Data Science L&L3

An Introduction to the Mathematics of Machine Learning, Through the (Movie) Lens of Collaborative Filtering

(MovieLens is a popular dataset used for collaborative filtering projects... it's a good pun!)

Where we left off…

We saw a lot of **vectors** last time.

But what really *is* a vector?

We'll have to talk about **linear algebra** (sorry not sorry).

Continuing the discussion

One problem

Recommending items to shoppers.

Two models

Item-item and user-item collaborative filtering.

What we'll learn from these 2 models

Item-item collaborative filtering

Vectors, vector spaces, dimensions.

User-item collaborative filtering

Matrices, derivatives, minimization, train/test split, bias-variance trade off.

Item-item based collaborative filtering

jupyter Item_Item_Similarity Last Checkpoint: 02/08/2019 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

Item-item similarity

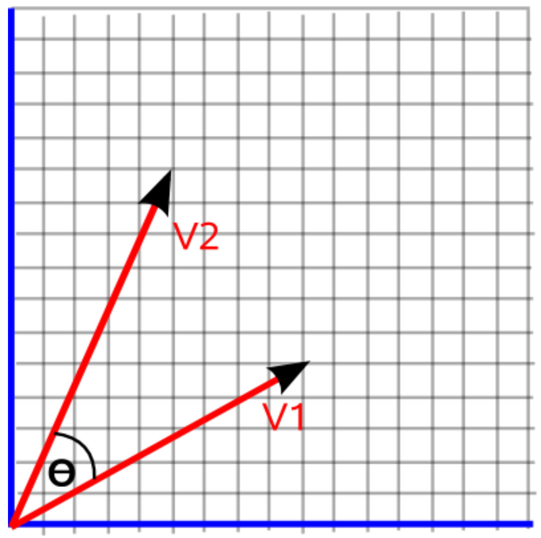
In [1]:
```
import pandas as pd
import numpy as np
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

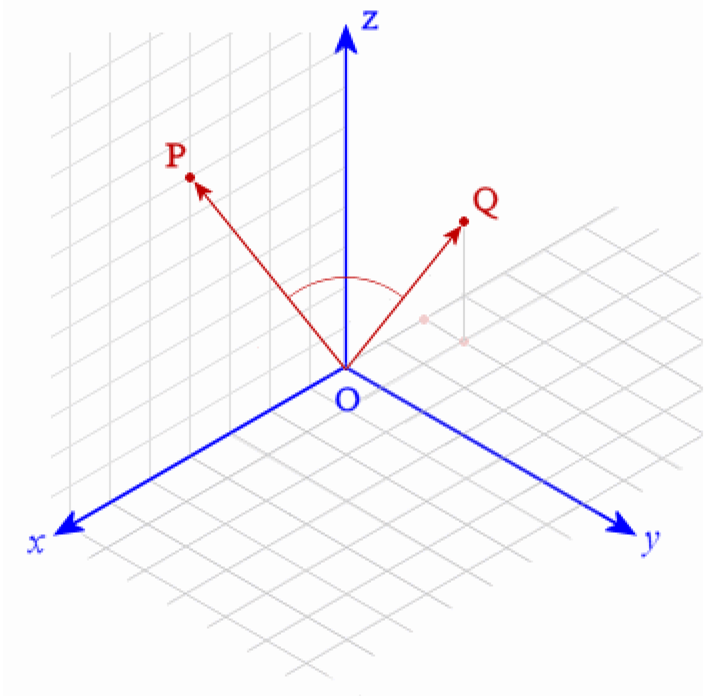
In [2]:
```
# Vector representations of 3 items (from a real model!)

