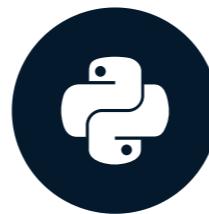


Matrix Multiplication

BUILDING RECOMMENDATION ENGINES WITH PYSPARK



Jamen Long

Data Scientist at Nike

Matrix Multiplication

1	2	3
4	5	6
7	8	9

•

9	8	7
6	5	4
3	2	1

Matrix Multiplication

1	2	3
4	5	6
7	8	9

•

9	8	7
6	5	4
3	2	1

=

(1*9)		

Matrix Multiplication

The diagram illustrates the multiplication of two 3x3 matrices. The first matrix (left) has columns labeled 1, 2, and 3. A red arrow points from the first column to the second column. The second matrix (middle) has rows labeled 9, 6, 3. A red arrow points from the top row to the middle row. The result (right) is shown as a green matrix with the expression $(1*9) + (2*6)$ in the top-left cell.

1	2	3
4	5	6
7	8	9

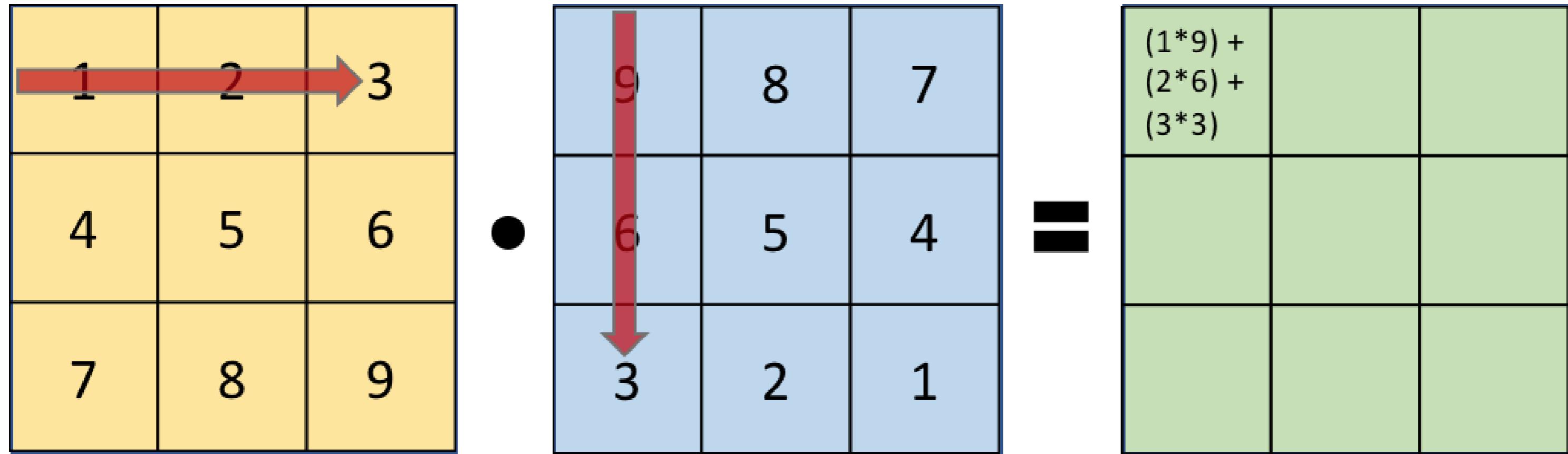
•

9	8	7
6	5	4
3	2	1

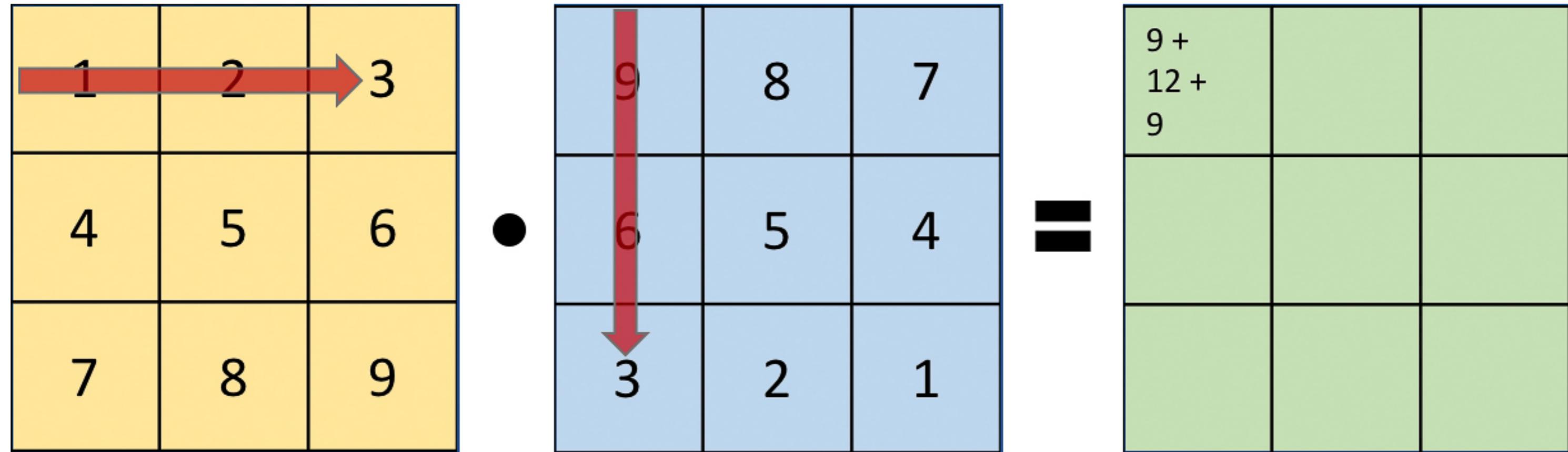
=

$(1*9) + (2*6)$		

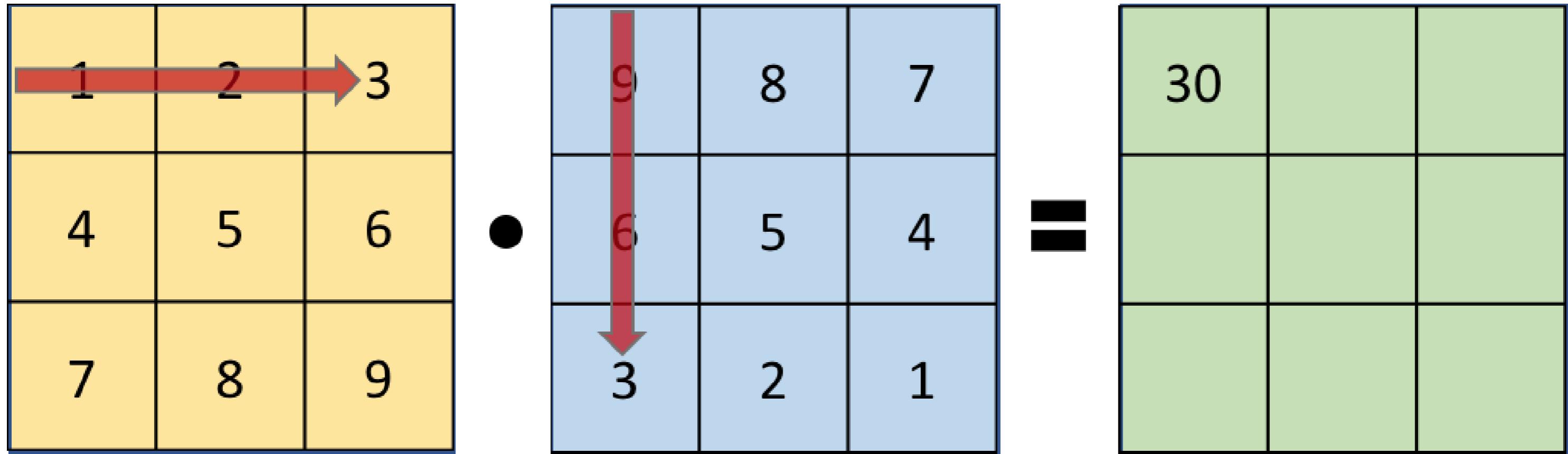
Matrix Multiplication



Matrix Multiplication



Matrix Multiplication



Matrix Multiplication

The diagram illustrates the multiplication of two 3x3 matrices. The first matrix, on the left, has columns labeled 1, 2, and 3. A red arrow points to the first column. The second matrix, in the center, has rows labeled 8 and 9. A red arrow points to the second row. The result of the multiplication is shown on the right, consisting of three rows. The first row contains the value 30 and the label $(1 * 8)$. The other two rows are empty.

1	2	3
4	5	6
7	8	9

•

9	8	7
6	5	4
3	2	1

=

30	$(1 * 8)$	

Matrix Multiplication

The diagram illustrates the multiplication of two 3x3 matrices. The first matrix (left) has columns labeled 1, 2, and 3. The second matrix (middle) has rows labeled 8, 5, and 2. The result (right) shows the product and the calculation for the top-left element.

