Making Better Features: Principal Components Analysis and Other Data Transformations

36-350: Data Mining

22 September 2008

Reading: Sections 2.4, 3.4, 3.5 and 3.6 in the textbook, *especially* Section 3.6 on principle component analysis.

Clustering, as we've seen, tries to group data points together based on the associations between feature vectors. However, it takes the choice of features as given. We've already seen some ideas for checking the value of features, mostly in connection with classification. This lecture is going to explain some of the main ways we can transform existing features into new ones, which may be more valuable.

1 This Week's Models

This week's dataset is 93 cars from the 1993 model year, each with 11 features:

Type	Small, Sporty, Compact, Midsize, Large, or Van
Price	Midrange Price (in $1,000)
MPG.highway	Highway miles per gallon by EPA rating
EngineSize	Engine size (liters)
Passengers	Passenger capacity (persons)
Length	Length (inches)
Wheelbase	Wheelbase (inches)
Width	Width (inches)
Weight	Weight (pounds)

All of the features except `Type` are numerical. Table 1 shows the first few lines from the data set.

We want to extract patterns from this data, like combinations of features which tend to appear together (clusters), and combinations of features which tend to *not* appear together (voids and anomalies).

2 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

```
               Type Price MPG.highway EngineSize Horsepower
Acura Integra Small  15.9          31        1.8        140
Dodge Colt    Small   9.2          33        1.5         92
Dodge Shadow  Small  11.3          29        2.2         93
Eagle Summit  Small  12.2          33        1.5         92
Ford Festiva  Small   7.4          33        1.3         63
              Passengers Length Wheelbase Width Turn.circle Weight
Acura Integra          5    177       102    68          37   2705
Dodge Colt             5    174        98    66          32   2270
Dodge Shadow           5    172        97    67          38   2670
Eagle Summit           5    174        98    66          36   2295
Ford Festiva           4    141        90    63          33   1845
```

Table 1: The first five lines of the `cars` data set.

invariance to scaling. Similarly, we want to be invariant to any simple transformation of an 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. 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

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

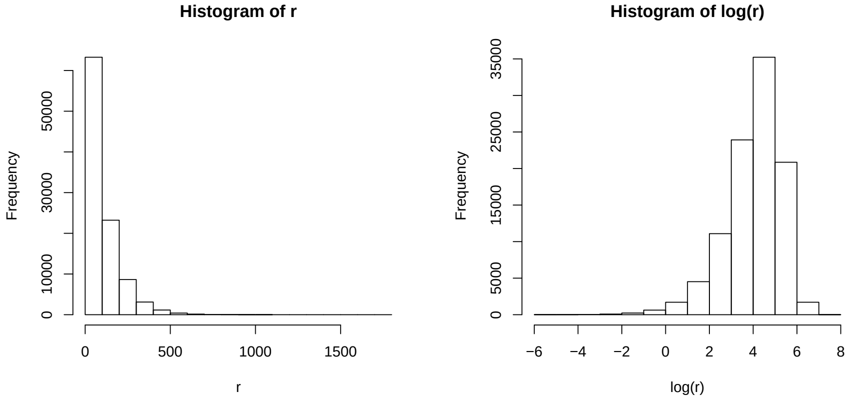

Figure 1: Taking the logarithm of the data values can make skewed distributions (left) more symmetric (right), as well as narrowing the range (note scales on horizontal axes).

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.

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

3 Projections: Low-Dimensional Summaries of High-Dimensional Data

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.[2] 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, projection is just matrix multiplication: if $\mathbf{x}$ is a p-dimensional vector (i.e. a $1 \times p$ matrix), a linear projection down into a k-dimensional vector

[2] 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).