bananas = np.asarray([ 0.0087136 ,  0.11107873, -0.03011765,  0.27452528, -0.01759873,
        0.18172793,  0.1278607 ,  0.100421  , -0.20177312,  0.42927584,
       -0.19039348,  0.03840672, -0.07849546, -0.12838618, -0.15439957,
        0.05887354,  0.25117505,  0.09944015, -0.23130412,  0.20598255,
       -0.1499348 , -0.00880967, -0.08055986,  0.22725877,  0.18821055,
       -0.06729905, -0.22903053, -0.10923043, -0.2247791 , -0.20998077,
       -0.23373747,  0.26395795, -0.16175163,  0.22598718, -0.04418046,
        0.3314703 ,  0.24617538,  0.11800054,  0.02048402,  0.05689507,
       -0.34176347,  0.10174422,  0.03350698,  0.09681564, -0.01070288,
        0.13358569,  0.22294806,  0.12837207, -0.08167729,  0.05172373,
       -0.1716046 , -0.20568007, -0.00381418, -0.12346429,  0.35753208,
        0.25467467, -0.05690626,  0.0649089 , -0.0691802 , -0.05593562,
       -0.05994233,  0.17475179, -0.06301752, -0.38442996,  0.36473778,
        0.25892147,  0.01931274, -0.03046758,  0.1152597 ,  0.02760088,
        0.01868668,  0.16121829,  0.0342854 , -0.3006919 , -0.12704842,
       -0.04112784,  0.01021937, -0.31746528,  0.07777109,  0.1126919 ,
       -0.1865313 , -0.10423375,  0.26352096, -0.17642814, -0.03994525,
        0.04652169,  0.19884254, -0.12860878, -0.10467128, -0.16291167,
        0.22563323,  0.21647538, -0.10508862, -0.22031806,  0.34358075,
       -0.0793887 , -0.10166975, -0.14907232,  0.06237672,  0.073644  ])
```

What is a vector?

A vector ***v*** has a magnitude and direction.

We can visualize this in 2D and 3D.





Vector Space Axioms

Definition: A nonempty set V is considered a vector space if the two operations: **1.** addition of the objects $\mathbf{u}$ and $\mathbf{v}$ that produces the sum $\mathbf{u}+\mathbf{v}$, and, **2.** multiplication of these objects $\mathbf{u}$ with a scalar a that produces the product $a\mathbf{u}$, are both defined and the ten axioms below hold. Furthermore, if V is a vector space then the objects in V are called vectors:

1. $\mathbf{u}+\mathbf{v}=\mathbf{v}+\mathbf{u}$ (Commutativity of vector addition).
2. $\mathbf{u}+(\mathbf{v}+\mathbf{w})=(\mathbf{u}+\mathbf{v})+\mathbf{w}$ (Associativity of vector addition).
