Big Data

(Collaborative Filtering with SVD++)

Instructor: Thanh Binh Nguyen

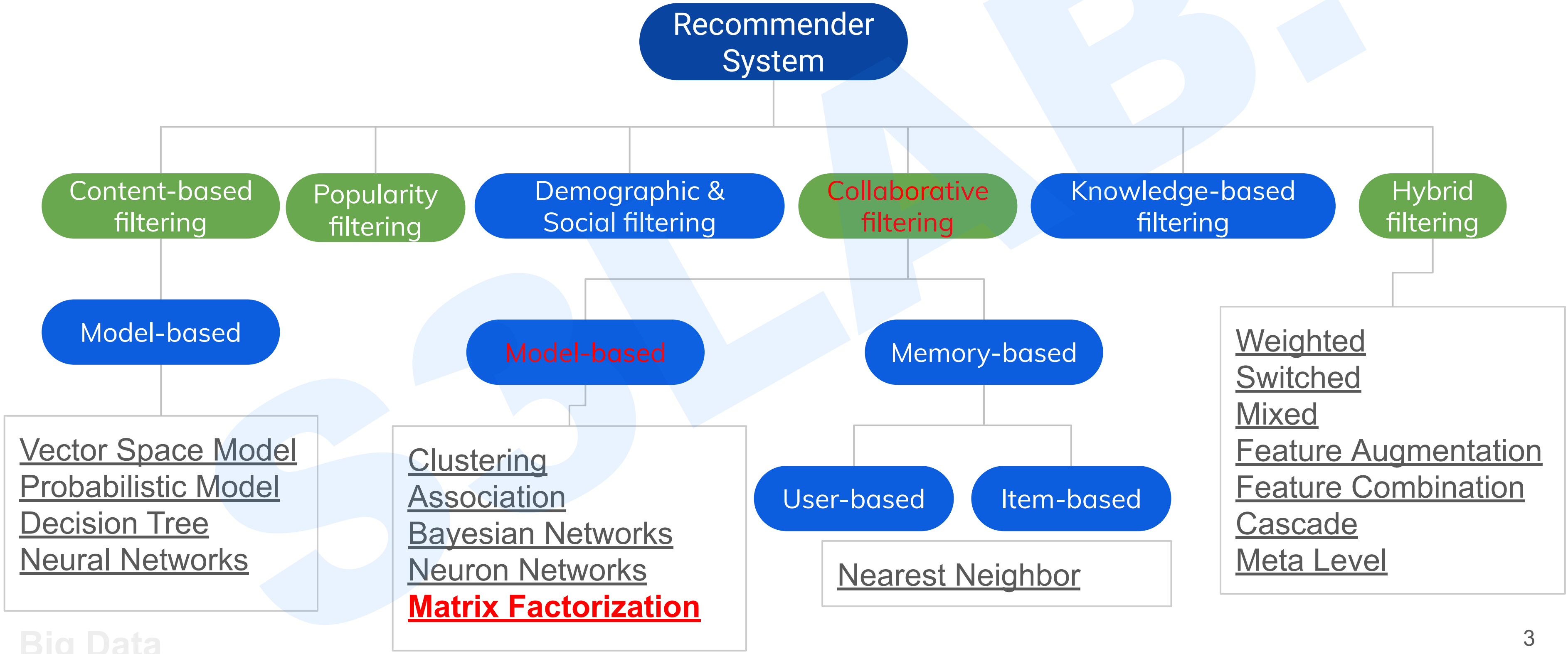
September 1st, 2019

S3Lab
Smart Software System Laboratory

“Big data is at the foundation of all the megatrends that are happening today, from social to mobile to cloud to gaming.”

– Chris Lynch, Vertica Systems

Approaches


Recommender System
Content-based filtering
Popularity filtering
Demographic & Social filtering
Collaborative filtering
Knowledge-based filtering
Hybrid filtering
Model-based
Vector Space Model
Probabilistic Model
Decision Tree
Neural Networks
Model-based
Clustering
Association
Bayesian Networks
Neuron Networks
Matrix Factorization
Memory-based
User-based
Item-based
Nearest Neighbor
Weighted
Switched
Mixed
Feature Augmentation
Feature Combination
Cascade
Meta Level

SVD++

Motivation

- An algorithm which support to latent factor models (a form of matrix factorization), one type of **collaborative filtering**.
- One of the key algorithms that contributed most to the winning algorithm of **Netflix Prize** winner.