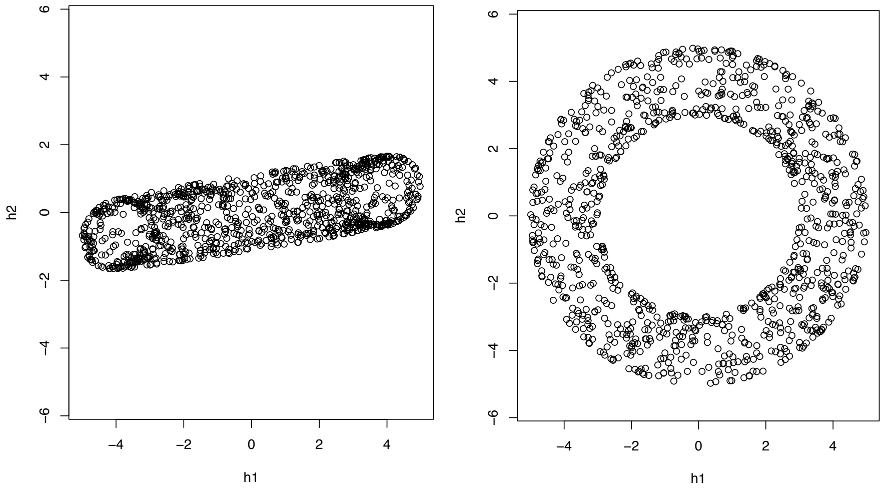

Figure 2: Two different 2D projections of a 3D torus. Left: projecting on to a plane nearly perpendicular to the equator; only faint traces of the structure are visible. Right: projecting on to a plane nearly parallel to the equator.

h is

$$\mathbf{h} = \mathbf{x}\mathbf{w}$$

where **w** is a $p \times k$ matrix of **weights** or **loadings**. Each component of h is a linear combination of components of x. 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

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

where **w** is the same as before.

The utility of a projection is going to depend on choosing the weights **w**. Figure 2, for instance, are 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.[3] 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.

If our data is high-dimensional (we have a lot of features), it would be very helpful to have a low-dimensional summary, especially if it preserves, or takes advantage of, any structure in the data set, or dependencies among the features. In effect, we are trying to trade in our old features for a new, smaller set which is built out of the old ones. There are lots of techniques which try to produce these

[3] 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.

low-dimensional summaries in an automated way. Our torus is really a two-dimensional surface, even though it's in a three-dimensional space: **manifold learning** methods try to automatically search for such structures. **Projection pursuit** (see chapter 11 of the textbook) encompasses a lot of specific techniques for searching for projections which have good properties. Some of them require us to pick k, the dimension of the projection, while others attempt to pick the best k. One particular method for finding the best projection, however, is by far the most common, robust and important.

4 Principal Components Analysis

Principal Component Analysis (PCA) is a method for maximizing the variance of the projected data. The output of PCA is a set of p orthogonal vectors in the original p-dimensional feature space, the **principal components**. These are ordered, from first to last. The first principal component is the direction in the feature space along which the data has the most variance. The second principal component is the direction orthogonal to the first component with the most variance. The third principal component is the direction orthogonal to the first two components with the most variance; and so on. To summarize the data, we pick $k < p$ and project the feature vectors on to the first k principal components. (If $k = p$, then "projection" is just an invertible linear transformation, some sort of rotation of the data.)

There are a number of reasons why PCA is useful.

- The principal components make sense as weighted combinations of the original features. They take into account the correlations among those features.

- The projections on to the principal components are uncorrelated.

- For the j^{th} component, we know what fraction λ_j of the variance in the original data it captures. (We will see next time where these numbers come from.) Because the projections on to the components are uncorrelated, we know that the first k components capture a fraction $\sum_{j=1}^{k} \lambda_j$ of the variance.

- No other projection on to k dimensions captures more of the variance.

PCA is often applied when the data have been standardized so that each feature has mean 0 and variance 1, but that's not necessary.

