

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:

$\text{dot}(\text{user 2, movie 1}) = \underline{\hspace{1cm}}$

$\text{dot}(\text{user 1, movie 2}) = \underline{\hspace{1cm}}$

$\text{dot}(\text{user 4, movie 3}) = \underline{\hspace{1cm}}$

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.).

$\text{Dot}(u1, m1) = 1.0$, $\text{Dot}(u1, m2) = 0.0$, $\text{Dot}(u1, m3) = -1.0$

$\text{Dot}(u2, m1) = 0.0$, $\text{Dot}(u2, m2) = -1.0$, $\text{Dot}(u2, m3) = 1.0$

Users

User 1			
User 2			

Movies

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

Part 3: Learning Embeddings

The previous section had a trivial solution because the movies were completely independent.

Now we limit the dimensionality. So you *won't* be able to find a perfect solution. But we can see how we can learn embeddings with gradient descent. See *tips* at the end.

Users

User 1	-1.0
User 2	1.0

Movies

Movie 1	
Movie 2	
Movie 3	

1. Pencil in -1.0 and 1.0 for the two users and all zeros for the movies. We'll try to learn the movie embeddings by gradient descent. (In a real situation we'd initialize both matrices randomly, but this will keep it simple enough to do by hand.)

2. Compute $\text{dot}(u_1, m_1)$. Compute the MSE loss on this “minibatch” by squaring its difference from the *desired* value (1.0) given above.
3. Compute the *gradient* of the loss with respect to the movie-1 embedding (m_1).
4. Use the gradient to determine what adjustment to the movie-1 embedding will reduce the loss.
5. Compute the loss again using the updated embedding. Make sure it went down.

Tips: Computing the gradient may seem tricky. But this problem is set up so that it is easy.

Suggested approach:

1. Compute $\text{pred} = \text{dot}(u_1, m_1)$. Notice that this has a particularly simple form in this case.
2. Compute $\text{diff} = \text{target} - \text{pred}$
3. Compute $\text{loss} = \text{diff}^2$
4. Compute the gradient with respect to *diff*. Check it by thinking about what would happen to the loss if you wiggled the diff a bit.
5. Use that to compute the gradient with respect to pred . Check again, the same way.
6. Use that to compute the gradient with respect to m_1 . Check again.