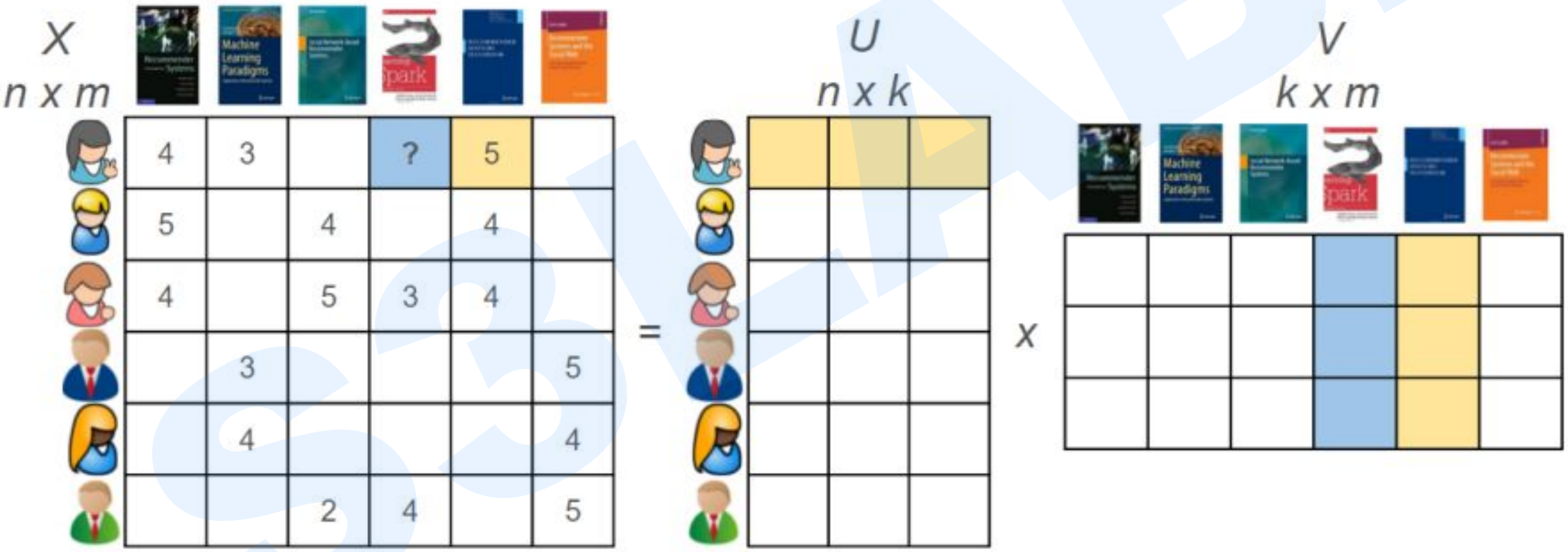
SVD++

Key Ideas

- A “**Singular Value**” (the movie rating) can be “**Decomposed**” or determined by a set of hidden latent factors (user preferences and movie features) that intuitively represent things like genre, actors, etc..
- **Latent factors** can be iteratively learned using **gradient descent** and **known movie ratings**.
- User / movie bias contribute to someone’s rating and are also learned.

SVD++

Key Ideas

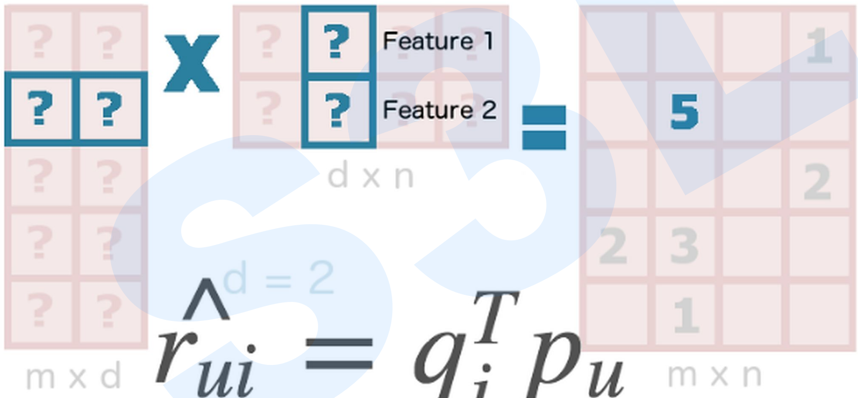


SVD++

Key Ideas

Matrix Factorization

m = number of users, n = number of items
choose d, the number of features



$$\hat{r}_{ui} = q_i^T p_u$$

$$\min_{q^*, p^*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

$$e_{ui} \stackrel{def}{=} r_{ui} - q_i^T p_u.$$

$$q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i)$$

$$p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u)$$

http://nicolas-hug.com/blog/matrix_facto_1

Movie Recommendation Problem

Model Inputs

Training epoch --->	50
Learning rate --->	0.01
L2 (lambda) penalty factor -->	0.001

Legend:

- Training data
- Test data

Model Evaluation

	Training	Test
RMSE = Root Mean Squared Error		

Movie embedding

feature1
feature2
feature3
bias

Latent Factors
(learned through training)

User embedding

feature1	feature2	feature3	bias

Latent Factors
(learned through training)

	Inside Out	Good Will Hunting	Mean Girls	Terminator	Titanic	Warrior
Tina Fey	3	1	5	1	?	1
Helen Mirren	2	?	?	2	5	1
Sylvester Stallone	1	3	1	4	2	5
Tom Hanks	?	3	1	?	4	3
George Clooney	2	2	1	3	1	4

Movie Recommendation Problem

Dataset

- Model uses 30 fictitious ratings (5 users x 6 movies).
- We'll use 25 ratings to train our model and 5 ratings to test the model's accuracy.
- Our goal is to build a system that works well on the **25 known ratings** (**training data**) and hope it predicts well on the **5 hidden** (**but known**) **ratings** (**test data**).
- If we had more data, we'd split our data into 3 groups — **training** (~70%), **validation** (~20%), and **test** (~10%) and we would use our validation set to validate our model.

https://machinelearningcoban.com/2017/03/04/overfitting/

Movie Recommendation Problem

User / Movie features

- Intuitively, these features represent things like genre, actors, length of movie, director, year of movie, etc. Even though we don't quite know what each feature is, we can intuitively guess what they could represent after we visualize them on a graph.
- I used **3 features** for simplicity, but an actual model could have 50, 100, or more. The # of features will grow in proportion to the complexity of your data. With too many features, the model will "**overfit/memorize**" your training data and it won't generalize well when it predicts the hidden (but known) test ratings.
- If a user had a high value for their 1st feature (let's assume it represents "comedies") and the the movie also had a high value for this "comedy" feature, this would likely result in a high rating for that movie.

Movie Recommendation Problem

User / Movie biases

- A user **bias** is how harsh or nice the movie critic is. If the **average** rating of all movies on Netflix is a 3.5 and the average of everything you rate is a 4.0, you would have a 0.5 bias. Movie bias can be thought of the same way. If the Titanic had an average rating of 4.25 across all users, it would have a 0.75 bias (= 4.25 – 3.50).

Movie Recommendation Problem

Model Inputs - Hyperparameter Tuning

- **Training epoch** — 1 epoch is 1 training loop through the the entire data set
- **Learning rate** — Controls how fast we adjust the weights / biases
- **L2 (lambda)** penalty factor — A term to help the model prevent **overfitting** or "**memorizing**" the training data so it can generalize on unseen test data.