1	2	3
4	5	6
7	8	9

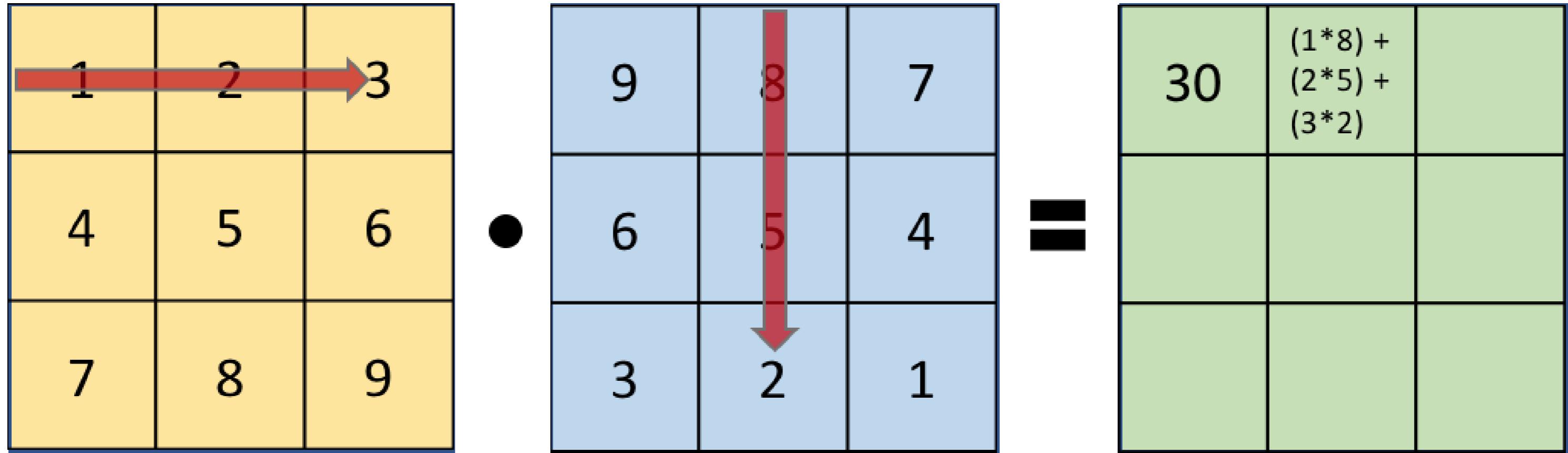
•

9	8	7
6	5	4
3	2	1

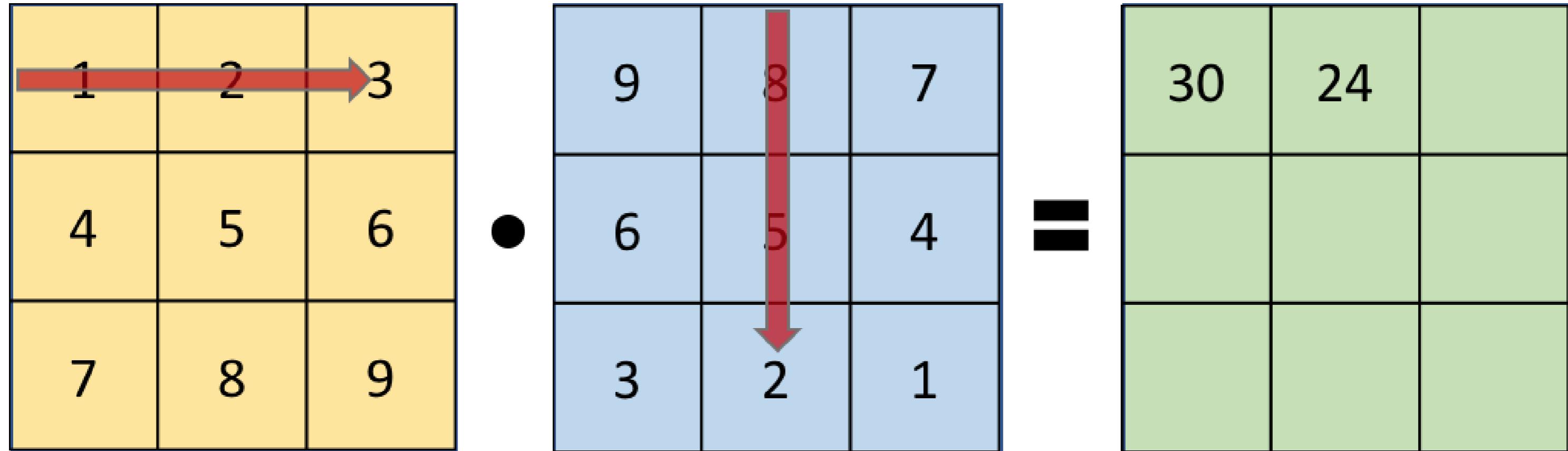
=

30	$(1*8) + (2*5)$	

Matrix Multiplication



Matrix Multiplication



Matrix Multiplication

1	2	3
4	5	6
7	8	9

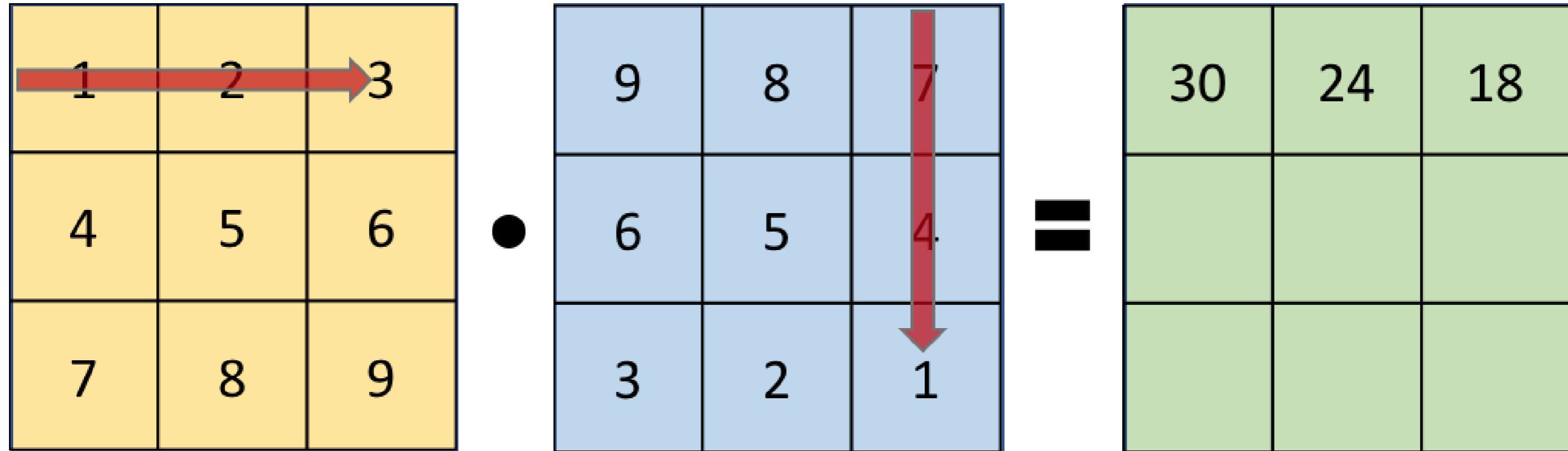
•

9	8	7
6	5	4
3	2	1

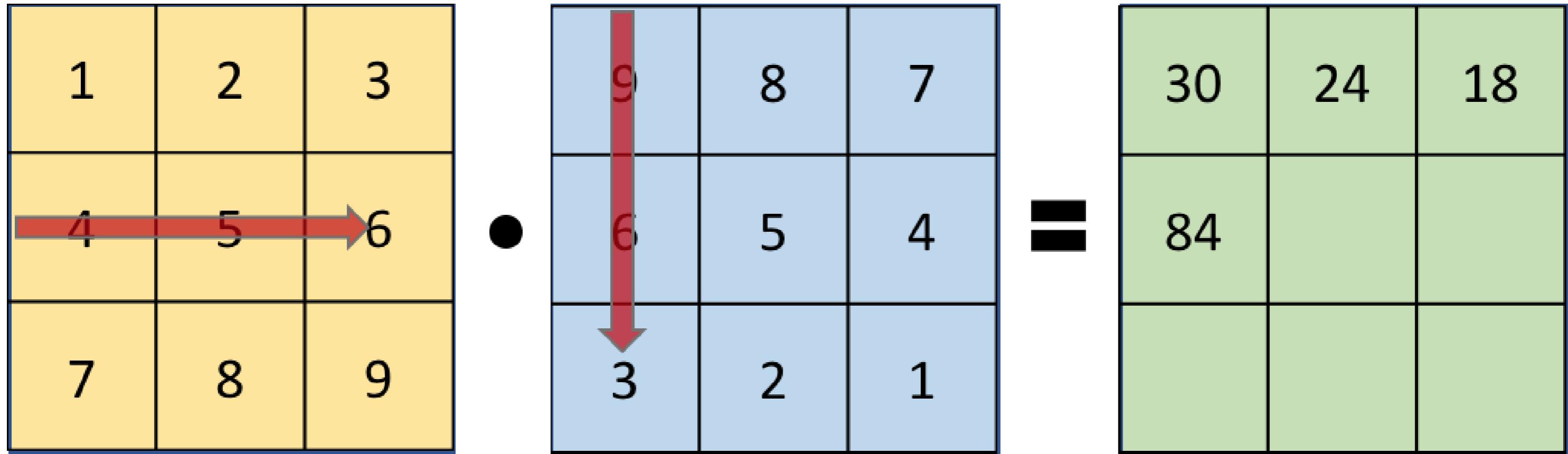
=

30	24	$(1*7) + (2*4) + (3*1)$

Matrix Multiplication



Matrix Multiplication

$$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix} \bullet \begin{matrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{matrix} = \begin{matrix} 30 & 24 & 18 \\ 84 & & \end{matrix}$$


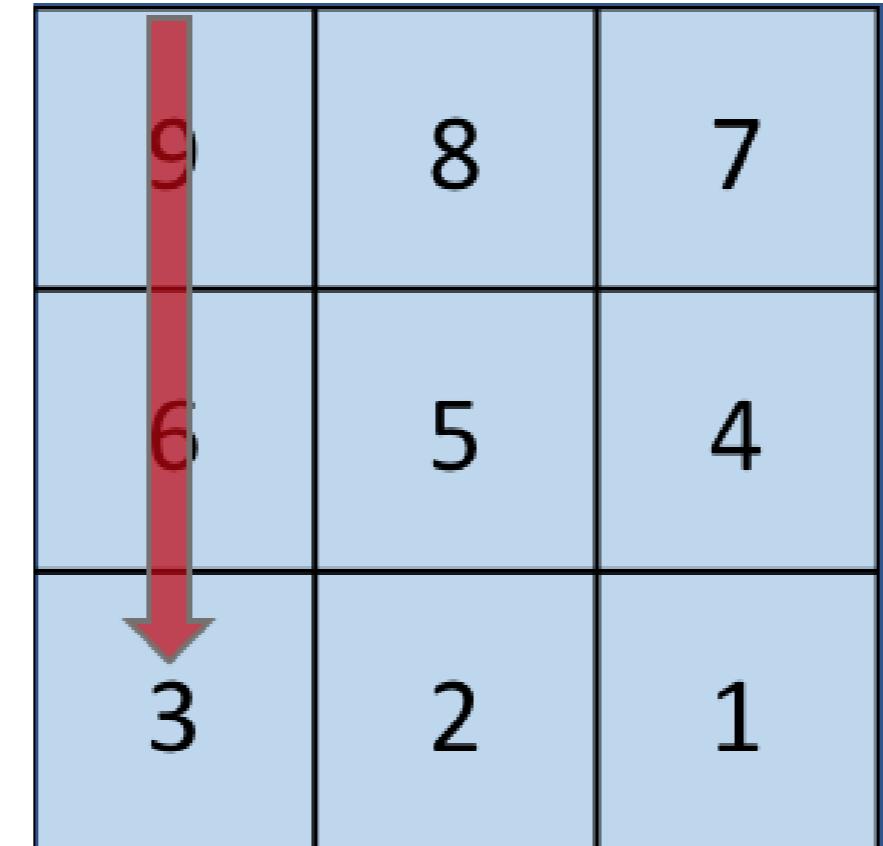
The diagram illustrates the multiplication of two 3x3 matrices. The first matrix has columns labeled 1, 2, 3 and rows labeled 4, 5, 6. The second matrix has columns labeled 9, 8, 7 and rows labeled 6, 5, 4. The result matrix has columns labeled 30, 24, 18 and rows labeled 84, . The middle column of the second matrix is highlighted in red, and a red arrow points from the bottom row of the first matrix to this column, indicating the calculation of the element at position (1,3) in the result matrix.

Matrix Multiplication

$$\begin{matrix} 1 & 2 & 3 \\ & & \\ 4 & 5 & 6 \\ & & \end{matrix} \bullet \begin{matrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \\ \end{matrix} = \begin{matrix} 30 & 24 & 18 \\ 84 & 69 & 54 \\ & & \end{matrix}$$

The diagram illustrates the multiplication of two 3x3 matrices. The first matrix has columns labeled 1, 2, 3 and rows labeled 4, 5, 6. The second matrix has columns labeled 9, 8, 7 and rows labeled 6, 5, 4. The result matrix has columns labeled 30, 24, 18 and rows labeled 84, 69, 54. Red arrows highlight the calculation of the first element in the result matrix: it is the sum of the products of the first row of the first matrix and the first column of the second matrix. Specifically, it is calculated as $(1 \cdot 9) + (2 \cdot 6) + (3 \cdot 3) = 30$.

