

Collaborative Filtering

Hal Daumé III

School of Computing
University of Utah

me@hal3.name

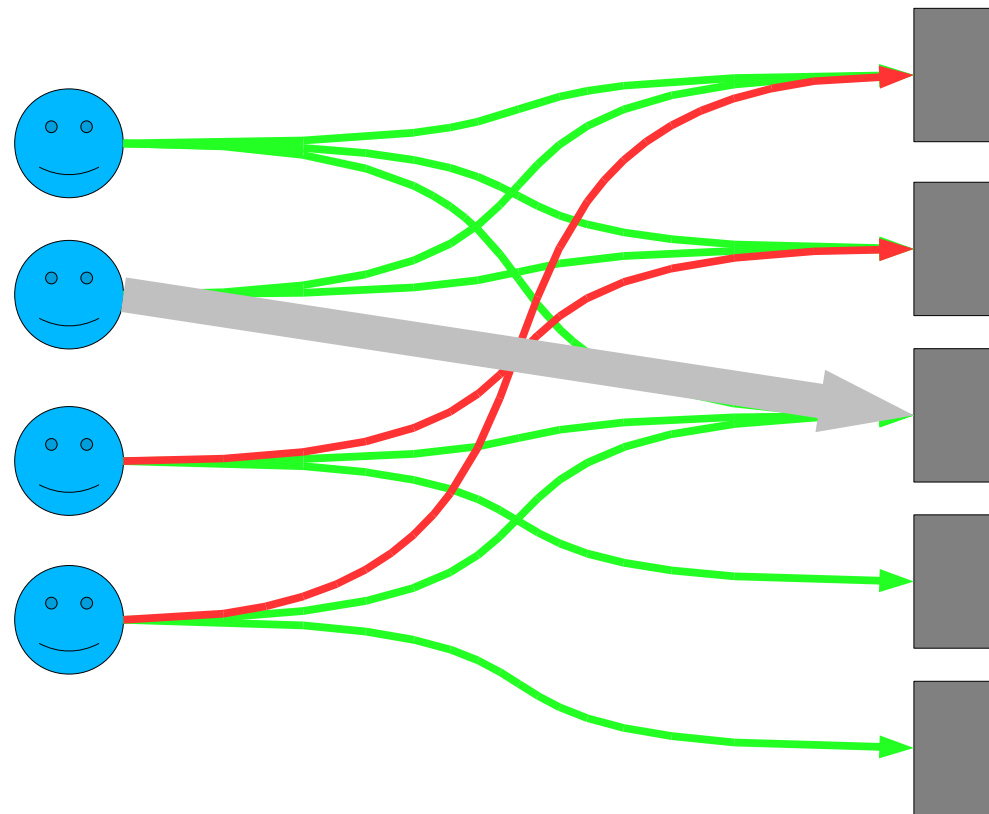


What do these have in common?



NetFlix Problem

- Given N users and D movies, with ratings on some small fraction of pairs
- Predict rating (1-5) of arbitrary user on arbitrary movie
- Real goal: maximize profit
- Approximate goal: minimize mean squared error



NetFlix Challenge

- Beat the current NetFlix system by 10%
- Reward: \$1m

Recommendations via Regression

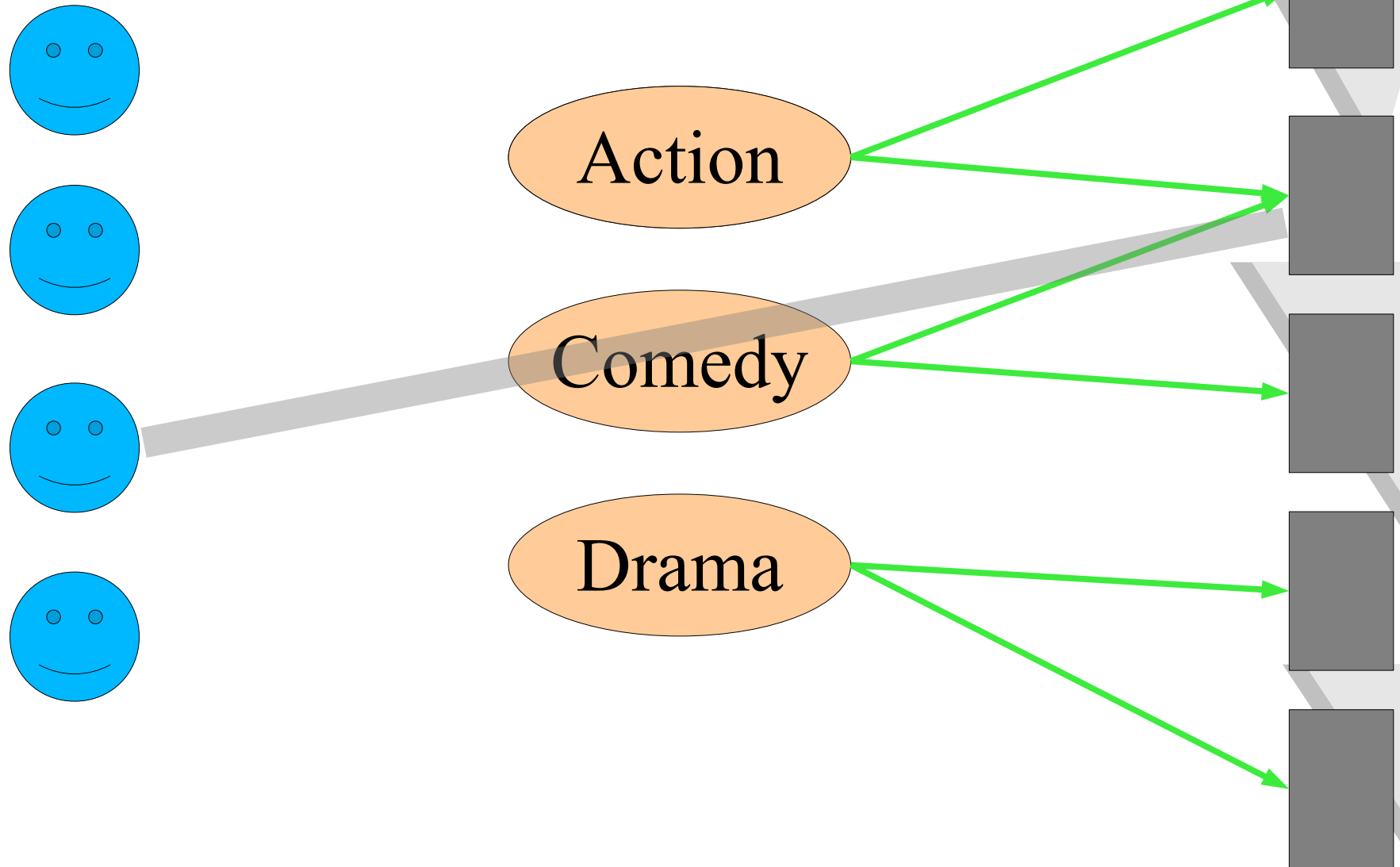
1	5	3	4	2	1
2	2	2	5	1	1
5	5	1	2	4	2
2	3	2	1	3	1
5	4	3	3	3	1
2	1	5	2	2	5
3	3	3	4	1	1