Model Inputs

Select training epoch --->	50
Select learning rate --->	0.300
Select L2 (lambda) penalty factor -->	0.001

Training or Learning

Gradient Descent + Derivatives

- Gradient descent is the iterative algorithm used during training to update the values of movie features, user preferences, and biases for better predictions. The general cycle is:
    - **Step 1** — Define a cost/loss function to minimize and initialize weights
    - **Step 2** — Calculate predictions
    - **Step 3** — Calculate gradients (change in cost with respect to each weight)
    - **Step 4** — Update each weight “justttt a little bit” (the learning rate) in the direction that will minimize the cost
    - **Step 5** — Repeat steps 2–4

Training or Learning

Step 1a. Define Cost Function

$$\min_{U,M} \underbrace{\frac{1}{2}\sum_{(i,j):r(i,j)=1}\left(\hat{r}_{i,j} - r_{i,j}\right)^2}_{\text{"Error"}} + \underbrace{\frac{1}{2}\lambda\left(\sum_i \|u_i\|^2 + \sum_j \|m_j\|^2\right)}_{\text{"Weight Penalty"}}$$

"Error" + "Weight Penalty"

Find the users' latent factors (U) and the movies' latent factors (M) which minimize the sum of the:

"Error" (squared difference between the predicted movie ratings and the actual movie ratings) + the
"Weight Penalty"(Lambda x (sum of the squared user factors + sum of the squared movie factors))

Where:

U = Matrix containing all users' latent factors
M = Matrix containing all movies' latent factors
i = each ith user (i.e. Tina, Helen, Sly, Tom, George)
j = each jth movie (i.e. Inside Out, Good Will Hunting, Mean Girls, Terminator, Titanic, Warrior)
$(i, j) : r(i, j) = 1$ Only consider if user i has rated movie j in the training data; otherwise, it is ignored
$\hat{r}_{i,j}$ = Predicted movie rating for user i (i.e. Tina, Helen, ...) and movie j (i.e. Inside Out, Titanic, ...)
$r_{i,j}$ = Actual movie rating for user i and movie j
$\sum_i \|u_i\|^2$ = Sum the squares of each (i^{th}) user latent factor (excludes bias)
$\sum_j \|m_j\|^2$ = Sum the squares of each (j^{th}) movie latent factor (excludes bias)
λ = L2 penalty factor (user set) to help the model generalize and ensure weights are not too big
1/2 = Terms added to make the gradient descent math easier (to be shown)

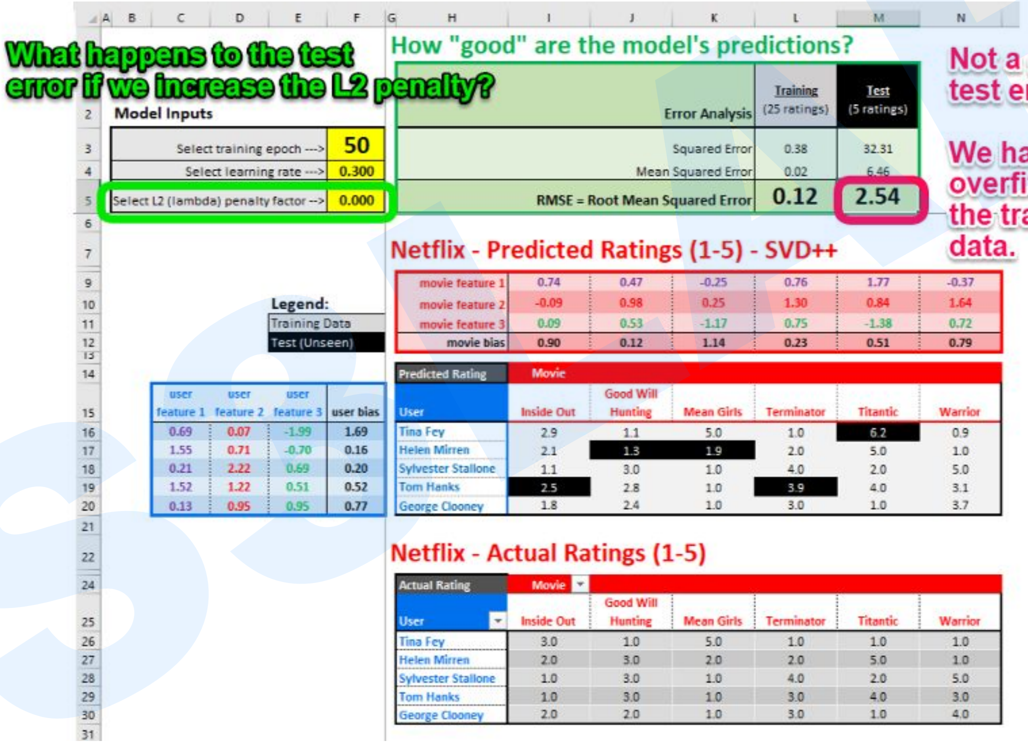
Training or Learning

Step 1a. Define Cost Function

- We add the weight penalty (L2 regularization or “ridge regression”) to prevent the latent factors from becoming too high. This ensures the model isn’t “over-fitting” (i.e. memorizing) the training data because it won’t perform well on the unseen test movies.
- Earlier, we trained the model using zero L2 regularization penalty and the RMSE training error was 0.12 after 50 epochs

Training or Learning

Step 1a. Define Cost Function



What happens to the test error if we increase the L2 penalty?

Model Inputs

Select training epoch --->	50
Select learning rate --->	0.300
Select L2 (lambda) penalty factor -->	0.000

How "good" are the model's predictions?

Error Analysis	Training (25 ratings)	Test (5 ratings)
Squared Error	0.38	32.31
Mean Squared Error	0.02	6.46
RMSE = Root Mean Squared Error	0.12	2.54

Legend:
Training Data
Test (Unseen)

Netflix - Predicted Ratings (1-5) - SVD++

movie feature 1	0.74	0.47	-0.25	0.76	1.77	-0.37
movie feature 2	-0.09	0.98	0.25	1.30	0.84	1.64
movie feature 3	0.09	0.53	-1.17	0.75	-1.38	0.72
movie bias	0.90	0.12	1.14	0.23	0.51	0.79

user feature 1	user feature 2	user feature 3	user bias
0.69	0.07	-1.99	1.69
1.55	0.71	-0.70	0.16
0.21	2.22	0.69	0.20
1.52	1.22	0.51	0.52
0.13	0.95	0.95	0.77

Predicted Rating / User	Movie: Inside Out	Good Will Hunting	Mean Girls	Terminator	Titantic	Warrior
Tina Fey	2.9	1.1	5.0	1.0	6.2	0.9
Helen Mirren	2.1	1.3	1.9	2.0	5.0	1.0
Sylvester Stallone	1.1	3.0	1.0	4.0	2.0	5.0
Tom Hanks	2.5	2.8	1.0	3.9	4.0	3.1
George Clooney	1.8	2.4	1.0	3.0	1.0	3.7

Netflix - Actual Ratings (1-5)

