

Chapter 2

Basic Structure of High-Dimensional Spaces

Data is naturally represented geometrically by associating each record with a point in the space spanned by the attributes. This idea, although simple, raises a number of challenging problems in practice.

2.1 Comparing Attributes

For any single attribute, it is not necessarily the case that the significance of the values it can take depends linearly on those values. For example, for an attribute that measures income, almost all of the values will be between 0 and a few hundred thousand (almost regardless of currency); however, there will be a few individuals whose income is three or four decimal orders of magnitude greater than this. Should this huge difference in income amount be treated as a huge difference in significance? Perhaps, but there's at least a case that it should not. One plausible way to address this would be to discretize attribute values into ranges with semantics; for example, incomes could be placed into a few categories such as *low*, *medium*, *well-off*, *wealthy*, and *super-rich*.

However, there is then the problem of how to measure differences in the values of a single attribute between two different records. The significance of a difference does not follow immediately from an understanding of the significance of a magnitude. For example, small differences may plausibly be treated as no difference at all.

The most common (dis)similarity measure based on the natural geometric embedding is *Euclidean distance*—but this measure is built from the *squares* of the differences in the values of each individual attribute. It therefore implicitly claims that, for differences, significance grows much faster (quadratically) than magnitude.

2.2 Comparing Records

When the data has more than one attribute, a new difficulty appears. To compare two records, we must compare the differences in the values of two or more attributes measuring different things. How can we combine the difference for each attribute into a difference between the whole records? There is no straightforward way to do this—it really is comparing apples and oranges. This applies even if both attributes are measuring the quantity or number of “the same” underlying objects, for example dollar amounts or word frequencies. Similarity depends on the range and distribution of the values that each attribute takes over the whole dataset, not just on the kind of objects that it describes.

The standard approach, although it is difficult to justify, is *normalization*. Normalization means converting the raw values for each attribute into a standardized form that is the same for all attributes. For example, a common approach to normalization, for each attribute, is to compute the mean and standard deviation of the values of that attribute across the entire dataset. In the data table or matrix, this means computing the mean and standard deviation of each column. The values in the column corresponding to the attribute are then converted by subtracting the column mean from each value, and dividing the result by the column standard deviation. For the entire set of values of each attribute, subtracting the mean centers them around zero, with roughly half positive and half negative. Dividing by the standard deviation makes the ranges of all of the attributes roughly comparable. If the original attribute values were drawn from a Gaussian distribution, this normalization maps two-thirds of the values to the range between -1 and $+1$. This normalization is called *z-scoring*.

Other normalizations are possible. For example, the values of each attribute could be mapped into the range $0-1$, but the effect depends on the maximum and minimum values taken by the attribute more than on the distribution of values it possesses in the dataset. Calculations of the mean could be trimmed, that is some of the largest and smallest values could be omitted to give a more robust estimate of the distribution of values.

2.3 Similarity

So now suppose that all of the attributes have been normalized in some reasonable way. There are still choices about which kind of function will be used to combine the per-attribute similarities.

By far the most common way to do this combining is to use Euclidean distance between the points corresponding to each record. Of course, distance is a *dissimilarity* measure since points that are far apart are not similar to one another—but the relationship between distance and similarity is straightforward and we can think about it either way.

Viewed geometrically, using Euclidean distance seems sensible; it’s the way we think about distance in the real world. But Euclidean distance between the points

corresponds to taking the difference in the values for each attribute, *squaring it*, and then adding up these squares and taking a square root. The squaring step means that two records with a large difference in the values of only a single attribute seem disproportionately far apart because of the impact on the sum of this one term. From this perspective, Euclidean distance seems less obvious.

Other common distance calculations include: Manhattan distance (sum the differences for each attribute), or Hamming distance (sum the number of times the values for each attribute are different, regardless of how much).

There is a practical problem with computing distances or similarities as well. If n is large (there are many records) then n^2 distance calculations need to be done, and this may simply be too expensive. Fortunately, for the purpose of understanding the global structure of the data, only the other points fairly close to each point need to be looked at, and this creates the opportunity for some optimizations that we will see later. But the quadratic complexity of computing all of the pairwise distances means that some common approaches do not scale to large, high-dimensional (many attribute) datasets.

