Making Better Features

36-350: Data Mining

16 September 2009

READING: Sections 2.4, 3.4 and 3.5 in the textbook.

Contents

Clustering is a way of finding structure in the data: it tries to group datapoints together by looking for similarities between feature vectors. However, clustering methods like k-means and agglomeration take the features themselves as given, and don't look for relationships between the features. When we used information theory to *select* features, we did look at relationships between variables (in the form of redundancy etc.), but just to pick out certain features from the set we started with. The next several lectures are going to be concerned with transforming those initial features into new ones. We do this partly to find and exploit structure in the data, namely relationships between the features, and partly because we hope that the new features will be more amenable to analysis than the ones we started with.

First, we'll look at some routine transformations of individual features to give them nicer descriptive statistics. Then we'll begin to examine how relationships between multiple features can be used to construct new features. The latter topic will occupy us for several lectures, and ultimately lead us through dimensionality reduction to new clustering methods.

1 Standardizing and Transforming

We want our results to be **invariant** to the units used to represent the features (pounds versus ounces versus kilograms, etc.). In other words, we want invariance to scaling. Similarly, we want to be invariant to any simple transformation of a feature, like miles per gallon vs. gallons per mile. This is done by **standardizing** the attributes to have similar distributions.

Here are some common ways to standardize quantitative features:

Rank conversion Replace all values with their rank in the dataset. This is invariant to any monotonic transformation, including scaling.

Scale to equalize variance Subtract the mean from attribute, and divide by its standard deviation (making it have mean 0, variance 1). This is invariant to changes in units and shift in the origin, but not other transformations.

Whitening Scale and subtract attributes from each other to make the variances 1 and covariances 0. This is invariant to taking linear combinations of the attributes.

These standardizations are special cases of **transforming** the features, which we do because their unmodified distributions are in some manner unhelpful or displeasing. Such transformations ought to be **invertible** — there should be a one-to-one mapping from the old to the new values, so that no information is lost. The three standardization methods I've just mentioned all tend to make different attributes more **comparable**, because they give them similar ranges, means and variances. Many people also like their data to have **symmetric** distributions, and transformations can increase the symmetry. In particular, data with highly skewed distributions over broad ranges often look nicer when we take their logarithms (Figure 1). However, the common emphasis on transforming to normality is rather misplaced. If you are going to plug your results into something like linear regression, it's generally much more important to have an approximately-linear relationship between your variables than for them to have nice, symmetric bell-curve distributions[1]. The old emphasis on normality had more justification back when all calculations had to be done by hand (or hand-cranked adding machine), and you could simplify many of those calculations by assuming that everything was Gaussian — so if it wasn't Gaussian, you made it so. This concern is an outmoded relic of a barbarous age.[2]

There are a number of algorithms which search for the "optimal" transformation of the data, according to various criteria (especially the "looks Gaussian" criterion). It's not at all clear that these are useful in practice, so we'll skip them.

2 Relationships Among Features and Low-Dimensional Summaries

Suppose we have two features which are perfectly co-linear (Figure 2*a*). Then one of them is redundant, both in the ordinary sense and in the technical,

[1] For more on this, and much else, see Berk (2004).

[2] An outstanding example of how it can lead to mischief comes from IQ testing. The raw scores people get on tests have asymmetric and generally quite skewed distributions. These are then transformed to scores which by *definition* have a mean of 100 and a standard deviation of 15. As a result, one of the most important facts about IQ tests, which is that average performance has been rising steadily all over the world, as far back as records go, was obscured for decades. See Flynn (2007).

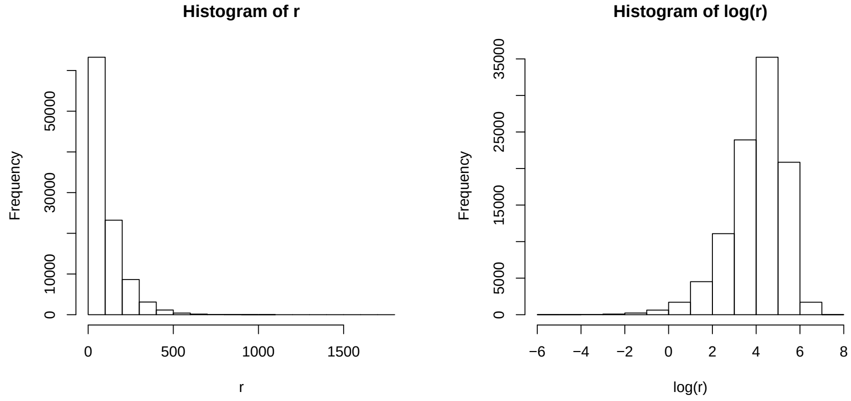

Figure 1: Taking the logarithm of the data values can make skewed distributions (left) more symmetric (right), as well as narrowing the range (note scales on horizontal axes).

information-theoretic sense: we could get rid of it without losing anything. Geometrically, instead of having data scattered over a two-dimensional plane, we have data falling on a one-dimensional line. If they are not perfectly co-linear, but almost (Figure 2*b*), then one of the features is nearly redundant, and the data cluster around a line in the two-dimensional space.

