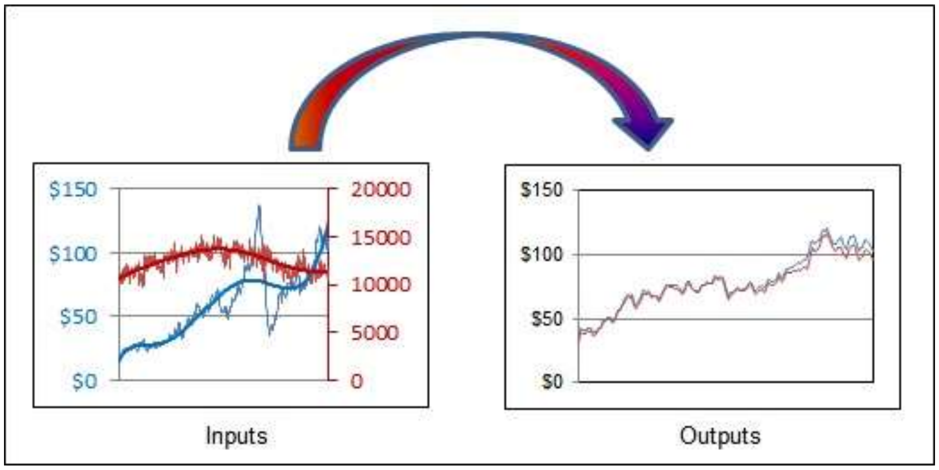
Data Conditioning: Handling Outliers

Behavioral models, which comprise a large class of computer models used to predict things like natural gas consumption, market prices, and the performance of the economy, are usually based on data. Their equations are estimated by statistical methods like linear regression or they "learn" from the data using non-parametric methods like artificial neural networks. A common criticism of data based models is that they can become unreliable once you get too far away from the data on which they are based. At best, confidence in their predictions goes down and novel situations raise justifiable questions. Fair enough. Here's a way to address part of the problem.

First, let's deal with some jargon for those of you who don't work with models very often. Inputs to the model are known as "exogenous variables" and outputs as "endogenous variables". For example, a model that predicts natural gas demand uses temperature, wind speed and direction, time of year, time of day, and other exogenous variables to predict gas use, the endogenous variable. But, there is another way of looking at the model. A model is really a map that takes you from the inputs to the outputs. The equations are the map and the data draw the map by means of statistics, learning, or whatever. The first figure depicts this important concept.

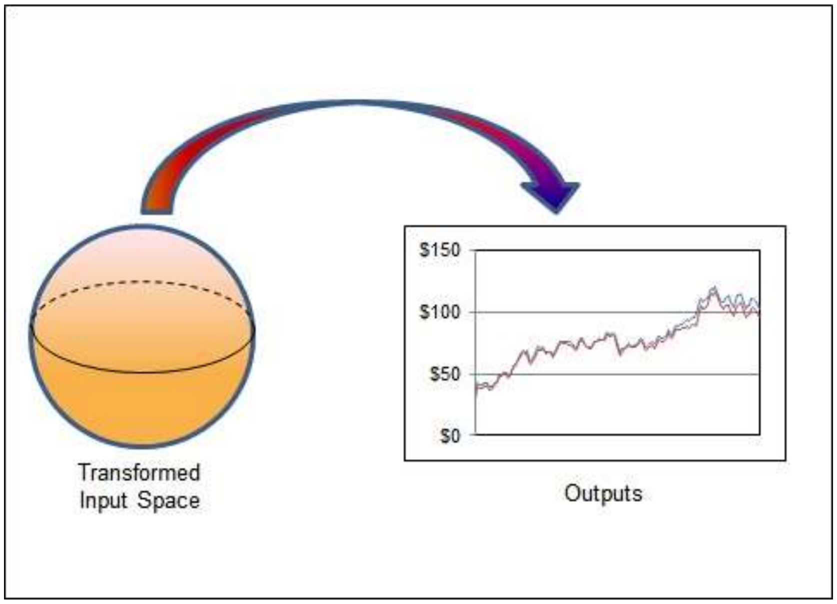


Drawing the map proceeds by the following steps:

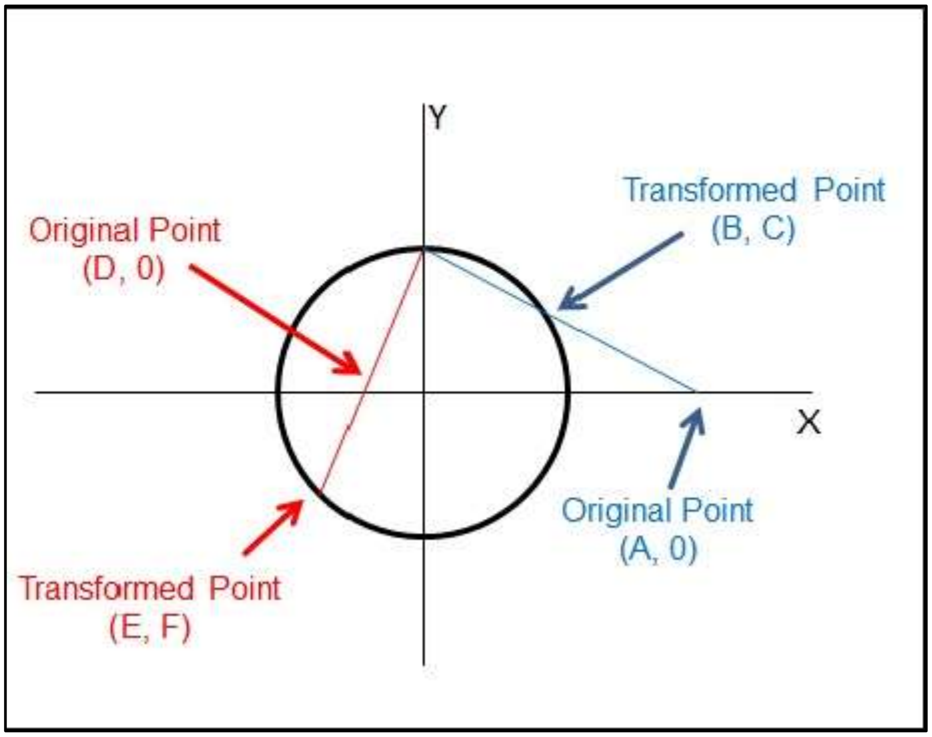
1. Gather the data.
2. Specify the form of the equations.
3. Estimate the equations (really, their parameter values).
4. Use the model to make predictions.

So what happens when the inputs you use to make the predictions (Step 4) differ markedly from the data you collected (Step 1) and used in Step 3? In other words, the inputs are some distance from the Step 1 data. Ah, you see the problem!

But suppose you introduce a new step, call it Step 1a, in which you transform the data so that no possible input can be too far away from the data used to estimate the equations in Step 3. The problem is sharply reduced, or maybe goes away entirely. We can do this by mapping our inputs onto the surface of a sphere. Because a sphere has a constant radius, no point on its surface can be more than one diameter (two radii) from any other. The second figure depicts the idea of how the mapping/model is changed by doing this.



But how do we accomplish this transformation? The final figure depicts the transformation concept for a very simple situation, one input. Our single input consists of points on the X axis (or "real line"). Next we add a second dimension, the Y axis, and draw a circle of some arbitrary radius (see below). To transform a point (data observation) on the X axis, draw a line from the north pole of the circle through the point. We now have a new point where the line intersects the circle. In the figure, we have two example points at coordinates (A, 0) and (D, 0) on the X axis. They transform into two new points at (B, C) and (E, F) which lie on the circle and therefore have the same length, one radius. Also, we started with one dimension and now we have two. We started with one input (on the X axis) and now we have two (X and Y) but both are uniquely dependent on our original input plus the radius.



How does this fit into the sphere concept (second figure)? By choosing the radius carefully, the transformed points will be more or less evenly scattered around the circle. If you use the transformed points as data to estimate your model – to build your map – you have a model that shows you how to go from the circle to the output, endogenous, variables. Because every set of inputs lies on the circle, no set of inputs will be too far from the data, you're never in completely novel territory. (Choosing the radius is a subject for another post but it is nearly never 1.0.)

Suppose there are many inputs, not just three needed to define a sphere? (Remember, a sphere is a three dimensional object. Or is it?) Not a problem. You can have a sphere in any number of dimensions, not just three. Called a "hyper-sphere", it is defined by the idea that all points on its surface are equidistant from the center. If you start with, say, three input variables, the transformation results in four inputs, or a point on a four dimensional hyper-sphere. The math – which we won't go into here – is quite simple and I usually do it in a spreadsheet for models using as many as 5,000 or so data points and tens of input variables.