Recommendations via Regression

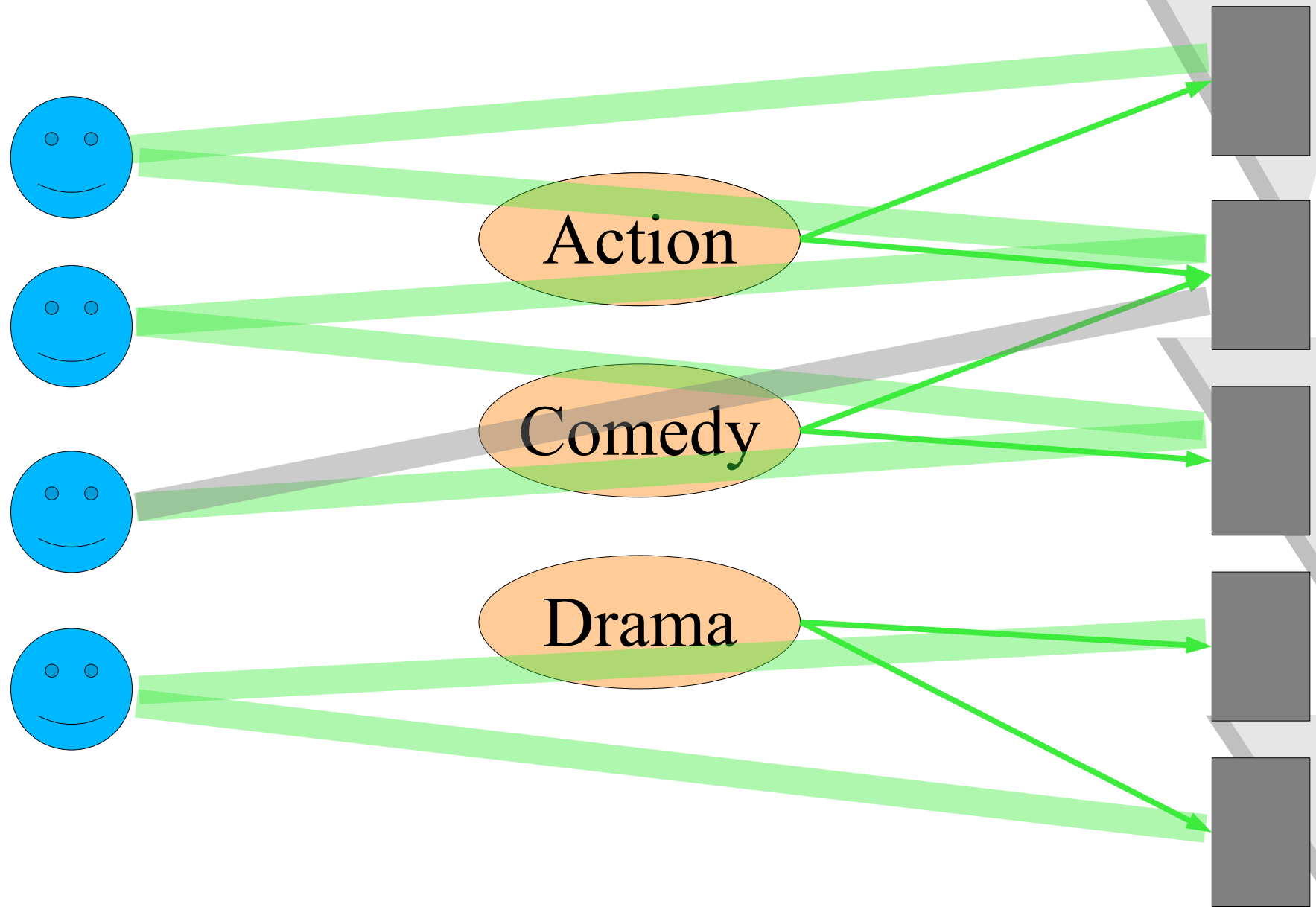
?	?	3	4	?	1
?	?	2	?	?	1
5	5	1	2	4	2
2	?	2	?	3	?
5	4	?	3	3	?
2	?	5	?	?	5
?	?	3	4	?	1

What features to make this prediction?