PCA has been rediscovered many times in many fields, so it is also known as the Karhunen-Loève transformation, the Hotelling transformation, the method of empirical orthogonal functions, and singular value decomposition[4]. My description of PCA makes it sound like an optimization problem — search for the

[4] Strictly speaking, singular value decomposition is a matrix algebra trick which is used in the most common algorithm for PCA.

first principal component, then the second, etc. One reason it's been rediscovered so often is that we can actually do the optimization analytically, without any search. Next time, I'll define the sense in which PCA is optimal more exactly, and show how to solve the optimization problem directly, with fast linear algebra.

Figure 3 shows the projections of the car data points on to the first two principal components (out of 10 in all); this accounts for 83% of the total variance. (This was done after standardizing the variables.) The weight matrix is

	h_1	h_2
Price	0.29	0.43
MPG.highway	-0.30	-0.05
EngineSize	0.35	0.06
Horsepower	0.30	0.49
Passengers	0.21	-0.68
Length	0.33	-0.10
Wheelbase	0.33	-0.26
Width	0.34	-0.07
Turn.circle	0.32	-0.08
Weight	0.37	0.01

The figure also shows, as red arrows, the projection of the original features on to the first two principal components. Changing the value of a feature moves it along that direction. Notice that all of the arrows point to the right (i.e., all of the h_1 weights are positive) except for the `MPG.highway` feature, which points left (and has a negative h_1 weight). This reflects the fact that all the features are positively correlated with each other, *except* for MPG, which is negatively correlated with everything else. Basically, the first component tells us whether we're looking at a big, expensive, gas-hungry car, or a small, cheap, fuel-efficient car. Notice also that about half of the arrows point up (positive weights on h_2) and about half point down (negative weights on h_2); this is a pretty common pattern. Looking at the weightings, the second most important distinction, once we control for over-all size, is between price, horsepower and engine size, and number of passengers, fuel efficiency and length — i.e., is it a sports car or a mini-van?[5]

References

Berk, Richard A. (2004). *Regression Analysis: A Constructive Critique*, vol. 11 of *Advanced Quantitative Techniques in the Social Sciences*. Thousand Oaks, California: Sage.

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

[5] You can see in Figure 5 that the cars with the highest value of the second component are the Corvette and the RX-7, and those with the lowest are the Chevy Astro and the Eurovan.

