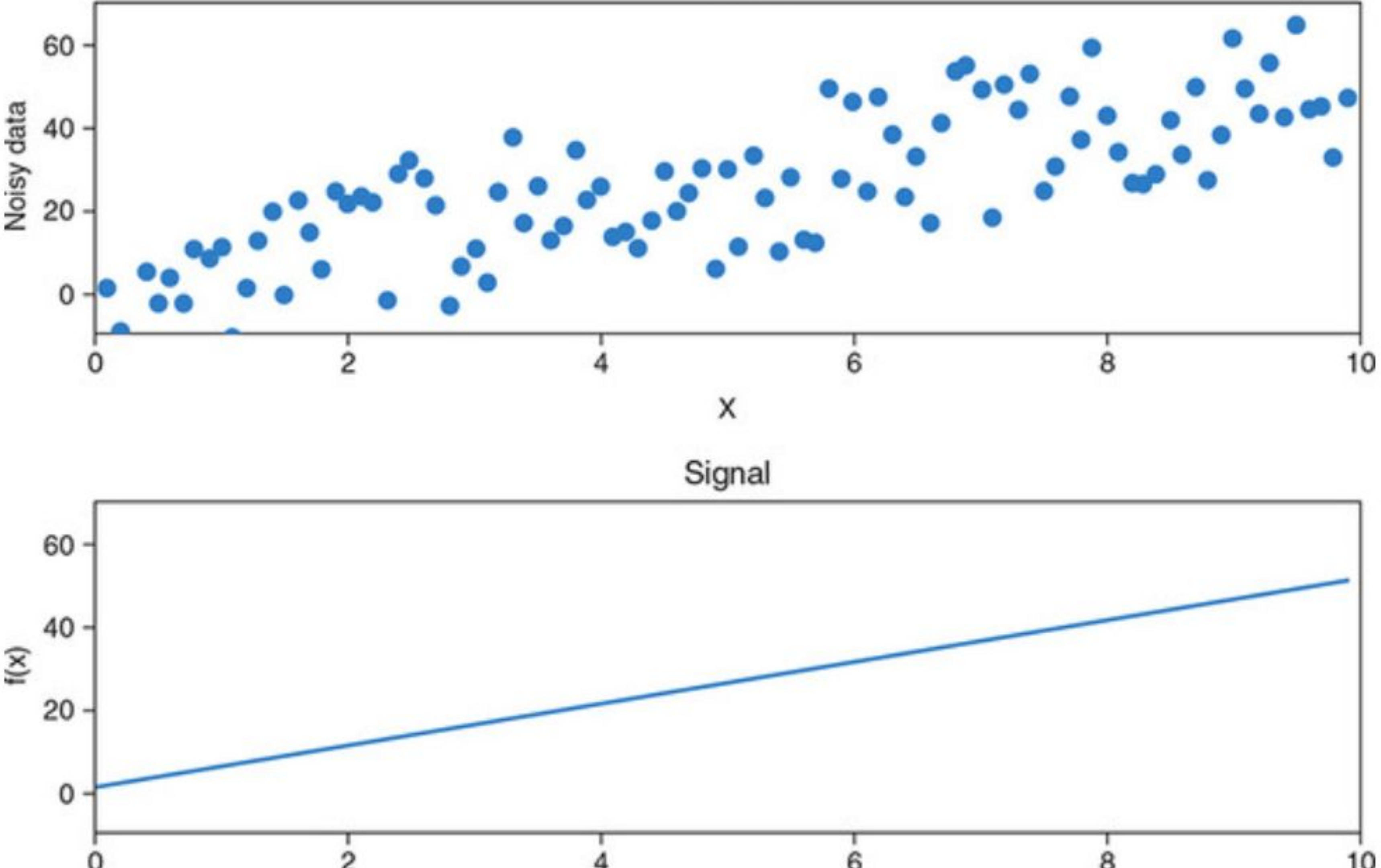
Matrix Factorization

•••

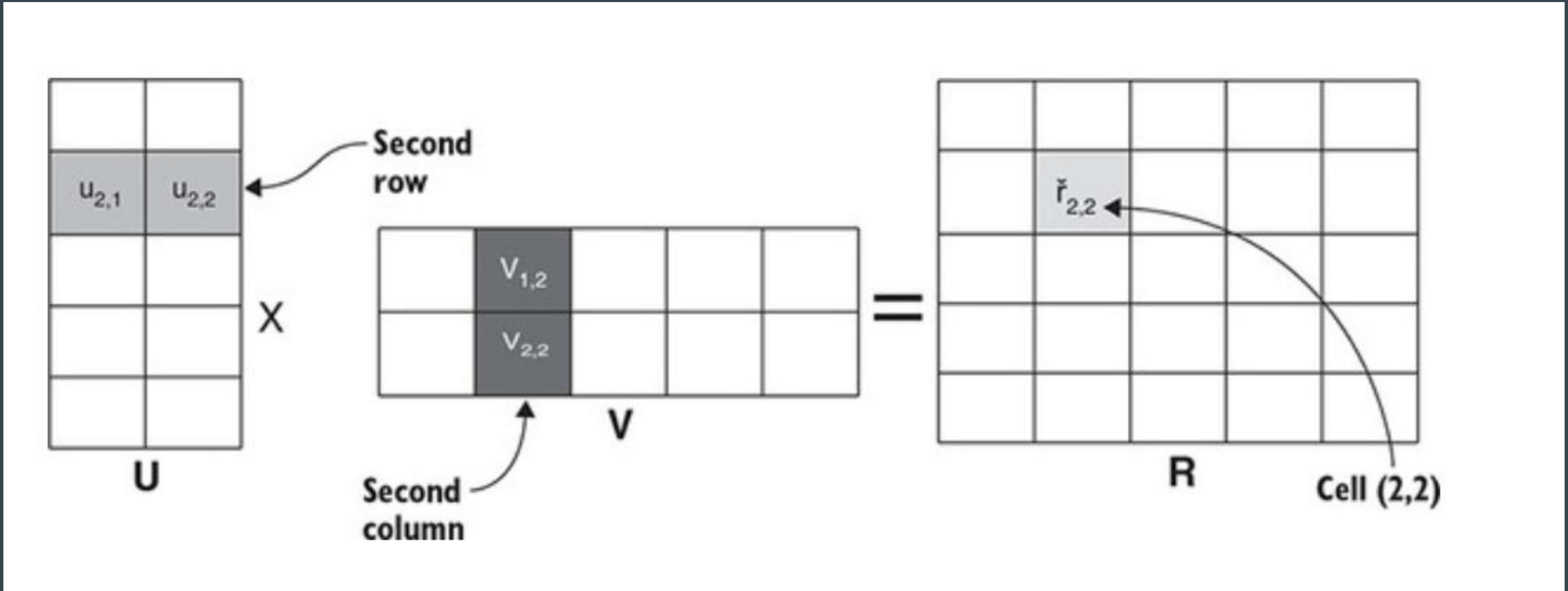
Why Matrix Factorization

- Dimension reduction for recommender algorithm
- Find latent factors in the data
- Latent because they're defined by something that an algorithm calculates
- It will not make sense to humans
- They are trends in the data that show or explain the user's taste
- Invented by Simon Funk during Netflix competition
- Find latent factors for ratings matrix