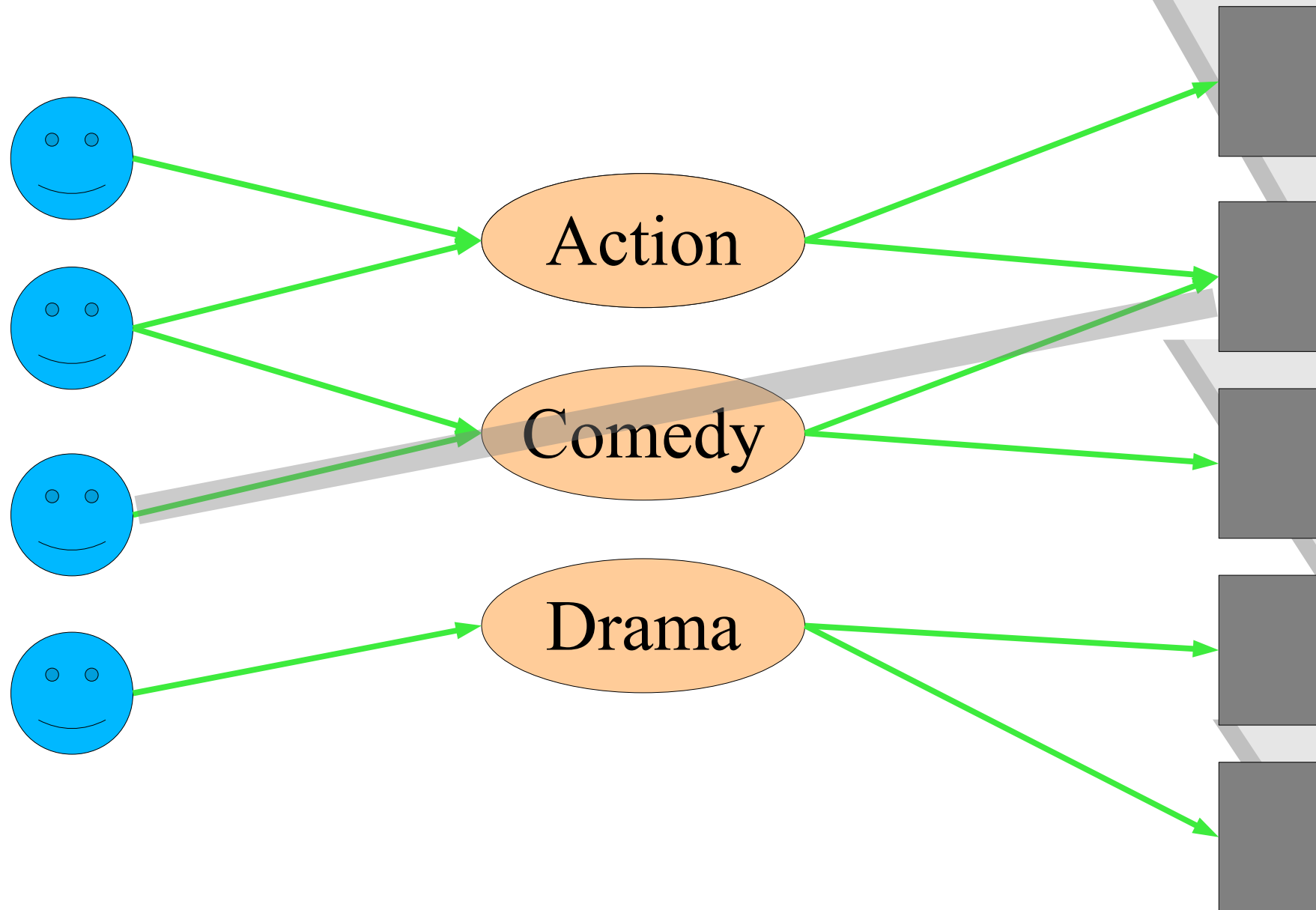
An Initial Attempt



An Initial Attempt

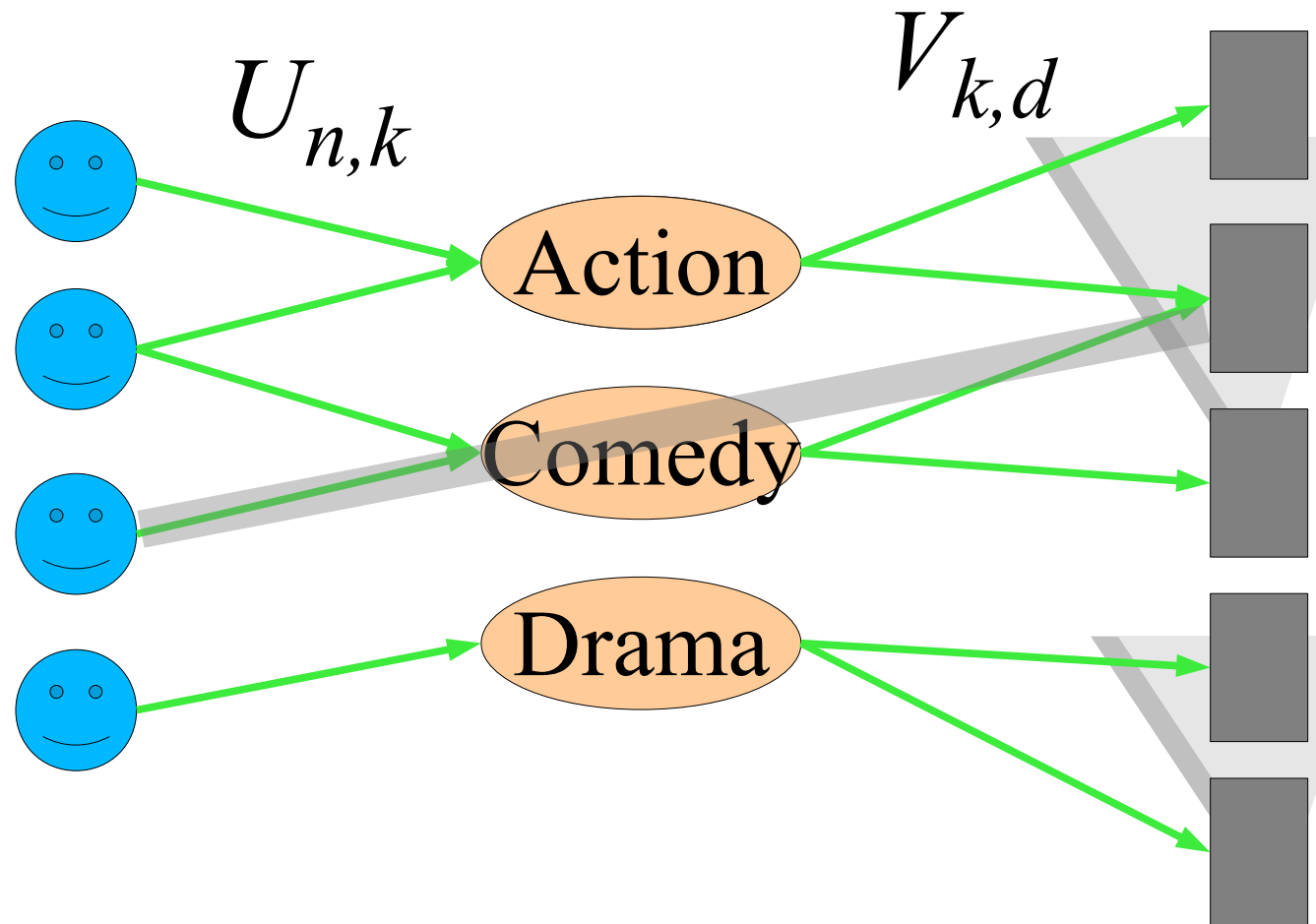


An Initial Attempt



Initial Predictions

$$\text{Score}(n, d) = \sum_k U_{n,k} V_{k,d}$$



What's Wrong?