Matrix Multiplication

$$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix} \bullet \begin{matrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{matrix} = \begin{matrix} 30 & 24 & 18 \\ 84 & 69 & 54 \\ 138 & & \end{matrix}$$


Matrix Multiplication

$$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix} \bullet \begin{matrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{matrix} = \begin{matrix} 30 & 24 & 18 \\ 84 & 69 & 54 \\ 138 & 114 & \end{matrix}$$

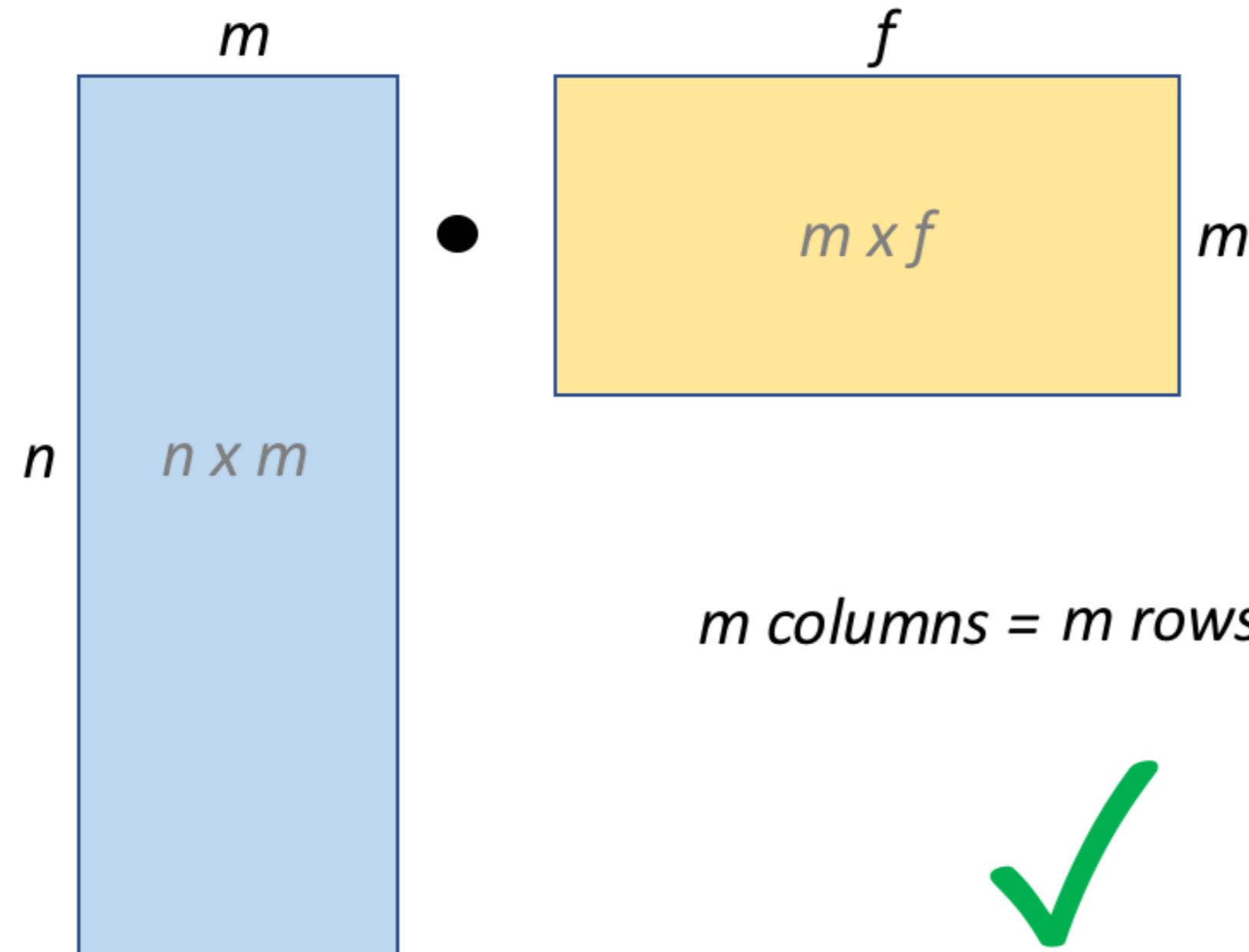
The diagram illustrates the multiplication of two 3x3 matrices. The first matrix has columns 1, 2, and 3. The second matrix has rows 1, 2, and 3. The result matrix has columns 1, 2, and 3. A red arrow points from the third column of the first matrix to the second row of the second matrix, indicating the calculation of the element at the intersection of the third column of the first matrix and the second row of the second matrix.

Matrix Multiplication

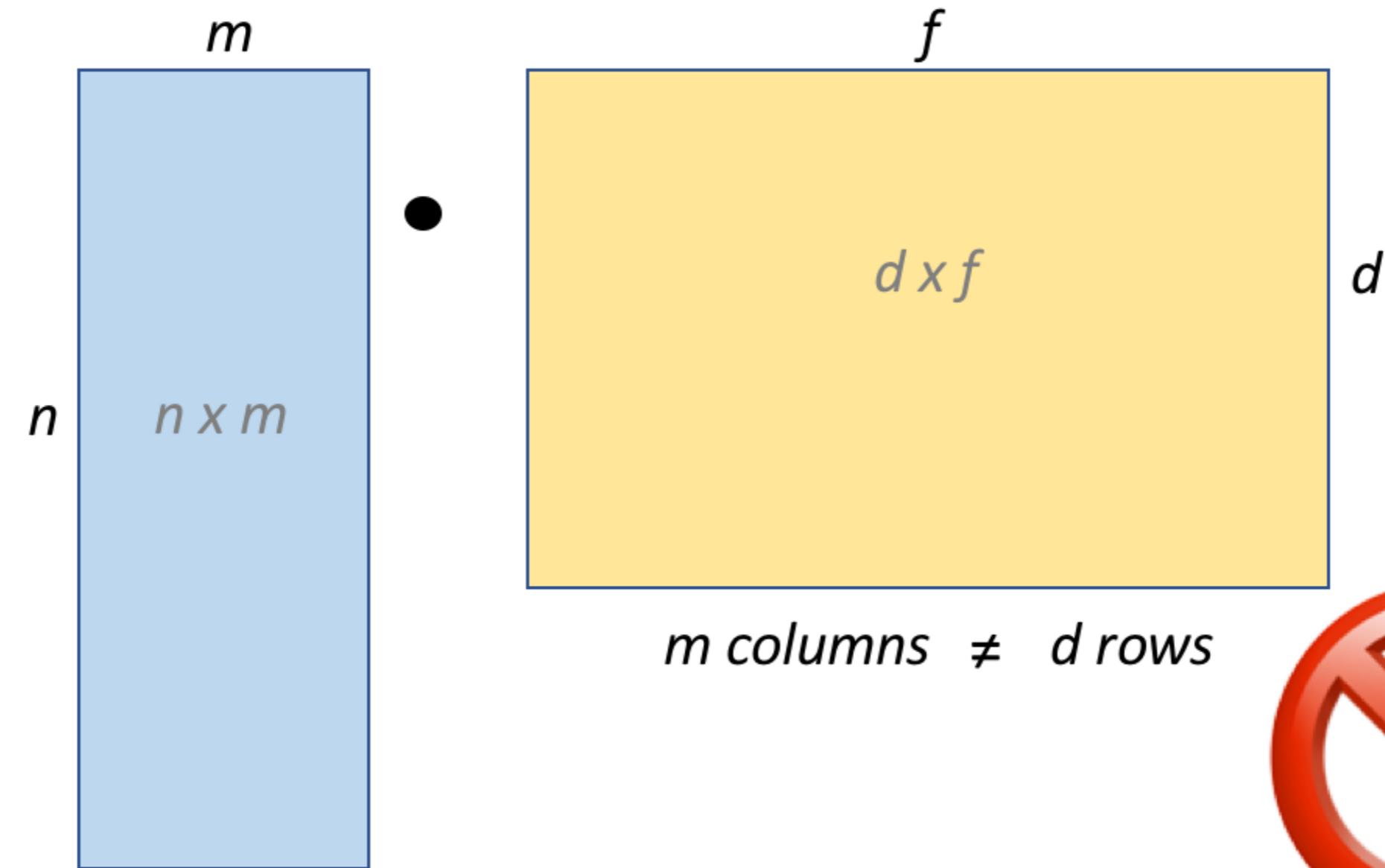
$$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix} \bullet \begin{matrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{matrix} = \begin{matrix} 30 & 24 & 18 \\ 84 & 69 & 54 \\ 138 & 114 & 90 \end{matrix}$$

The diagram illustrates the multiplication of two 3x3 matrices. The first matrix has columns 1, 2, and 3. The second matrix has rows 1, 2, and 3. The result matrix has columns 1, 2, and 3. A red arrow points from the third column of the first matrix to the third row of the second matrix, indicating the calculation of the element at the intersection of the third row and third column of the result matrix. The value 9 is highlighted in red in both the third column of the first matrix and the third row of the second matrix.

Matrix Multiplication



Matrix Multiplication

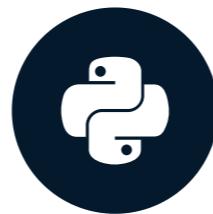


Let's practice!

