An Application of Self Organizing Maps

Methods Paper

Lee Panter

The University of Colorado-Denver

Department of Mathematical and Statistical Sciences

Math 6330: Workshop in Statistical Consulting

Contents

1 Introduction

Self Organizing Maps (SOMs) are hybrid statistical learning algorithms first introduced by Dr. Kohonen Teuvo in 1982 in his paper “Self-organized formation of topologically correct feature maps” [1]. SOMs are best characterized as hybrid processes because they are useful for unsupervised learning in the absence of outcome information; and once this information has been made available, SOMs can be extended into classification algorithms. SOMs are used extensively for clustering, classification, pattern recognition, segmentation and visualization [2].

1.1 Background

Self Organizing Maps emulate biological information processing in the human brain, specifically with respect to compartmentalized processing. The human brain has dedicated neuronal pathways designed to process specific stimuli. Signals and inputs will be processed in completely separate areas of the brain based upon whether they have sensory, memory, or regulatory effects. The human brain pre-processes inputs, organizing them into similar clusters prior to evaluation.

Statistical learning algorithms are established as solid platforms for regression, prediction, and classification. Familiar approaches use bi-directional information flow, which consequently resembles “learning by example”. Algorithms are provided “examples” (data) and the algorithms are corrected until reasonable answers are provided. Although useful, these bi-directional flow algorithms have shown a tendency to be input-sensitive; and in theory, this type of learning is not replicating human intelligence. [3]

A closer approximation to human intelligence would be an algorithm that learned by observation. Learning by observation would require an algorithm to improve outcomes without back propagation of information from a response [3]. SOMs are feed-forward algorithms, that improves outcomes using a three-step approach: Competition, Cooperation, and Adaptation.

1.2 Objective and Outcome

The SOM algorithm is useful as an unsupervised method for clustering, and generating visualizations of high-dimensional data that can be used for inference since the dimension-reduction methods preserve the topological characteristics of the high dimensional relationships in the lower-dimensional representations.

In clustering applications Self Organizing Maps are useful for developing an understanding how the whole data set, represented as a large, highly variable population can be represented as a set of smaller, more homogeneous sub-populations. Visualizations can also show how the high dimensional representations of the data compare across each group.

The SOM algorithm is also useful for classifying high dimensional inputs into low-dimensional outcomes. The process of constructing clusters and high dimensional visualizations has "trained" a SOM map so that new, unseen observations are readily associated to the same outcome space.

The low-dimensional groupings that are the result of the SOM are further useful as the input layer of an Artificial Neural Network [4]. This approach is a useful data reduction and efficiency measure employed in training extensive deep learning algorithms. As an alternative representation of the original observations, a variety of techniques can be applied to these values.

Self Organizing Maps are useful for characterizing observational tendencies, and predictor relationships even when no outcome is present and/or when non-linear techniques are needed. The SOM algorithm can assist in visualizing relationships in high dimensional spaces, and is used as a data reduction method for inputs to Artificial Neural Networks.

SOMs are useful when an order structure can be associate with the predictor space. This situation is common in cases where predictor variables are measurements of the same “type” of variable. Some examples of these predictor spaces would include:

- 10 questions with integer-valued answers
- Measurements of height, width, and length
- Sports statistics with player percentage values

The SOM algorithm heuristically calls for 10% of the total observation count as a starting point for the number of lower-dimensional clusters used. This implies that SOMs require a minimum of around 20 total observations (two nodes for classification). It should be noted that Self Organizing Maps are dependent on initialization parameters. Results can be dependent on parameters provided to the algorithm and convergence issues.

2 Case Study Description

2.1 Data

An application of a Self Organizing Map will be applied to Edgar Anderson’s Iris Data. This data contains 150 observations of three species of Iris flowers. The three species of flowers are Iris Setosa (denoted “S”), Versicolor (denoted “Ve”), and Virginica (denoted “Vi”). There are 50 instances of each species type. There are four quantitative measurements in addition to species, corresponding to lengths on the flowers. A summary table of the measurements separated by species is listed in Table 1 below.

Sum of Values and *Product of Values* are calculated by performing the corresponding operation (sum or product) on all four measurements (Sepal Length, Sepal Width, Pedal Length, and Pedal Width) within each observation. The values of *Sum of Values* and *Product of Values* are plotted against all four measurements and against each other in Appendix A.