3. There exists a zero vector $\mathbf{0}$ such that $\mathbf{0}+\mathbf{u}=\mathbf{u}+\mathbf{0}=\mathbf{u}$ (Existence of an additive identity).
4. For every $\mathbf{u}\in V$, there exists a vector $-\mathbf{u}$ such that $\mathbf{u}+(-\mathbf{u})=(-\mathbf{u})+\mathbf{u}=\mathbf{0}$ (Existence of an additive inverses).
5. $a(b\mathbf{u})=(ab)\mathbf{u}$. (Associativity of scalar multiplication)
6. $1\mathbf{u}=\mathbf{u}$ (Existence of a multiplicative identity).
7. $a(\mathbf{u}+\mathbf{v})=a\mathbf{u}+a\mathbf{v}$ (Distributivity of a scalar multiplication over vector addition).
8. $(a+b)\mathbf{u}=a\mathbf{u}+b\mathbf{u}$. (Distributivity of scalar multiplication over field addition)
9. If $\mathbf{u},\mathbf{v}\in V$, then $(\mathbf{u}+\mathbf{v})\in V$ (Closure under addition).
10. If a is any scalar and $\mathbf{u}\in V$, then $a\mathbf{u}\in V$ (Closure under scalar multiplication).

So what really are vector spaces?

A **vector space R**n is a mathematical object that contains vectors of dimension ***n***.

If our vector space is $\mathbf{R}^3$, then all the vectors contained in it will look something like:

$$\begin{pmatrix} 19 \\ 45 \\ 58 \end{pmatrix}, \begin{pmatrix} 94.8 \\ 53.2 \\ -0.41 \end{pmatrix} \in \mathbf{R}^3$$

But 3D is boring.

How can we have multiple dimensions?

We can picture 3D, but we can’t picture 4D. But **most things in life are multiple dimensions**.

If we’re representing Serena, we can represent her as:

$$\begin{pmatrix} nice \\ positive \\ happy \\ smart \end{pmatrix}$$

Why stop at 4? We shouldn’t put people in a box.

Why do we use vectors in ML?

Vectors allow us to represent many attributes (dimensions) in **one mathematical object**.

We can choose as **few** or as **many** attributes (dimensions) as we want.

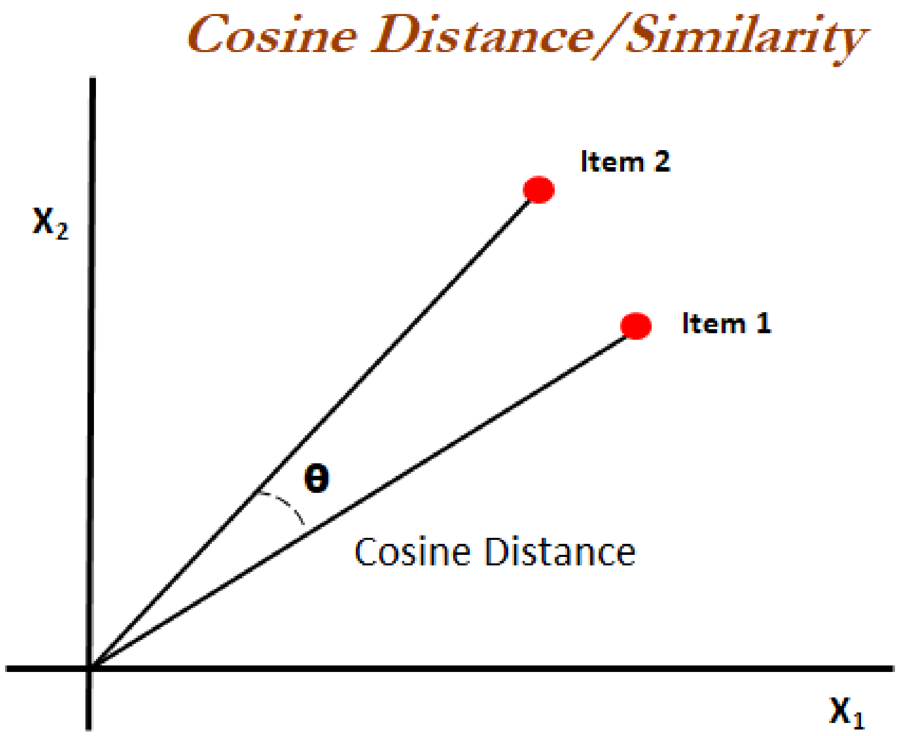
$$\mathbf{v_i} \in \mathbf{R}^n, n = \text{whatever}.$$

How do we compare vectors?

Cosine similarity is a popular option.

If we don't care about magnitude…

Given two vectors, how far apart are they? What is the **angle** between the two?



First, some more math

The **dot product** of two vectors is the sum of the product of all entries:

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

Over the real numbers, the dot product is the same as the **inner product**:

$$< a, b >$$

The **norm** of a vector is:

$$\|\boldsymbol{x}\|_2 := \sqrt{x_1^2 + \cdots + x_n^2}.$$

Generally we deal with the **L2** norm. Meaning we take the **square root**.

Why are these terms important?

Dot product, inner product, and norm are **fundamental concepts** that play into all equations in machine learning.

$$< [1, 3, -5], [4, -2, -1] > =$$

$$\begin{aligned}[1, 3, -5] \cdot [4, -2, -1] &= (1 \times 4) + (3 \times -2) + (-5 \times -1) \\ &= 4 - 6 + 5 \\ &= 3\end{aligned}$$

Finally, cosine similarity is:

$$\textbf{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

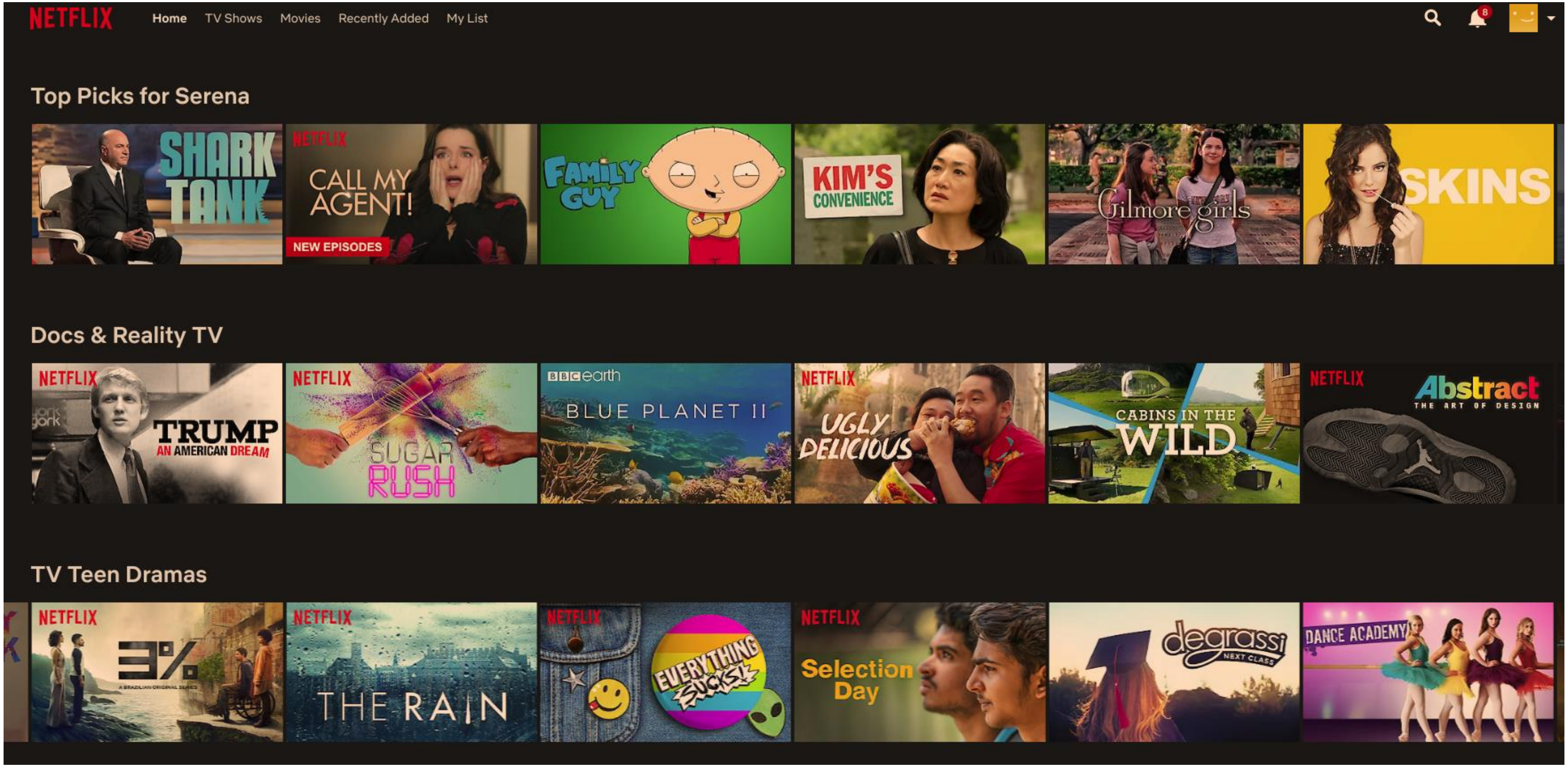
Item-item similarity with full notation

Let $v_i \in \mathbf{R}^n$ be our vector representation of product i, where $i = 1, \ldots, m$, and we have m products.

Then we can compute the similarity, sim, between products v_i, v_j as:

$$sim(i, j) = \frac{v_i \cdot v_j}{||v_i||||v_j||}.$$

User-item based collaborative filtering



Motivation for matrices

$$[1, 3, -5], [4, -2, -1] \Rightarrow \begin{bmatrix} 1 & 3 & -5 \\ 4 & -2 & -1 \end{bmatrix}$$

If we stack our vectors $v_1, v_2, \ldots v_m$ from the previous slide, we obtain an $m \times n$ matrix, since we had m products, with vectors of dimension n.

$$\mathrm{M} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ \vdots & & \ddots & \\ x_{m1} & \cdots & & x_{mn} \end{bmatrix}$$

More formally, what is a matrix?

If **v** in $\mathbf{R}^n$ and we have **m** objects, then from these vectors we can create a matrix **M** of dimension $m \times n$ with **m** rows, **n** columns.

It's like a **rectangle of vectors**.

The set of $m \times n$ matrices denoted M_{mn} is a **vector space**. It satisfies all the vector space axioms we saw previously.

A glimpse at the proof

Whenever we make claims like "the set of m by n matrices is a vector space," we need to back it up with **proof**. Let's prove that **u + v = v + u**, where **u**, **v** in $m \times n$ space.