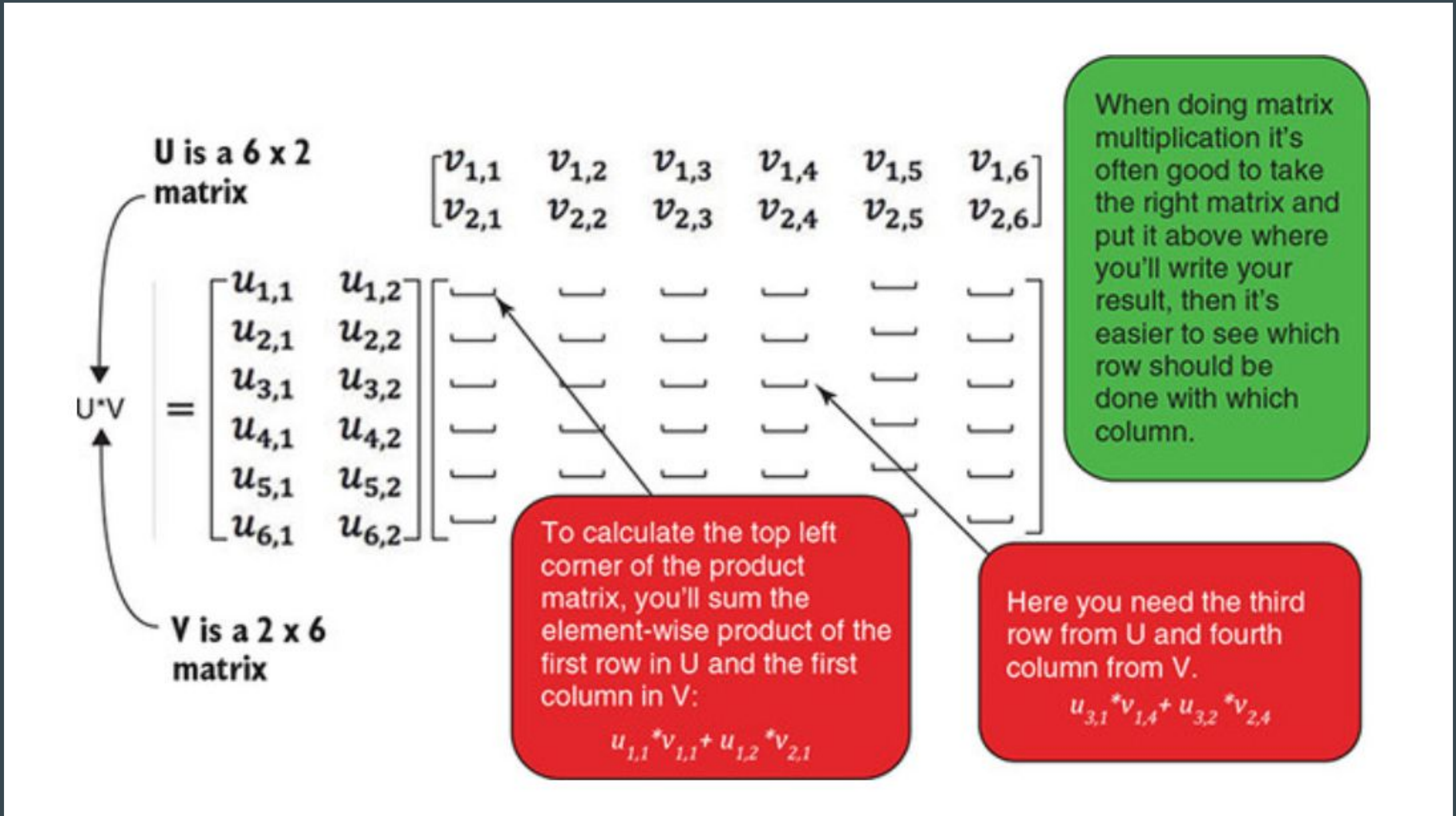


60
40
20
0
Noisy data
0
2
4
6
8
10
X
Signal
60
40
20
0
f(x)
0
2
4
6
8
10

Matrix Multiplication



Matrix Multiplication



Factoring the Matrix

- Splitting the matrix
- 100 = 2 x 2 x 5 x 5
- Factor a matrix into a product of matrices
- Rating Matrix = UV

UV Decomposition with 2 Dimensions

Rating Matrix (R) dimensions = Users x Items

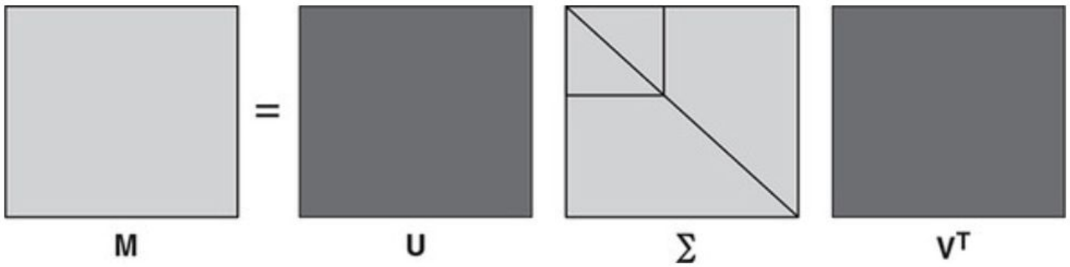
R = UV

User feature matrix (U) = Userx x Dimension (D)

