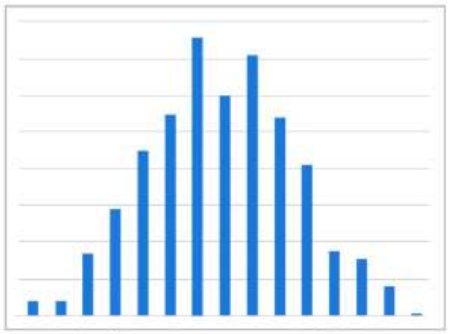
Advanced Data Prep

Thursday, September 26, 2024 10:46 AM

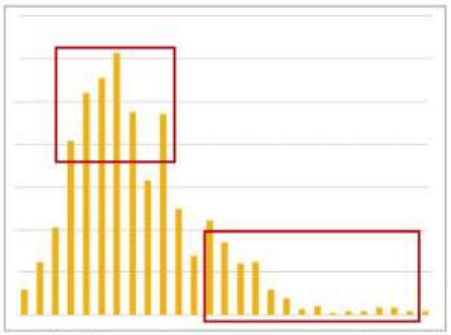
Box-Cox Transformations

Sometimes it is most effective to transform data before using it to train a model. Certain models assume that the data is normally distributed, which causes the results to have bias if that assumption is wrong.

For example, say we want to fit two models. One model is to predict the height of American women in their 40s (a data set that fits the normal distribution pretty closely) and the other model is to predict the weight of those same women (a data set that is somewhat normal for lower weight women, but not normal for higher weight women). The second data set has a big difference in variance, which is known as Heteroscedasticity. If we try to fit a regression model on this data set, we may end up with bias since the higher variance at the upper end of the data set can make the estimation errors larger which will push the model to fit those datapoints better than the rest of the data set.



Height of American women in their 40s



Weight of same American women in their 40s

One way to deal with heteroscedasticity is to use a Box-Cox transformation: a statistical technique that involves logarithmically transforming the target variable so the data follows a normal distribution. It does this by:

- Stretching out the smaller range to enlarge its variability
- Shrinking the larger range to reduce its variability

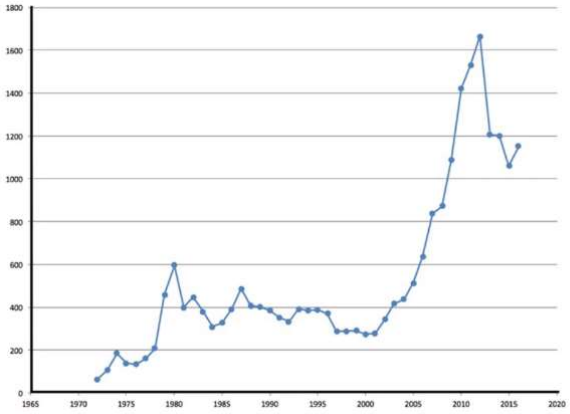
The goal of the Box-Cox transformation is to take the vector of responses *y* and find the best value of λ such that the transformed vector of responses *t(y)* approximates the normal distribution:

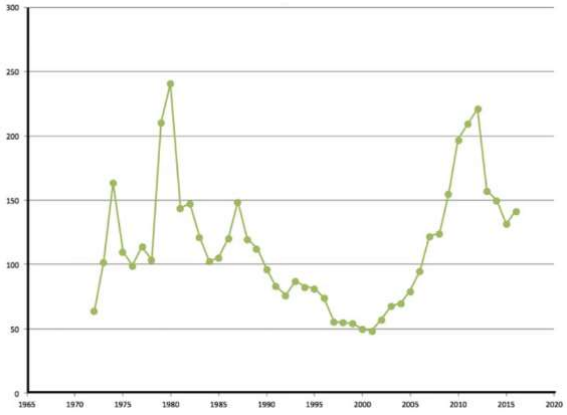
- $t(y) = \frac{y^{\lambda} - 1}{\lambda}$

Good statistical software can perform this transformation for you, but the first step is to always check to see the shape of the data first so you know if the transformation is even needed. One way to do this is to use a Q-Q Plot, which will tell you if a data set is normally distributed.