$$u+v=\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ u_{21} & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1} & u_{m2} & \cdots & u_{mn} \end{bmatrix}+\begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \cdots & v_{mn} \end{bmatrix}=\begin{bmatrix} u_{11}+v_{11} & u_{12}+v_{12} & \cdots & u_{1n}+v_{1n} \\ u_{21}+v_{21} & u_{22}+v_{22} & \cdots & u_{2n}+v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1}+v_{m1} & u_{m2}+v_{m2} & \cdots & u_{mn}+v_{mn} \end{bmatrix}$$

$$=\begin{bmatrix} v_{11}+u_{11} & v_{12}+u_{12} & \cdots & v_{1n}+u_{1n} \\ v_{21}+u_{21} & v_{22}+u_{22} & \cdots & v_{2n}+u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1}+u_{m1} & v_{m2}+u_{m2} & \cdots & v_{mn}+u_{mn} \end{bmatrix}=\begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \cdots & v_{mn} \end{bmatrix}+\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ u_{21} & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1} & u_{m2} & \cdots & u_{mn} \end{bmatrix}=v+u$$

Why do we use matrices?

As with vectors, we use matrices to **represent our data** as **mathematical objects**. It’s convenient to represent our data by a simple **M**.

Linear algebra and matrix notation is powerful because it gives us **short hand, well defined notation** that everyone understands and **agrees on**.

Why do we decompose matrices?

Matrices are great for representing lots of data, but we don't always want all that data. If we can represent our objects using a **smaller representation**, we'll save on **computation** and **space**.

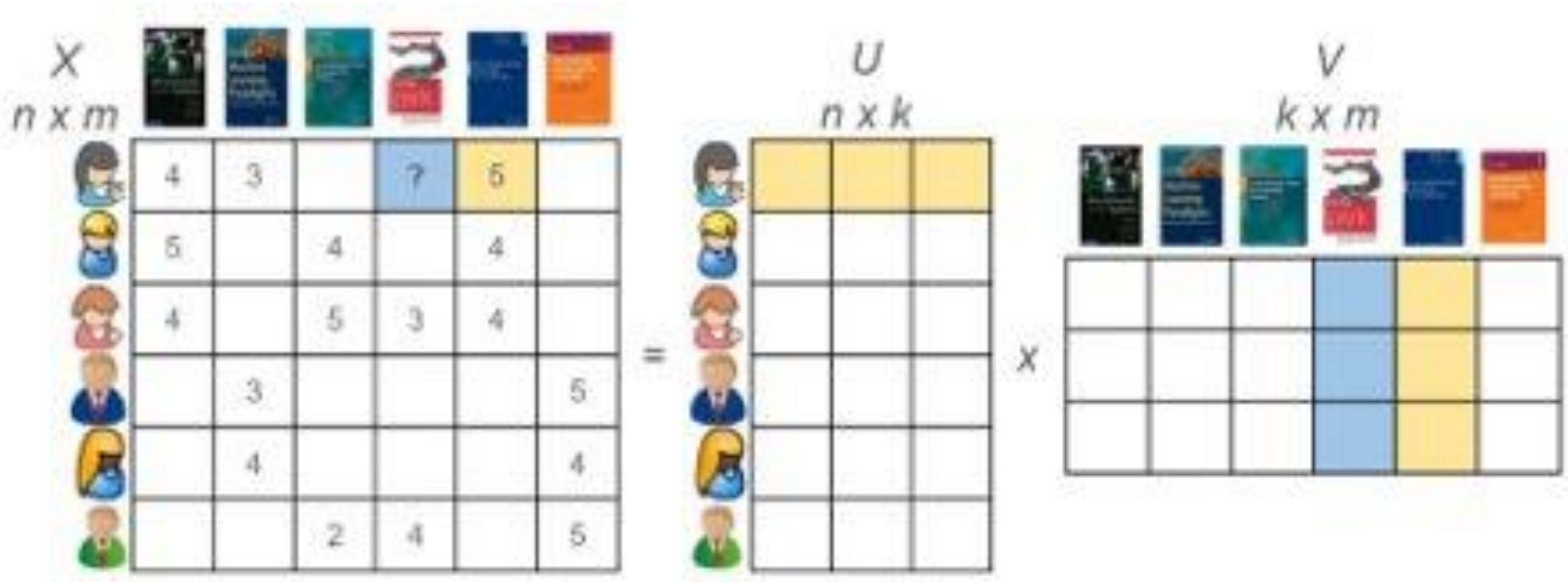
If a model can compress an object's vector attributes into a **smaller** dimension, then the model is **more efficient**.

The **simplest** solution is the **best** solution! (Why use a huge number of dimensions when we could use just a few?)

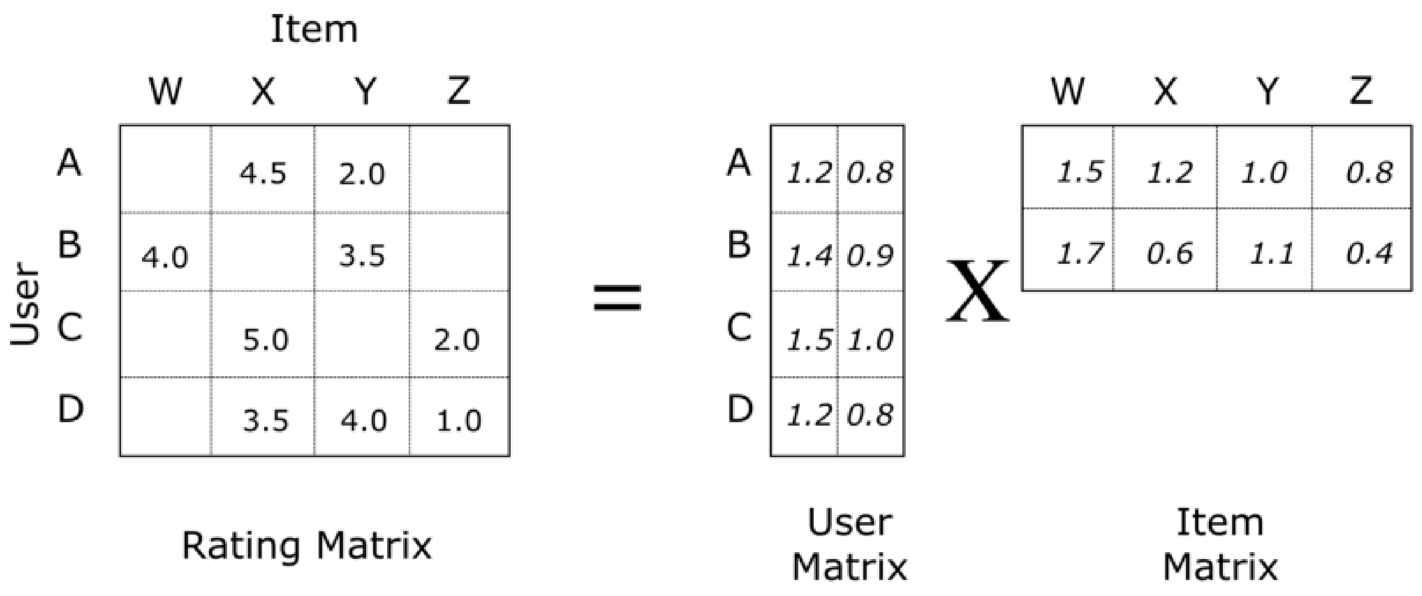
Matrix decomposition

The goal is to obtain **two matrices** whose **product** give us the original matrix.

One matrix represents **users**, the other represents **items**. We need to solve for both these matrices, somehow.



Common methods, big assumptions



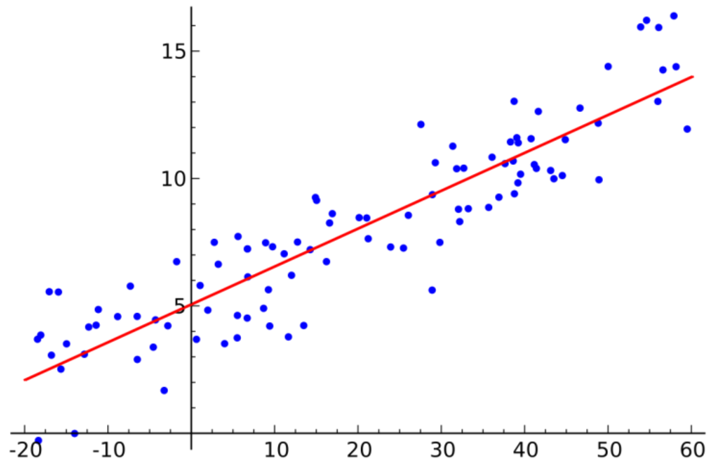
The absense of an interaction is often treated as a rating of zero, but **an absence of interaction does not indicate dislike.**