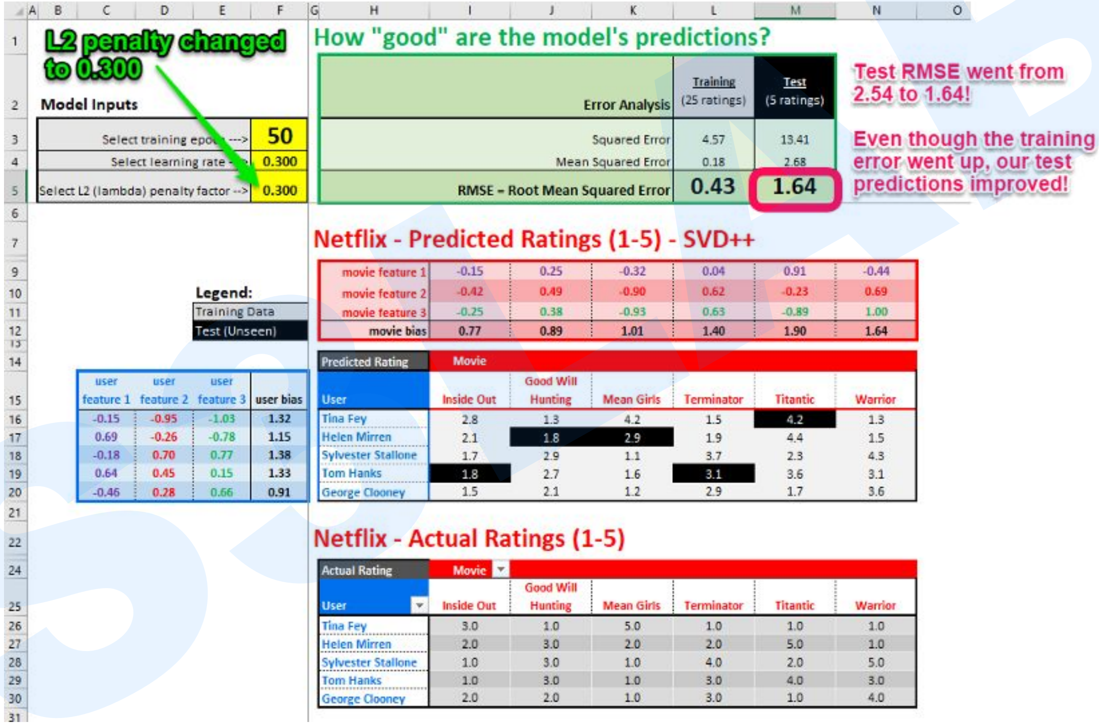
Actual Rating / User	Movie: Inside Out	Good Will Hunting	Mean Girls	Terminator	Titantic	Warrior
Tina Fey	3.0	1.0	5.0	1.0	1.0	1.0
Helen Mirren	2.0	3.0	2.0	2.0	5.0	1.0
Sylvester Stallone	1.0	3.0	1.0	4.0	2.0	5.0
Tom Hanks	1.0	3.0	1.0	3.0	4.0	3.0
George Clooney	2.0	2.0	1.0	3.0	1.0	4.0

Not a good test error....

We have overfit the training data.

Training or Learning

Step 1a. Define Cost Function



L2 penalty changed to 0.300

Model Inputs

Select training epochs --->	50
Select learning rate --->	0.300
Select L2 (lambda) penalty factor -->	0.300

How "good" are the model's predictions?

Error Analysis	Training (25 ratings)	Test (5 ratings)
Squared Error	4.57	13.41
Mean Squared Error	0.18	2.68
RMSE – Root Mean Squared Error	0.43	1.64

Test RMSE went from 2.54 to 1.64!

Even though the training error went up, our test predictions improved!

Netflix - Predicted Ratings (1-5) - SVD++

movie feature 1	-0.15	0.25	-0.32	0.04	0.91	-0.44
movie feature 2	-0.42	0.49	-0.90	0.62	-0.23	0.69
movie feature 3	-0.25	0.38	-0.93	0.63	-0.89	1.00
movie bias	0.77	0.89	1.01	1.40	1.90	1.64

Legend:
Training Data
Test (Unseen)

user feature 1	user feature 2	user feature 3	user bias
-0.15	-0.95	-1.03	1.32
0.69	-0.26	-0.78	1.15
-0.18	0.70	0.77	1.38
0.64	0.45	0.15	1.33
-0.46	0.28	0.66	0.91

Predicted Rating	Movie					
User	Inside Out	Good Will Hunting	Mean Girls	Terminator	Titantic	Warrior
Tina Fey	2.8	1.3	4.2	1.5	4.2	1.3
Helen Mirren	2.1	1.8	2.9	1.9	4.4	1.5
Sylvester Stallone	1.7	2.9	1.1	3.7	2.3	4.3
Tom Hanks	1.8	2.7	1.6	3.1	3.6	3.1
George Clooney	1.5	2.1	1.2	2.9	1.7	3.6

Netflix - Actual Ratings (1-5)

Actual Rating	Movie					
User	Inside Out	Good Will Hunting	Mean Girls	Terminator	Titantic	Warrior
Tina Fey	3.0	1.0	5.0	1.0	1.0	1.0
Helen Mirren	2.0	3.0	2.0	2.0	5.0	1.0
Sylvester Stallone	1.0	3.0	1.0	4.0	2.0	5.0
Tom Hanks	1.0	3.0	1.0	3.0	4.0	3.0
George Clooney	2.0	2.0	1.0	3.0	1.0	4.0

Training or Learning

Step 1b. Weight Initialization

- At the start of training, weights are randomly assigned to the user/movie features and then the algorithm learns the optimal weights during training. 2 weight initialization approaches:
    - **Simple** — I randomly chose 0.1, 0.2, and 0.3 for the user features and left 0.1 for everything else.
    - “**Kaiming He**” — A more formal/better initialization approach which randomly chooses weights across a Gaussian distribution (“bell curve”) using a mean of zero and a standard deviation determined by the # of features

Training or Learning

Step 1b. Kaiming He

- Weights = Random sample from a normal distribution using a mean of 0 and a standard deviation of **(=SquareRoot(2 / # of features))**.
- Excel formula used to find the values in the spreadsheet: **=NORMINV(RAND(),0,SQRT(2/3))**

$$W_l \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n_l}}\right) \text{ and } \mathrm{b} = 0.$$

Training or Learning

Step 1b. Weight Initialization

Kaiming He - Predicted Ratings vs. Actual

How "good" are the model's predictions?