What are the right “genres”

How many “genres”

Generalization to Digg/Amazon/Match.com

How to find “ U ” values

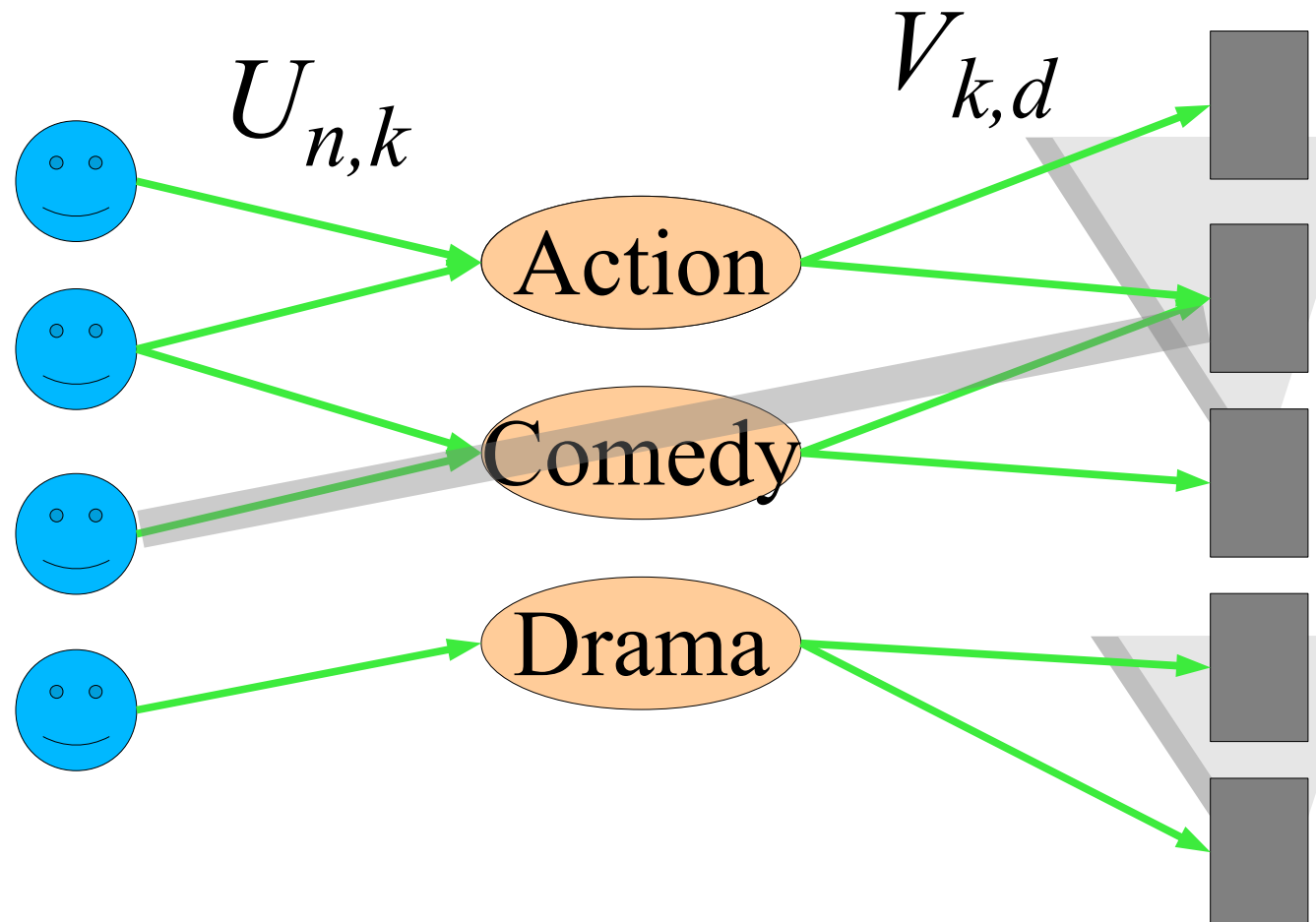
Labor-intensive creation of “ V ” values

