Gradient Descent: Matrix Factorization Setup

Goal: Find $\mathbf{W} \in \mathbb{R}^{n \times k}$ and $\mathbf{H} \in \mathbb{R}^{k \times m}$ such that:

$$\mathbf{A} \approx \mathbf{WH}$$

Objective Function:

$$L(\mathbf{W}, \mathbf{H}) = \sum_{(i,j) \in \Omega} (a_{ij} - \mathbf{w}_i^\top \mathbf{h}_j)^2$$

Initialize:

- $\mathbf{W}$: $n \times k$ matrix with small random values
- $\mathbf{H}$: $k \times m$ matrix with small random values

Gradient Descent: Matrix Factorization Setup

Goal: Find $\mathbf{W} \in \mathbb{R}^{n \times k}$ and $\mathbf{H} \in \mathbb{R}^{k \times m}$ such that:

$$\mathbf{A} \approx \mathbf{WH}$$

Objective Function:

$$L(\mathbf{W}, \mathbf{H}) = \sum_{(i,j) \in \Omega} (a_{ij} - \mathbf{w}_i^\top \mathbf{h}_j)^2$$

Initialize:

- $\mathbf{W}$: $n \times k$ matrix with small random values
- $\mathbf{H}$: $k \times m$ matrix with small random values

Gradient Descent Loop:

- For $t = 1$ to T:
  - $\mathbf{W} \leftarrow \mathbf{W} - \alpha \cdot \nabla_{\mathbf{W}} L(\mathbf{W}, \mathbf{H})$
  - $\mathbf{H} \leftarrow \mathbf{H} - \alpha \cdot \nabla_{\mathbf{H}} L(\mathbf{W}, \mathbf{H})$

Gradient Descent: Matrix Factorization Setup

Goal: Find $\mathbf{W} \in \mathbb{R}^{n \times k}$ and $\mathbf{H} \in \mathbb{R}^{k \times m}$ such that:

$$\mathbf{A} \approx \mathbf{W}\mathbf{H}$$

Objective Function:

$$L(\mathbf{W}, \mathbf{H}) = \sum_{(i,j) \in \Omega} (a_{ij} - \mathbf{w}_i^\top \mathbf{h}_j)^2$$

Initialize:

- $\mathbf{W}$: $n \times k$ matrix with small random values
- $\mathbf{H}$: $k \times m$ matrix with small random values

Gradient Descent Loop:

- For $t = 1$ to T:
  - $\mathbf{W} \leftarrow \mathbf{W} - \alpha \cdot \nabla_{\mathbf{W}} L(\mathbf{W}, \mathbf{H})$
  - $\mathbf{H} \leftarrow \mathbf{H} - \alpha \cdot \nabla_{\mathbf{H}} L(\mathbf{W}, \mathbf{H})$

Note: Gradients can be computed either over all $(i, j) \in \Omega$ (Batch GD) or stochastically.

Algorithm Comparison and Practical Considerations

ALS vs SGD: Head-to-Head Comparison

Aspect	**ALS**	**SGD**
Updates	Alternating	Simultaneous
Convergence	Faster, more stable	Slower, can oscillate
Parallelization	Excellent	Limited
Memory	Higher	Lower
Implementation	Complex	Simple
Hyperparameters	Few (rank r)	Many (α, schedule)
Scalability	Very good	Good

ALS vs SGD: Head-to-Head Comparison

Aspect	**ALS**	**SGD**
Updates	Alternating	Simultaneous
Convergence	Faster, more stable	Slower, can oscillate
Parallelization	Excellent	Limited
Memory	Higher	Lower
Implementation	Complex	Simple
Hyperparameters	Few (rank r)	Many (α, schedule)
Scalability	Very good	Good

When to Use Which?

- **ALS:** Large-scale, production systems (Spark, distributed)

ALS vs SGD: Head-to-Head Comparison

Aspect	**ALS**	**SGD**
Updates	Alternating	Simultaneous
Convergence	Faster, more stable	Slower, can oscillate
Parallelization	Excellent	Limited
Memory	Higher	Lower
Implementation	Complex	Simple
Hyperparameters	Few (rank r)	Many (α, schedule)
Scalability	Very good	Good

When to Use Which?