Species	Measurement	Min	Median	Mean	Max
S	Sepal Length	4.3	5	5.006	5.8
S	Sepal Width	2.3	3.4	3.428	4.4
S	Pedal Length	1	1.5	1.462	1.9
S	Pedal Width	0.1	0.2	0.246	0.6
S	Sum of Values	8.4	10.2	10.14	12
S	Product of Values	1.419	5.151	6.457	16.8
Species	Measurement	Min	Median	Mean	Max
Ve	Sepal Length	4.9	5.9	5.936	7
Ve	Sepal Width	2	2.8	2.77	3.4
Ve	Pedal Length	3	4.35	4.26	5.1
Ve	Pedal Width	1	1.3	1.326	1.8
Ve	Sum of Values	11.5	14.35	14.29	16.4
Ve	Product of Values	35	92.72	97.35	170.85
Species	Measurement	Min	Median	Mean	Max
Vi	Sepal Length	4.9	6.5	6.588	7.9
Vi	Sepal Width	2.2	3	2.974	3.8
Vi	Pedal Length	4.5	5.55	5.552	6.9
Vi	Pedal Width	1.4	2	2.026	2.5
Vi	Sum of Values	13.6	17.2	17.14	20.4
Vi	Product of Values	93.71	228.2	227.45	431.29

Table 1: Summary of Iris case Study Data.

The data be downloaded directly from the CRAN website by loading the `datasets` package and loading `data(iris)` into the R working environment.

2.2 Case Study Objectives

In an effort to outline the process and potential benefits of using Self Organizing Maps, a 4×4 SOM will be trained on a subset of the Iris data. This SOM will be used to obtain visualizations of the 16 clusters. Following this clustering, a hierarchy will be imposed on the 16 clusters, and as set of three clusters formed from the hierarchy will be ordered, and chosen to represent the three Species in the Iris data. The remaining observations (leftover from the original data after training) will then be mapped to these "Synthetic Species" and an accuracy estimate will be obtained.

3 Model and Methods

3.1 Topology Preserving Maps

Suppose that (X, d_X) and (Y, d_Y) are Metric Spaces and that the observation, $\mathbf{x}_i = (x_{i1}, \ldots, x_{iP})$ resides in the predictor space, where $i = 1, \ldots, N$ represents the number of observations.

A Self Organizing Map is a function that preserves the relationships between elements in a topological space [5].

$$\mathbf{F} : (X, d_X) \subseteq \mathbb{R}^P \longrightarrow (Y, d_y) \subseteq \mathbb{R}^Q$$

where $Q = 1, 2$, is the outcome space used for visualization.

Since continuous functions preserve topological relationships, it is sufficient to require:

$$\forall \epsilon > 0,\ \exists \delta > 0 :\ \mathbf{F}\left(B_\delta\left(a\right)\right) \subset B_\epsilon\left(\mathbf{F}(a)\right)$$

where

$$B_\delta\left(a\right) = \left\{x \in (X, d_x) : d_x\left(a, x\right) < \delta\right\}$$

and

$$B_\epsilon\left(\mathbf{F}(a)\right) = \left\{y \in (Y, d_y) : d_y\left(\mathbf{F}(a),\ y\right) < \epsilon\right\}$$

3.2 Algorithm

Let the outcome space, (Y, d_Y) be an $(M \times K)$ lattice structure with nodes enumerated sequentially, so that node $j \in \{1, \ldots, M \times K\}$

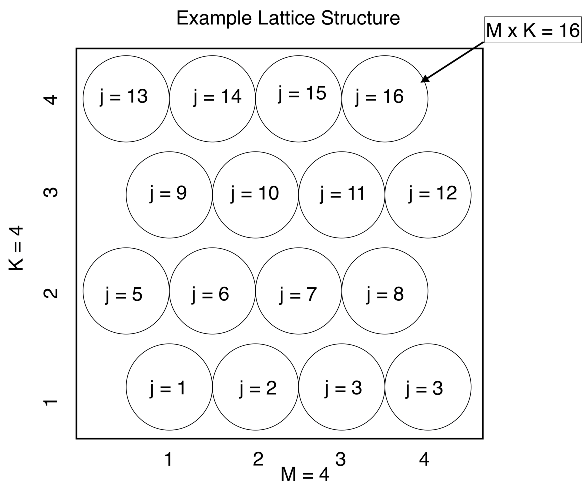

Figure 1: Counting two dimensional lattice structures $j \in \{1, \ldots, M \times K\}$

For each node there will be a weight vector, $\mathbf{W}_{ij}$ associated with each outcome:

$$\mathbf{W}_{ij} = (W_{ij1}, \ldots, W_{ijP})$$

for $i = 1, \ldots, N$ and $j = 1, \ldots, M \times K$, a total of $N \times M \times K$ weight vectors.