Item feature matrix (V) = Dimension (D) x Items

$$\begin{bmatrix} 5 & 3 & 0 & 2 & 2 & 2 \\ 4 & 3 & 4 & 0 & 3 & 3 \\ 5 & 2 & 5 & 2 & 1 & 1 \\ 3 & 5 & 3 & 0 & 1 & 1 \\ 3 & 3 & 3 & 2 & 4 & 5 \\ 2 & 3 & 2 & 3 & 5 & 5 \end{bmatrix} = \begin{bmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \\ u_{3,1} & u_{3,2} \\ u_{4,1} & u_{4,2} \\ u_{5,1} & u_{5,2} \\ u_{6,1} & u_{6,2} \end{bmatrix} \begin{bmatrix} v_{1,1} & v_{1,2} & v_{1,3} & v_{1,4} & v_{1,5} & v_{1,6} \\ v_{2,1} & v_{2,2} & v_{2,3} & v_{2,4} & v_{2,5} & v_{2,6} \end{bmatrix}$$

Factorization using SVD



You call them:

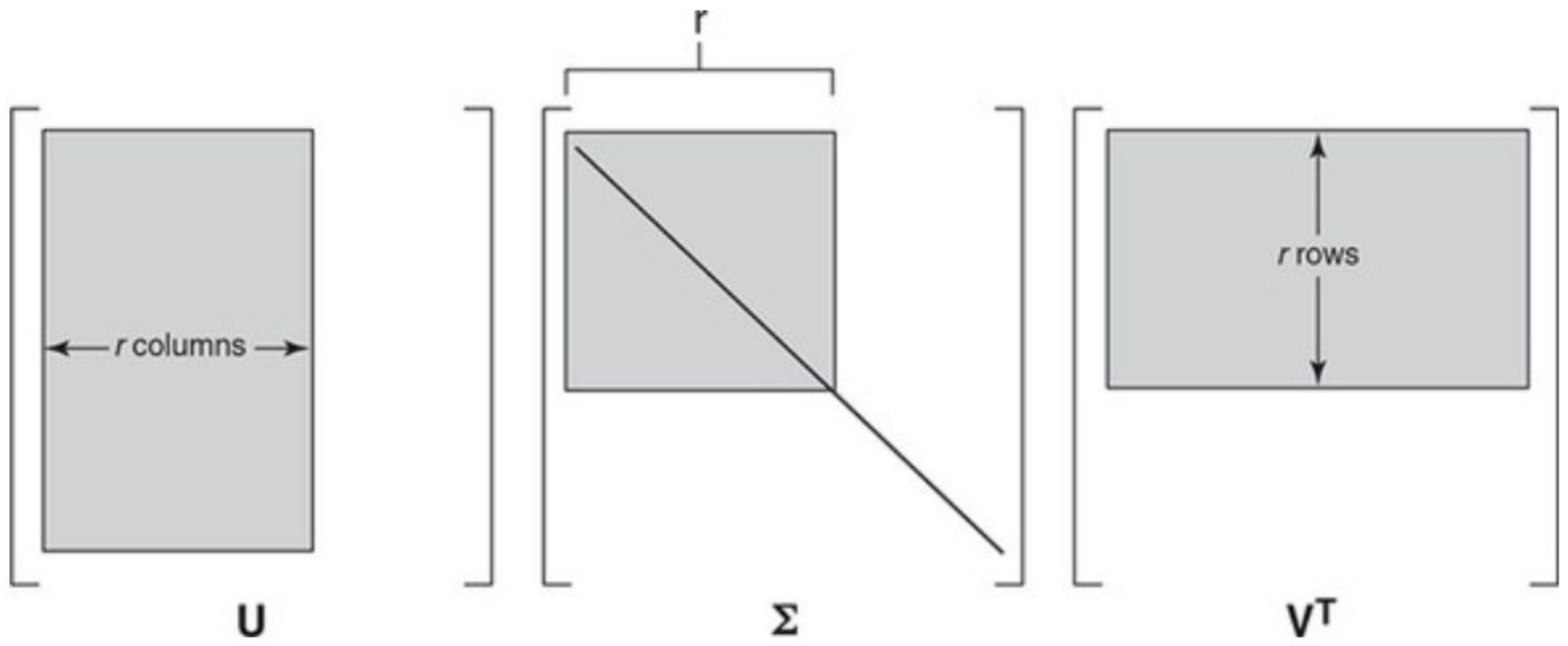
- *M*—A matrix you want to decompose; in your case, it's the rating matrix.
- *U*—User feature matrix.
- Σ—Weights diagonal.
- *VT*—Item feature matrix.