Model Inputs

Select training epoch --->	50
Select learning rate --->	0.300
Select L2 (lambda) penalty factor -->	0.000

Error Analysis	Training (25 ratings)
Squared Error	0.38
Mean Squared Error	0.02
RMSE = Root Mean Squared Error	**0.12**

Legend:

- Training Data
- Test (Unseen)

Netflix - Predicted Ratings (1-5) - SVD++

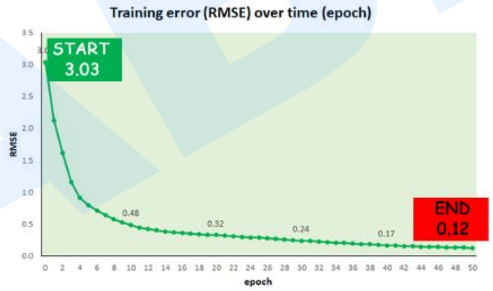
movie feature 1	0.74	0.47	-0.25	0.76	1.77	-0.37
movie feature 2	-0.09	0.98	0.25	1.30	0.84	1.64
movie feature 3	0.09	0.53	-1.17	0.75	-1.38	0.72
movie bias	0.90	0.12	1.14	0.23	0.51	0.79

user feature 1	user feature 2	user feature 3	user bias	Predicted Rating / User	Movie: Inside Out	Good Will Hunting	Mean Girls	Terminator	Titantic	Warrior
0.69	0.07	-1.99	1.69	Tina Fey	2.9	1.1	5.0	1.0	6.2	0.9
1.55	0.71	-0.70	0.16	Helen Mirren	2.1	1.3	1.9	2.0	5.0	1.0
0.21	2.22	0.69	0.20	Sylvester Stallone	1.1	3.0	1.0	4.0	2.0	5.0
1.52	1.22	0.51	0.52	Tom Hanks	2.5	2.8	1.0	3.9	4.0	3.1
0.13	0.95	0.95	0.77	George Clooney	1.8	2.4	1.0	3.0	1.0	3.7

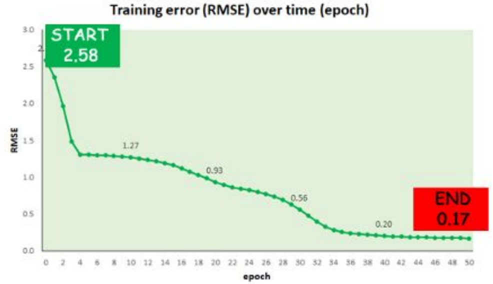
Netflix - Actual Ratings (1-5)

Actual Rating / User	Movie: Inside Out	Good Will Hunting	Mean Girls	Terminator	Titantic	Warrior
Tina Fey	3.0	1.0	5.0	1.0	1.0	1.0
Helen Mirren	2.0	3.0	2.0	2.0	5.0	1.0
Sylvester Stallone	1.0	3.0	1.0	4.0	2.0	5.0
Tom Hanks	1.0	3.0	1.0	3.0	4.0	3.0
George Clooney	2.0	2.0	1.0	3.0	1.0	4.0

Weight Initialization - Kaiming He



Weight Initialization - Simple



Training or Learning

Step 2. Calculate Predictions

$$\hat{r}_{i,j} = ((u_1 m_1) + (u_2 m_2) + (u_3 m_3) + u_{bias} + m_{bias})$$

Where:

$\hat{r}_{i,j}$ = Predicted movie rating for user i (i.e. Tina, Helen, ...) and movie j (i.e. Inside Out, Titanic, ...)

u_1, u_2, u_3 = Users' latent factors

m_1, m_2, m_3 = Movies' latent factors

u_{bias} = User bias

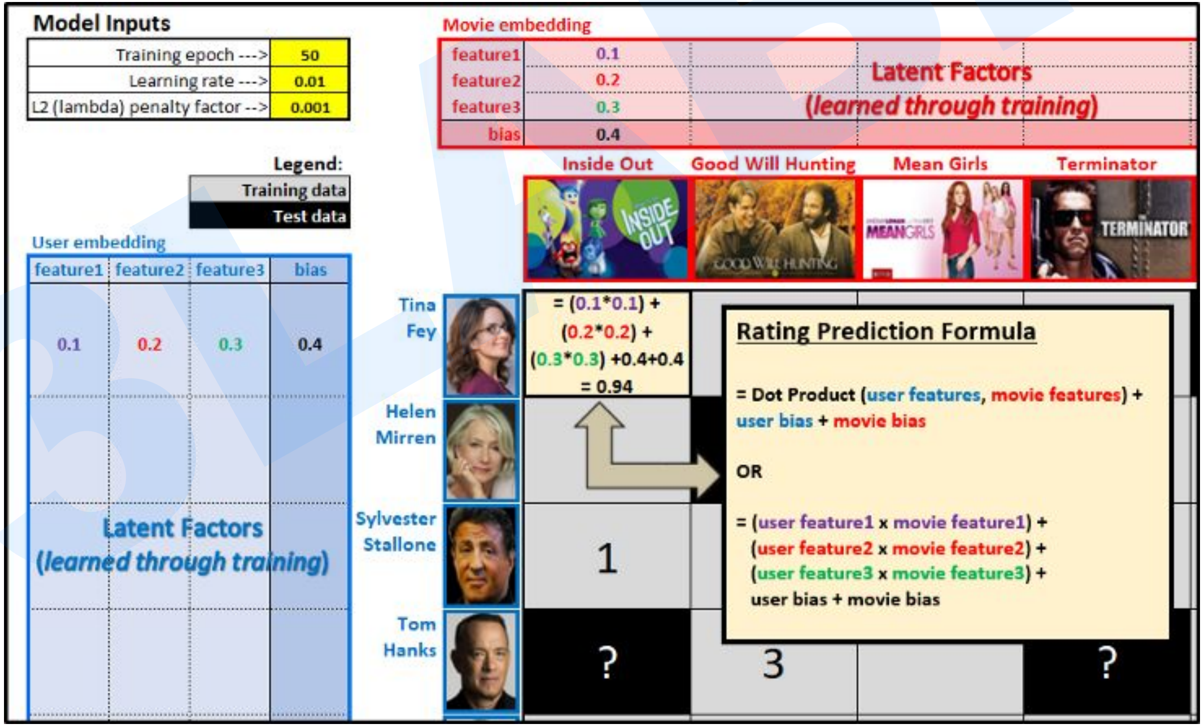
m_{bias} = Movie bias

SUM | =IF($I11="Ignore",0,SUMPRODUCT(J11:L11,N11:P11)+M11+Q11)