- **ALS:** Large-scale, production systems (Spark, distributed)
- **SGD:** Online learning, real-time updates, research

Advanced Considerations: Regularization

Problem: Basic matrix factorization can overfit to training data

Advanced Considerations: Regularization

Problem: Basic matrix factorization can overfit to training data
Solution: Add regularization terms to control complexity

$$\text{minimize}_{\mathbf{W},\mathbf{H}} \sum_{(i,j)\in\Omega} (a_{ij} - \mathbf{w}_i^T \mathbf{h}_j)^2 + \lambda(\|\mathbf{W}\|_F^2 + \|\mathbf{H}\|_F^2)$$

Advanced Considerations: Regularization

Problem: Basic matrix factorization can overfit to training data
Solution: Add regularization terms to control complexity

$$\text{minimize}_{\mathbf{W},\mathbf{H}} \sum_{(i,j)\in\Omega} (a_{ij} - \mathbf{w}_i^T \mathbf{h}_j)^2 + \lambda(\|\mathbf{W}\|_F^2 + \|\mathbf{H}\|_F^2)$$

What this does:

- Penalizes large feature values

Advanced Considerations: Regularization

Problem: Basic matrix factorization can overfit to training data
Solution: Add regularization terms to control complexity

$$\text{minimize}_{\mathbf{W},\mathbf{H}} \sum_{(i,j)\in\Omega} (a_{ij} - \mathbf{w}_i^T \mathbf{h}_j)^2 + \lambda(\|\mathbf{W}\|_F^2 + \|\mathbf{H}\|_F^2)$$

What this does:

- Penalizes large feature values
- Prevents overfitting to observed ratings

Advanced Considerations: Regularization

Problem: Basic matrix factorization can overfit to training data
Solution: Add regularization terms to control complexity

$$\text{minimize}_{\mathbf{W},\mathbf{H}} \sum_{(i,j)\in\Omega} (a_{ij} - \mathbf{w}_i^T \mathbf{h}_j)^2 + \lambda(\|\mathbf{W}\|_F^2 + \|\mathbf{H}\|_F^2)$$

What this does:

- Penalizes large feature values
- Prevents overfitting to observed ratings
- λ controls regularization strength

Advanced Considerations: Regularization

Problem: Basic matrix factorization can overfit to training data
Solution: Add regularization terms to control complexity

$$\text{minimize}_{\mathbf{W},\mathbf{H}} \sum_{(i,j)\in\Omega} (a_{ij} - \mathbf{w}_i^T \mathbf{h}_j)^2 + \lambda(\|\mathbf{W}\|_F^2 + \|\mathbf{H}\|_F^2)$$

What this does:

- Penalizes large feature values
- Prevents overfitting to observed ratings
- λ controls regularization strength
- Helps generalization to unseen ratings

Advanced Considerations: Bias Terms

Real-world insight: Not all ratings differences are due to preferences!

Advanced Considerations: Bias Terms

Real-world insight: Not all ratings differences are due to preferences!

Bias sources:

- **Global bias μ:** Average rating across all users/movies

Advanced Considerations: Bias Terms

Real-world insight: Not all ratings differences are due to preferences!

Bias sources:

- **Global bias μ:** Average rating across all users/movies
- **User bias b_i:** Some users rate higher than others

Advanced Considerations: Bias Terms

Real-world insight: Not all ratings differences are due to preferences!

Bias sources:

- **Global bias μ:** Average rating across all users/movies
- **User bias b_i:** Some users rate higher than others
- **Item bias b_j:** Some movies are generally better rated

Advanced Considerations: Bias Terms

Real-world insight: Not all ratings differences are due to preferences!

Bias sources:

- **Global bias μ:** Average rating across all users/movies
- **User bias b_i:** Some users rate higher than others
- **Item bias b_j:** Some movies are generally better rated

Advanced Considerations: Bias Terms

Real-world insight: Not all ratings differences are due to preferences!

Bias sources:

- **Global bias μ:** Average rating across all users/movies
- **User bias b_i:** Some users rate higher than others
- **Item bias b_j:** Some movies are generally better rated

Enhanced model:

$$\hat{a}_{ij} = \mu + b_i + b_j + \mathbf{w}_i^T \mathbf{h}_j$$

Advanced Considerations: Bias Terms

Real-world insight: Not all ratings differences are due to preferences!

