Open in app Get started

Published in bag of words

You have **1** free member-only story left this month. Sign up for Medium and get an extra one

Brett Vintch Follow
May 12, 2020 · 5 min read ★ · Listen

Save

What similarity metric should you use for your recommendation system?

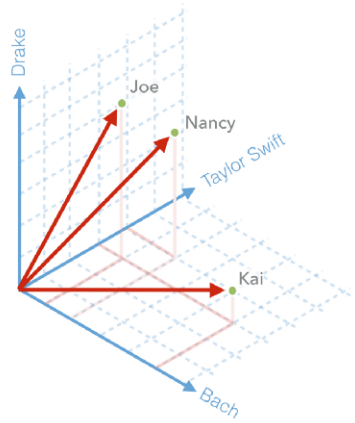
credit: microassist

There isn't a single right answer. It depends on many things, including the model you choose and the type and distribution of your data. However, understanding the mechanics of each similarity metric from a geometric perspective can help make the process of choosing a metric more transparent.

Open in app Get started

will look at inner product similarity, cosine similarity, Pearson correlation, and Euclidean proximity. The first three are particularly interesting because they are all variations of inner products.

Representing user ratings as vectors



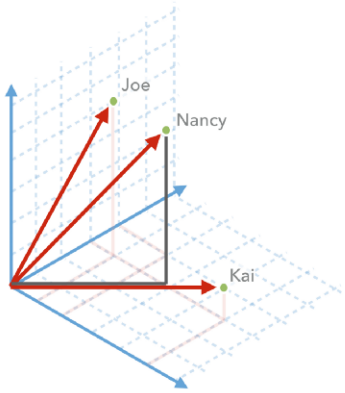
User ratings can be represented as vectors in Euclidean space. Take for example a music service where users rate artists with between 1 and 5 stars. Joe doesn't much care for Bach, is indifferent to Taylor Swift, and loves Drake; thus, for the item set [Bach, Taylor Swift, Drake] Joe's ratings vector is [1, 3, 5]. Two other users, Nancy and Kai, have vectors [2, 4, 3] and [5, 3 1], respectively. These user vectors are depicted in the figure above, and in this toy scenario the average rating for each user is the same: 3 stars.

Note that in this particular example, all three users have rated all three artists; there are no null values. While this is not typical of most real-life settings where item catalogs are large, this geometric interpretation is still valid for both Collaborative Filtering and Matrix Factorization. For Collaborative Filtering algorithms, this is because we only compare users over the set of items for which each user has provided a rating (i.e. the intersection of items). Therefore, the specific subspace of items is different for every two pairs of users and there are no null values. For Matrix Factorization, this is because most existing models learn dense embeddings for every user and every item.

Open in app Get started

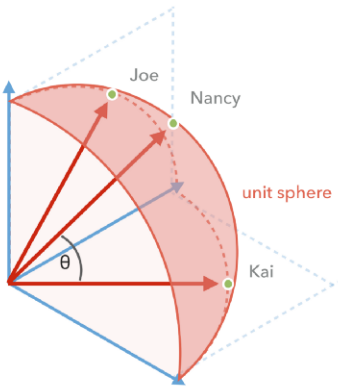
interaction between users and items is usually explicitly modeled as an inner product.

inner product



$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos(\theta)$$

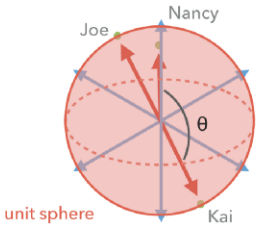
cosine similarity



$$= \cos(\mathbf{a}, \mathbf{b})$$

$$= \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}$$

Pearson Correlation



$$= \frac{\mathbf{a}_c \cdot \mathbf{b}_c}{\|\mathbf{a}_c\| \|\mathbf{b}_c\|}$$

$$= \cos(\mathbf{a}_c, \mathbf{b}_c)$$

From the figure above, it should be apparent that each of the similarity metrics is some version of a normalized inner product. Cosine similarity is identical to an inner product if both vectors are unit vectors (i.e. the norm of **a** and **b** are 1). This also means that cosine similarity can be calculated by first projecting every vector to the unit sphere, and then taking the inner product between the resulting vectors. Furthermore, Pearson correlation is the same thing as cosine similarity if the vectors are centered first by removing the vector mean (denoted with a subscripted *c*). Then, the same practice of projecting to the unit sphere and then taking the inner product applies.

If we view these metrics as operations on pairs of user ratings vectors, then we can summarize them as follows:

inner product	cosine similarity	Pearson Correlation
• not centered	• not centered	• centered
• vector length matters	• vector length *does not* matter	• vector length *does not* matter
• vector direction matters	• vector direction matters	• vector direction matters

Open in app Get started

Collaborative Filtering or Matrix Factorization. This may be because the operation is less efficient than inner product-based metrics, but it may also be because odd things can happen for short vectors near the origin. The nearest neighbor of a vector with a small magnitude could be a vector 180 degrees away, reflected across the origin. In cases where the vector direction doesn't have a lot of meaning this may not be very important, but this is usually not the case for recommendation systems.

So which metric should you choose?

There's no simple rule of thumb. The best way to choose a metric is to optimize predictions offline with cross-validation, or optimize them online with an A/B test. The right metric for one data set may not be the right metric for another because of the distribution of feedback values over users, items, and interactions. However, combining the geometric interpretation above with a bit of experience allows us to make a few observations:

- Cosine similarity and Pearson correlation naturally normalize rating scales across users because they don't care about the magnitude of the user vectors.
- It doesn't make sense to use Cosine similarity on unary or binary data, which is common in the case of implicit feedback, or for ordinal data that is not normalized or centered. This is because these rating vectors will always be in the first quadrant for every user, which vastly limits the dynamic range of the Cosine metric. That is, for two vectors in this quadrant, they can be a maximum of 90 degrees from one another which gives a floor of 0 to the similarity metric, versus 180 degrees, or -1, for centered data.

If we are seeking the nearest items for a user with vectors obtained from Matrix Factorization, there are few other things to note:

- The most common formulation for Matrix Factorization models ratings as the dot product between the user vector and the item vector. Therefore, when looking for the best items for a user the natural similarity metric should be a dot product: the


Open in app Get started

because these items and users have high average ratings, and long vectors are more likely to give a high predicted rating when multiplied.

- Using Cosine similarity or Pearson correlation helps to mitigate the bias towards popular items, but can also end up recommending very unpopular, niche items.
- Euclidean proximity gives a nice balance between being sensitive to vector direction (for vectors far enough from the origin) without being overtly biased to popular items. In my own work I've found that optimizing models that represent distance with Euclidean proximity often gives the most intuitive results.

Your mileage may vary. The most important things to remember are: 1) the performance of each metric is dependent upon the distribution of your data, and 2) your specific business objectives may help clarify the choice (for example, you may be willing to accept a slight dip in accuracy if the results are more diverse). It's worth noting that there are many other ways to measure distance and we didn't cover metrics well suited to implicit feedback, such as Jaccard distance, or its nearest neighbor approximation MinHash. Even then, a geometric approach can yield a new perspective and aid in the decision making process.

33

About Help Terms Privacy

Get the Medium app

Open in app Get started