Analysis of the Baby Food market using Kohonnen Self Organizing Maps

Brian McFadden
James D. McMillian
Logan Flewellen

Department of Computer Science, Undergraduate
University of Pittsburgh – Pittsburgh, PA

27 April 2012

Introduction

A self-organizing map (SOM), also known as a Kohonen map, is a specific implementation of an artificial neural network which uses an unsupervised learning algorithm to produce a clustered graphing of instance data. The plotting is usually 2-dimensional for easy visualization, which is arguably the purpose of self-organizing maps. The cluster organization is determined algorithmically via features of the instance data. Although the data is often labeled, as it is usually useful to the user for them to be so, the algorithm is truly unsupervised. That is, there is no regiment of testing following training. The algorithm does not use the labeling in its placement of the instances into the graph; the process is entirely dependent upon feature values.

The alternate name of the model, Kohonen map, comes from Dr. Teuvo Kohonen, Professor Emeritus at the Academy of Finland, Helsinki, Finland. It is he who first conceived of using artificial neural networks for the purpose of topological sorting. His book which describes the algorithm in detail is titled Self-organizing Maps (3rd edition, 2001).

In order to test our implementation, we decided to select one of several possible applications and design the process around solving that problem. Our problem was organizing food based on nutritional values being the features. The nature of the problem seemed very complementary to the notion of self-organization based on features. In order to further evaluate how useful self-organization could be, we decided to make an application of that concept specifically. The idea that we used was locating nutritional "holes" in the baby-food market. This part of our study was not intended to be a study on baby food nutrition; it is strictly a demonstration of what can be found within the plotting. We were able to reach our goal, and it so happens that foods similar to cheeseburgers and pizza are underrepresented in the baby-food market.

The Interest in Self-Organizing Maps

Self-organizing maps are interesting because they are rather unlike many other machine learning systems that we experienced in the course. Additionally, everybody in the group thought a system with such visual appeal would be fun to demonstrate, with results that could be easily viewed and mostly understood by others. We were also curious as to the type of algorithm which would provide such an intuitive experience. Finally, we speculated that such a generalized system, naturally organizing data based on inherent, emergent patterns, could be widely applicable.

As stated, the algorithm is unsupervised. With an unsupervised learning scheme, new relationships in the data can be found. The trade-off is that the quality of the algorithm can be difficult to judge, though we were interested in this open-ended approach to machine learning despite that shortcoming.

Practical Application

Since the output of a SOM is both intuitive and informative, many pragmatic uses come to mind. We decided to sort food, with features being various nutritional values, in our specific implementation. Food is a topic with which everybody is familiar, which easily lends itself to the notion of features, and which continues to surprise the population when it yields previously-undiscovered information.

Expounding on the point of discovering information, there is a very strong presence of food awareness in society. Food permeates many sectors of life, from public health and academia to life and culture. Many people of many backgrounds are interested in the relationship between food and the body, and one of the ways to understand that relationship is by using the relationship between all of the food in one's diet.

Since self-organizing maps reveal hidden relationships of data in an intuitive way, we felt that sorting food, with its popularity, was a perfect match. A SOM using food could possibly be used to identify either abundances or paucities in nutritional groups for food consumed by a single person, food offered by a restaurant, or food consumed by nations on average. In those three examples, the topics of personal health, business, and public health are present. With many parties of different sophistications interested, the wide appeal of the visual presentation paired with the robust set of use cases of this machine learning system is not only sensible, but preferable.

To demonstrate the practical use of a SOM, we decided to single out one of the example usages and actually apply the system as described. As a case study, we chose to take on the analysis of a group of food, looking for gaps which could indicate possible exploitable areas of that particular food market. Baby food was the group of food selected for this, because a large proportion of the target population, namely, the babies, may live entirely on store bought baby food. Our study is an example application of a self-organizing map and does not take into consideration things like recommended baby nutrition. However, it does hint at the direction into which researchers could look if they are trying to satisfy the span of the nutritional spectrum of baby food, and we do speculate that similar approaches could be used in clinical research.

Previous Work

Because a SOM reveals emergent relationships within the data, it has been used for tasks requiring such revelations in the past. A search through academic papers showed that there were many ideas for purposing this algorithm. We were not actually able to find any papers detailing goals similar to ours, which is evidence for the algorithm being applicable to many situations.