Bias sources:

- **Global bias μ:** Average rating across all users/movies
- **User bias b_i:** Some users rate higher than others
- **Item bias b_j:** Some movies are generally better rated

Enhanced model:

$$\hat{a}_{ij} = \mu + b_i + b_j + \mathbf{w}_i^T \mathbf{h}_j$$

Example: Example

Mean rating = 3.5, Arjun rates 0.5 higher, Sholay gets 0.8 higher

$\hat{a}_{\text{Arjun,Sholay}} = 3.5 + 0.5 + 0.8 + \mathbf{w}_{\text{Arjun}}^T \mathbf{h}_{\text{Sholay}}$

Advanced Considerations: Implicit Feedback

Beyond explicit ratings: Many systems have only implicit feedback

Advanced Considerations: Implicit Feedback

Beyond explicit ratings: Many systems have only implicit feedback

Examples of implicit feedback:

- User clicked on movie (binary: 0 or 1)

Advanced Considerations: Implicit Feedback

Beyond explicit ratings: Many systems have only implicit feedback

Examples of implicit feedback:

- User clicked on movie (binary: 0 or 1)
- User watched movie for 5 minutes vs 2 hours

Advanced Considerations: Implicit Feedback

Beyond explicit ratings: Many systems have only implicit feedback

Examples of implicit feedback:

- User clicked on movie (binary: 0 or 1)
- User watched movie for 5 minutes vs 2 hours
- User added to watchlist vs ignored

Advanced Considerations: Implicit Feedback

Beyond explicit ratings: Many systems have only implicit feedback

Examples of implicit feedback:

- User clicked on movie (binary: 0 or 1)
- User watched movie for 5 minutes vs 2 hours
- User added to watchlist vs ignored

Advanced Considerations: Implicit Feedback

Beyond explicit ratings: Many systems have only implicit feedback
Examples of implicit feedback:

- User clicked on movie (binary: 0 or 1)
- User watched movie for 5 minutes vs 2 hours
- User added to watchlist vs ignored

Confidence weighting:

$$\text{Confidence: } c_{ij} = 1 + \alpha \cdot \text{frequency}_{ij}$$

Advanced Considerations: Implicit Feedback

Beyond explicit ratings: Many systems have only implicit feedback
Examples of implicit feedback:

- User clicked on movie (binary: 0 or 1)
- User watched movie for 5 minutes vs 2 hours
- User added to watchlist vs ignored

Confidence weighting:

$$\text{Confidence: } c_{ij} = 1 + \alpha \cdot \text{frequency}_{ij}$$

Key Points:

Idea: More interactions = higher confidence in preference

Advanced Considerations: Cold Start Problem

The Challenge: What about new users or movies with no ratings?

Advanced Considerations: Cold Start Problem

The Challenge: What about new users or movies with no ratings?
Strategies:

- **Content-based features:** Use movie genres, actors, directors

Advanced Considerations: Cold Start Problem

The Challenge: What about new users or movies with no ratings?
Strategies:

- **Content-based features:** Use movie genres, actors, directors
- **Demographic information:** Age, location, gender of users

Advanced Considerations: Cold Start Problem

The Challenge: What about new users or movies with no ratings?
Strategies:

- **Content-based features:** Use movie genres, actors, directors
- **Demographic information:** Age, location, gender of users
- **Hybrid approaches:** Combine collaborative + content-based

Advanced Considerations: Cold Start Problem

The Challenge: What about new users or movies with no ratings?
Strategies:

- **Content-based features:** Use movie genres, actors, directors
- **Demographic information:** Age, location, gender of users
- **Hybrid approaches:** Combine collaborative + content-based
- **Popular items:** Recommend trending content initially

Advanced Considerations: Cold Start Problem

The Challenge: What about new users or movies with no ratings?
Strategies:

- **Content-based features:** Use movie genres, actors, directors
- **Demographic information:** Age, location, gender of users
- **Hybrid approaches:** Combine collaborative + content-based
- **Popular items:** Recommend trending content initially

Advanced Considerations: Cold Start Problem

The Challenge: What about new users or movies with no ratings?
Strategies:

- **Content-based features:** Use movie genres, actors, directors
- **Demographic information:** Age, location, gender of users
- **Hybrid approaches:** Combine collaborative + content-based
- **Popular items:** Recommend trending content initially