A More Generic Approach

Think of U as a $N \times K$ matrix,
 V as a $K \times D$ matrix

Matrix
Multiplication

$$\sum_k U_{n,k} V_{k,d}$$



A More Generic Approach

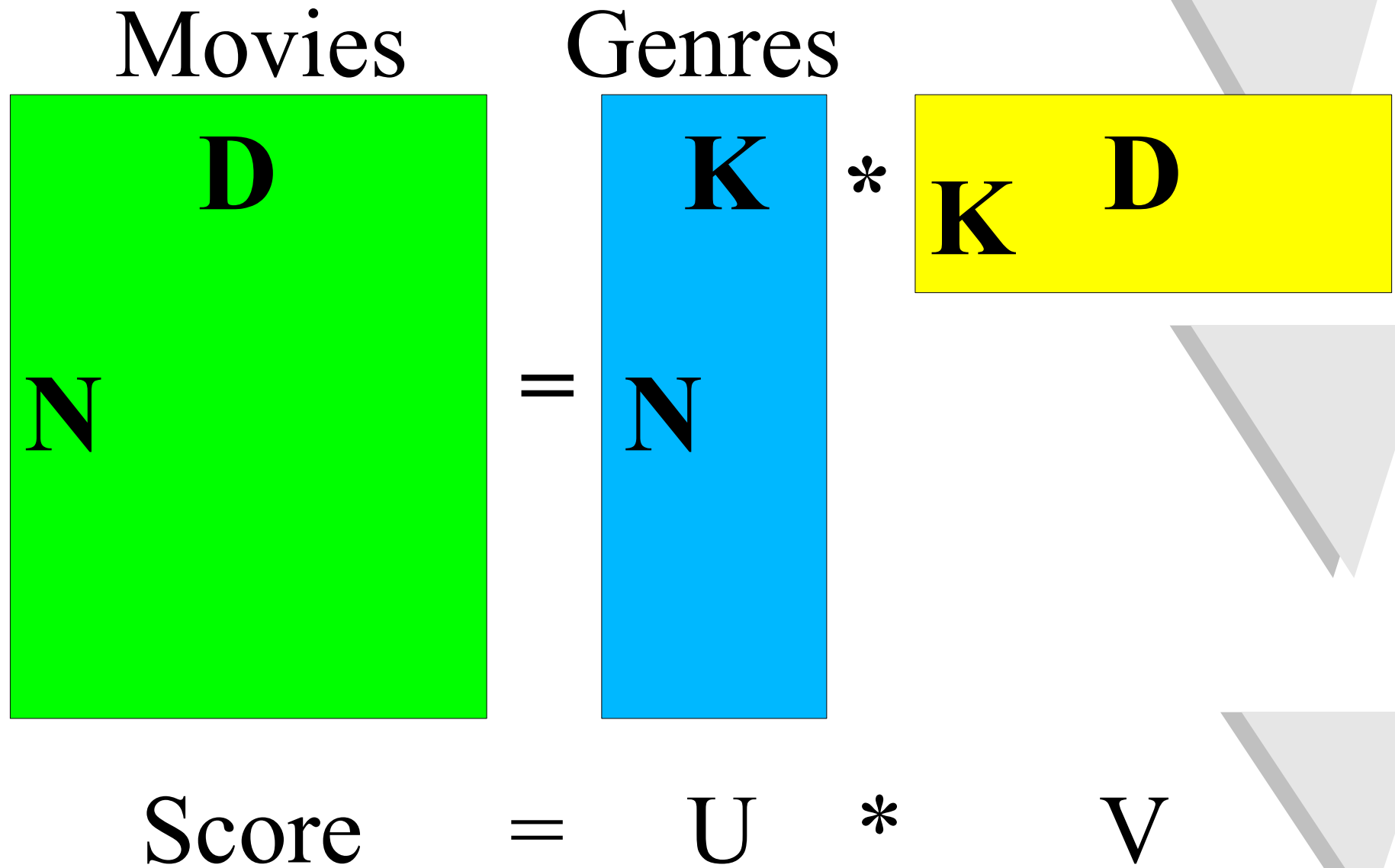
$$\begin{matrix} & \mathbf{D} \\ \mathbf{N} & \end{matrix} = \begin{matrix} & \mathbf{K} \\ \mathbf{N} & \end{matrix} * \begin{matrix} & \mathbf{D} \\ \mathbf{K} & \end{matrix}$$

$$\text{Score}_{n,d} = \sum_k U_{n,k} V_{k,d}$$

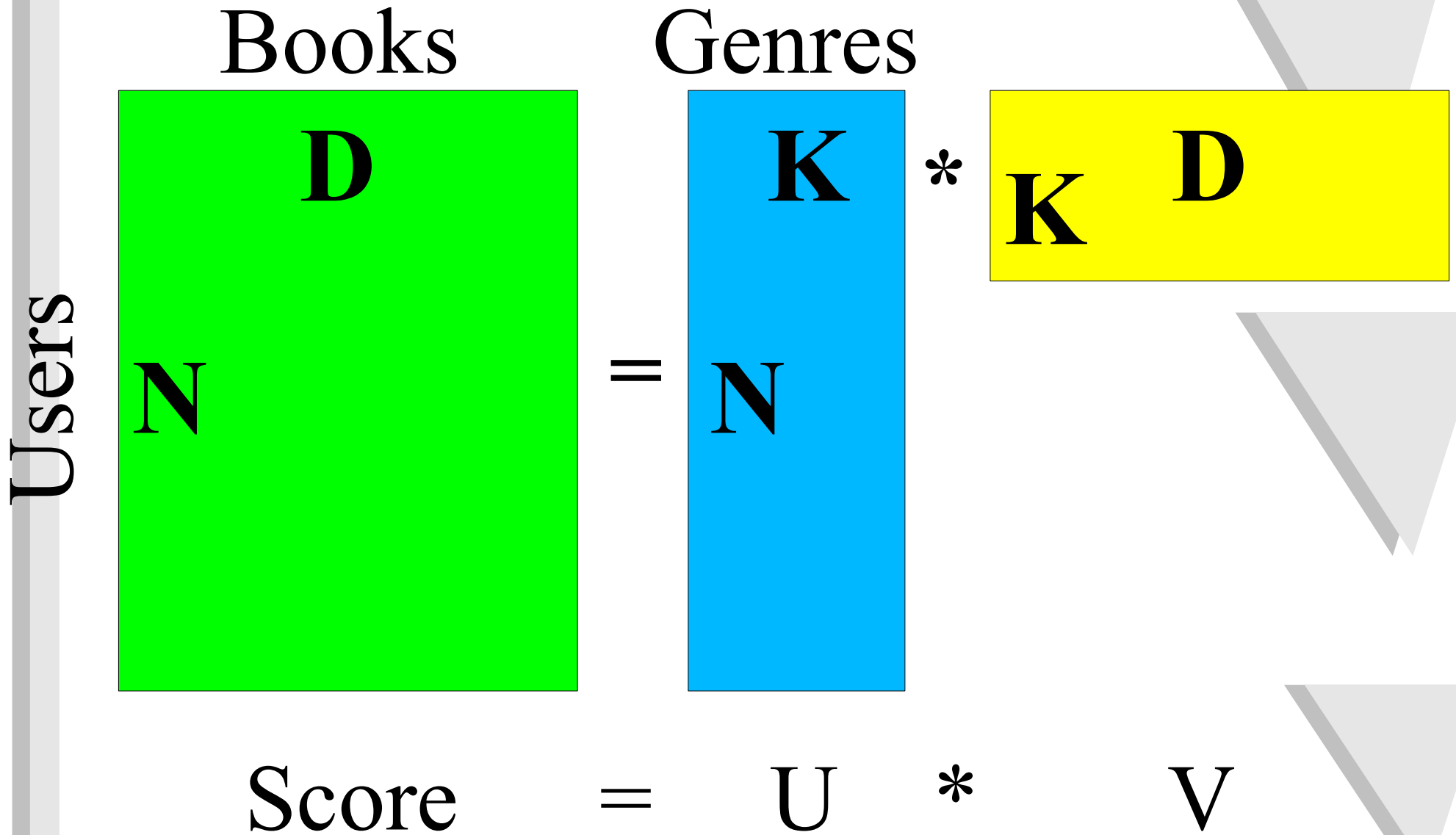
$$\text{Score} = \mathbf{U} * \mathbf{V}$$