BUILDING RECOMMENDATION ENGINES WITH PYSPARK

Overview of matrix factorization

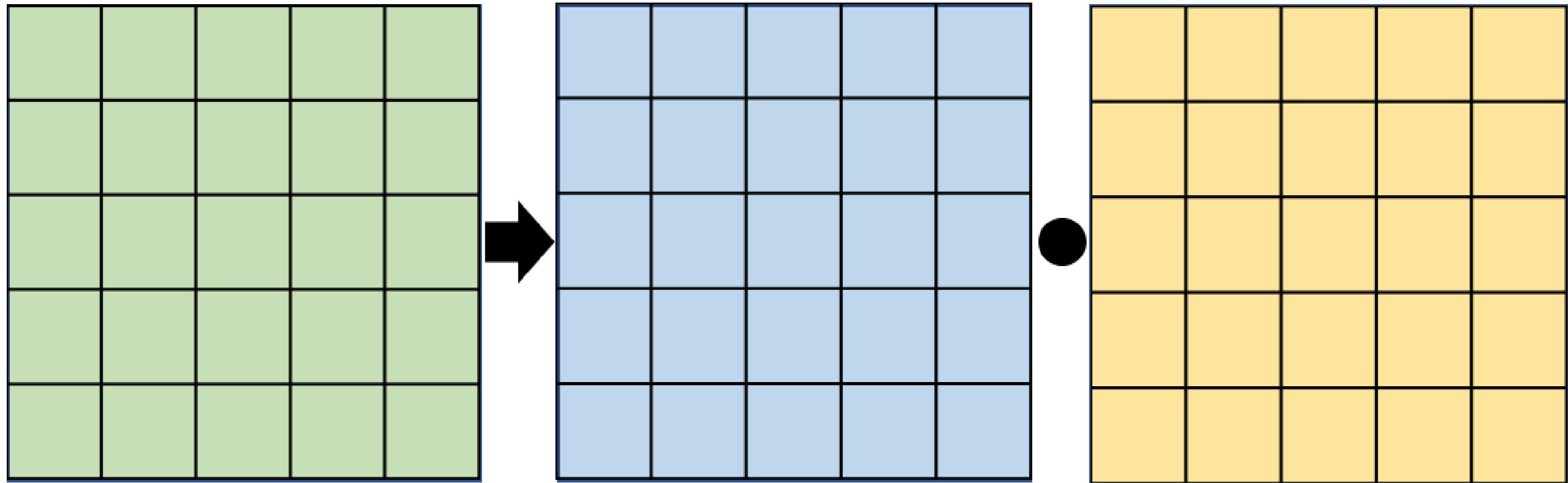
BUILDING RECOMMENDATION ENGINES WITH PYSPARK



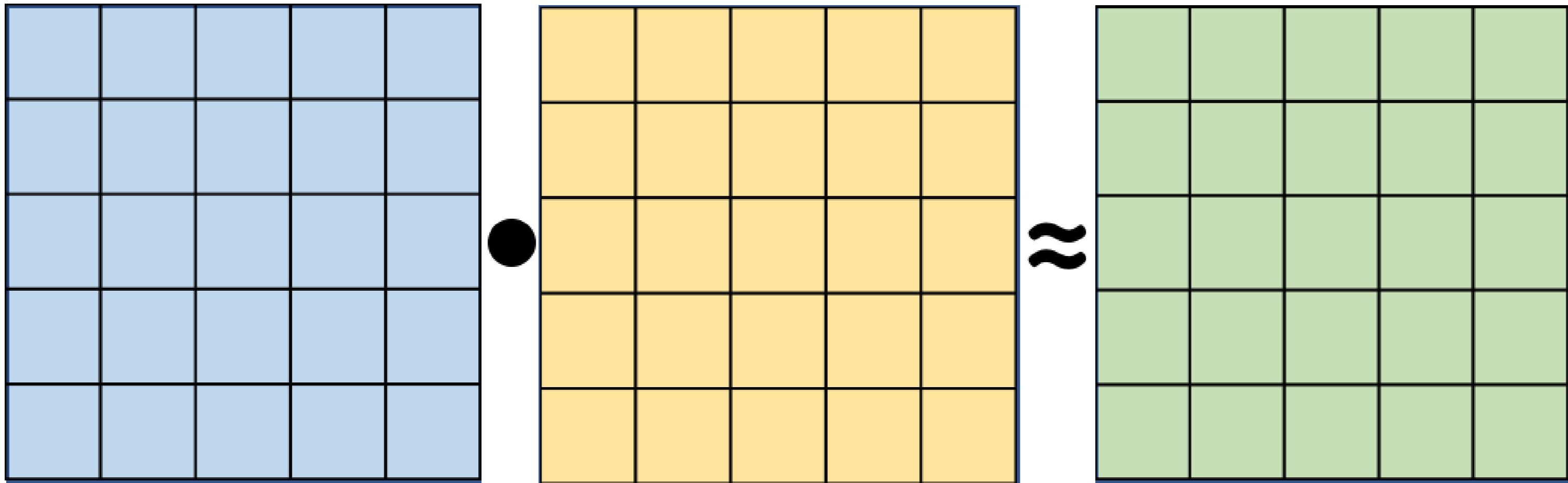
Jamen Long

Data Scientist at Nike

Matrix Factorization



Matrix Factorization



Matrix Factorization

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

Matrix Factorization

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



1	0	0	0	0
2/5	1	0	0	0
1/5	19/8	1	0	0
2/5	1	2/9	1	0
3/5	17/8	7/9	2/43	1



5	1	4	3	3
0	8/5	12/5	9/5	4/5
0	0	-9/2	-7/8	5/2
0	0	0	43/36	-5/9
0	0	0	0	-18/43

Matrix Factorization

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



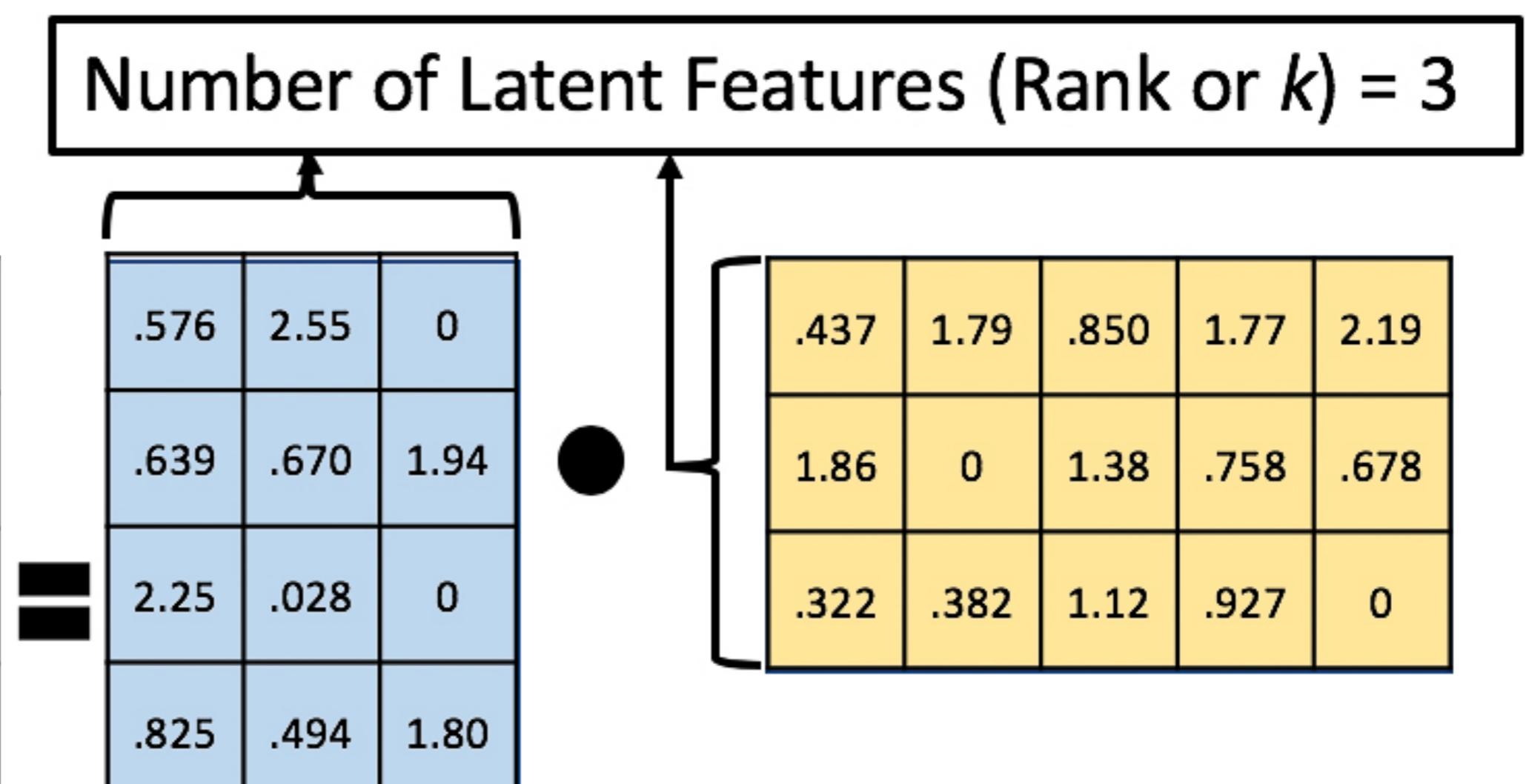
.576	2.55	0
.639	.670	1.94
2.25	.028	0
.825	.494	1.80
1.99	.982	.838



.437	1.79	.850	1.77	2.19
1.86	0	1.38	.758	.678
.322	.382	1.12	.927	0

Rank of Factor Matrices

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



84	-	48	-
-	50	-	-
-	-	51	-
91	-	-	107

Filling in the Blanks

84	-	48	-
-	50	-	-
-	-	51	-
91	-	-	107

=

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

•

6	2	3	7
7	8	4	8
8	0	5	7
5	3	3	9
4	2	2	6

Filling In the Blanks

84	-	48	-
-	50	-	-
-	-	51	-
91	-	-	107

=

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

•

6	2	3	7
7	8	4	8
8	0	5	7
5	3	3	9
4	2	2	6

Filling In the Blanks

84	-	48	-
-	50	-	-
-	-	51	-
91	-	-	107

=

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

•

6	2	3	7
7	8	4	8
8	0	5	7
5	3	3	9
4	2	2	6

Filling In the Blanks

84	-	48	-
-	50	-	-
-	-	51	-
91	-	-	107

=

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

•

6	2	3	7
7	8	4	8
8	0	5	7
5	3	3	9
4	2	2	6

Filling In the Blanks

84	-	48	-
-	50	-	-
-	-	51	-
91	-	-	107

=

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

•

6	2	3	7
7	8	4	8
8	0	5	7
5	3	3	9
4	2	2	6

Filling In the Blanks

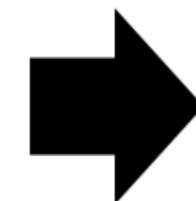
84	-	48	-
-	50	-	-
-	-	51	-
91	-	-	107

=

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

•

6	2	3	7
7	8	4	8
8	0	5	7
5	3	3	9
4	2	2	6



84	40	48	110
96	50	54	112
89	60	51	115
91	30	51	107

Let's practice!

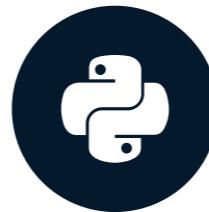
BUILDING RECOMMENDATION ENGINES WITH PYSPARK

How ALS alternates to generate predictions

BUILDING RECOMMENDATION ENGINES WITH PYSPARK

Jamen Long

Data Scientist at Nike



movieId	1	2	3	4	5	6
userId						
1	-	-	-	-	-	-
2	-	-	-	-	-	-
3	-	-	-	-	-	-
4	-	-	-	-	-	-
5	-	-	-	-	-	-
6	-	-	-	-	-	-
7	3	-	-	-	-	-
8	-	-	-	-	-	-
9	4	-	-	-	-	-
10	-	-	-	-	-	-
11	-	-	-	-	-	-
12	-	-	-	-	-	-

movieId	1	2	3	4	5	6
userId						
1	-	-	-	-	-	-
2	-	-	-	-	-	-
3	-	-	-	-	-	-
4	-	-	-	-	-	-
5	-	-	4	-	-	-
6	-	-	-	-	-	-
7	3	-	-	-	-	-
8	-	-	-	-	-	-
9	4	-	-	-	-	-
10	-	-	-	-	-	-
11	-	-	-	-	-	-
12	-	-	-	-	-	-

	U_LF_0	U_LF_1	U_LF_2
User_0	0.000000	0.092497	0.007393
User_1	0.000000	0.000000	1.322806
User_2	0.293443	0.000000	0.362894
User_3	0.140157	1.074063	1.332295
User_4	0.495512	0.075752	0.529940
User_5	0.219477	0.124221	0.000000
User_6	0.022552	0.162423	1.088869
User_7	0.999537	0.109175	0.372134
User_8	0.124147	0.180746	0.276849
User_9	0.144918	0.154747	0.196846
User_10	0.177815	0.025850	0.015767
User_11	0.066618	0.117502	0.064694
User_12	0.367768	0.000000	0.322991

	Movie_0	Movie_1	Movie_2	Movie_3	Movie_4	Movie_5	Movie_6
M_LF_0	1.296350	0.290096	0.000000	0.000000	0.044772	0.372374	0.000000
M_LF_1	0.297598	0.000000	0.008054	0.005646	0.036239	0.296742	0.172025
M_LF_2	1.265775	0.980059	0.471187	0.096935	0.465978	0.687785	0.419522

movieId	1	2	3	4	5	6
userId						
1	-	-	-	-	-	-
2	-	-	-	-	-	-
3	-	-	-	-	-	-
4	-	-	-	-	-	-
5	-	-	-	-	-	-
6	-	-	-	-	-	-
7	3	-	-	-	-	-
8	-	-	-	-	-	-
9	4	-	-	-	-	-
10	-	-	-	-	-	-
11	-	-	-	-	-	-
12	-	-	-	-	-	-

Constant

	U_LF_0	U_LF_1	U_LF_2
User_0	0.000000	0.092497	0.007393
User_1	0.000000	0.000000	1.322806
User_2	0.293443	0.000000	0.362894
User_3	0.140157	1.074053	1.332295
User_4	0.495522	0.071752	0.529940
User_5	0.219477	0.121221	0.000000
User_6	0.022552	0.162423	1.088869
User_7	0.999537	0.109175	0.372134
User_8	0.114117	0.180746	0.276849
User_9	0.144918	0.154747	0.196846
User_10	0.177815	0.025850	0.015767
User_11	0.066618	0.117502	0.064694
User_12	0.367768	0.000000	0.322991