Important: Real-World Solution

Most production systems use hybrid approaches combining multiple signals

Hands-On Understanding

Let's Build Intuition: Small Example

Our 3×3 rating matrix:

$$\mathbf{A} = \begin{bmatrix} 5 & ? & 2 \\ 4 & 4 & ? \\ ? & 5 & 1 \end{bmatrix}$$

Let's Build Intuition: Small Example

Our 3×3 rating matrix:

$$\mathbf{A} = \begin{bmatrix} 5 & ? & 2 \\ 4 & 4 & ? \\ ? & 5 & 1 \end{bmatrix}$$

Goal: Find $\mathbf{W} \in \mathbb{R}^{3\times 2}$ and $\mathbf{H} \in \mathbb{R}^{2\times 3}$ such that:

$$\mathbf{A} \approx \mathbf{WH} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \end{bmatrix}$$

Let's Build Intuition: Small Example

Our 3×3 rating matrix:

$$\mathbf{A} = \begin{bmatrix} 5 & ? & 2 \\ 4 & 4 & ? \\ ? & 5 & 1 \end{bmatrix}$$

Goal: Find $\mathbf{W} \in \mathbb{R}^{3\times 2}$ and $\mathbf{H} \in \mathbb{R}^{2\times 3}$ such that:

$$\mathbf{A} \approx \mathbf{WH} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \end{bmatrix}$$

Constraint: Only minimize error on observed entries!

Step-by-Step ALS Solution

Iteration 1: Initialize randomly

$$\mathbf{W}^{(0)} = \begin{bmatrix} 0.5 & 0.3 \\ 0.4 & 0.6 \\ 0.2 & 0.8 \end{bmatrix}, \quad \mathbf{H}^{(0)} = \begin{bmatrix} 1.0 & 0.5 & 0.2 \\ 0.3 & 1.2 & 0.8 \end{bmatrix}$$

Step-by-Step ALS Solution

Iteration 1: Initialize randomly

$$\mathbf{W}^{(0)} = \begin{bmatrix} 0.5 & 0.3 \\ 0.4 & 0.6 \\ 0.2 & 0.8 \end{bmatrix}, \quad \mathbf{H}^{(0)} = \begin{bmatrix} 1.0 & 0.5 & 0.2 \\ 0.3 & 1.2 & 0.8 \end{bmatrix}$$

Update User 1: Only use observed ratings (positions 1,3)

$$\mathbf{y}_1 = [5, 2]^T \tag{4}$$

$$\mathbf{X}_1 = \begin{bmatrix} 1.0 & 0.3 \\ 0.2 & 0.8 \end{bmatrix} \text{ (columns 1,3 of } \mathbf{H}^{(0)\,T}) \tag{5}$$

Step-by-Step ALS Solution

Iteration 1: Initialize randomly

$$\mathbf{W}^{(0)} = \begin{bmatrix} 0.5 & 0.3 \\ 0.4 & 0.6 \\ 0.2 & 0.8 \end{bmatrix}, \quad \mathbf{H}^{(0)} = \begin{bmatrix} 1.0 & 0.5 & 0.2 \\ 0.3 & 1.2 & 0.8 \end{bmatrix}$$

Update User 1: Only use observed ratings (positions 1,3)

$$\mathbf{y}_1 = [5, 2]^T \tag{4}$$

$$\mathbf{X}_1 = \begin{bmatrix} 1.0 & 0.3 \\ 0.2 & 0.8 \end{bmatrix} \text{ (columns 1,3 of } \mathbf{H}^{(0)\,T}\text{)} \tag{5}$$

Solve: $\mathbf{w}_1^{(1)} = \mathrm{LS}(\mathbf{X}_1, \mathbf{y}_1)$
Continue for all users and movies...

Pop Quiz #3: Netflix Engineering Challenge

Answer this!

You're Netflix's lead ML engineer. You have:

- 200M users, 15K movies
- 20B ratings (0.67% filled)
- Need real-time recommendations
- New users/movies arrive daily

Pop Quiz #3: Netflix Engineering Challenge

Answer this!