It's best to use the information we know, and avoid assumptions.

Alternating least squares to the rescue!

We will be **alternating** solving for each matrix, and will treat it like a **least squares** problem. The algorithm lets us use **only what we know**.

Least squares is a common method used to solve **linear regression** problems.



$$f(U,M) = \sum_{(i,j)\in I} (r_{ij} - \mathbf{u}_i^T \mathbf{m}_j)^2 + \lambda \left(\sum_i n_{u_i} \|\mathbf{u}_i\|^2 + \sum_j n_{m_j} \|\mathbf{m}_j\|^2 \right)$$

$R = \{r_{ij}\}_{n_u \times n_m}$ denote the user-movie matrix, where each element r_{ij} represents the rating score of movie j rated by user i

let $U = [\mathbf{u_i}]$ be the user feature matrix, where $\mathbf{u_i} \subseteq \mathbb{R}^{n_f}$ for all $i = 1 \ldots n_u$, and let $M = [\mathbf{m_j}]$ be the movie feature matrix, where $\mathbf{m_j} \subseteq \mathbb{R}^{n_f}$ for all $j = 1 \ldots n_m$. Here n_f is the dimension of the feature space

If user ratings were fully predictable, we could expect that $r_{ij} = < \mathbf{u_i}, \mathbf{m_j} >, \forall\ i, j$.

n_{u_i} and n_{m_i} denote the number of ratings of user i and movie j

I is the index set of the known ratings

How do we go from the math to a solution?

$$f(U, M) = \sum_{(i,j)\in I} (r_{ij} - \mathbf{u}_i^T \mathbf{m}_j)^2 + \lambda \left(\sum_i n_{u_i} \|\mathbf{u}_i\|^2 + \sum_j n_{m_j} \|\mathbf{m}_j\|^2 \right)$$

$\Downarrow$

????!!!?

Item

User	W	X	Y	Z
A		4.5	2.0	
B	4.0		3.5	
C		5.0		2.0
D		3.5	4.0	1.0

Rating Matrix

=

A	1.2	0.8
B	1.4	0.9
C	1.5	1.0
D	1.2	0.8

User Matrix

X

W	X	Y	Z
1.5	1.2	1.0	0.8
1.7	0.6	1.1	0.4

Item Matrix

Some terms you may encounter in machine learning

Objective function: The function we want to optimize.

Loss function: What we call the objective function when we are optimizing a single point.

Cost function: What we call the objective function when we are optimizing all our data. It’s the sum of all our loss functions.

By **optimizing** the objective function, we find parameter values that help us best approximate our **training** data.

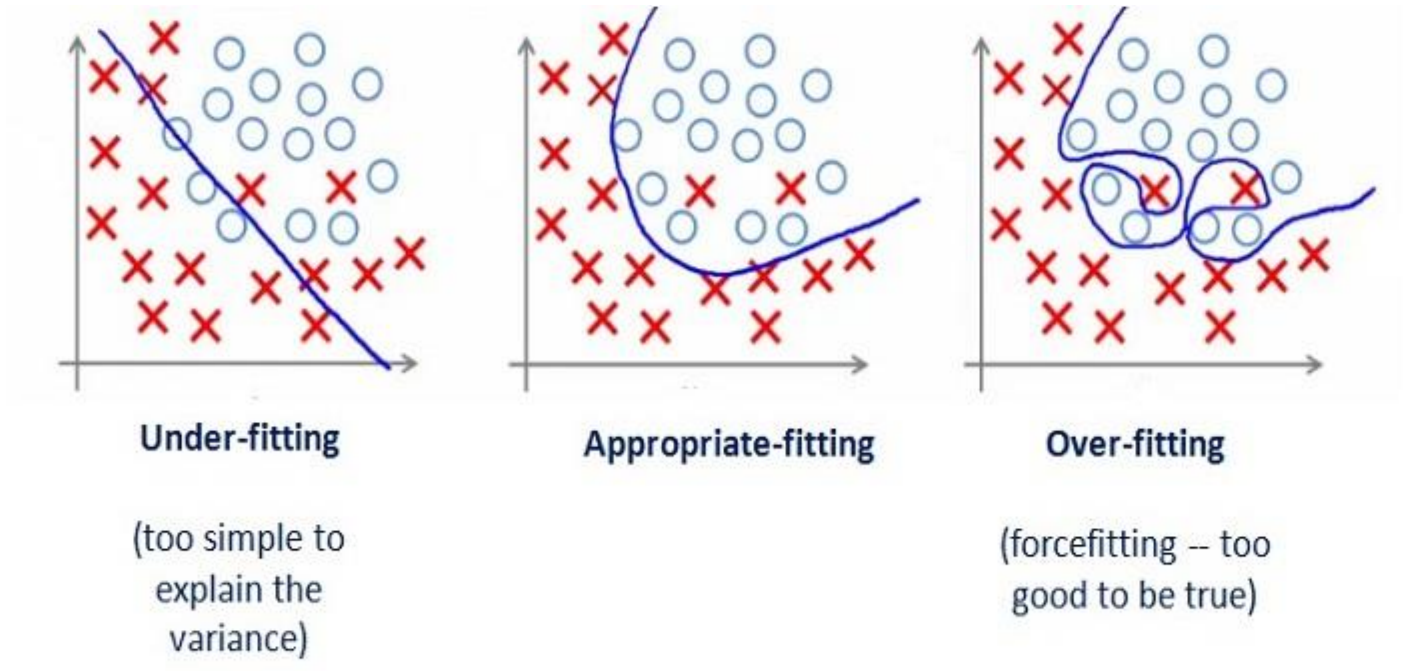