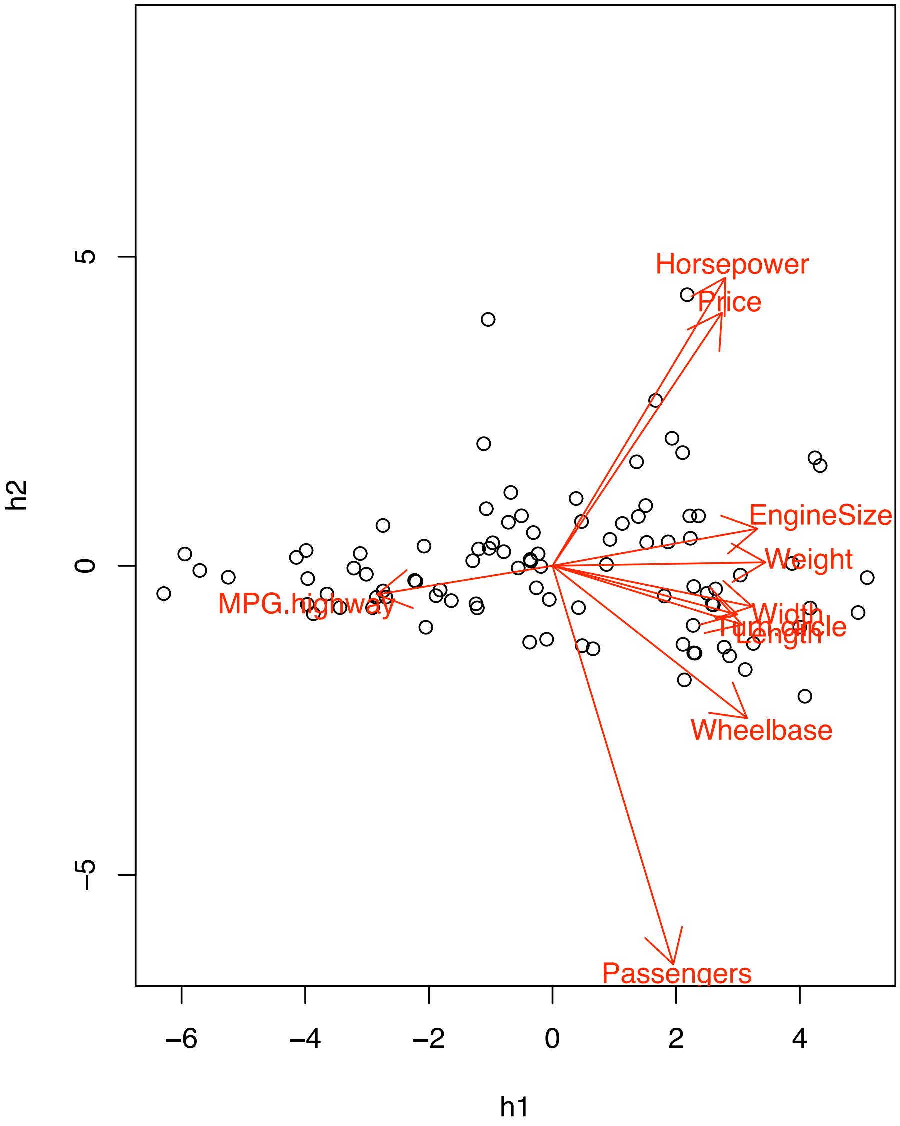

Figure 3: Principal components of the cars data-set, with the projection of the original variables onto the principal components. Notice the wider spread (greater variance) along the first principal component (h1) than the second (h2). Notice also how all of the original features have positive projections onto the first principal component (their vectors point to the right), *except* for MPG, which was negatively correlated with all the other features.

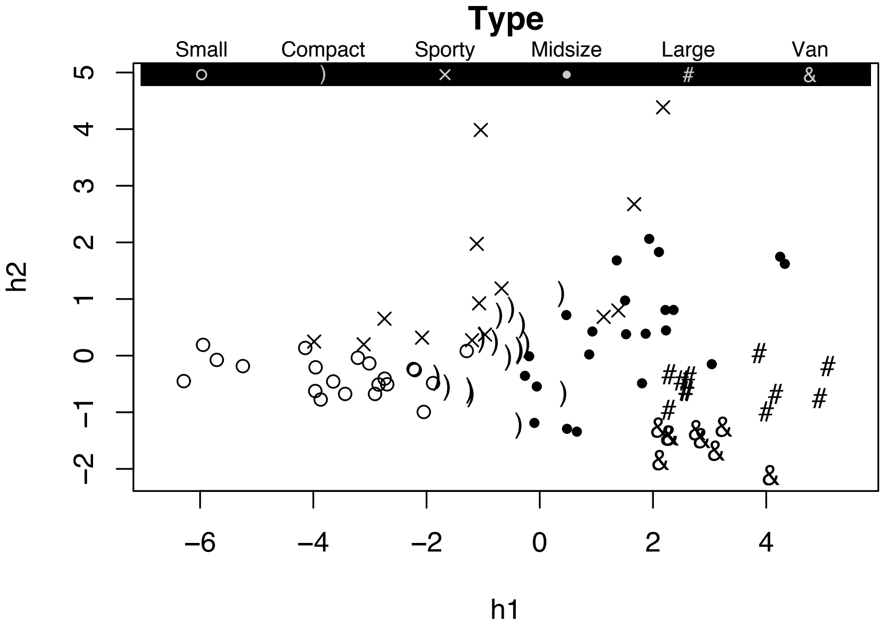

Figure 4: Projection of cars onto the first two principal components, and their classification (`Type`). Different regions in the plane correspond pretty cleanly to different types, even though that attribute was not used to find the principal components.