You're Netflix's lead ML engineer. You have:

- 200M users, 15K movies
- 20B ratings (0.67% filled)
- Need real-time recommendations
- New users/movies arrive daily

Design your recommendation system:

1. Which algorithm: ALS or SGD? Why?

Pop Quiz #3: Netflix Engineering Challenge

Answer this!

You're Netflix's lead ML engineer. You have:

- 200M users, 15K movies
- 20B ratings (0.67% filled)
- Need real-time recommendations
- New users/movies arrive daily

Design your recommendation system:

1. Which algorithm: ALS or SGD? Why?
2. What rank r would you choose?

Pop Quiz #3: Netflix Engineering Challenge

Answer this!

You're Netflix's lead ML engineer. You have:

- 200M users, 15K movies
- 20B ratings (0.67% filled)
- Need real-time recommendations
- New users/movies arrive daily

Design your recommendation system:

1. Which algorithm: ALS or SGD? Why?
2. What rank r would you choose?
3. How to handle new users?

Pop Quiz #3: Netflix Engineering Challenge

Answer this!

You're Netflix's lead ML engineer. You have:

- 200M users, 15K movies
- 20B ratings (0.67% filled)
- Need real-time recommendations
- New users/movies arrive daily

Design your recommendation system:

1. Which algorithm: ALS or SGD? Why?
2. What rank r would you choose?
3. How to handle new users?
4. How to handle the scale?

Pop Quiz #3: Netflix Engineering Challenge

Answer this!

You're Netflix's lead ML engineer. You have:

- 200M users, 15K movies
- 20B ratings (0.67% filled)
- Need real-time recommendations
- New users/movies arrive daily

Design your recommendation system:

1. Which algorithm: ALS or SGD? Why?
2. What rank r would you choose?
3. How to handle new users?
4. How to handle the scale?

Pop Quiz #3: Netflix Engineering Challenge

Answer this!

You're Netflix's lead ML engineer. You have:

- 200M users, 15K movies
- 20B ratings (0.67% filled)
- Need real-time recommendations
- New users/movies arrive daily

Design your recommendation system:

1. Which algorithm: ALS or SGD? Why?
2. What rank r would you choose?
3. How to handle new users?
4. How to handle the scale?

Suggested Solution:

Summary and Key Takeaways

Key Insights: Part 1

1. **Sparsity** $\Rightarrow$ **Factorization**: Sparse matrices can be approximated by low-rank factorizations

Key Insights: Part 1

1. **Sparsity** $\Rightarrow$ **Factorization**: Sparse matrices can be approximated by low-rank factorizations
2. **Latent Features**: Users and items characterized by hidden factors

Key Insights: Part 1

1. **Sparsity** ⇒ **Factorization**: Sparse matrices can be approximated by low-rank factorizations
2. **Latent Features**: Users and items characterized by hidden factors
3. **Bilinear Problem**: Non-convex jointly, convex individually

Key Insights: Part 2

4. **Scale Matters**: Algorithm choice depends on data size

Key Insights: Part 2

4. **Scale Matters**: Algorithm choice depends on data size
5. **Real-World Complexity**: Need regularization, bias terms, cold start solutions

Key Insights: Part 2

4. **Scale Matters**: Algorithm choice depends on data size
5. **Real-World Complexity**: Need regularization, bias terms, cold start solutions

Key Insights: Part 2

4. **Scale Matters**: Algorithm choice depends on data size
5. **Real-World Complexity**: Need regularization, bias terms, cold start solutions

The Mathematical Beauty:

$$\boxed{\text{Collaborative Filtering} = \text{Matrix Factorization} = \text{Dimensionality Reduct}}$$

Extensions: Interpretable Factorization

Non-negative Matrix Factorization (NMF):

Extensions: Interpretable Factorization

Non-negative Matrix Factorization (NMF):

Definition: NMF Constraint

All factors must be non-negative: $\mathbf{W} \geq 0, \mathbf{H} \geq 0$

Extensions: Interpretable Factorization

Non-negative Matrix Factorization (NMF):

Definition: NMF Constraint

All factors must be non-negative: $\mathbf{W} \geq 0, \mathbf{H} \geq 0$

Why this matters:

- Factors represent "parts" or "components"

Extensions: Interpretable Factorization

Non-negative Matrix Factorization (NMF):

Definition: NMF Constraint

All factors must be non-negative: $\mathbf{W} \geq 0, \mathbf{H} \geq 0$

Why this matters:

- Factors represent "parts" or "components"
- No negative contributions $\rightarrow$ interpretable

Extensions: Interpretable Factorization

Non-negative Matrix Factorization (NMF):

Definition: NMF Constraint

All factors must be non-negative: $\mathbf{W} \geq 0, \mathbf{H} \geq 0$

Why this matters:

- Factors represent "parts" or "components"
- No negative contributions $\rightarrow$ interpretable
- Example: Genre weights are always positive

Extensions: Interpretable Factorization

Non-negative Matrix Factorization (NMF):

Definition: NMF Constraint

All factors must be non-negative: $\mathbf{W} \geq 0, \mathbf{H} \geq 0$

Why this matters:

- Factors represent "parts" or "components"
- No negative contributions $\rightarrow$ interpretable
- Example: Genre weights are always positive

Extensions: Interpretable Factorization

Non-negative Matrix Factorization (NMF):

Definition: NMF Constraint

All factors must be non-negative: $\mathbf{W} \geq 0, \mathbf{H} \geq 0$

Why this matters:

- Factors represent "parts" or "components"
- No negative contributions $\rightarrow$ interpretable
- Example: Genre weights are always positive

Example: Movie Example

If factor $1 =$ "Action-ness", then $h_{1j} \geq 0$ means movie j has some amount of action (never "negative action")

Extensions: Deep Learning Approaches

Deep Matrix Factorization:

Extensions: Deep Learning Approaches

Deep Matrix Factorization:
Limitation of linear factorization:

$$\hat{a}_{ij} = \mathbf{w}_i^T \mathbf{h}_j \text{ (only linear interactions)}$$

Extensions: Deep Learning Approaches

Deep Matrix Factorization:
Limitation of linear factorization:

$$\hat{a}_{ij} = \mathbf{w}_i^T \mathbf{h}_j \text{ (only linear interactions)}$$

Deep learning solution:

$$\hat{a}_{ij} = f_{\mathrm{NN}}(\mathbf{w}_i, \mathbf{h}_j)$$

where f_{NN} is a neural network

Extensions: Deep Learning Approaches

Deep Matrix Factorization:
Limitation of linear factorization:

$$\hat{a}_{ij} = \mathbf{w}_i^T \mathbf{h}_j \text{ (only linear interactions)}$$

Deep learning solution:

$$\hat{a}_{ij} = f_{\text{NN}}(\mathbf{w}_i, \mathbf{h}_j)$$

where f_{NN} is a neural network

Key Points:

Captures complex, non-linear user-item interaction patterns!

Extensions: Deep Learning Approaches

Deep Matrix Factorization:
Limitation of linear factorization:

$$\hat{a}_{ij} = \mathbf{w}_i^T \mathbf{h}_j \text{ (only linear interactions)}$$

Deep learning solution:

$$\hat{a}_{ij} = f_{\text{NN}}(\mathbf{w}_i, \mathbf{h}_j)$$

where f_{NN} is a neural network

Key Points:

Captures complex, non-linear user-item interaction patterns!

Examples:

- Neural Collaborative Filtering

Extensions: Deep Learning Approaches

Deep Matrix Factorization:
Limitation of linear factorization:

$$\hat{a}_{ij} = \mathbf{w}_i^T \mathbf{h}_j \text{ (only linear interactions)}$$

Deep learning solution:

$$\hat{a}_{ij} = f_{\text{NN}}(\mathbf{w}_i, \mathbf{h}_j)$$

where f_{NN} is a neural network

Key Points:

Captures complex, non-linear user-item interaction patterns!

Examples:

- Neural Collaborative Filtering
- AutoRec (Autoencoder-based)

Extensions: Deep Learning Approaches

Deep Matrix Factorization:
Limitation of linear factorization:

$$\hat{a}_{ij} = \mathbf{w}_i^T \mathbf{h}_j \text{ (only linear interactions)}$$

Deep learning solution:

$$\hat{a}_{ij} = f_{\text{NN}}(\mathbf{w}_i, \mathbf{h}_j)$$

where f_{NN} is a neural network

Key Points:

Captures complex, non-linear user-item interaction patterns!