Detrending

Recall that with time series data, trend refers to data that is increasing or decreasing over time. Sometimes we want to remove a certain trend so that we can see how the data behaves in relation to other factors. For example, the price of gold over time appears to have a strong overall upward trend. If we adjust the data to remove the effect of inflation, the story changes quite a bit. If we keep the trend in, it may mess up a factor-based analysis. If we wanted to estimate the price of gold in a regression model as a function of other factors, keeping the inflation trend in would likely cause the predictions to be much higher than they should be.



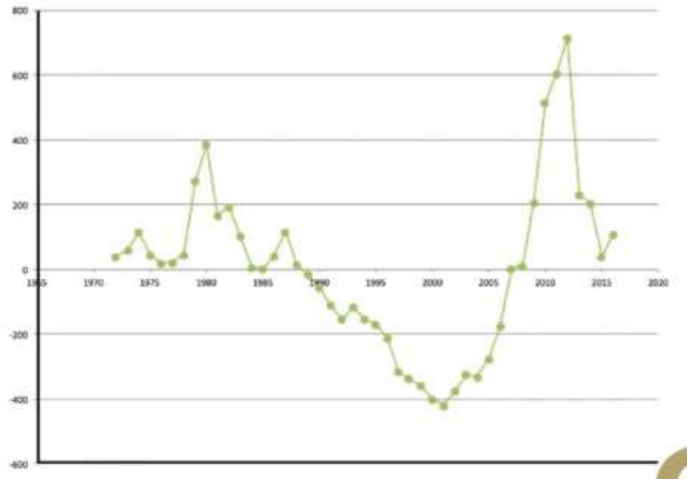


Detrending is the process of removing the effects of a trend from a data set to show only the differences in values from the trend. Detrending can be done on the Response variable and the Predictors and you should always consider detrending if you are building a factor-based model like regression or SVM.

One way to detrend is factor-by-factor; to fit a one-dimensional regression to it. For example, to detrend the gold data with this method, we would subtract the one dimensional regression for each data point from the actual price to get the de-trended price. This method isn't perfect since it assumes a constant rate of inflation over the years, but it approximates well enough.

$$price = -45600 + 23.2 * year$$

$$detrended\ price = inflated\ price\ - (-45600 + 23.2 * year)$$



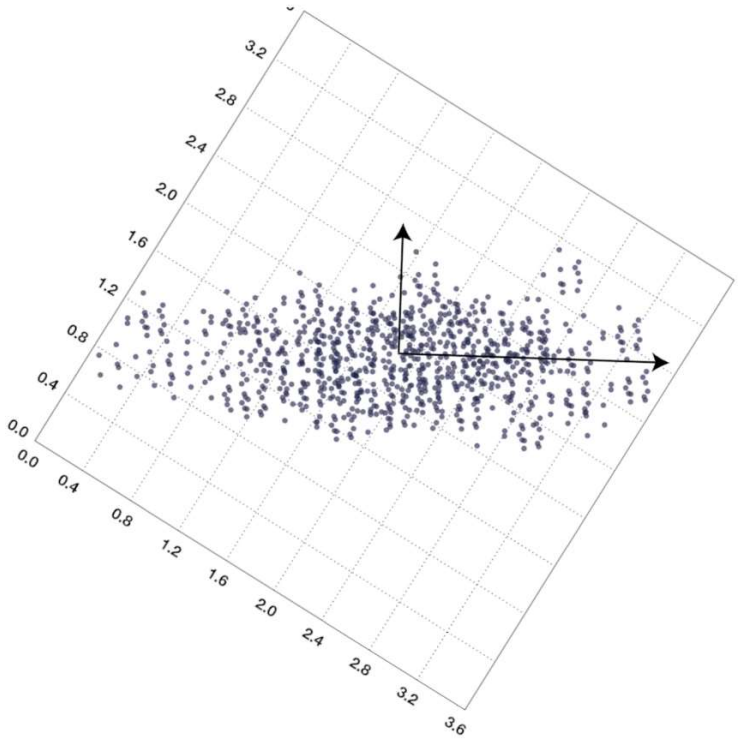
Intro to Principal Component Analysis

When dealing with a model that has a lot of factors, we will want to know which of them are the most important for predicting the response. Some reasons for doing this include:

- Working with massively huge datasets is computationally expensive and time consuming
- If years' worth of data is necessary, it may not capture how the situation that created that data has changed over time
- There may be very high correlation between some of the predictors, so you're capturing the same basic information within multiple factors; it's redundant

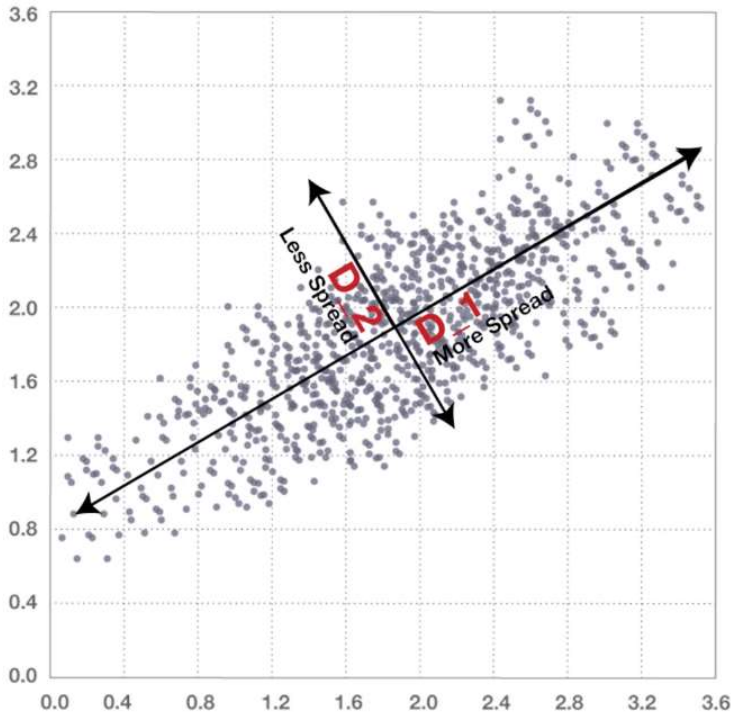
These last two issues can be addressed with Principal Component Analysis (PCA): a dimensionality reduction and machine learning method used to simplify a large data set into a smaller data set while maintaining significant patterns and trends. PCA transforms the data to remove correlations within the data and rank coordinates by importance. The more important principal components are likely to have a higher signal-to-noise ratio, meaning that their variance is more likely caused by actual effects rather than random effects. So if we use only the first *n* principal components, it will reduce the effect of randomness.

For example, the two dimensional data set below clearly has high correlation. If we change the coordinate system, there is no longer correlation between the two factors.



These new dimensions, labeled D1 and D2, are orthogonal to each other and do not correlate but we can see that D1 has more spread than D1. PCA recognizes this

spread and makes the dimension with bigger spread D1. Thus if we wanted to use just a one dimensional factor, D1 is a better choice than D2.



Using Principal Component Analysis

Here is the math behind PCA.

Say that:

- X: $initial\ matrix\ of\ data$
- x_{ij} = $jth\ factor\ of\ data\ point\ i$

The data is scaled so that for each factor *j*, the average value of x_{ij} is shifted to be zero: $\frac{1}{m}\sum_i x_{ij} = \mu_j = 0$

Next, we find all the eigenvectors of $X^T X$:

- V: matrix of eigenvectors, sorted by eigenvalue
- $V = [V_1\ V_2\ \dots]$ where V_j is the jth eigenvector of $X^T X$

Then the PCA-Linear transformation is as follows:

- XV_1: first component
- XV_2: second component

Etc.