Instantiated for NetFlix

Users

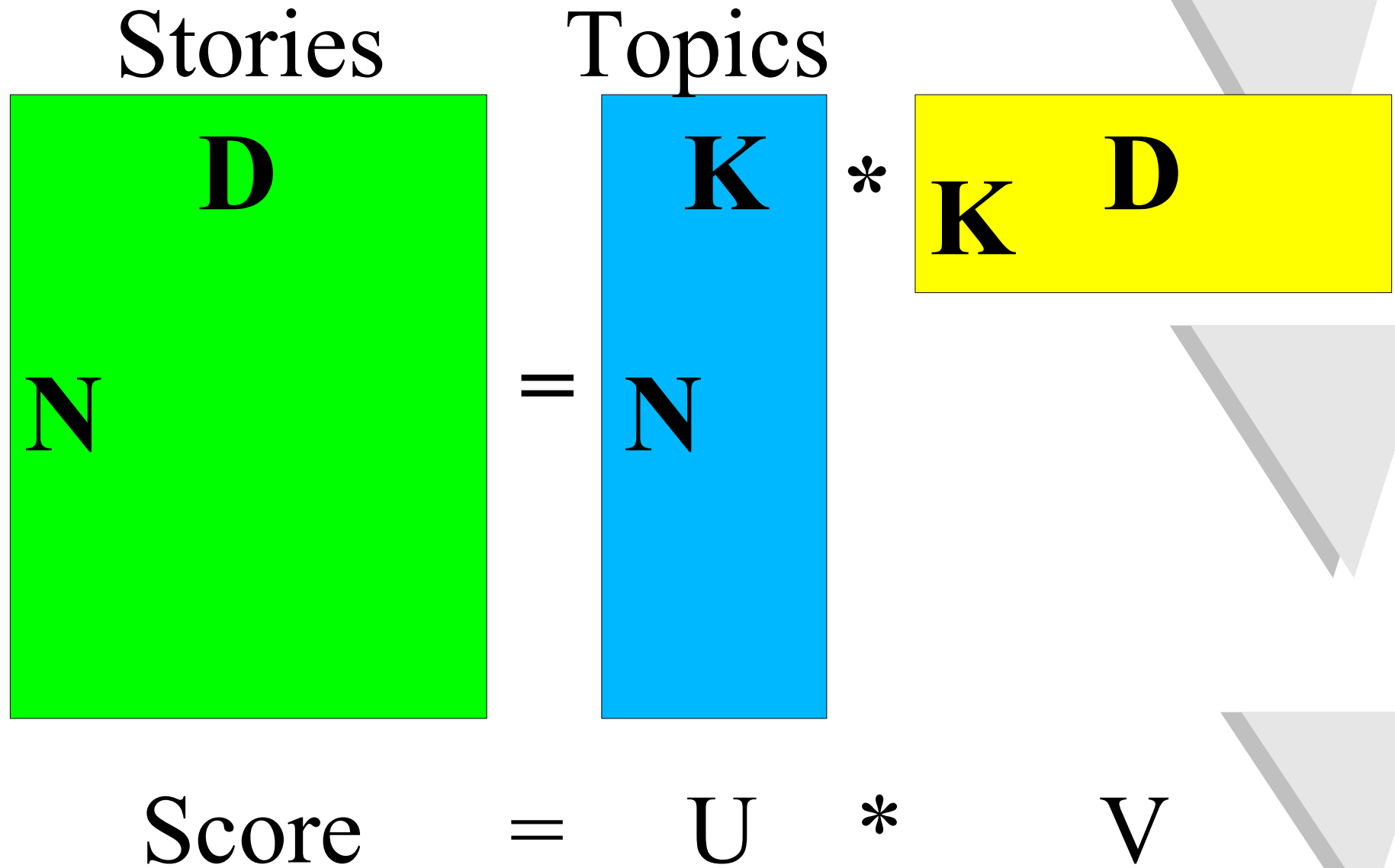


Instantiated for Amazon



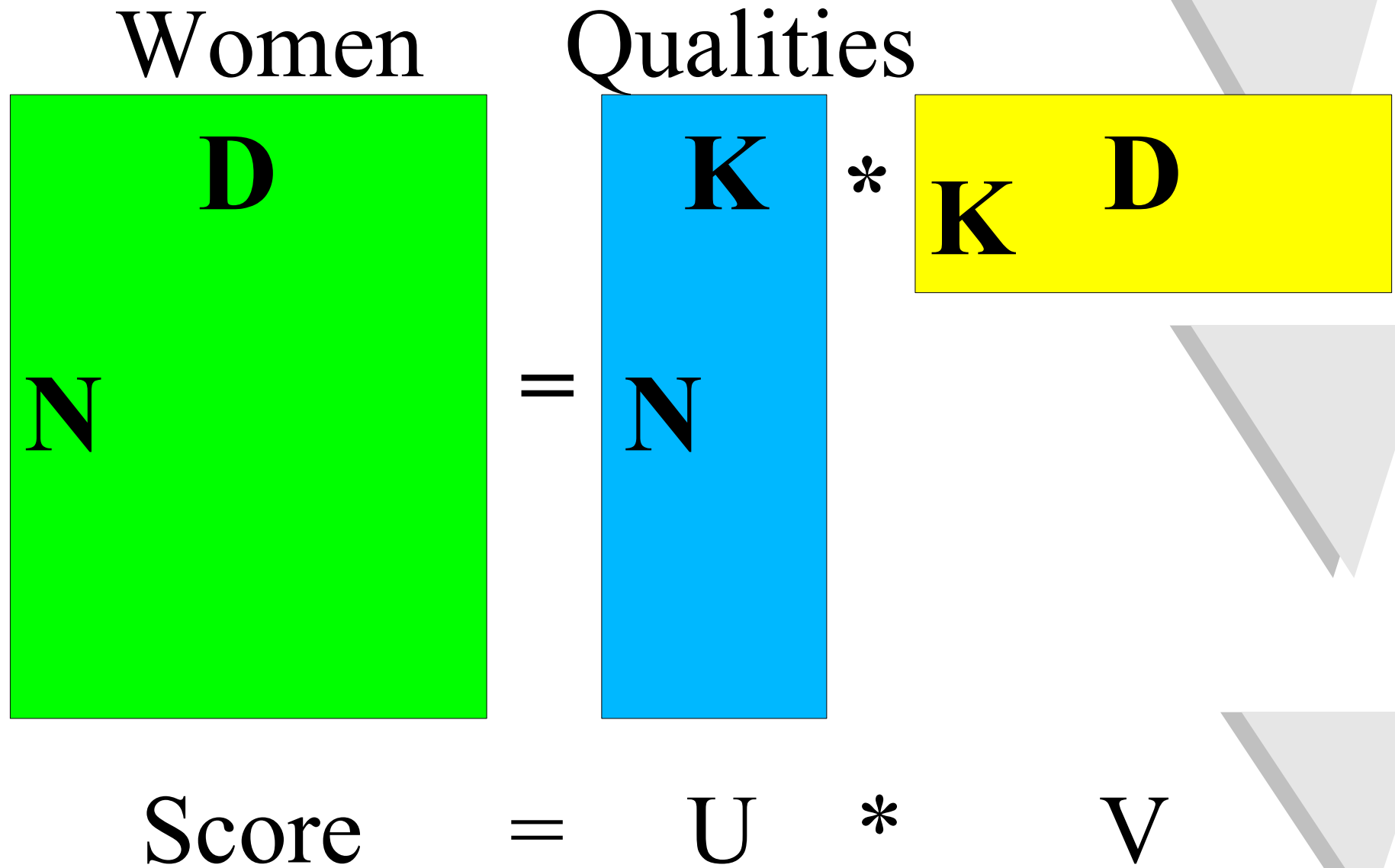
Instantiated for Digg

Users



Instantiated for Match.com

Men



How many factors can there be?

- What happens as K approaches 1?
- What happens as K approaches D ?

The diagram illustrates the matrix equation $\text{Score} = U * V$. The matrix Score is represented by a green rectangle with dimensions N (height) and D (width). The matrix U is represented by a blue rectangle with dimensions N (height) and K (width). The matrix V is represented by a yellow rectangle with dimensions K (height) and D (width). The equation is shown as $\text{Score} = U * V$, with the matrices and their dimensions labeled accordingly.

The Linear Algebra Approach

- Suppose X were completely known
- Task: Find U, V such that $X=U*V$ and U, V have rank K

Essentially a
“singular value
decomposition”
problem

$$\begin{array}{c}
 \text{D} \\
 \text{N}
 \end{array}
 X
 =
 \begin{array}{c}
 \text{K} \\
 \text{N}
 \end{array}
 U
 *
 \begin{array}{c}
 \text{K} \\
 \text{D}
 \end{array}
 V$$

The Linear Algebra Approach

- Suppose X is **not** completely known
- Task: Find U, V such that the observed $X=U*V$ and U, V have rank K