Diagonal Matrix

- It has zeros except in the diagonal
- Central diagonal matrix contains elements that are sorted from the largest to smallest
- The elements are called singular values and they indicate how much value this feature produces for the data set

$$\begin{bmatrix} 9 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Setting unimportant singular values to Zero



Demo - Matrix Factorization Notebook

-0.34	0.05	0.91	0.11	0.19	-0.00
-0.43	0.16	-0.31	-0.12	0.74	0.35
-0.39	0.56	-0.19	0.63	-0.32	0.02
-0.33	0.42	0.02	-0.76	-0.37	-0.05
-0.48	-0.34	-0.18	0.03	0.10	-0.78
-0.46	-0.61	-0.06	0.02	–0.40	0.51

U

17.27	0	0	0	0	0
0	5.84	0	0	0	0
0	0	3.56	0	0	0
0	0	0	3.13	0	0
0	0	0	0	1.67	0
0	0	0	0	0	0.56

Σ

-0.50	-0.44	-0.41	-0.22	-0.40	-0.43
0.46	0.17	0.42	-0.22	-0.49	-0.55
0.50	0.22	-0.78	0.26	-0.08	-0.13
0.34	-0.77	0.17	0.51	-0.02	-0.01
0.41	-0.36	-0.16	-0.76	0.19	0.25
-0.01	-0.03	0.01	-0.02	0.75	-0.66

V^t

How much should the matrix be reduced

- Rule of thumb - Retain 90% of the information

Predicting a rating

	mib	st	av	b	ss	lm
Sara	4.87	3.11	0.05	2.24	1.94	1.92
Jesper	3.49	3.46	4.19	0.95	2.62	2.82
Therese	5.22	1.80	4.92	1.59	1.10	1.14
Helle	3.25	4.77	2.90	-0.47	1.14	1.13
Pietro	2.93	3.05	3.03	2.11	4.30	4.67
Ekaterina	2.27	2.77	1.89	2.50	4.92	5.35

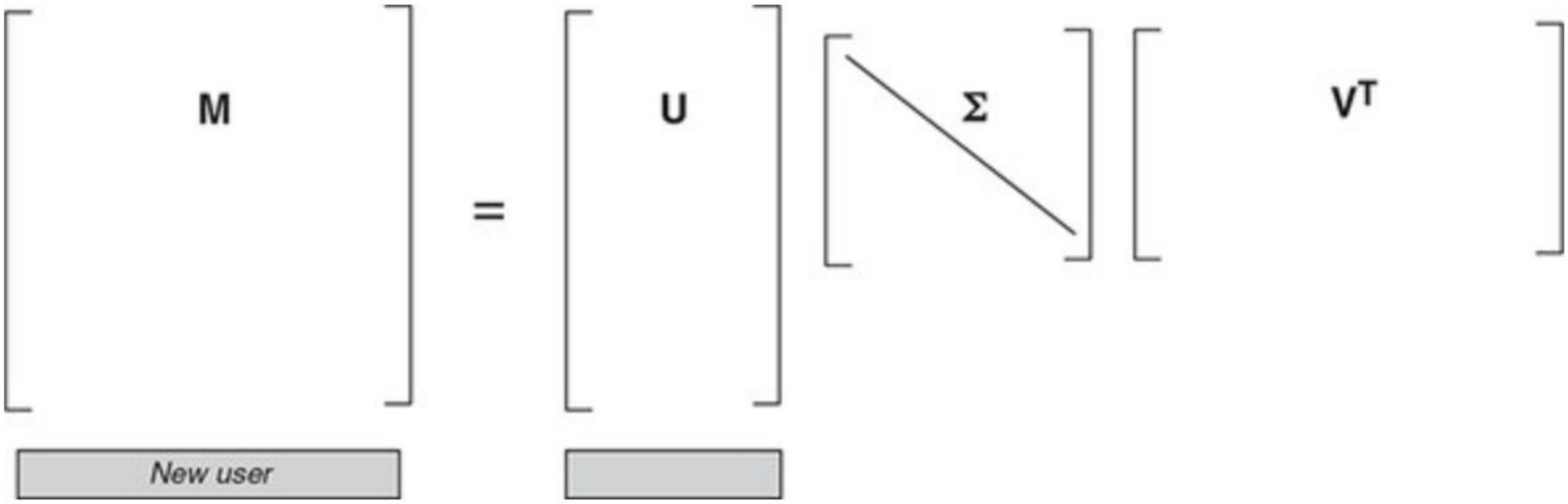
Demo - Notebook Calculation for reduced matrix