The weight vectors are adjusted in a three step process, and the process is repeated until sufficient weights do not move, or move very little with additional iteration.

In each iteration, a single observation is fed into the algorithm, then output lattice structure (nodes) and the weights associated with them:

1. **Compete** for selection as the "winner" that maximizes proximity within the *output lattice*, which then
2. **Cooperates** with the other output nodes to varying degrees according to a neighborhood function. Nodes closer to the winning node are adjusted more than those considered further away from it.
3. **Adapt** the weights of the output nodes, adjusting the values so that they become more similar to the winning value.

3.2.1 Competition

Given a single observation $\mathbf{x}_i = (x_{i1}, \ldots, x_{iP})$ where $i \in \{1, \ldots, N\}$ weight values are calculated for each value of $j = 1, \ldots, M \times K$

$$\mathbf{W}_{ij} = d_X(\mathbf{x}_i, \mathbf{W}_j)$$

and a value of $\mathbf{W}_i^*$ is defined based upon:

$$\mathbf{W}_i^* = \arg\min_j \; d_X(\mathbf{x}_i, \mathbf{W}_j)$$

this is called the “winning node”

3.2.2 Cooperation

A neighborhood function $h_{i^*}(j, \epsilon(t), \alpha(t))$ to determine how many (and which) node weights in the vicinity of i^* will be altered, where $t = 0, 1, 2$ represents the algorithm iteration number (discrete time).

$N_{\epsilon(t_0)}(i^*)$ is an open epsilon-ball around i^* at time $t = t_0$ is the set of nodes given by:

$$N_{\epsilon(t_0)}(i^*) = \{j \in 1, \ldots, M \times K \mid d_Y(i^*, j) < \epsilon(t)\}$$

$\epsilon(t)$ is a monotonically decreasing function of t that represents the largest cross-sectional width/radius of $N_\epsilon(i^*)$.

$\alpha(t)$ is a monotonically decreasing function of t that represents the learning rate. $\alpha(t) \in [0, 1] \; \forall t \in \mathbb{N}$

Two neighborhood functions are:

$$h_{i^*}(j, \epsilon(t), \alpha(t)) = \begin{cases} h_{i^*}(j, \epsilon(t), \alpha(t)) & \text{if} \quad j \in N_\epsilon(i^*) \\ 0 & \text{if} \quad j \notin N_\epsilon(i^*) \end{cases}$$

or they can be continuous, as in:

$$h_{i^*}(j, \epsilon(t), \alpha(t)) = exp\left(\frac{-d^2(i^*, j)}{2\epsilon^2(t)}\right)$$

$\forall j \in \{1, \ldots, M \times K\}$

3.2.3 Adaptation

Following the calculations of the winner $\mathbf{W}_i^*$ and weights $\mathbf{W}_{ij}$, the neighborhood calculations $h_{i^*}(j, \epsilon(t), \alpha(t))$ the set of weights are updated to reflect the information gained by minimizing the distance (the competition step), using the update formula:

$$\mathbf{W}_{ij}^{t_1} = \mathbf{w}_{ij}^{t_0} + \alpha(t_0) \times h_{i^*}(j, \epsilon(t_0), \alpha(t_0)) \left[\mathbf{x}(t_0) - \mathbf{W}_{ij}^{t_0}\right]$$

for each value of $j = 1, \ldots, M \times K$

where the discrete time step is defined as $t_1 = t_0 + 1$, and the algorithm is repeated over the discrete time variable until convergence is seen in the weights for each value observation, and the Competition, Cooperation, and Adaptation portions of the algorithm are repeated for each observation in the data until convergence is seen in the output layer. Re-circulations through the data set may also be needed.

3.3 Assumptions

One of the major benefits of implementing Self Organizing Maps is a general lack of assumptions. The SOM algorithm is dependent on being able to order observations within the predictor space, which means that there can only be limited unordered categorical values. Response values that cannot be associated with a magnitude or generalized hierarchy are not well suited to comparison, and alternative methods must be considered. Additionally, the SOM algorithm makes an implicit assumption that inputs are equally weighted prior to training. This assumption is critical to the related assumption that weight values are bounded, and restricts the types of data that the SOM algorithm can accommodate further to data that is standardizable.