	A	B	I	J	K	L	M	N	O	P	Q	R
1			epoch -->	0	0	0	0	0	0	0	0	0
3		learning rate	0.300									
4		L2 penalty factor	0.000									
6		Total Squared Error										
7		Mean Squared Error										
8		RMSE = Root Mean Squared Error		Initial wt	Initial wt	Initial wt	Initial wt	Initial wt	Initial wt	Initial wt	Initial wt	
10	User	Movie	actual	user feature 1	user feature 2	user feature 3	user bias	movie feature 1	movie feature 2	movie feature 3	movie bias	prediction
11	Tina Fey	Inside Out	3.0	-0.66	-0.56	-1.17	0.10	0.56	-0.14	0.36	0.10	=IF($I11="
12	Tina Fey	Good Will Hunting	1.0	-0.66	-0.56	-1.17	0.10	-0.02	0.95	0.21	0.10	-0.56
13	Tina Fey	Mean Girls	5.0	-0.66	-0.56	-1.17	0.10	0.61	0.63	0.59	0.10	-1.24
14	Tina Fey	Terminator	1.0	-0.66	-0.56	-1.17	0.10	-0.23	0.19	0.52	0.10	-0.36
15	Tina Fey	Titantic	Ignore	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
16	Tina Fey	Warrior	1.0	-0.66	-0.56	-1.17	0.10	1.07	0.64	0.96	0.10	-1.98

Training or Learning

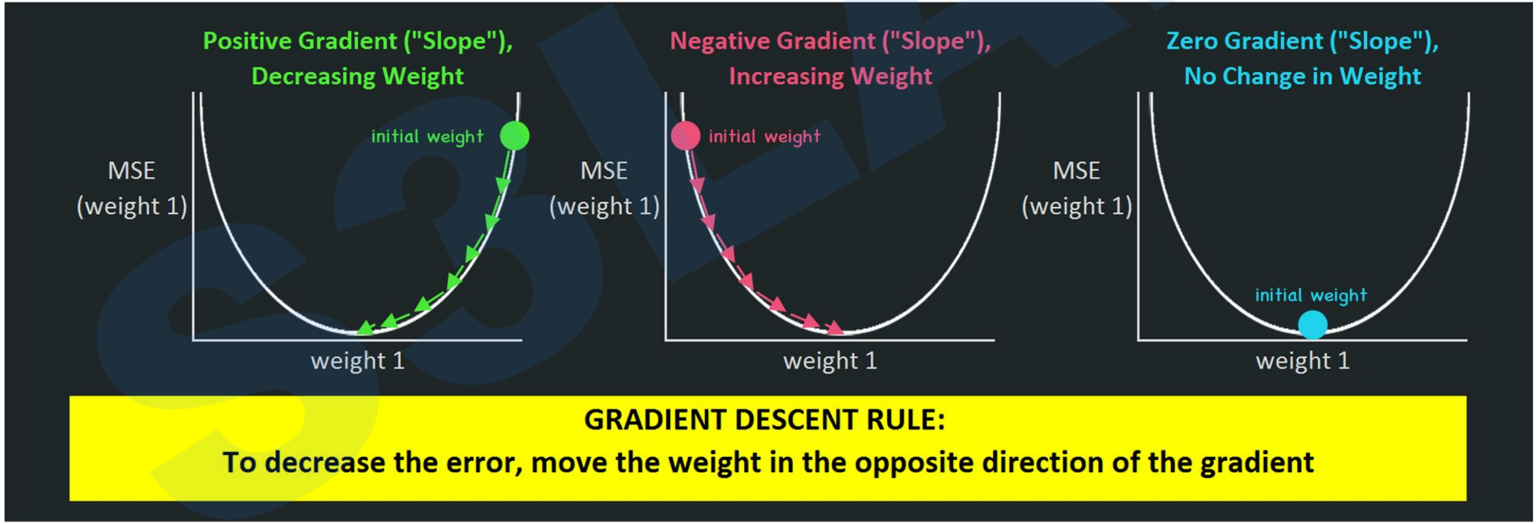
Step 2. Calculate Predictions



Training or Learning

Step 3. Calculate Gradients

- The goal is to find the gradient ("slope") of the error with respect to the weight you are updating.



Training or Learning

Step 3. Calculate Gradients

Step 1: Calculate the gradient of the cost with respect to the 1st latent factor for Tina Fey (u_1).

1.1 Re-write the cost objective function and take the partial derivative of the cost with respect to the 1^{st} latent factor (u_1) for Tina Fey.

$$\frac{\partial J(cost)}{\partial u_1} = \frac{1}{2}\sum_{(i,j):r(i,j)=1}\left(\hat{r}_{i,j} - r_{i,j}\right)^2 + \frac{1}{2}\lambda\left(\sum_i \|u_i\|^2 + \sum_j \|m_j\|^2\right)$$

1.2 Re-write the predicted rating function and re-write the sum of the users' latent factors squared and the movies' latent factors squared.

$$\frac{\partial J}{\partial u_1} = \frac{1}{2}\sum(((u_1 m_1) + (u_2 m_2) + (u_3 m_3) + u_{bias} + m_{bias}) - r_{i,j})^2 + \frac{1}{2}\lambda\sum\left(u_1^2 + u_2^2 + u_3^2\right) + \frac{1}{2}\lambda\sum\left(m_1^2 + m_2^2 + m_3^2\right)$$

1.3 Treat u_1 as a constant in each part of the 3 parts of the formula to find the partial derivative of u_1 with respect to the cost in each part of the formula.