And the k*th* new factor value for the i*th* data point is: $t_{ik} = \sum_{j=1}^{m} x_{ij} v_{jk}$

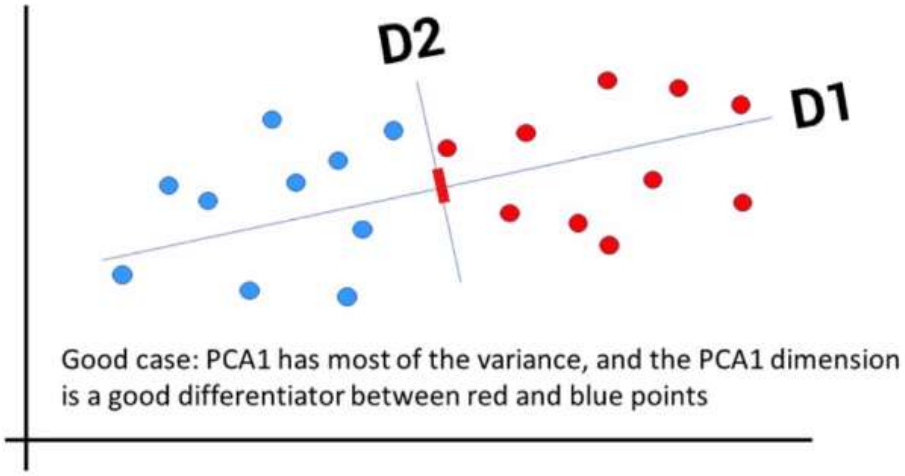
The math above assumes that we're looking for a linear transformation, but statistical software will be able to deal with nonlinear functions using Kernels.

Say we are using PCA with a regression model. PCA finds L new dimension factors $\{t_{ik}\}$ and regression finds coefficients $\{b_0 \dots b_L\}$ for those factors. We can interpret the new model in terms of the original factors because we can calculate the implied regression coefficient for the original factors x_j: $a_j = \sum_{k=1}^{L} b_k\, v_{jk}$

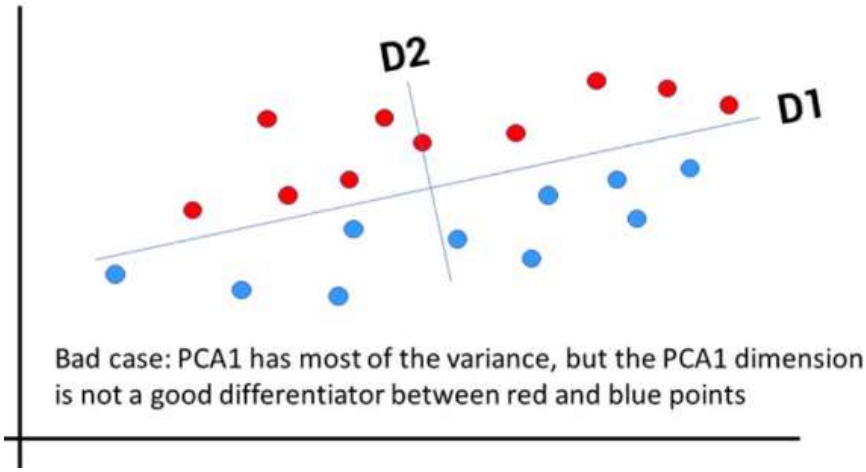
PCA Good and Bad

PCA transforms the data with a change of coordinates to create uncorrelated factors sorted by amount of variability explained by the factors. It allows for the use of a smaller number of variables, usually by picking the ones that explain the most variability.

PCA is really helpful in certain cases, such as when one of the principle components has most of the variance and that same component is also a good differentiator.



But PCA is less helpful in some cases, like when one of the principle components has most of the variance but it is not the best differentiator.



Dimensions that have variability are often good differentiators because they contain more information, but that isn't always the case. Often it's helpful to use PCA to reduce the dimensions without losing too much information.