Constant

	Movie_0	Movie_1	Movie_2	Movie_3	Movie_4	Movie_5	Movie_6
M_LF_0	1.296350	0.290096	0.000000	0.000000	0.044772	0.372374	0.000000
M_LF_1	0.297598	0.000000	0.008954	0.005646	0.036239	0.296742	0.172025
M_LF_2	1.265775	0.980059	0.471181	0.096935	0.465978	0.687785	0.419522

Adjusted

movieId	1	2	3	4	5	6
userId	-	-	-	-	-	-
1	-	-	-	-	-	-
2	-	-	-	-	-	-
3	-	-	-	-	-	-
4	-	-	-	-	-	-
5	-	-	-	-	-	-
6	-	-	-	-	-	-
7	3	4	5	6	7	8
8	4	5	6	7	8	9
9	5	6	7	8	9	10
10	6	7	8	9	10	11
11	7	8	9	10	11	12
12	8	9	10	11	12	-

Constant

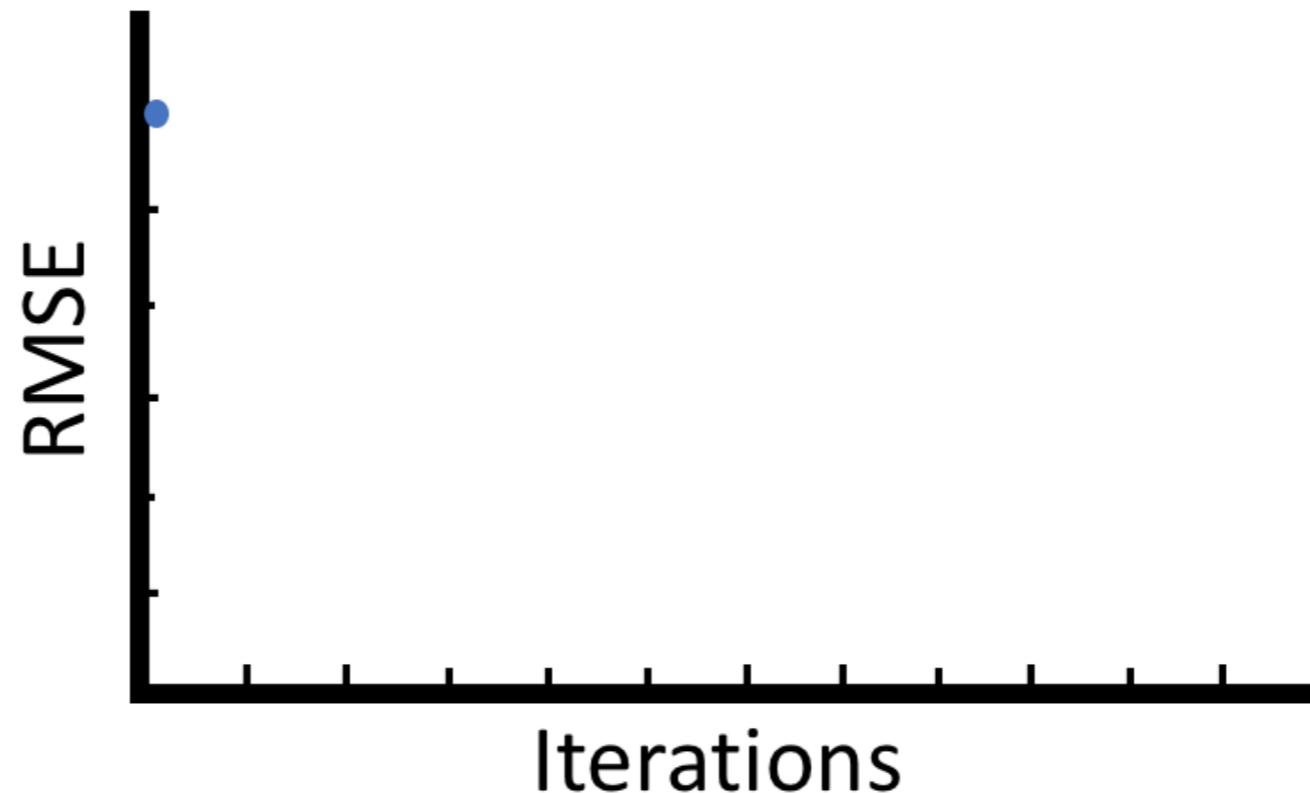
	U_LF_0	U_LF_1	U_LF_2
User_0	0.000000	0.092497	0.007393
User_1	0.000000	0.000000	1.322806
User_2	0.293443	0.000000	0.362894
User_3	0.140157	1.074029	1.332295
User_4	0.495522	0.000752	0.529940
User_5	0.219477	0.124221	0.000000
User_6	0.022552	0.162423	1.088869
User_7	0.999517	0.109175	0.372134
User_8	0.114117	0.180746	0.276849
User_9	0.144918	0.154747	0.196846
User_10	0.177815	0.025850	0.015767
User_11	0.066618	0.117502	0.064694
User_12	0.367768	0.000000	0.322991

Constant

	Movie_0	Movie_1	Movie_2	Movie_3	Movie_4	Movie_5	Movie_6
M_LF_0	1.296350	0.290096	0.000000	0.000000	0.044772	0.372374	0.000000
M_LF_1	0.297598	0.000000	0.008054	0.005646	0.036239	0.296742	0.172025
M_LF_2	1.265775	0.080059	0.471817	0.096935	0.465978	0.687785	0.419522

Adjusted

Iteration: 1
RMSE = 12,000



movieId	1	2	3	4	5	6
userId	-	-	-	-	-	-
1	-	-	-	-	-	-
2	-	-	-	-	-	-
3	-	-	-	-	-	-
4	-	-	-	-	-	-
5	-	-	-	-	-	-
6	-	-	-	-	-	-
7	3	-	-	-	-	-
8	4	-	-	-	-	-
9	-	-	-	-	-	-
10	-	-	-	-	-	-
11	-	-	-	-	-	-
12	-	-	-	-	-	-

Constant

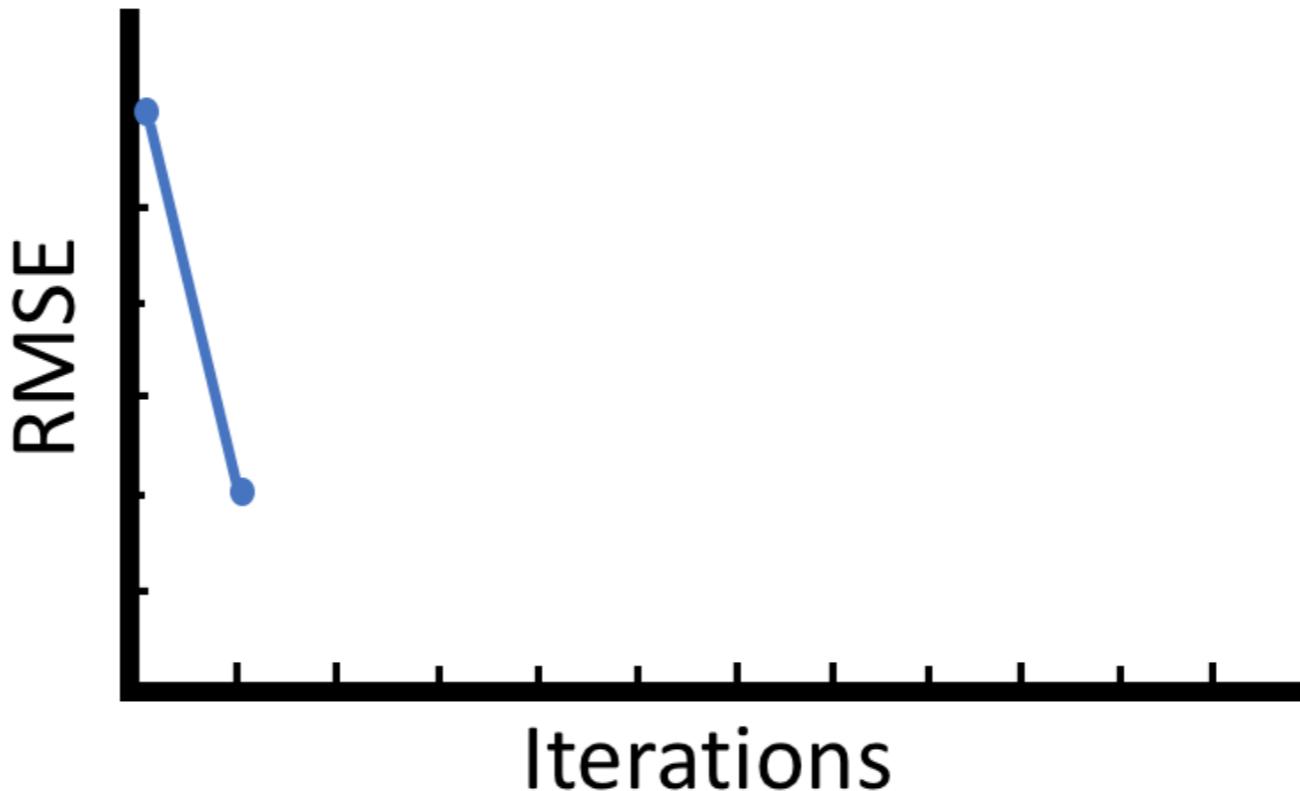
	U_LF_0	U_LF_1	U_LF_2
User_0	0.000000	0.092497	0.007393
User_1	0.000000	0.000000	1.322806
User_2	0.293443	0.000000	0.362894
User_3	0.140157	1.074063	0.332295
User_4	0.495522	0.045752	0.529940
User_5	0.219477	0.121221	0.000000
User_6	0.022552	0.162423	1.088869
User_7	0.999537	0.109175	0.372134
User_8	0.124147	0.180746	0.276849
User_9	0.244918	0.154747	0.196846
User_10	0.177815	0.025850	0.015767
User_11	0.066618	0.117502	0.064694
User_12	0.367768	0.000000	0.322991

Adjusted

	Movie_0	Movie_1	Movie_2	Movie_3	Movie_4	Movie_5	Movie_6
M_LF_0	1.296350	0.290096	0.000000	0.000000	0.044772	0.372374	0.000000
M_LF_1	0.297598	0.000000	0.000000	0.005646	0.036239	0.296742	0.172025
M_LF_2	1.265775	0.080059	0.4718	0.096935	0.465978	0.687785	0.419522

Constant

Iteration: 2
RMSE = 4,000



movieId	1	2	3	4	5	6
userId	-	-	-	-	-	-
1	-	-	-	-	-	-
2	-	-	-	-	-	-
3	-	-	-	-	-	-
4	-	-	-	-	-	-
5	-	-	-	-	-	-
6	-	-	-	-	-	-
7	3	-	-	-	-	-
8	-	-	-	-	-	-
9	4	-	-	-	-	-
10	-	-	-	-	-	-
11	-	-	-	-	-	-
12	-	-	-	-	-	-