Predictors and additional covariate information should be considered in close coordination with the definition of metrics chosen for the underlying mapping input and output spaces. Data with order can be accommodated, even if the the values are not quantitative, but careful metric definitions be evaluated.

The SOM algorithm is extremely flexible with respect to parametric assumptions. SOMs do not assume any properties of a distribution prior to mapping or clustering observations. This lack of a distributional assumption makes SOMs a particularly useful tool for data with missing observations. SOMs construct mappings based on empirical distributions that contain all informational attributes in the data (even those that are missing). In this way, a missing value is considered information itself.

---

4 Analysis and Results

A $4X4$ Self Organizing Map is fit to normalized Iris data, using a hexagonal topology arrangement of the output node layer. The SOM fitting process was restricted to 400 epochs (iterations through the entire data set). The SOM mapping is displayed below in Figure 2:

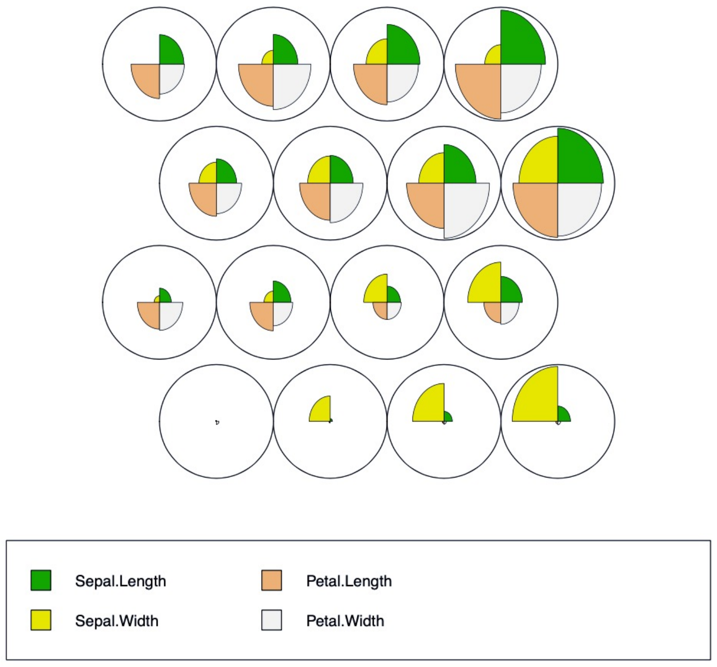

Figure 2: The self organizing map output layer node show the representation of the "winning weight value" inside each circle of the 4x4 hexagonal topology

Following the initial training of the SOM, a hierarchical structure is imposed over the 16 SOM output cluster. The hierarchy here is a representation of Euclidean distance as measured with respect to the characteristics of the winning node. A cut value of three clusters is chosen to represent the three Iris classifications types

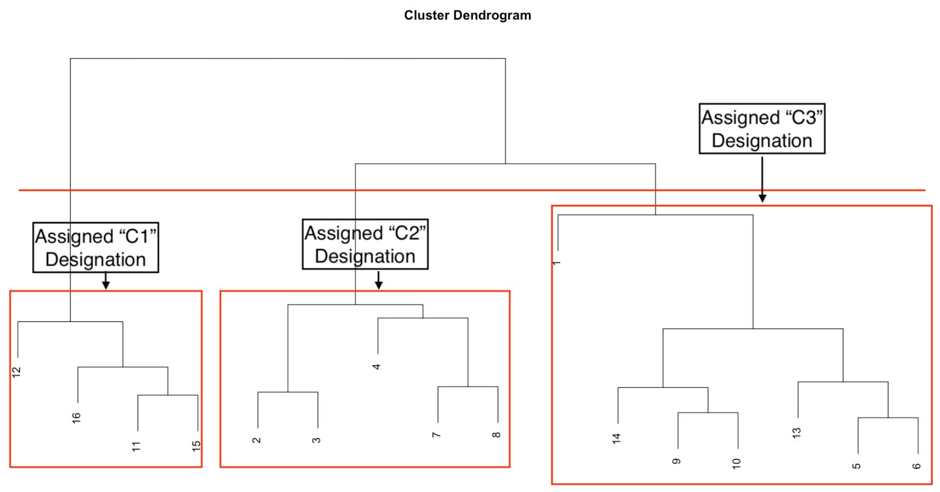
Cluster Dendrogram
Assigned “C1” Designation
Assigned “C2” Designation
Assigned “C3” Designation
12
16
11
15
2
3
4
7
8
1
14
9
10
13
5
6

Figure 3: The self organizing map output layer nodes are organized into a hierarchy using a Euclidean metric measurement between representative "winning weight values".