Now suppose we have p-dimensional data, but it's nearly co-linear. We won't, in general, be able to pick out one variable to just eliminate the way we could before, but it will still be the case that the data will cluster around a $p-1$ dimensional surface in the p-dimensional space. Instead of just dropping one of the features, we could set up a new coordinate system on this surface, and use it to describe the data. (This works in the 2D case as well; the new coordinate is "distance along the line".) And of course this process could be repeated — points in the $p-1$-dimensional sub-space might cluster around a $p-2$-dimensional sub-sub-space, etc.

The idea here is to exploit relationships among the features to eliminate some of them — to get low-dimensional summaries of high-dimensional data by eliminating dependencies. Nothing *conceptually* requires these relationships to be linear. Data which cluster around a circle are just as effectively one-dimensional as data which cluster around a line (Figure 3. Techniques which look for low-dimensional structure in high-dimensional data perform **dimensionality reduction** through **manifold learning**. The math we need to describe non-linear relationships, like the circle, does however build out from the math for linear relationships, so we will start with the latter, which are based on projections.

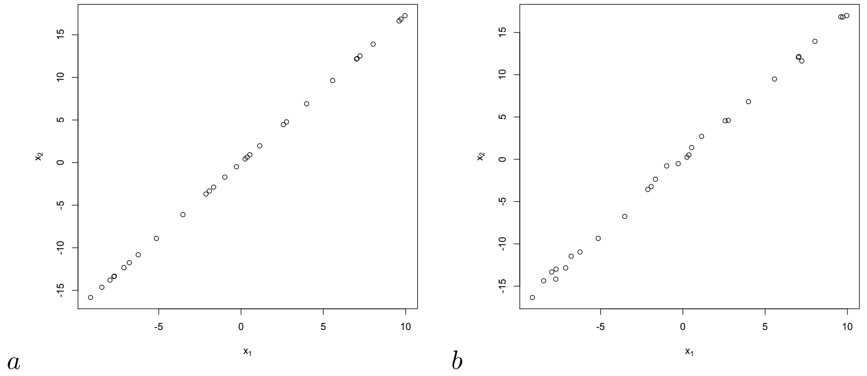

Figure 2: *a* (left): Perfect collinearity between two features. *b* (right): Approximate collinearity.

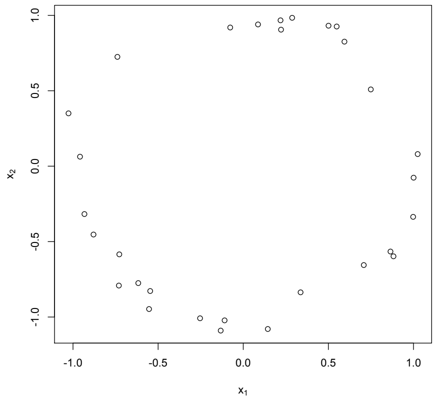

Figure 3: Data clustered around a ring. There is a one-dimensional structure latent in the two-dimensional data.

3 Projections: Linear Dimensionality Reduction

In geometry, a **projection** is a transformation taking points in a high-dimensional space into corresponding points in a low-dimensional space. The most familiar examples are maps and perspective drawings, both of which project three-dimensional points onto two-dimensional surfaces. Maps use **nonlinear projections**, which introduce warping and discontinuities. Perspective is an example of a **linear projection**, which doesn't warp but isn't invertible.[3] Linear projections are a lot more common, and usually "projection" by itself means "linear projection", so from now on we'll drop the "linear", too.

Algebraically, a projection is defined by a **projection operator**, say **P**, a square matrix with the properties that

$$\mathbf{P}\mathbf{P} = \mathbf{1}$$

and

$$(\mathbf{1} - \mathbf{P})\mathbf{P} = 0$$

For any vector $\vec{x}$, $\vec{x}\mathbf{P}$ is a new vector which is a linear function of $\vec{x}$. Geometrically, what happens is that each $\mathbf{P}$ corresponds to a linear sub-space, and $\vec{x}\mathbf{P}$ is the closest approximation to $\vec{x}$ within that subspace. If $\vec{x}$ happens to be in the subspace, then $\vec{x}\mathbf{P} = \vec{x}$, since the closest approximation is the vector itself. This is why we require $\mathbf{P}^2 = \mathbf{1}$. On the other hand, we can write any vector as

$$\vec{x} = \vec{x}\mathbf{P} + \vec{x}(\mathbf{1} - \mathbf{P})$$

so we require the residual, $\vec{x}(\mathbf{1} - \mathbf{P})$, to be orthogonal to the subspace, i.e., to have zero projection on to it.