Part 1

$$\frac{\partial J}{\partial u_1} = \frac{1}{2}\sum(((u_1 m_1) + (u_2 m_2) + (u_3 m_3) + u_{bias} + m_{bias}) - r_{i,j})^2 +$$

Part 2 *Part 3*

$$\frac{1}{2}\lambda\sum\left(u_1^2 + u_2^2 + u_3^2\right) + \frac{1}{2}\lambda\sum\left(m_1^2 + m_2^2 + m_3^2\right)$$

Training or Learning

Step 3. Calculate Gradients

1.3.1 (Part 1 of 3) - Use the 'chain rule' to find the partial derivative. The chain rule means we take the ((derivative of the outer function) x the inner function) x (the derivative of the inner function).

<u>Derivative of outer function x inner function</u>

$$= \frac{1}{2}\sum(((u_1m_1) + (u_2m_2) + (u_3m_3) + u_{bias} + m_{bias}) - r_{i,j})^2 \leftarrow \textit{power rule}$$

$$= 2 \times \frac{1}{2}(((u_1m_1) + (u_2m_2) + (u_3m_3) + u_{bias} + m_{bias}) - r_{i,j})^1$$

$$= (\textit{predicted rating} - \textit{actual rating})$$

$$= (\textit{error})$$

<u>Derivative of inner function</u>

$$= \frac{1}{2}\sum((u_1m_1) + (u_2m_2) + (u_3m_3) + u_{bias} + m_{bias}) - r_{i,j} \leftarrow \textit{constant rule}$$

$$((u_1m_1) + (0) + (0) + 0 + 0) - 0$$

$$= m_1$$

1.3.1 = $(\textit{error } x \; m_1)$

Training or Learning

Step 3. Calculate Gradients

1.3.1 (Part 2 of 3) Use the 'power rule' to find the partial derivative. Using the power rule, we identify the power (2), multiply it by the coefficient, $\frac{1}{2}$, and reduce the power by 1. u_2 and u_3 are treated as constants and become zero.

$$= \frac{1}{2}\lambda\sum\left(u_1^{\ 2} + u_2^{\ 2} + u_3^{\ 2}\right)$$

$$= 2 \times \frac{1}{2} \times \lambda\left(u_1^{\ 1} + 0 + 0\right)$$

$$= \lambda \times u_1$$

1.3.2 = $(\lambda \times u_1)$

1.3.3 (Part 3 of 3) Use the 'constant rule' to find the partial derivative in part 3.

$$= \frac{1}{2}\lambda\sum\left(m_1^{\ 2} + m_2^{\ 2} + m_3^{\ 2}\right)$$

$$= 0$$

Since u_1 has no impact on any of these terms, this becomes zero.

1.3.3 = 0

1.4 Combine part 1.3.1, 1.3.2, and 1.3.3 to find the final partial derivative of the cost with respect to u1.

Part 1 + *Part 2* + *Part 3*

$$\frac{\partial J}{\partial u_1} = (error\ x\ m_1) + (\lambda \times u_1) + 0$$

$$= (error\ x\ m_1) + (\lambda \times u_1)$$

Training or Learning

Step 3. Calculate Gradients

Step 2: For each movie in the training data which Tina has seen, calculate the gradient using the formula from step 1.4 and then calculate the average gradient for all movies that she saw.

	Inside Out	*Good Will Hunting*	*Mean Girls*	*Terminator*	*Titanic*	*Warrior*	**Average**
predicted rating	(0.52)	(0.56)	(1.24)	(0.36)	Ignore	(1.98)	
actual rating	3.0	1.0	5.0	1.0	Ignore	1.0	
error	(3.52)	(1.56)	(6.24)	(1.36)	Ignore	(2.98)	
m_1	0.56	(0.02)	0.61	(0.23)	Ignore	1.07	
$(error \times m_1)$	(1.98)	0.03	(3.80)	(0.31)	Ignore	(3.19)	
λ	0.300	0.300	0.300	0.300	Ignore	0.300	
u_1	(0.66)	(0.66)	(0.66)	(0.66)	Ignore	(0.66)	
$(\lambda \times u_1)$	(0.20)	(0.20)	(0.20)	(0.20)	Ignore	(0.20)	
$gradient = (error \times m_1) + (\lambda \times u_1)$	(2.18)	(0.17)	(3.99)	0.11	Ignore	(3.39)	**(1.92)** = ((2.18)+ (0.17)+ (3.99)+ 0.11+ (3.39)) / 5

Training or Learning

Step 4. Update the Weights

Using Tina's old u_1, the learning rate (α), and the average gradient calculated above, update u_1. The learning rate we'll use is 0.300.

Gradient descent formula:

$$New\ u_1 = old\ u_1 - \alpha\ (average\ gradient)$$
$$New\ u_1 = (0.66) - 0.3\ ((1.92))$$
$$New\ u_1 = (0.66) + 0.58$$

New u_1 = (0.08)

Training or Learning

Step 4. Update the Weights

SUM

```
=IF($I11="Ignore",0, J11-$I$3*((2*1/2)*
   (SUMPRODUCT((($11:$16)^(2-1)),N11:N16))+
   SUMPRODUCT(T11:T16,J11:J16))/
   COUNTIF($I11:$I16,"<>"&"Ignore"))
```

	A	B	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1			epoch -->	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3		learning rate	0.300															
4		L2 penalty factor	0.300															
6		Total Squared Error											229.73					
7		Mean Squared Error											9.19				*minimize*	
8		**RMSE = Root Mean Squared Error**		*Initial Wt*	*Initial Wt*	*Initial Wt*	*Initial Wt*	*Initial Wt*	*Initial Wt*	*Initial Wt*	*Initial Wt*		**3.03**				*this*	
10	User	Movie	actual	user feature 1	user feature 2	user feature 3	user bias	movie feature 1	movie feature 2	movie feature 3	movie bias	prediction	error (prediction - actual)	L2 penalty factor	Sum of all user features squared	Sum of all movie features squared	squared error + L2	user feature 1
11	Tina Fey	Inside Out	3.0	-0.66	-0.56	-1.17	0.10	0.56	-0.14	0.36	0.10	-0.52	-3.52	0.30	2.11	0.47	13.13	COUNT
12	Tina Fey	Good Will Hunting	1.0	-0.66	-0.56	-1.17	0.10	-0.02	0.95	0.21	0.10	-0.56	-1.56	0.30	2.11	0.95	3.36	-0.08
13	Tina Fey	Mean Girls	5.0	-0.66	-0.56	-1.17	0.10	0.61	0.63	0.59	0.10	-1.24	-6.24	0.30	2.11	1.12	39.95	-0.08
14	Tina Fey	Terminator	1.0	-0.66	-0.56	-1.17	0.10	-0.23	0.19	0.52	0.10	-0.36	-1.36	0.30	2.11	0.35	2.59	-0.08
15	Tina Fey	Titantic	Ignore	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
16	Tina Fey	Warrior	1.0	-0.66	-0.56	-1.17	0.10	1.07	0.64	0.96	0.10	-1.98	-2.98	0.30	2.11	2.48	10.29	-0.08