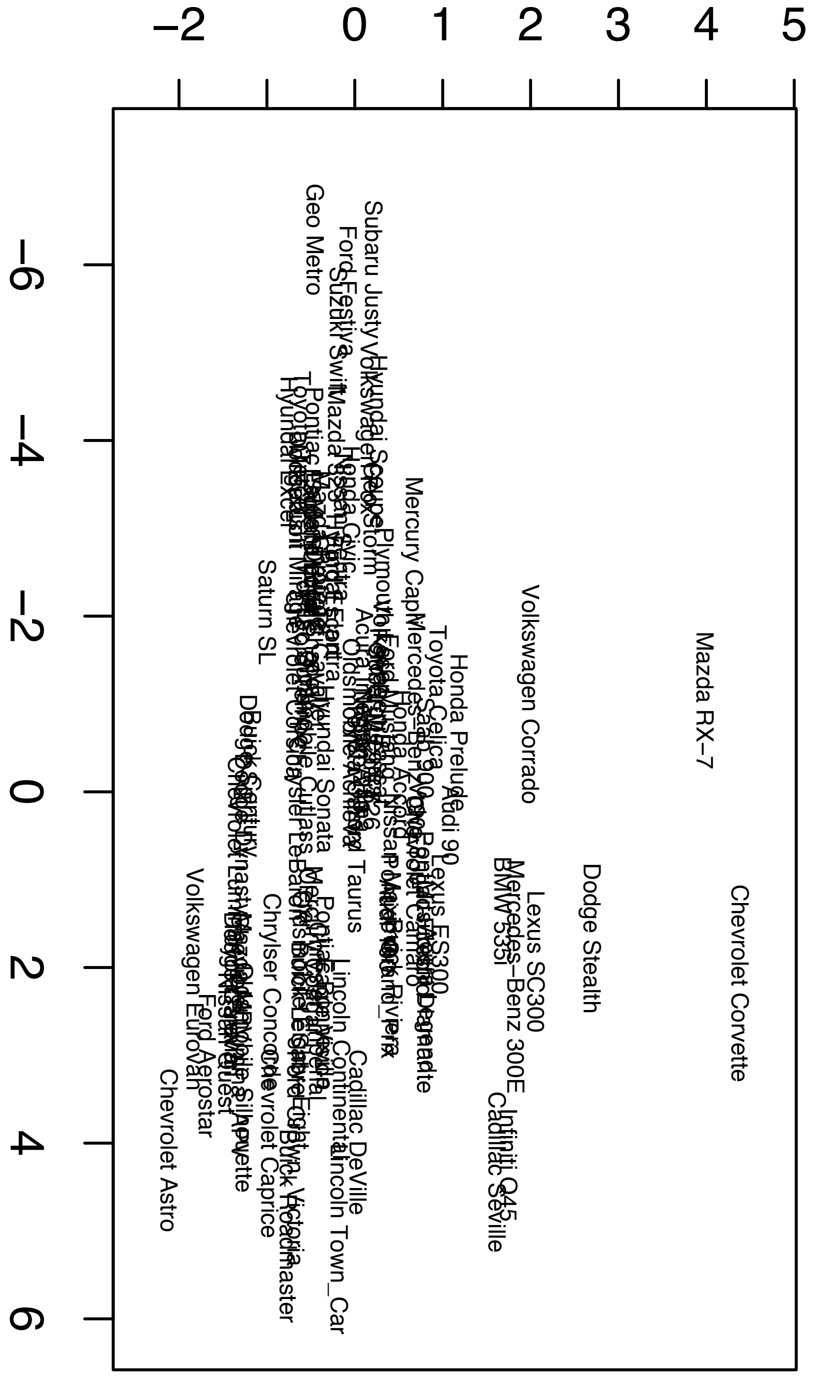

Figure 5: Cars and their PCA projections.