Rating Matrix Imputation

- Only a few cells in the rating matrix will have values
- Mean imputation for the User or Item
- Normalize each row centered on zero

Adding a New User or Item

- Users and Items can be added (though not before there have been interactions)



- $u_{kim} = r_k V^t \Sigma^{-1}$

where

- u_{kim} is the user vector in the reduced space to represent the new user.
- r_k is the new user's rating vector.
- Σ^{-1} is the inverse of the Sigma matrix.
- V^T is the item matrix.

```
from numpy.linalg import inv

r_kim = np.array([4.0, 5.0, 0.0, 3.0, 3.0, 0.0])
u_kim = r_kim *Vt_reduced.T* inv(Sigma_reduced)
```

Adding New Items

$$\hat{i}_{new} = r^T_{new\ item} U \Sigma^{-1}$$

where

- i_{new} is the vector in the reduced space to represent the new item.
- $r_{new\ item}$ is the new items user ratings vector.
- Σ^{-1} is the inverse of the sigma matrix.
- U—The user matrix.

Why should we update SVD when we are able to add users and items?

Updating SVD

- Latent factors are not updated when we add users and items
- Update it depending on the number of new users or items

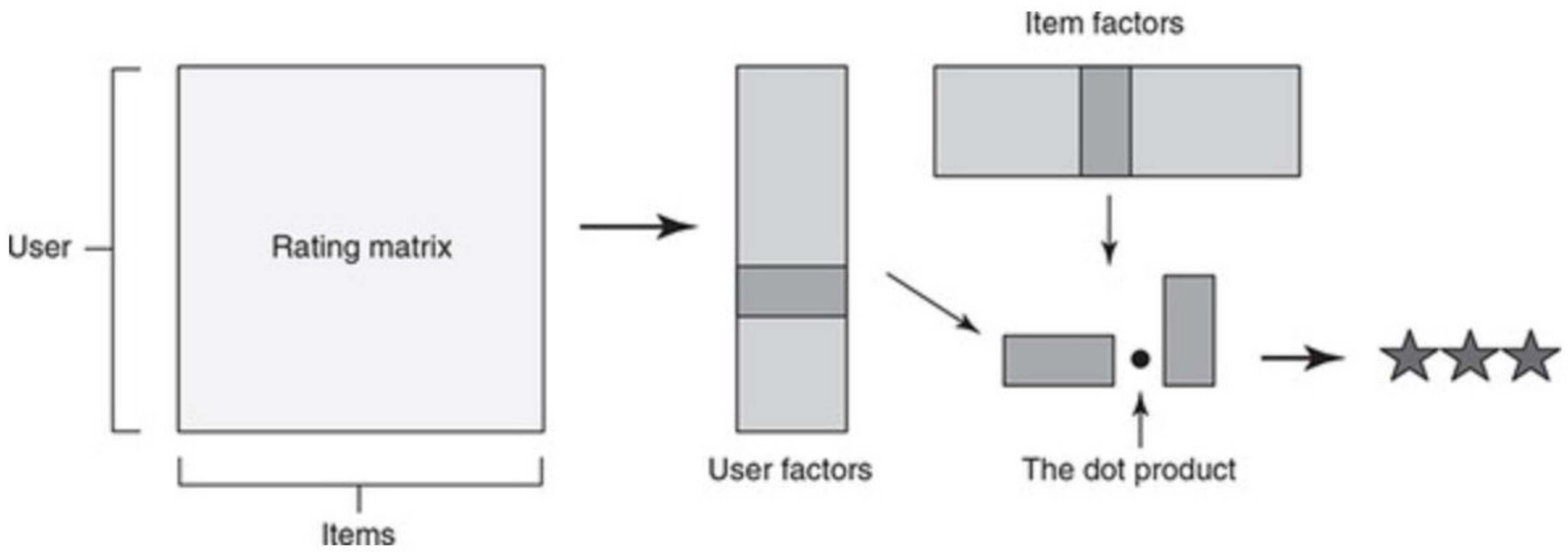
Challenges with SVD

It needs imputation with unfilled cells

Slow to calculate large matrices

SVD should be updated as often as possible

SVD isn’t at all explainable



Item factors
User
Rating matrix
Items
User factors
The dot product