The goal is now to impose another hierarchy onto this grouping of clusters.

Cluster	Mean Sum All	Mean Product All
C1	3.600	1.320
C2	2.680	1.010
C2	0.039	0.030

Table 2: Mean sum of all variable and Mean product of all variables grouped by the clusters in Figure 3

Figure 4 below indicates that, although not perfectly separable, the Iris classes are distinguishable based on the fact that there seems to be a quadratic trend in the value of *Product of All Variables* when plotted against *Sum of All Variables.* Additionally, this trend can be approximately partitioned according to Species with

$$\text{S} \leq \text{Ve} \leq \text{Vi} \qquad \text{(EQ 1)}$$

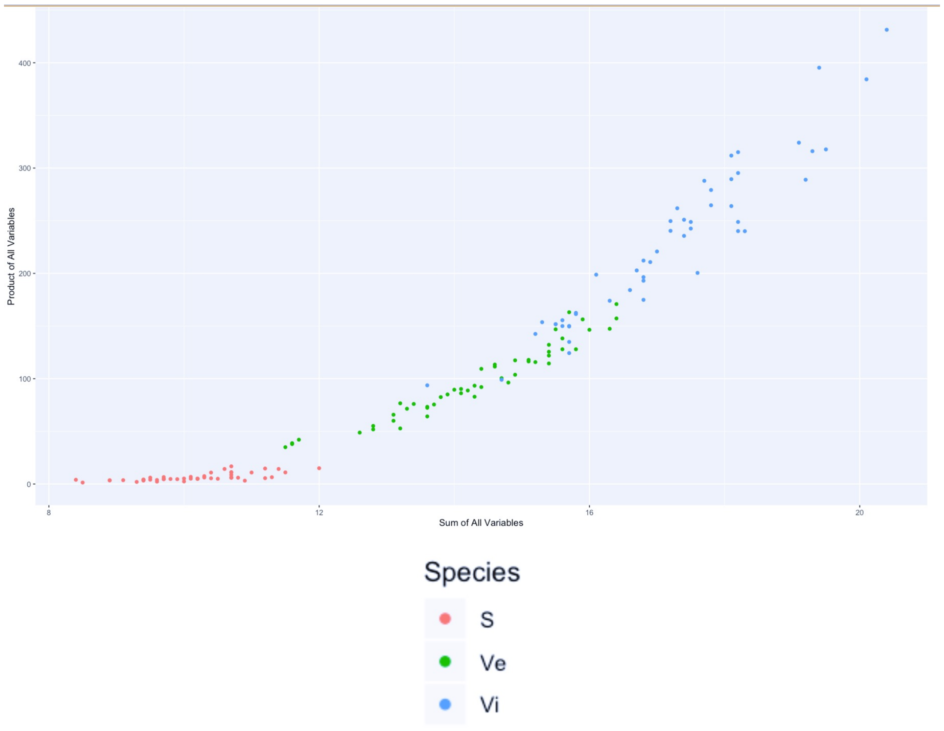

Figure 4: Plot of all four variables vs Product of All Variables

Equation (1) combined with Figure 4 motivates the assignments:

Cluster	Assigned Cluster
C1	Vi
C2	Ve
C2	S

Table 3: Assignments from meta variable summaries

5 Discussion

This analysis has classified Edgar Anderson's Iris data using a 4x4 Self Organizing Map. Observations were first separated into test and training data, with 75% of the observations being used in the training set. The 16 SOM outcomes were then clustered into three groups using a hierarchical criteria based on the Euclidean distance from representative nodes. An Iris species assignment was associated to each of the three hierarchical clusters by deducing an order present in the data that was also found in the unlabeled clusters formed from hierarchical clustering.

After assigning a Species label to the training data values, the cluster assignments are found to be 77.68% accurate in classify the correct training observation, and after evaluating the pre-trained SOM on the test data in a similar calculation (with only 38 observations to classify) a 98% classification accuracy is estimated.

5.1 Next Steps

A 98% test classification accuracy values is not reasonable when considering 78% estimated for the training set. This is all in addition to the fact that minimum classification accuracy should be considered 66% for any algorithm attempting to classify a three-level set of observations. Additional consideration needs to be paid to accuracy estimates conducted over differing training/testing sizes.