Why not use all the data? More is better, right?

More data is great, but you shouldn't train on all your data. At the very least, you need to leave some **test** data out to test your results.

If you don't, you might get **overfitting**.

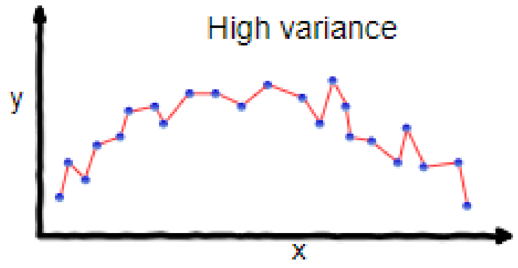
What is overfitting?

Overfitting occurs when our model learns the training data so well that **it isn't able to generalize** to new, unseen cases.

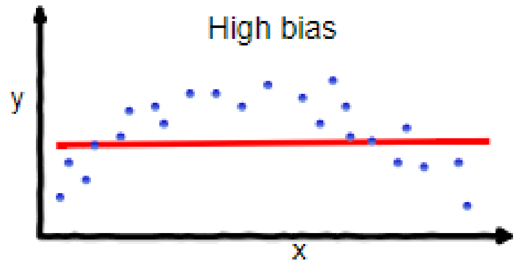


The infamous bias-variance trade off

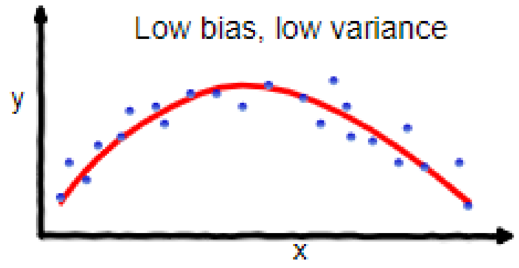
You may hear the term “bias-variance trade off.” **Bias** is underfitting, and **variance** is overfitting. It really is a trade off!



overfitting



underfitting



Good balance

Back to the equation

This is our objective function, so we want to minimize it, so we obtain **u** and **m** that are of dimension **k**, such that $r_{ij} = < \mathbf{u_i}, \mathbf{m_j} >, \forall\ i, j$.

$$f(U, M) = \sum_{(i,j)\in I} (r_{ij} - \mathbf{u}_i^T \mathbf{m}_j)^2 + \lambda \left(\sum_i n_{u_i} \|\mathbf{u}_i\|^2 + \sum_j n_{m_j} \|\mathbf{m}_j\|^2 \right)$$

Solving the ALS equation

To minimize, we take the derivative first solving for **u**, then for **m**.

$$
\begin{aligned}
&\frac{1}{2}\frac{\partial f}{\partial u_{ki}} = 0, \quad \forall i, k \\
\Rightarrow &\sum_{j \in I_i} (\mathbf{u}_i^T \mathbf{m}_j - r_{ij}) m_{kj} + \lambda n_{u_i} u_{ki} = 0, \quad \forall i, k \\
\Rightarrow &\sum_{j \in I_i} m_{kj} \mathbf{m}_j^T \mathbf{u}_i + \lambda n_{u_i} u_{ki} = \sum_{j \in I_i} m_{kj} r_{ij}, \quad \forall i, k \\
\Rightarrow &\left(M_{I_i} M_{I_i}^T + \lambda n_{u_i} E\right) \mathbf{u}_i = M_{I_i}\ R^T(i, I_i), \quad \forall i \\
\Rightarrow &\mathbf{u}_i = A_i^{-1} V_i, \quad \forall i
\end{aligned}
$$

$$f(U,M) = \sum_{(i,j)\in I} (r_{ij} - \mathbf{u}_i^T \mathbf{m}_j)^2 + \lambda \left(\sum_i n_{u_i} \|\mathbf{u}_i\|^2 + \sum_j n_{m_j} \|\mathbf{m}_j\|^2 \right)$$

$$\frac{\partial f}{\partial u_{ki}} = 0 \qquad \forall\ i,k$$

$$\Rightarrow \frac{1}{2}\frac{\partial f}{\partial u_{ki}} = \sum_{j\in I_i} (\underset{1\times k}{\vec{u}_i^T}\ \underset{k\times 1}{\vec{m}_j} - \underset{\substack{1\times 1\\(a\ \#)}}{r_{ij}})\ \underset{1\times 1}{m_{kj}} + \lambda \sum_i \underset{1\times 1}{n_{u_i}} \underset{1\times 1}{u_{ki}}$$