RMSE - Root Mean Squared Error

- “on average, how many stars (1–5) did your predicted ratings differ vs. the actual ratings”?
- We only care about the **absolute differences**. A prediction that is 1 higher than the actual rating has the same error, 1, as a prediction that is 1 lower than the actual rating.
- **RMSE** is an **average of magnitude** of the error which isn’t the same as the **absolute average error**. In our example above, the **absolute average error** was **0.75** (1 + 1 + 0.25 = 2.25/3 = 0.75), but the **RMSE** was **0.8292**. RMSE gives a higher weight to large errors which is useful when large errors are undesirable.

RMSE - Root Mean Squared Error

			RMSE in 4 Steps	
			Step 1 = Error Calculate Error (Prediction - Actual)	**Step 2 = Squared** Square the Error
	Actual Ratings	Predictions		
				
Tina Fey 	3	2	-1	1
Helen Mirren 	2	3	1	1
Sylvester Stallone 	1	1.25	0.25	0.0625

Total	2.0625	**Step 3 = Mean** Sum the squared errors and calculate the mean ("average")
Mean	0.6875	
RMSE	**0.8292**	**Step 4 = Root** Take square root of the mean

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{(i,j):r(i,j)=1}^{n}\left(\hat{r}_{i,j} - r_{i,j}\right)^2}$$

Where:

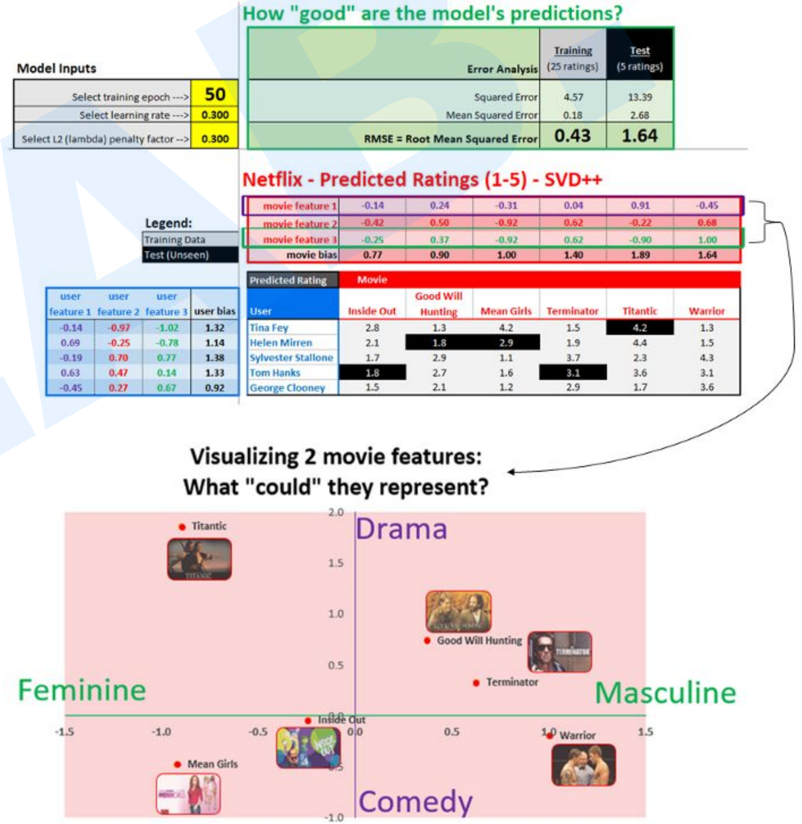
$(i, j): r(i, j) = 1$ Only consider if user i has rated movie j in the training data; otherwise, it is ignored

$\hat{r}_{i,j}$ = Predicted movie rating for user i (i.e. Tina, Helen, ...) and movie j (i.e. Inside Out, Titanic, ...)

$r_{i,j}$ = Actual movie rating for user i and movie j

Model Evaluation & Visualizations

- If our model was more complex and it had 10, 20, or 50+ latent factors, we could use a technique called “Principal component analysis (PCA)” to extract the most important features and then visualize them on a graph.



Model Inputs

Select training epoch --->	50
Select learning rate --->	0.300
Select L2 (lambda) penalty factor -->	0.300

How "good" are the model's predictions?

Error Analysis	Training (25 ratings)	Test (5 ratings)
Squared Error	4.57	13.39
Mean Squared Error	0.18	2.68
RMSE = Root Mean Squared Error	0.43	1.64

Netflix - Predicted Ratings (1-5) - SVD++

Legend:
Training Data
Test (Unseen)

movie feature 1	-0.14	0.24	-0.31	0.04	0.91	-0.45
movie feature 2	-0.42	0.50	-0.92	0.62	-0.22	0.68
movie feature 3	-0.25	0.37	-0.92	0.62	-0.90	1.00
movie bias	0.77	0.90	1.00	1.40	1.89	1.64

user feature 1	user feature 2	user feature 3	user bias	Predicted Rating / User	Movie: Inside Out	Good Will Hunting	Mean Girls	Terminator	Titantic	Warrior
-0.14	-0.97	-1.02	1.32	Tina Fey	2.8	1.3	4.2	1.5	4.2	1.3
0.69	-0.25	-0.78	1.14	Helen Mirren	2.1	1.8	2.9	1.9	4.4	1.5
-0.19	0.70	0.77	1.38	Sylvester Stallone	1.7	2.9	1.1	3.7	2.3	4.3
0.63	0.47	0.14	1.33	Tom Hanks	1.8	2.7	1.6	3.1	3.6	3.1
-0.45	0.27	0.67	0.92	George Clooney	1.5	2.1	1.2	2.9	1.7	3.6

Q & A

Cảm ơn đã theo dõi

Chúng tôi hy vọng cùng nhau đi đến thành công.