---

6 Appendix

6.1 Appendix A: Data Plots

Summary Plots of *Sum Of All Variables*

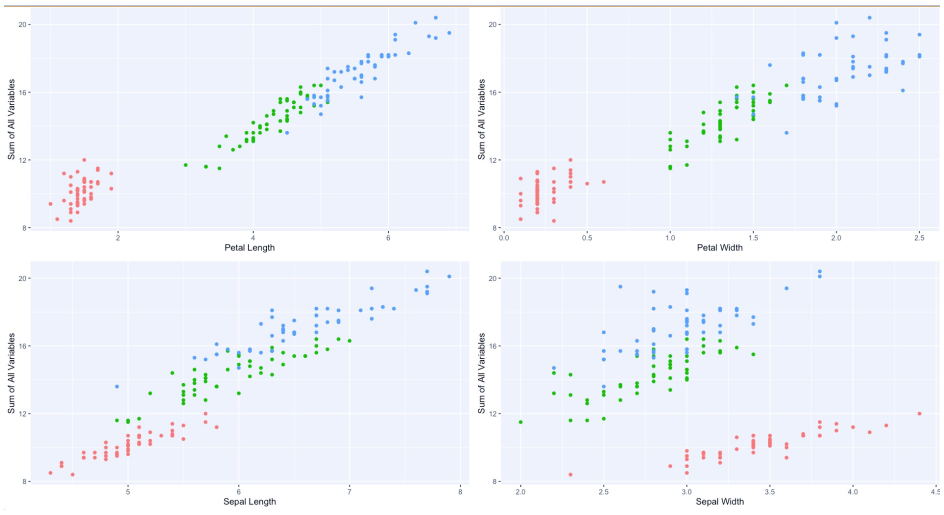

Figure 5: Plot of all four variables vs Sum of All Variables

Summary Plots of *Product Of All Variables*

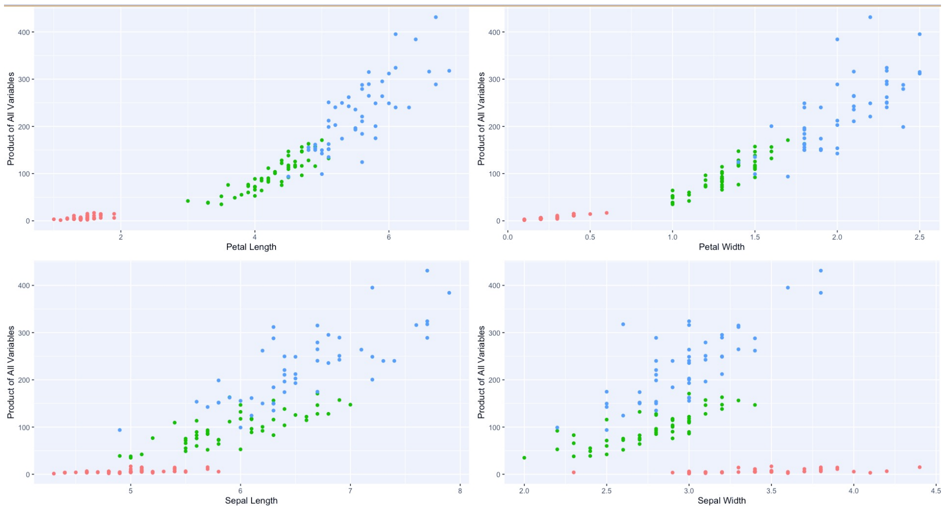

Figure 6: Plot of all four variables vs Product of All Variables

6.2 List of Figures and Tables

List of Figures

List of Tables

6.3 Data and Code Availability

Access to code, with instructions, and all data used for the analysis completed here is available for download at:

`https://github.com/leepanter/SelfOrganizingMaps.git`

or

`git@github.com:leepanter/SelfOrganizingMaps.git`

6.4 References

1. Teuvo K (1982) Self-organized formation of topologically correct feature maps. *Biological cybernetics* 43: 59–69.

2. Asan U, Ercan S (2012) An introduction to self-organizing maps, In, *Computational intelligence systems in industrial engineering*, Springer, 295–315.

3. (2020) *Coloradoedu*.

4. Larose DT, Larose CD (2014) Discovering knowledge in data: An introduction to data mining, John Wiley & Sons.

5. Belavkin R Lecture 13: Self-organising maps.