$$m_j^T u_i = u_i^T m_j$$

$$\Rightarrow \sum_{j\in I_i} \underset{1\times 1}{m_{kj}}\ \underset{1\times k}{m_j^T}\ \underset{k\times 1}{\vec{u}_i} + \lambda \sum_i \underset{\substack{1\times 1\\(a\ \#)}}{n_{u_i}} \underset{1\times 1}{u_{ki}} = \sum_{j\in I_i} \underset{1\times 1}{r_{ij}}\ \underset{1\times 1}{m_{kj}}$$

M_{I_i} = movie mtx M where only movies that user i rated are included.

M is $k\times n_m$ so M_{I_i} is $k\times n_{u_i}$

apply summation of j over I_i, then $\forall\ k$:

$$\Rightarrow \underbrace{\underset{k\times n_{u_i}}{M_{I_i}}\ \underset{n_{u_i}\times k}{M_{I_i}^T}}_{\substack{\shortparallel\\ k\times k}}\ \underset{k\times 1}{\vec{u}_i} + \underbrace{\lambda\ \underset{(a\#)}{n_{u_i}}\ \underset{k\times k}{E}\ \underset{k\times 1}{\vec{u}_i}}_{\substack{\shortparallel\\ k\times 1}} = \underbrace{\underset{k\times n_{u_i}}{M_{I_i}}\ \underset{n_{u_i}\times 1}{R^T(i,I_i)}}_{\substack{\shortparallel\\ k\times 1}}$$

$R(i,I_i)$ is row vector w/ cols that are ratings where user i rated the movie.

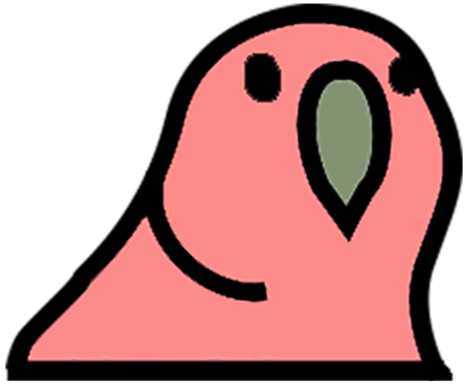
R is $n_u \times n_m$, and $R(i,I_i)$ considers just user i and movies $j\in I_i$, so $R(i,I_i)$ is $1\times n_{u_i}$

E is $k\times k$ Identity mtx

then $u_i = A_i^{-1} v_i \ \forall\ i$, where $A_i = M_{I_i} M_{I_i}^T + \lambda n_{u_i} E$, $V_i = M_{I_i} R^T(i, I_i)$

How do we go from the math to a solution?

$$f(U,M) = \sum_{(i,j)\in I} (r_{ij} - \mathbf{u}_i^T \mathbf{m}_j)^2 + \lambda \left(\sum_i n_{u_i} \|\mathbf{u}_i\|^2 + \sum_j n_{m_j} \|\mathbf{m}_j\|^2 \right)$$



Item

User	W	X	Y	Z
A		4.5	2.0	
B	4.0		3.5	
C		5.0		2.0
D		3.5	4.0	1.0

Rating Matrix

=

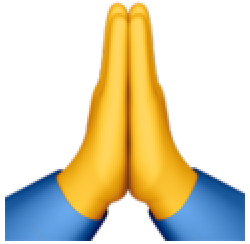
A	1.2	0.8
B	1.4	0.9
C	1.5	1.0
D	1.2	0.8

User Matrix

X

W	X	Y	Z
1.5	1.2	1.0	0.8
1.7	0.6	1.1	0.4

Item Matrix



In conclusion, finally…

Focusing on collaborative filtering, we walked through an introduction to the math of machine learning. We touched on the following topics:

1. Linear algebra
2. Similarity measures
3. Test/train split
4. Bias-variance tradeoff
5. Optimization methods