Constant

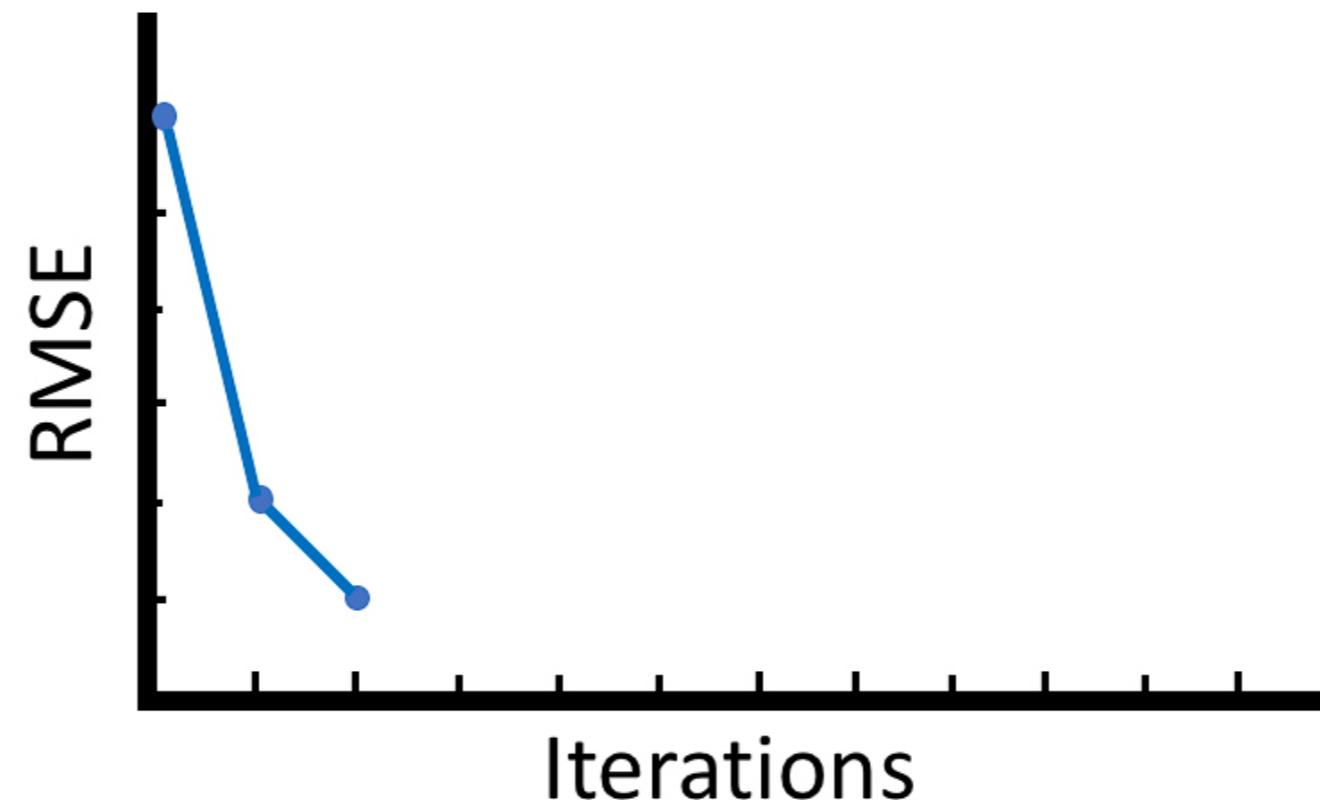
	U_LF_0	U_LF_1	U_LF_2
User_0	0.000000	0.092497	0.007393
User_1	0.000000	0.000000	1.322806
User_2	0.293443	0.000000	0.362894
User_3	0.140157	1.074003	1.332295
User_4	0.495522	0.077532	0.529940
User_5	0.219477	0.121221	0.000000
User_6	0.022552	0.162423	1.088869
User_7	0.999537	0.109175	0.372134
User_8	0.114117	0.180746	0.276849
User_9	0.144918	0.154747	0.196846
User_10	0.177815	0.025850	0.015767
User_11	0.066618	0.117502	0.064694
User_12	0.367768	0.000000	0.322991

Constant

	Movie_0	Movie_1	Movie_2	Movie_3	Movie_4	Movie_5	Movie_6
M_LF_0	1.296350	0.290096	0.000000	0.000000	0.044772	0.372374	0.000000
M_LF_1	0.297598	0.000000	0.008054	0.005646	0.036239	0.296742	0.172025
M_LF_2	1.265775	0.980059	0.471801	0.096935	0.465978	0.687785	0.419522

Adjusted

Iteration: 3
RMSE = 2,000



movieId	1	2	3	4	5	6
userId	-	-	-	-	-	-
1	-	-	-	-	-	-
2	-	-	-	-	-	-
3	-	-	-	-	-	-
4	-	-	-	-	-	-
5	-	-	-	-	-	-
6	-	-	-	-	-	-
7	3	-	-	-	-	-
8	4	-	-	-	-	-
9	-	-	-	-	-	-
10	-	-	-	-	-	-
11	-	-	-	-	-	-
12	-	-	-	-	-	-

Constant

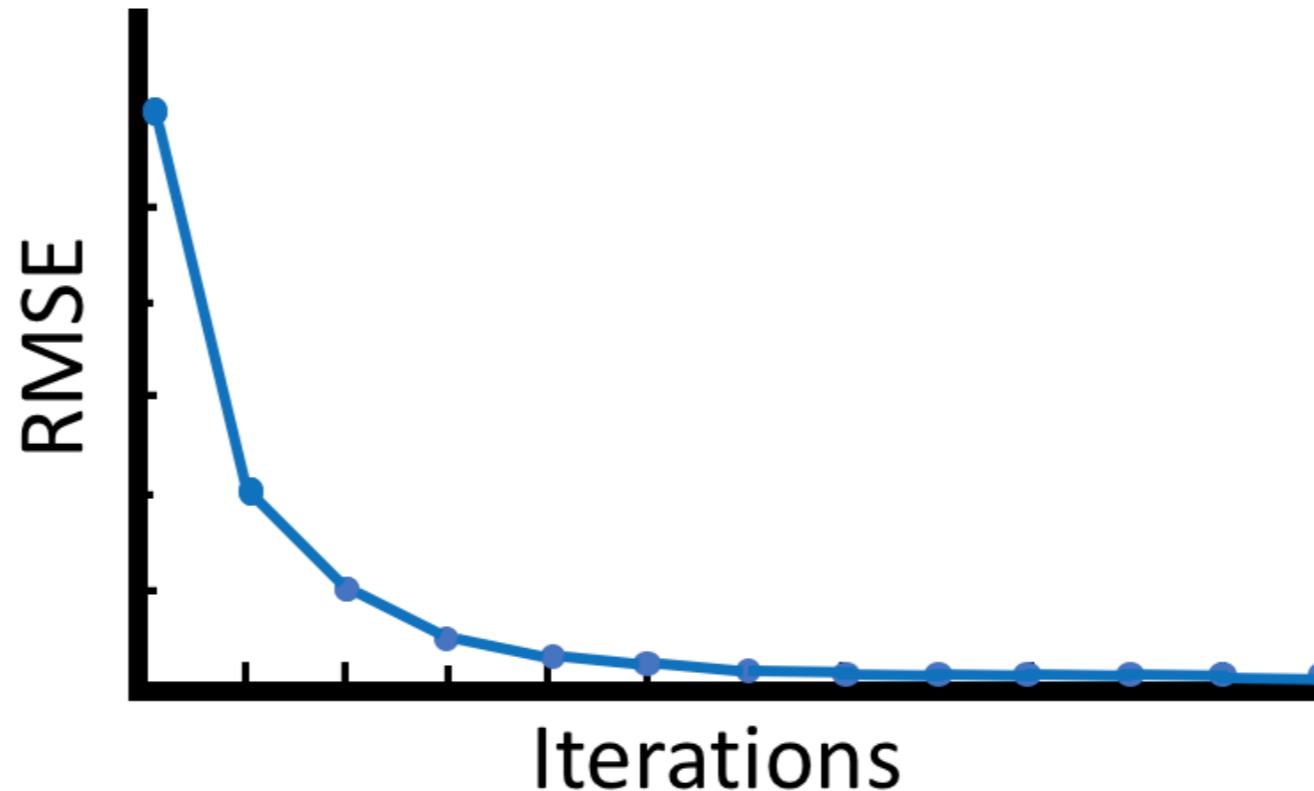
	U_LF_0	U_LF_1	U_LF_2
User_0	0.000000	0.092497	0.007393
User_1	0.000000	0.000000	1.322806
User_2	0.293443	0.000000	0.362894
User_3	0.140157	1.074063	0.332295
User_4	0.495512	0.075752	0.529940
User_5	0.219477	0.121221	0.000000
User_6	0.022552	0.162423	1.088869
User_7	0.999531	0.109175	0.372134
User_8	0.124147	0.180746	0.276849
User_9	0.244918	0.154747	0.196846
User_10	0.177815	0.025850	0.015767
User_11	0.066618	0.117502	0.064694
User_12	0.367768	0.000000	0.322991

Adjusted

	Movie_0	Movie_1	Movie_2	Movie_3	Movie_4	Movie_5	Movie_6
M_LF_0	1.296350	0.290096	0.000000	0.000000	0.044772	0.372374	0.000000
M_LF_1	0.297598	0.000000	0.000554	0.005646	0.036239	0.296742	0.172025
M_LF_2	1.265775	0.080059	0.4718	0.096935	0.465978	0.687785	0.419522

Constant

Iteration: n
RMSE = 0.6



movieId	1	2	3	4	5	6
userId						
1	-	-	-	-	-	-
2	-	-	-	-	-	-
3	-	-	-	-	-	-
4	-	-	-	-	-	-
5	-	4	-	-	-	-
6	-	-	-	-	-	-
7	3	-	-	-	-	-
8	-	-	-	-	-	-
9	4	-	-	-	-	-
10	-	-	-	-	-	-
11	-	-	-	-	-	-
12	-	-	-	-	-	-



Original Sparse Matrix

	Movie_1	Movie_2	Movie_3	Movie_4	Movie_5	Movie_6
User_1	1.296428	3.937481	0.128226	0.616399	0.909806	0.554946
User_2	0.440784	0.170991	0.035177	0.182239	0.358864	2.782786
User_3	1.346387	0.636410	0.135210	3.421250	1.287243	0.743692
User_4	0.663118	0.250311	0.051797	0.271871	2.867512	0.235352
User_5	0.063669	0.001000	4.000000	0.014328	0.118589	0.021369
User_6	1.073698	1.343627	0.106466	0.514285	0.805503	0.484745
User_7	3.000000	0.176224	0.036689	0.222114	0.660547	0.174899
User_8	0.307343	0.131904	0.027857	0.141114	3.989767	0.147237
User_9	4.000000	0.093997	0.019955	0.103822	0.235271	0.109201
User_10	0.067036	0.007638	1.452647	0.016245	0.084729	0.011062
User_11	0.082730	0.031429	0.006935	0.037387	0.104170	2.039812
User_12	0.423238	0.152189	0.031309	0.166972	4.001211	0.135502

Completed Prediction Matrix



	movie1	movie2	movie3	movie4	movie5
user1	-	-	3	-	5
user2	3	-	-	-	1
user3	-	-	-	3	2
user4	-	-	1	-	-
user5	-	2	-	-	-

R

	movie1	movie2	movie3	movie4	movie5
user1	-	-	3	-	5
user2	3	-	-	-	1
user3	-	-	-	3	2
user4	-	-	1	-	-
user5	-	2	-	-	-

U

	u_lf_1	u_lf_2	u_lf_3	u_lf_4
user1	1.0	0.0	2.0	1.0
user2	1.0	3.0	0.0	0.0
user3	0.0	3.0	0.0	0.0
user4	1.0	1.0	3.0	0.0
user5	2.0	1.0	0.0	0.0

P

	movie1	movie2	movie3	movie4	movie5
m_lf_1	1.0	0.0	1.0	2.0	1.0
m_lf_2	1.0	1.0	1.0	0.0	0.0
m_lf_3	0.0	0.0	1.0	0.0	1.0
m_lf_4	0.0	2.0	0.0	2.0	3.0

R

	movie1	movie2	movie3	movie4	movie5
user1	-	-	3	-	5
user2	3	-	-	-	1
user3	-	-	-	3	2
user4	-	-	1	-	-
user5	-	2	-	-	-

U

	u_lf_1	u_lf_2	u_lf_3	u_lf_4
user1	1.0	0.0	2.0	1.0
user2	1.0	3.0	0.0	0.0
user3	0.0	3.0	0.0	0.0
user4	1.0	1.0	3.0	0.0
user5	2.0	1.0	0.0	0.0