- Difficult optimization problem: iterate between finding U and V

$$\begin{array}{c}
 \text{D} \\
 \text{N} \times \text{D} \\
 \text{X}
 \end{array}
 =
 \begin{array}{c}
 \text{K} \\
 \text{N} \times \text{K} \\
 \text{U}
 \end{array}
 *
 \begin{array}{c}
 \text{K} \times \text{D} \\
 \text{K} \text{ D} \\
 \text{V}
 \end{array}$$

It's all about *Regularization*

- A solution with $K=D$ will be uninteresting. Why?
- K acts as a *regularizer*: prevents overfitting by limiting the complexity of the model

$$\begin{array}{c}
 \boxed{\begin{array}{cc} \mathbf{D} & \\ \mathbf{N} & \end{array}} \\
 \mathbf{X}
 \end{array}
 =
 \begin{array}{c}
 \boxed{\begin{array}{c} \mathbf{K} \\ \mathbf{N} \end{array}} \\
 \mathbf{U}
 \end{array}
 *
 \begin{array}{c}
 \boxed{\begin{array}{cc} \mathbf{K} & \mathbf{D} \end{array}} \\
 \mathbf{V}
 \end{array}$$

Infinite (LinAlg) Matrices

- As K approaches infinity, bad things happen
- Need another way of constraining complexity
- Idea: don't let any entry of U or V get too big

Results in a semi-definite programming problem using the trace norm

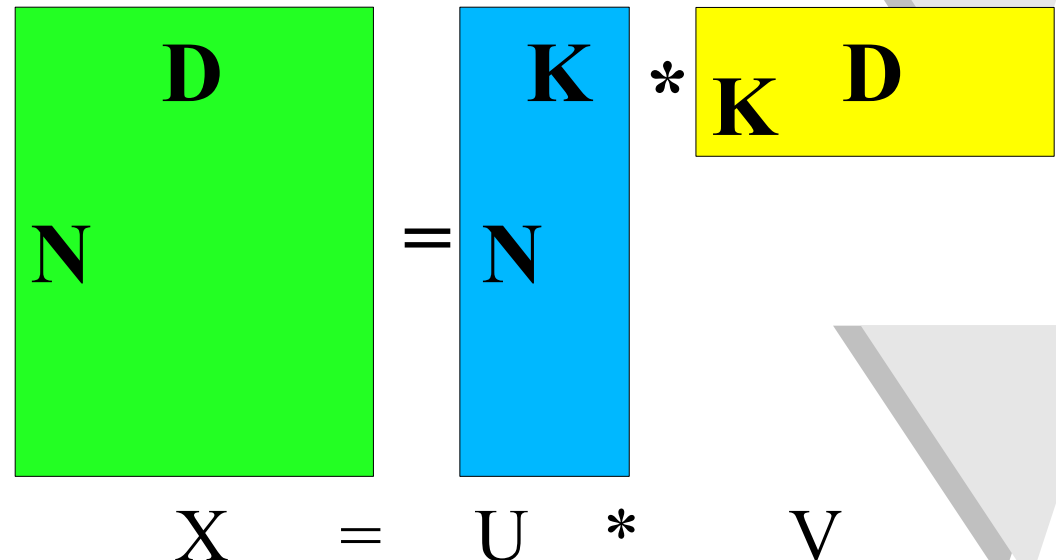
$$\begin{array}{c}
 \text{D} \\
 \text{N}
 \end{array}
 X
 =
 \begin{array}{c}
 \text{K} \\
 \text{N}
 \end{array}
 U
 *
 \begin{array}{c}
 \text{K} \\
 \text{D}
 \end{array}
 V$$