The projection operator gives us a new vector in the original feature space, but one which is confined to the subspace. (It forces data points to be exactly on the line, and gives us the coordinates of those points.) What we really want is to go from p dimensions to $q < p$ dimensions, so we want a different matrix, say $\mathbf{w}$, such that

$$\vec{y} = \vec{x}\mathbf{w}$$

where $\mathbf{w}$ is a $p \times q$ matrix, and $\vec{y}$ is a q-dimensional vector, giving coordinates in the q-dimensional subspace.[4] ($\vec{y}$ just says how far along the line we are.) $\mathbf{w}$ is called the matrix of **weights** or **loadings**. Picking $\mathbf{w}$ is equivalent to picking $\mathbf{P}$, but the former is all we really need for dimensionality reduction, so we'll focus on it.

If we have n points we want to project, then we make each point its own row, and get an $n \times p$ matrix $\mathbf{X}$, and its **image** is

$$\mathbf{Y} = \mathbf{X}\mathbf{w}$$

[3] This isn't just because map-makers haven't been clever enough. If there *were* a continuous, invertible transformation between the map and the globe, that would mean that they really had the same dimension, and so that $2 = 3$. To follow up this thought, read McCleary (2006).

[4] All this is assuming that $\vec{x}$ is being written as a $1 \times p$ matrix, i.e., a row vector. If we wanted it to be a column vector, we'd just transpose everything, so the weight matrix would go on the left.

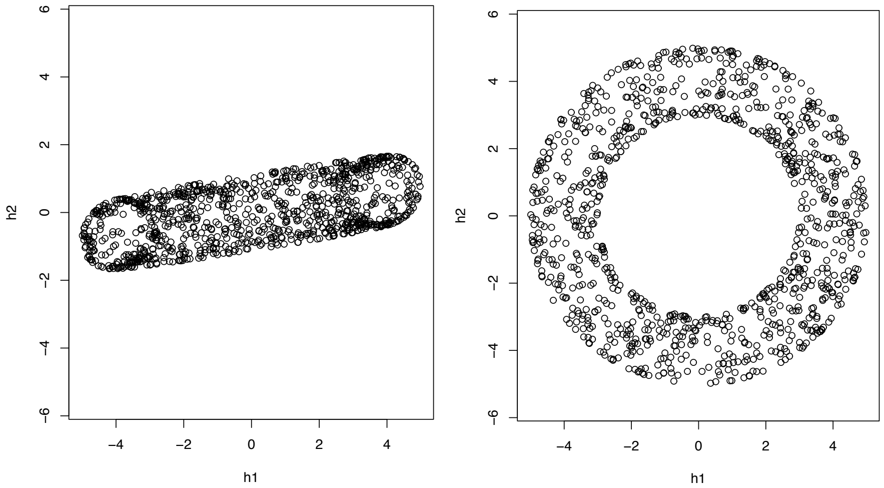

Figure 4: Two different 2D projections of a 3D torus. Left: projecting on to a plane nearly perpendicular to the equator; only faint traces of the structure are visible. Right: projecting on to a plane nearly parallel to the equator.

where **w** is the same as before.[5]

The utility of a projection depends on the fit between the data and the subspace we're projecting into — between **X** and **w**. Figure 4, for instance, shows two different projections of a three-dimensional torus onto two-dimensional planes. In one (where the projection is on to a plane nearly perpendicular to the equator of the torus), the structure is almost completely obscured, and all we see is a tube of points.[6] In the other, where the projection is on to a plane nearly parallel to the equator, where we at least see that it's a ring. One projection shows us that some combinations of features never happen; the other obscures this.

We want to take our large collection of high-dimensional, inter-dependent features and trade them in for a smaller set of more-nearly-independent features. Doing this by projection means picking **w**. Doing this well means picking a **w** which respects the structure of **X**. Which is what we are trying to learn. There are various ways of escaping this circle by making **w** a function of **X**. Generally, these involve trying to find the best projection according to various criteria. Of these optimal-projection methods, one is by far the most common, robust and important. This is **principal components analysis**, where the criterion is to try to preserve the variance of the original vectors.

[5] This is why we wrote $\vec{x}$ as a row vector.

[6] There are two circles at either end of the tube where the points are less dense than elsewhere, which reflects the fact that the torus was hollow.

References

Berk, Richard A. (2004). *Regression Analysis: A Constructive Critique*. Thousand Oaks, California: Sage.

Flynn, James R. (2007). *What Is Intelligence? Beyond the Flynn Effect*. Cambridge, England: Cambridge University Press.

McCleary, John (2006). *A First Course in Topology: Continuity and Dimension*, vol. 31 of *Student Mathematical Library*. Providence, Rhode Island: American Mathematical Society.