The next layer of complexity comes from the choice of attributes. For most datasets, it is not clear from the beginning which attributes will turn out to be important, so there is a natural tendency to collect any attributes that *might* be. The presence of these extra attributes makes the natural geometric space seem to be of higher dimension than it really is—and this, of course, alters the apparent similarity between each pair of records. For example, a single extra attribute with uniform random values for each entry will pull all of the records slightly further apart than they “should” be, but by a random amount, so blurring the similarity structure.

A more subtle problem happens when a subset of the attributes are measuring almost the same property but in different ways. At a macroscopic scale, this means that the values of each pair of attributes in the subset must be highly correlated. The effect of this redundant subset is that records that differ in this underlying property seem much more different than they should be, because the difference gets added into the sum multiple times.

If the set of attributes are well correlated then the solution is obvious—remove all but one of the attributes from the dataset. This tends not to work in practice—often two attributes will be strongly correlated over much of their range but uncorrelated over the rest, and it is not clear whether this latter range is important. Correlation is also not transitive, which further complicates trying to remove redundant attributes.

Finally, most real-world clusters are actually *biclusters*, that is, within each subset of the records similarity depends on only a small subset of the attributes, and this subset is different for each cluster. Ignoring this property, and computing distances using all of the attributes, blurs the tightness of each cluster and makes them all harder to detect. Datasets that contain biclusters are often analyzed with algorithms that simultaneously try to cluster the records *and* the attributes. There is an inherent symmetry in the data in this case: records are similar because of a particular subset of the attributes, but attributes are also similar because of a particular subset of the records.

Fig. 2.1 Two biclusters inhabiting different subsets of dimensions

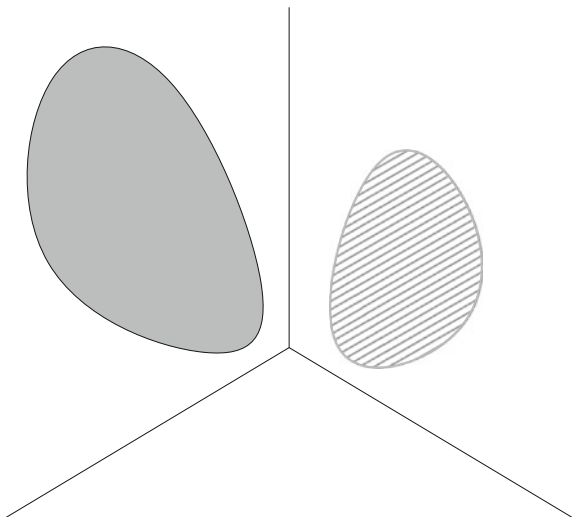


Figure 2.1 shows two biclusters and illustrates some of the difficulties. Naive analysis, perhaps based on distance, makes the bottom of both clusters seem similar so such an algorithm might discover three clusters, the top of each of the visible clusters, and a third consisting of the bottom halves of both.

2.4 High-Dimensional Spaces

Now we turn to issues that arise because of the high-dimensional nature of the natural space, issues which test our intuition developed in three-dimensional space. First of all, in high-dimensional spaces, distances do not behave quite as we expect. Let us suppose a dataset with m attributes (so an m -dimensional space) and suppose that the attribute values in each record are divided into three simple categories: large and positive, large and negative, and close to zero. Now consider records whose attribute values are equally likely to be one of these three categories. How likely is it that such a record will be close to the origin in the m -dimensional space? This can only happen if all of its attribute values are in the close-to-zero category, and the chance of this is $(1/3)^m$. This is an extremely small probability when m is large. In other words, if the records have uniformly chosen entries, almost all of them will lie far from the origin.

Now consider any two records. How likely is it that they are close to each other? This can only happen if their entries, for every attribute, match; that is, they are both large and positive, both large and negative, or both close to zero. The probability of this is, again, $(1/3)^m$ and so extremely small when m is large.