P

	movie1	movie2	movie3	movie4	movie5
m_lf_1	1.0	0.0	1.0	2.0	1.0
m_lf_2	1.0	1.0	1.0	0.0	0.0
m_lf_3	0.0	0.0	1.0	0.0	1.0
m_lf_4	0.0	2.0	0.0	2.0	3.0

R

	movie1	movie2	movie3	movie4	movie5
user1	-	-	3	-	5
user2	3	-	-	-	1
user3	-	-	3	2	-
user4	-	-	1	-	-
user5	-	2	-	-	-

U

	u_lf_1	u_lf_2	u_lf_3	u_lf_4
user1	1.0	0.0	2.0	1.0
user2	1.0	3.0	0.0	0.0
user3	0.0	3.0	0.0	0.0
user4	1.0	1.0	3.0	0.0
user5	2.0	1.0	0.0	0.0

P

	movie1	movie2	movie3	movie4	movie5
m_lf_1		1.0	0.0	1.0	2.0
m_lf_2		1.0	1.0	1.0	0.0
m_lf_3		0.0	0.0	1.0	0.0
m_lf_4		0.0	2.0	0.0	2.0
					3.0

R

	movie1	movie2	movie3	movie4	movie5
user1	-	-	3	-	5
user2	3	-	-	-	1
user3	-	-	-	3	2
user4	-	-	1	-	-
user5	-	2	-	-	-

U

	u_lf_1	u_lf_2	u_lf_3	u_lf_4
user1	1.0	0.0	2.0	1.0
user2	1.0	3.0	0.0	0.0
user3	0.0	3.0	0.0	0.0
user4	1.0	1.0	3.0	0.0
user5	2.0	1.0	0.0	0.0

P

	movie1	movie2	movie3	movie4	movie5
m_lf_1	1.0	0.0	1.0	2.0	1.0
m_lf_2	1.0	1.0	1.0	0.0	0.0
m_lf_3	0.0	0.0	1.0	0.0	1.0
m_lf_4	0.0	2.0	0.0	2.0	3.0

	movie1	movie2	movie3	movie4	movie5
user1	3	-	5		
user2	3	-	-	1	
user3	3	2			
user4	1	-	-		
user5	2	-	-		

	u_lf_1	u_lf_2	u_lf_3	u_lf_4
user1	1.0	0.0	2.0	1.0
user2	1.0	3.0	0.0	0.0
user3	0.0	3.0	0.0	0.0
user4	1.0	1.0	3.0	0.0
user5	2.0	1.0	0.0	0.0

	movie1	movie2	movie3	movie4	movie5
m_lf_1	1.0	0.0	1.0	2.0	1.0
m_lf_2	1.0	1.0	1.0	0.0	0.0
m_lf_3	0.0	0.0	1.0	0.0	1.0
m_lf_4	0.0	2.0	0.0	2.0	3.0

	movie1	movie2	movie3	movie4	movie5
user1	3	-	3	5	
user2	3	-	-	1	
user3	-	-	3	2	
user4	-	-	1	-	-
user5	-	2	-	-	-

	u_lf_1	u_lf_2	u_lf_3	u_lf_4
user1	1.0	0.0	2.0	1.0
user2	1.0	3.0	0.0	0.0
user3	0.0	3.0	0.0	0.0
user4	1.0	1.0	3.0	0.0
user5	2.0	1.0	0.0	0.0

	movie1	movie2	movie3	movie4	movie5
m_lf_1	1.0	0.0	1.0	2.0	1.0
m_lf_2	1.0	1.0	1.0	0.0	0.0
m_lf_3	0.0	0.0	1.0	0.0	1.0
m_lf_4	0.0	2.0	0.0	2.0	3.0



	movie1	movie2	movie3	movie4	movie5
user1	-	-	3	-	5
user2	3	-	-	-	1
user3	-	-	-	3	2
user4	-	-	1	-	-
user5	-	2	-	-	-

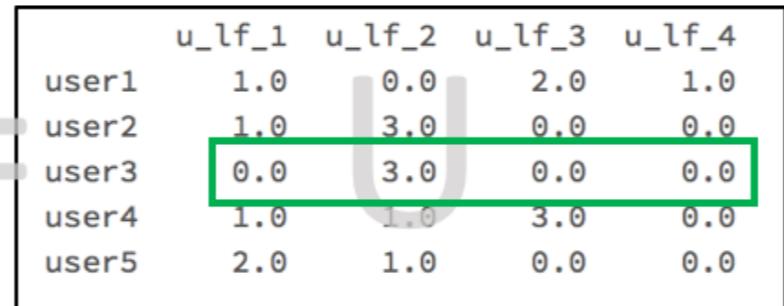


	u_lf_1	u_lf_2	u_lf_3	u_lf_4
user1	1.0	0.0	2.0	1.0
user2	1.0	3.0	0.0	0.0
user3	0.0	3.0	0.0	0.0
user4	1.0	1.0	3.0	0.0
user5	2.0	1.0	0.0	0.0

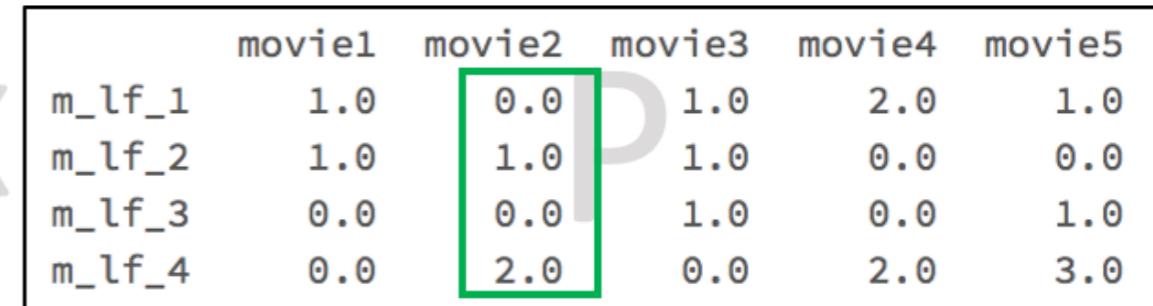


	movie1	movie2	movie3	movie4	movie5
m_lf_1	1.0	0.0	1.0	2.0	1.0
m_lf_2	1.0	1.0	1.0	0.0	0.0
m_lf_3	0.0	0.0	1.0	0.0	1.0
m_lf_4	0.0	2.0	0.0	2.0	3.0

	movie1	movie2	movie3	movie4	movie5
user1	-	-	3	-	5
user2	3	-	-	-	1
user3	-	-	-	3	2
user4	-	-	1	-	-
user5	-	2	-	-	-



	u_lf_1	u_lf_2	u_lf_3	u_lf_4
user1	1.0	0.0	2.0	1.0
user2	1.0	3.0	0.0	0.0
user3	0.0	3.0	0.0	0.0
user4	1.0	1.0	3.0	0.0
user5	2.0	1.0	0.0	0.0



	movie1	movie2	movie3	movie4	movie5
m_lf_1	1.0	0.0	1.0	2.0	1.0
m_lf_2	1.0	1.0	1.0	0.0	0.0
m_lf_3	0.0	0.0	1.0	0.0	1.0
m_lf_4	0.0	2.0	0.0	2.0	3.0

	movie1	movie2	movie3	movie4	movie5
user1	-	3	-	5	
user2	3	-	-	-	1
user3	-	-	3	2	
user4	-	-	1	-	-
user5	-	2	-	-	-

R

	u_lf_1	u_lf_2	u_lf_3	u_lf_4
user1	1.0	0.0	2.0	1.0
user2	1.0	3.0	0.0	0.0
user3	0.0	3.0	0.0	0.0
user4	1.0	1.0	3.0	0.0
user5	2.0	1.0	0.0	0.0

U

	movie1	movie2	movie3	movie4	movie5
m_lf_1	1.0	0.0	1.0	2.0	1.0
m_lf_2	1.0	1.0	1.0	0.0	0.0
m_lf_3	0.0	0.0	1.0	0.0	1.0
m_lf_4	0.0	2.0	0.0	2.0	3.0

P

	movie1	movie2	movie3	movie4	movie5
user1	1.0	2.0	3.0	4.0	6.0
user2	4.0	3.0	4.0	2.0	1.0
user3	3.0	3.0	3.0	1.0	1.0
user4	2.0	1.0	5.0	1.0	4.0
user5	3.0	1.0	3.0	4.0	2.0

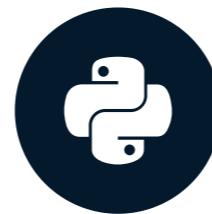
=

Let's practice!

BUILDING RECOMMENDATION ENGINES WITH PYSPARK

Data preparation for Spark ALS

BUILDING RECOMMENDATION ENGINES WITH PYSPARK



Jamen Long

Data Scientist at Nike

Conventional Dataframe

userId	Good Will H...	Batman For...	Incredibles	Shawshank Redemption	Coco	...
z097s3	2	3	null		4	4
z176c4	1	null	4		3	4
m821i6	3	4	null		3	5
t872c7	1	2	4		5	null
b728q0	2	null	5		2	null
f540n1	2	1	null		3	1
w066f1	5	null	5		2	5
v081u6	1	null	5		1	1
j197o6	3	2	2		4	null
n202j1	2	null	2		null	2

Row-based data format

userId	variable	rating
z097s3	Good Will Hunting	2
z097s3	Batman Forever	3
z097s3	The Shawshank Red...	4
z097s3	Coco	4
z176c4	Good Will Hunting	1
z176c4	The Incredibles	4
z176c4	The Shawshank Red...	3
z176c4	Coco	4
m821i6	Good Will Hunting	3
m821i6	Batman Forever	4

Row-based data format (cont.)

```
+-----+-----+-----+
|userId|           variable|rating|
+-----+-----+-----+
z097s3 |z097s3|  Good Will Hunting| 2|
|----> |z097s3|    Batman Forever| 3|
|----> |z097s3|The Shawshank Red...| 4|
|----> |z097s3|                  Coco| 4|
z176c4 |z176c4|  Good Will Hunting| 1|
|----> |z176c4|    The Incredibles| 4|
|----> |z176c4|The Shawshank Red...| 3|
|----> |z176c4|                  Coco| 4|
m821i6 |m821i6|  Good Will Hunting| 3|
|----> |m821i6|    Batman Forever| 4|
```

```
df.printSchema()
```

```
root
|-- userId: string (nullable = true)
|-- variable: string (nullable = false)
|-- rating: long (nullable = true)
```

Must be integers

```
df.printSchema()
```

```
root
|-- userId: string (nullable = true)
|-- variable: string (nullable = false)
|-- rating: long (nullable = true)
```

Must be integers!