The Probabilistic Approach

- Pretend X arose because U, V were generated from some distribution, multiplied and then some noise was added

$$Pr(U, V | X) = \frac{Pr(U)Pr(V)P(X|U, V)}{P(X)}$$

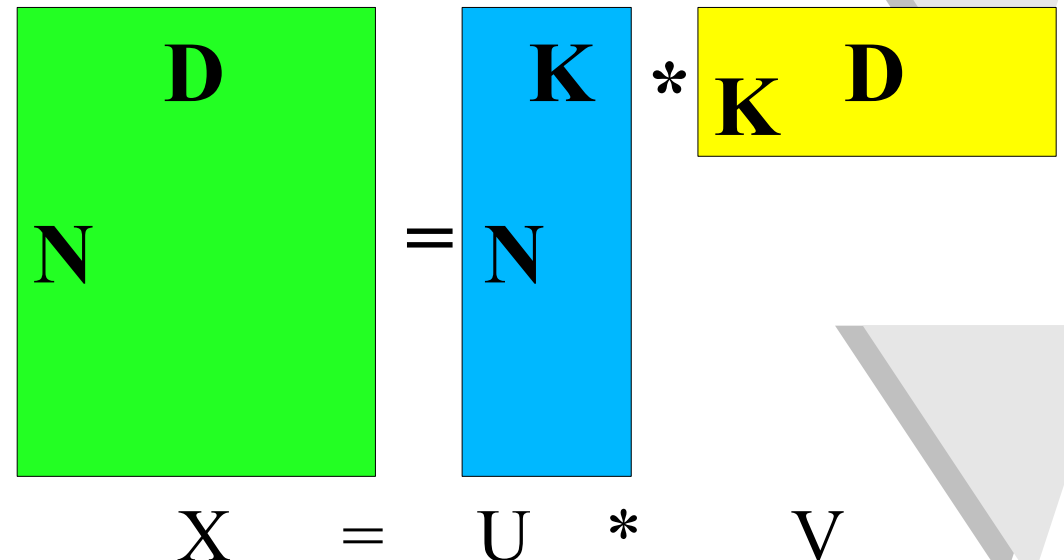
Find U, V that maximize their joint probability



Regularization as Prior Beliefs

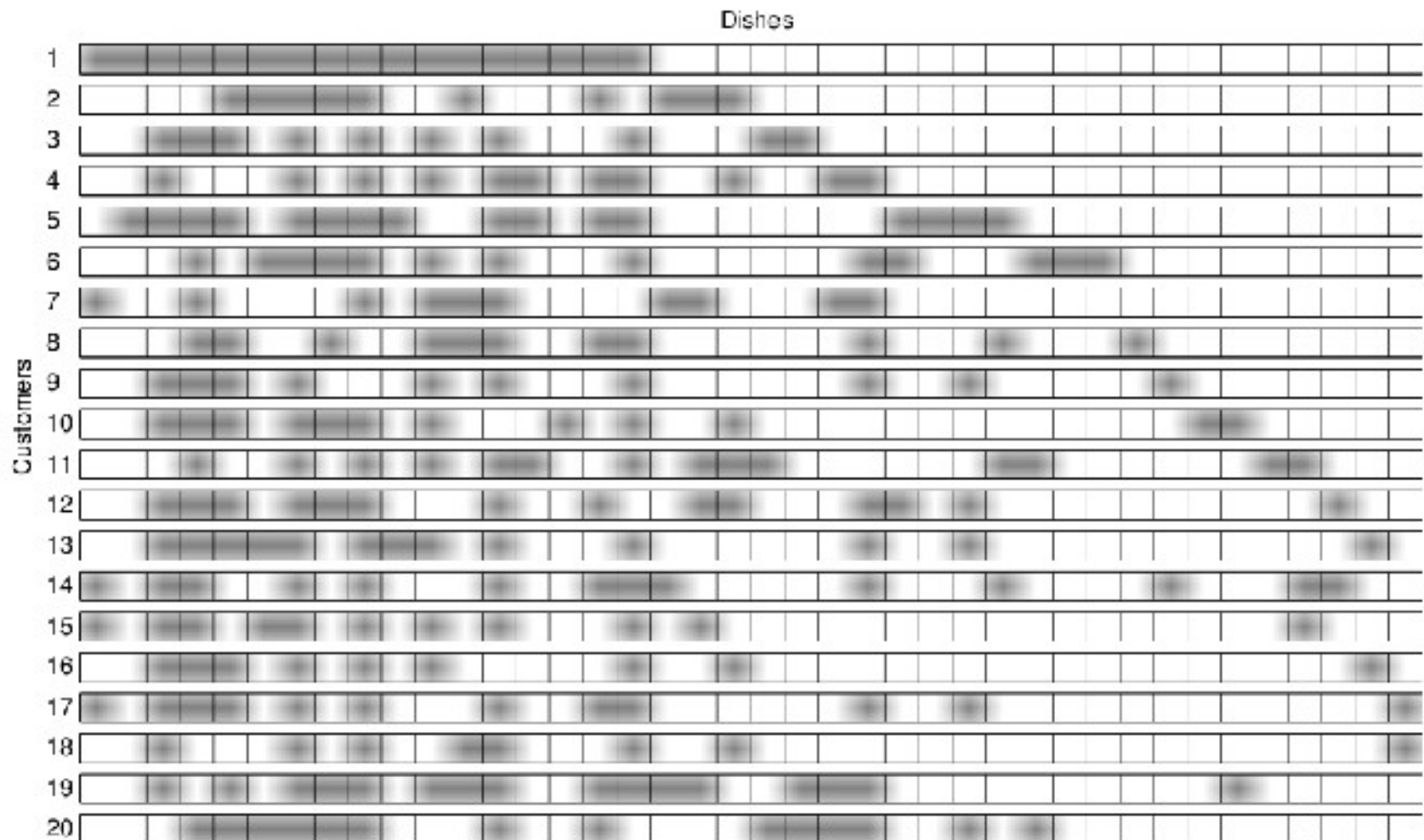
- So long as $Pr(U)$ and $Pr(V)$ favor small matrices, we'll be okay!

$$Pr(U, V | X) = \frac{Pr(U) Pr(V) P(X | U, V)}{P(X)}$$



Infinite (Probabilistic) Matrices

- Let U be generated by a *Beta Process*
- Intuitively, follows the “Indian Buffet” model



Does It Work?

- Team “BellKor” just won \$50k using these techniques.

Discussion

Recommender systems are everywhere

The best use fancy machine learning techniques

Applications abound (eg., in biology)

You could win \$1m