Part 1: Using Embeddings

Suppose we're using a dot product (no bias) collaborative filtering model:

Users:

User 1	2	0	-2
User 2	1	-1	-1
User 3	0	1	-1
User 4	0	-1	0
User 5	-1	1	-1

Movies:

Movie 1	0	-1	0
Movie 2	-1	-2	0
Movie 3	1	1	1

Compute the dot products to score how much each user likes each movie:

dot(user 2, movie 1) = ___
dot(user 1, movie 2) = ___
dot(user 4, movie 3) = ___

Part 2: Constructing Embeddings

Now let's construct embeddings. Fill in numerical values for the vectors below so that the following relationships hold (where u1 means User 1, etc.):

Dot(u1, m1) = 1.0, Dot(u1, m2) = 0.0, Dot(u1, m3) = 1.0
Dot(u2, m1) = 0.0, Dot(u2, m2) = 1.0, Dot(u2, m3) = 1.0

Users

User 1			
User 2			

Movies

Movie 1			
Movie 2			
Movie 3			

Part 3: Learning Embeddings

For the u1 and m1 vectors you constructed above:

1. Is there an element of u1 that has zero effect on dot(u1, m1)? How could you tell?
2. For each element of u1, what is the gradient of dot(u1, m1) with respect to that element? Now put those gradients together as a vector. Can you express that vector symbolically using u1 and m1?
3. Repeat the previous question for m1.