In other words, in high-dimensional spaces, uniformly randomly distributed points are all far from each other. The impact on data with more structure is that the relative distances from a point to its nearest neighbor and its furthest neighbor are similar, especially in relation to the absolute distances to both. This is why indexing schemes such as k-d trees that allow nearest neighbors to be found in low-dimensional spaces do not scale to high-dimensional spaces—they have an often-ignored exponential dependence on the dimensionality.

One approach to this problem is to view the space spanned by the attributes as a vector space instead of a Cartesian space. In this view, each point is regarded as the endpoint of a vector from the origin. Each point, therefore, has two associated properties: its *distance* from the origin, and its *direction* from the origin. These aren't new properties, just a different way of looking at the same set of points, just like converting from Cartesian to polar coordinates.

However, this new view makes it possible to see that two points (vectors) in the same direction from the origin might be considered similar even though they end at quite different distances. Two such points have the same *pattern* of attribute values but differ only in the *magnitudes* of these values, and in a proportional way. The records are alike in some deeper sense, but one has “more of the same” than the other. In some situations, this kind of similarity makes a great deal of sense. It is called *cosine similarity* because the cosine of the angle between the vectors to two points a and b is given by

$$\cos \theta = \frac{a \cdot b}{|a| |b|}$$

where the dot denotes the dot product of the two vectors, $a_1 \times b_1 + a_2 \times b_2 + \dots + a_m \times b_m$, and $|a|$ is the norm (length) of the vector a .

Equivalently, each row of the dataset can be divided by its length, as another form of normalization of the data. This has the effect of mapping the data points into a kind of hypersphere around the origin. The angle between two vectors is now just their dot product (since the norms are all now 1). Two vectors that point in roughly the same direction have a large dot product (close to +1) so the cosine of the angle between them is large and θ is close to 0° . Two vectors that are close to orthogonal have a dot product close to zero and θ is close to 90° . Two vectors that point in roughly opposite directions have a dot product close to -1 and θ is close to 180° .

So cosine similarity gives a different view of similarity, and so of clustering, based on projecting the data points onto a unit hypersphere. The sphere is still high-dimensional, so the problems of working in high-dimensional space have not gone away, but this form of similarity might be more appropriate for some data.

We are interested in datasets in which m is large, but the value of n , the number of records, is also relevant. There are three cases:

- n is about the same size as m so the dataset matrix is roughly square.
- n is much bigger than m . This tends to be the common case, for in many applications it is always possible to collect more data, perhaps just by waiting a little longer. Large values of n make it difficult to compute all pairwise distances, so better

algorithms are required to elicit the relationships among points. For example, if the distance from a to b is large and c is close to b , then it follows that a is far from c and this distance does not have to be explicitly calculated.

- n is smaller than m . In this case, the data cannot occupy more than an n -dimensional subspace of the m -dimensional space; in other words, it must be the case that the data occupies a lower-dimensional manifold in the high-dimensional space. This is not directly helpful because it is not, in general, possible to know how the lower-dimensional space is oriented inside the high-dimensional one, but it does help with some of the structure-discovery algorithms we will describe later.

It is common in the literature to claim that, in this case, a substantial number of the attributes can be discarded *a priori*. Although this is sometimes the case, more often than not it is impossible to discard attributes without deeper analysis.

2.5 Summary

The mapping from dataset records to points in the space spanned by the attributes is a natural and appealing one, but it contains hidden complexities. For local similarities to be sensible after such an embedding, choices need to be made about the relative scaling of the axes (the relationship between magnitude and significance for each attribute *and* the relative magnitudes between pairs of attributes). A similarity function needs to be defined—Euclidean distance seems natural in the geometric setting, but seems less so when its meaning for record similarity is considered. And high-dimensional spaces are unintuitive, with distances behaving in unexpected ways.

All of the choices made about embedding and similarity have a large impact on the resulting models—but there is usually no principled way to make many of them, at least until the domain is well understood. Such problems are ubiquitous in knowledge discovery; the usual solution is to iterate the embedding/model-building cycle; and that is usually what is required here.



<http://www.springer.com/978-3-642-33397-2>

Understanding High-Dimensional Spaces

Skillicorn, D.B.

2012, IX, 108 p. 29 illus., Softcover

ISBN: 978-3-642-33397-2