Row-Based Data Format

userId	variable	rating
z097s3	Good Will Hunting	2
z097s3	Batman Forever	3
z097s3	The Shawshank Red...	4
z097s3	Coco	4
z176c4	Good Will Hunting	1
z176c4	The Incredibles	4
z176c4	The Shawshank Red...	3
z176c4	Coco	4
m821i6	Good Will Hunting	3
m821i6	Batman Forever	4
m821i6	The Shawshank Red...	3
m821i6	Coco	5
t872c7	Good Will Hunting	1

Conventional Dataframe

```
ratings.show()
```

userId	Good Will H...	Batman For...	Incredibles	Shawshank Redemption	Coco	...
z097s3	2	3	null		4	4
z176c4	1	null	4		3	4
m821i6	3	4	null		3	5
t872c7	1	2	4		5	null
b728q0	2	null	5		2	null
f540n1	2	1	null		3	1
w066f1	5	null	5		2	5
v081u6	1	null	5		1	1

Wide to long function

```
# Function to convert conventional datafame into row-based ("long") dataframe  
wide_to_long
```

```
<function __main__.to_long>
```

```
# Function to convert conventional datafame into row-based ("long") dataframe
long_ratings = wide_to_long(ratings)
long_ratings.show()
```

```
+-----+-----+-----+
|userId|      variable|rating|
+-----+-----+-----+
|z097s3| Good Will Hunting|    2|
|z097s3|     Batman Forever|    3|
|z097s3|The Shawshank Red...|    4|
|z097s3|          Coco|    4|
|z176c4| Good Will Hunting|    1|
|z176c4|     The Incredibles|    4|
|z176c4|The Shawshank Red...|    3|
|z176c4|          Coco|    4|
```

Steps to get integer ID's

1. Extract unique `userIds` and `movieIds`
2. Assign unique integers to each id
3. Rejoin unique integer id's back to the ratings data

Extracting distinct user IDs

```
users = long_ratings.select('userId').distinct()  
users.show()
```

```
+-----+  
|userId|  
+-----+  
|j197o6|  
|m821i6|  
|g025o2|  
|z176c4|  
|a610z0|  
|c460j6|  
|w066f1|  
|v081u6|  
|t791a0|
```

Monotonically increasing ID

```
from pyspark.sql.functions import monotonically_increasing_id
```

Coalesce method

```
from pyspark.sql.functions import monotonically_increasing_id  
users = users.coalesce(1)
```

Persist method

```
from pyspark.sql.functions import monotonically_increasing_id
users = users.coalesce(1)
users = users.withColumn(
    "userIntId", monotonically_increasing_id()).persist()
users.show()
```

```
+-----+
|userId|userIntId|
+-----+
|j197o6|      0|
|m821i6|      1|
|g025o2|      2|
|z176c4|      3|
|a610z0|      4|
|c460j6|      5|
|w066f1|      6|
```

Movie integer IDs

```
movies = long_ratings.select("variable").distinct()
movies = movies.coalesce(1)
movies = movies.withColumn(
    "movieId", monotonically_increasing_id()).persist()
movies.show()
```

```
+-----+-----+
|      variable|movieId|
+-----+-----+
| The Incredibles|     0|
|          Coco|     1|
| The Shawshank Red...|     2|
|   Good Will Hunting|     3|
|    Batman Forever|     4|
+-----+-----+
```

Joining UserIds and Movields

```
ratings_w_int_ids = long_ratings.join(  
    users, "userId", "left").join(movies, "variable", "left")  
  
ratings_w_int_ids.show()
```

```
+-----+-----+-----+-----+  
|      variable|userId|rating|userIntId|movieId|  
+-----+-----+-----+-----+  
| Good Will Hunting|z097s3| 2| 16| 3|  
| Batman Forever|z097s3| 3| 16| 4|  
| The Shawshank Red...|z097s3| 4| 16| 2|  
|          Coco|z097s3| 4| 16| 1|  
| Good Will Hunting|z176c4| 1| 3| 3|  
| The Incredibles|z176c4| 4| 3| 0|  
| The Shawshank Red...|z176c4| 3| 3| 2|  
|          Coco|z176c4| 4| 3| 1|
```

```
from pyspark.ml.functions import col

ratings_data = ratings_w_int_ids.select(
    col("userIntId").alias("userid"),
    col("variable").alias("movieId"),
    col("rating"))

ratings_data.show()
```

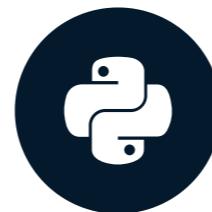
```
+----+----+----+
|userId|movieId|rating|
+----+----+----+
|   16|      3|     2|
|   16|      4|     3|
|   16|      2|     4|
|   16|      1|     4|
|    3|      3|     1|
|    3|      0|     4|
|    -|      -|     -|
```

Let's practice!

BUILDING RECOMMENDATION ENGINES WITH PYSPARK

ALS parameters and hyperparameters

BUILDING RECOMMENDATION ENGINES WITH PYSPARK



Jamen Long

Data Scientist at Nike

Example ALS model code

```
als_model = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
                 rank=25, maxIter=100, regParam=.05, alpha=40,
                 nonnegative=True,
                 coldStartStrategy="drop",
                 implicitPrefs=False)
```

Column names

```
als_model = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
                 rank=25, maxIter=100, regParam=.05, alpha=40,
                 nonnegative=True,
                 coldStartStrategy="drop",
                 implicitPrefs=False)
```

Arguments

- `userCol` : Name of column that contains user id's
- `itemCol` : Name of column that contains item id's
- `ratingCol` : Name of column that contains ratings

Original Ratings Matrix

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6	Movie 7	Movie ...	Movie n
User 1									
User 2									
User 3									
User 4									
User 5									
User...									
User m									

ALS

Factor Matrices

	LF1	LF...	LF k
User 1			
User 2			
User 3			
User 4			
User 5			
User..			
User m			

U

X

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie...	Movie n
LF1							
LF...							
LF k							

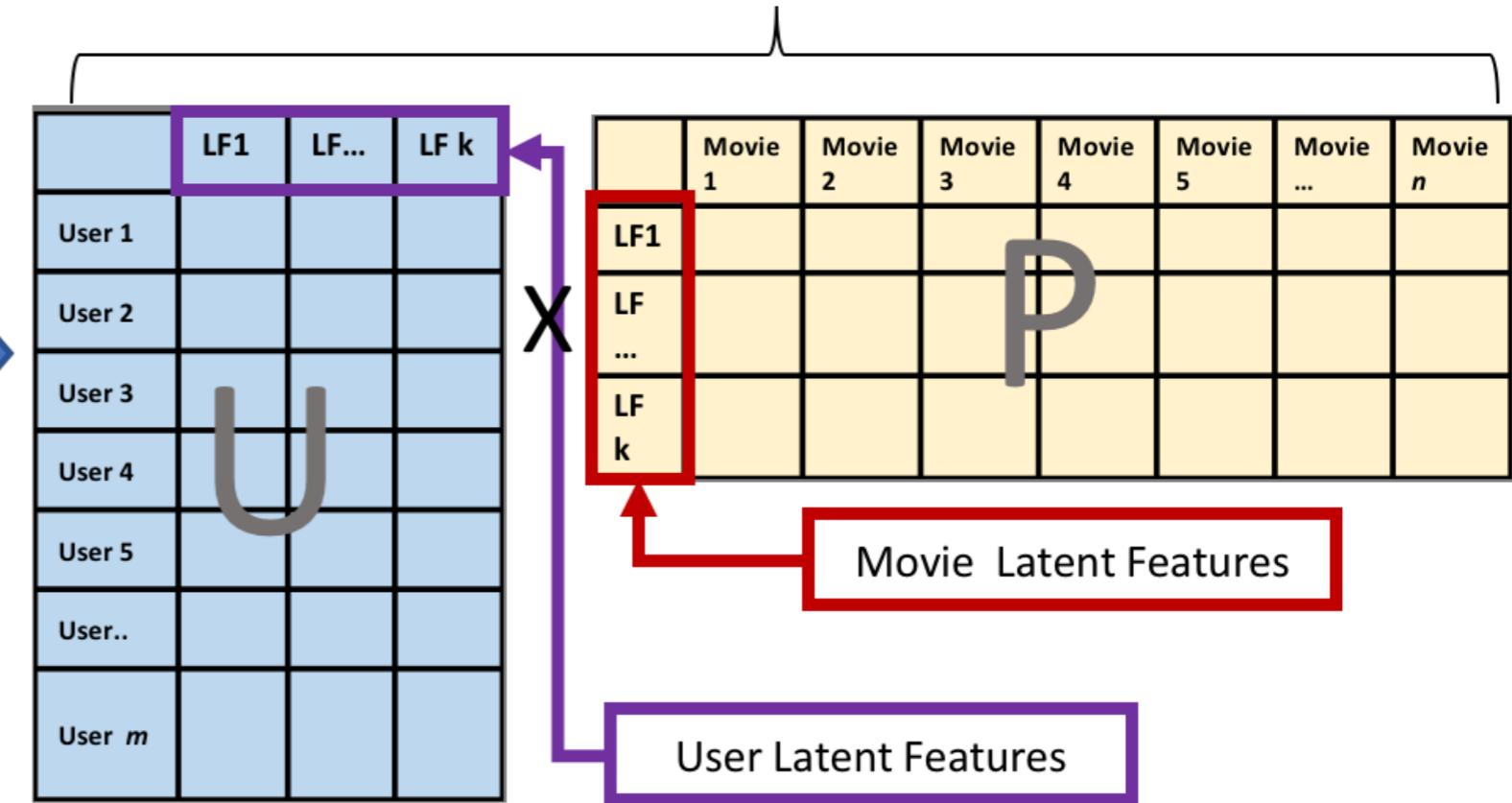
P

Original Ratings Matrix

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6	Movie 7	Movie ...	Movie n
User 1									
User 2									
User 3									
User 4									
User 5									
User...									
User m									

ALS

Factor Matrices



Rank

```
als_model = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
    rank=25, maxIter=100, regParam=.05, alpha=40,
    nonnegative=True,
    coldStartStrategy="drop",
    implicitPrefs=False)
```

Hyperparameters

- `rank`, k : number of latent features

MaxIter

```
als_model = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
                 rank=25, maxIter=100, regParam=.05, alpha=40,
                 nonnegative=True,
                 coldStartStrategy="drop",
                 implicitPrefs=False)
```

Hyperparameters

- `rank`, k : number of latent features
- `maxIter` : number of iterations

RegParam

```
als_model = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
                 rank=25, maxIter=100, regParam=.05, alpha=40,
                 nonnegative=True,
                 coldStartStrategy="drop",
                 implicitPrefs=False)
```

Hyperparameters

- `rank`, k : number of latent features
- `maxIter` : number of iterations
- `regParam` : Lambda

Alpha

```
als_model = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
                 rank=25, maxIter=100, regParam=.05, alpha=40,
                 nonnegative=True,
                 coldStartStrategy="drop",
                 implicitPrefs=False)
```

Hyperparameters

- `rank`, k : number of latent features
- `maxIter` : number of iterations
- `regParam` : Lambda
- `alpha` : Discussed later. Only used with implicit ratings.

Non-negative

```
als_model = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
                 rank=25, maxIter=100, regParam=.05, alpha=40,
                 nonnegative=True,
                 coldStartStrategy="drop",
                 implicitPrefs=False)
```

Additional Arguments

- `nonnegative = True` : Ensures positive numbers

Cold start strategy

```
als_model = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
                 rank=25, maxIter=100, regParam=.05, alpha=40,
                 nonnegative=True,
                 coldStartStrategy="drop",
                 implicitPrefs=False)
```

Additional Arguments

- `nonnegative = True` : Ensures positive numbers
- `coldStartStrategy = "drop"` : Addresses issues with test/train split

Implicit preferences

```
als_model = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
                 rank=25, maxIter=100, regParam=.05, alpha=40,
                 nonnegative=True,
                 coldStartStrategy="drop",
                 implicitPrefs=False)
```

Additional Arguments

- `nonnegative = True` : Ensures positive numbers
- `coldStartStrategy = "drop"` : Addresses issues with test/train split
- `implicitPrefs = True` : True/False depending on ratings type

Sample ALS model build

```
als = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
          rank=25, maxIter=100, regParam=.05,
          nonnegative=True,
          coldStartStrategy="drop",
          implicitPrefs=False)
```

Fit and transform methods

```
# Fit ALS to training dataset  
model = als.fit(training_data)  
  
# Generate predictions on test dataset  
predictions = model.transform(test_data)
```

Let's practice!

BUILDING RECOMMENDATION ENGINES WITH PYSPARK