Examples:

- Neural Collaborative Filtering
- AutoRec (Autoencoder-based)
- Deep Factorization Machines

Extensions: Advanced Interaction Modeling

Factorization Machines: Handle multi-way interactions

Extensions: Advanced Interaction Modeling

Factorization Machines: Handle multi-way interactions

Example: Beyond User-Movie

Traditional: User $\times$ Movie
FM: User $\times$ Movie $\times$ Time $\times$ Device $\times$ Location $\times$ Weather

Extensions: Advanced Interaction Modeling

Factorization Machines: Handle multi-way interactions

Example: Beyond User-Movie

Traditional: User $\times$ Movie
FM: User $\times$ Movie $\times$ Time $\times$ Device $\times$ Location $\times$ Weather

Key Points:

Enable richer context-aware recommendations

Extensions: Neural Approaches

Variational Autoencoders:

- Probabilistic approach with uncertainty estimates

Extensions: Neural Approaches

Variational Autoencoders:

- Probabilistic approach with uncertainty estimates
- Can generate diverse, novel recommendations

Extensions: Neural Approaches

Variational Autoencoders:

- Probabilistic approach with uncertainty estimates
- Can generate diverse, novel recommendations

Extensions: Neural Approaches

Variational Autoencoders:

- Probabilistic approach with uncertainty estimates
- Can generate diverse, novel recommendations

Graph Neural Networks:

- Model interactions as graph structure

Extensions: Neural Approaches

Variational Autoencoders:

- Probabilistic approach with uncertainty estimates
- Can generate diverse, novel recommendations

Graph Neural Networks:

- Model interactions as graph structure
- Examples: GraphRec, NGCF, LightGCN

Extensions: Exploration and Real-World Applications

Multi-armed Bandits: Balance exploration vs exploitation

Extensions: Exploration and Real-World Applications

Multi-armed Bandits: Balance exploration vs exploitation

Important: The Dilemma

Recommend known good items vs explore new possibilities?

Extensions: Exploration and Real-World Applications

Multi-armed Bandits: Balance exploration vs exploitation

Important: The Dilemma

Recommend known good items vs explore new possibilities?

Key Points:

Balance accuracy with discovery!

Real-World Applications

Matrix factorization is everywhere:

Real-World Applications

Matrix factorization is everywhere:

- **E-commerce:** Amazon recommendations

Real-World Applications

Matrix factorization is everywhere:

- **E-commerce:** Amazon recommendations
- **Music streaming:** Spotify Discover Weekly

Real-World Applications

Matrix factorization is everywhere:

- **E-commerce:** Amazon recommendations
- **Music streaming:** Spotify Discover Weekly
- **Social media:** Facebook friend suggestions

Real-World Applications

Matrix factorization is everywhere:

- **E-commerce:** Amazon recommendations
- **Music streaming:** Spotify Discover Weekly
- **Social media:** Facebook friend suggestions
- **Advertising:** Targeted ads

Real-World Applications

Matrix factorization is everywhere:

- **E-commerce:** Amazon recommendations
- **Music streaming:** Spotify Discover Weekly
- **Social media:** Facebook friend suggestions
- **Advertising:** Targeted ads

Real-World Applications

Matrix factorization is everywhere:

- **E-commerce:** Amazon recommendations
- **Music streaming:** Spotify Discover Weekly
- **Social media:** Facebook friend suggestions
- **Advertising:** Targeted ads

Key Points:

Same principles apply across domains!

Pop Quiz #4: Matrix Factorization Fundamentals

Answer this!

True or False?

1. Matrix factorization can only work with explicit ratings
2. ALS always converges to the global optimum
3. A rank-1 factorization means all users have identical preferences

Pop Quiz #4: Answers - Fundamentals

Answer this!

Answers:

1. **False** - Works with implicit feedback too
2. **False** - Converges to local optimum only
3. **False** - Rank-1 means one pattern, not identical

Pop Quiz #5: Advanced Topics

Answer this!

True or False?

1. Adding regularization always improves recommendations
2. SGD is better than ALS for all applications

Bonus: What are the main trade-offs between ALS and SGD?

Pop Quiz #5: Answers - Advanced Topics

Answer this!

Answers:

1. **False** - Too much regularization causes underfitting
2. **False** - Choice depends on data scale and needs

Bonus: ALS: Fast, parallel. **SGD:** Online, low memory.