One example was organizing documents for classification within a bibliographical database (Guerrero Bote, Moya Anegón, and Herrero Solana, 2000). Although we did not pursue classification with our experiments, it would have been possible if we considered item proximity to clusters of previously-identified items. Their goal was to organize documents in such a way that exact search terms would no longer be required to find documents in a database, because relatively close search terms

would locate the correct cluster within the topologically-sorted data set. Their experiment, while not very similar to ours, shows the versatility of the algorithm.

There have been studies more focused on data mining and emergent patterns, like ours, as well. One interesting study was predicting future wind patterns based on a history of previous patterns (Fayos, Fayos, 2006). Instead of looking for holes in the data like we were, their work was focused on making future predictions. Once again, this is not very similar to our goal, but it is another natural application of the algorithm.

Planning

The planning began shortly after we agreed that self-organizing maps were fun, interesting, and applicable. During research, it was discovered that a common advisory was not to try to implement the system on one's own. There are many pitfalls that many programmers often fail to avoid when implementing the algorithm. Likewise, Professor Hwa suggested using a pre-constructed algorithm, like one provided by Weka, and focusing on collecting data, tweaking the parameters, and feature selection.

After getting to know Weka and struggling to find a SOM package for the purpose of clustering, we decided to try another library. Encog is a machine learning package provided by Heaton Research on the Java and .NET platforms. We began to research how to use Encog. Due to some difficulties working with Encog's packaging system, and due to continued interest in exploring the algorithm at the source code level, a Java implementation of the algorithm was adapted from a selection of C♯ code from the site Dynamic Notions (Wakefield, 2008). Additionally, with nutritional data being very commonly-sought, the data collection did not take as long as it may have taken other groups with more niche types of data. Nutritional information data sources are plentiful and comprehensive. We found data provided by the US Department of Agriculture very early in the project and decided that it was a strong source for the baseline testing of the SOM on large sets of food.

Data Preparation

The USDA set comprises 7, 907 data points. Each datum in the set was comprised of the following 51 features: water, kilocalories, protein, lipids, ash, carbohydrates, fiber, sugar, calcium, iron, magnesium, phosphorus, potassium, sodium, zinc, copper, manganese, selenium, vitamin C, thiamin, riboflavin, niacin, panto acid, vitamin B6, folate, folic acid, food folate, folate DFE, choline, vitamin B12, vitamin A, vitamin A RAE, retinol, alpha carot, beta carot, beta crypt, lycopene, lut-zea, vitamin D, vitamin E, Vi. vitamin D IU, vitamin K, saturated fat, monounsaturated fat, polyunsaturated fat, cholesterol, GmWt_1[1], GmWt_Desc1[2], GmWt_2, GmWt_Desc2, Refuse_Pct[3]. However, we selected only 6 of these features to use in the SOM despite the wealth available to select from. For the purposes of this exploration; kilocalories, grams of carbohydrates, milligrams of cholesterol, grams of lipids (fats), grams of protein, and milligrams of sodium were selected. These 6 were chosen because they are the

[1] GmWt_1 - First household weight listed for this entry.
[2] GmWt_Desc1 – Description of the type of measurement for GmWt_1
[3] Percent of refuse matter found in database entry

best-known and most-often advertised on Nutritional Facts labels. All of the data are labeled, although labels bear no significance on the algorithmic level, only the analytical level.

The feature values in the original database are based on 1 serving, which we retained in our data set. After considering normalizing the amounts to 1 gram, we realized that portions of "1 serving" were actually more fair with respect to one of our practical goals, which is monitoring what people eat. Consider a teaspoon of butter, which amounts to approximately 5 grams. 1 teaspoon may constitute a serving of food if the food is butter, but a teaspoon does not come close to approaching the serving size for steak. Additionally, the lipid ratio between a gram of butter and a gram of steak is very high, but the lipid ratio between servings of each food is similar. Another example would be a tablespoon of table salt (almost entirely sodium) being compared with tablespoon of whipped cream. A tablespoon of whipped cream may be close to what somebody puts on a pie, but a tablespoon of salt is not close to what one puts on anything. It wouldn't be fair or interesting to see how similar amounts of food, in mass, compare to each other because we do not eat all foods in the same quantities. The serving size system was created to be a fair comparator between foods, and fortunately, was the standard measure for all the database entries to the best of our determination.

The data file is formatted in XML. Here is an excerpt of an item from the XML file, with fields restricted to those of interest regarding our experiment to save space:

```
<Nutrition_Entry>
        <NDB_No>01001</NDB_No>
        <Shrt_Desc>BUTTER, WITH SALT</Shrt_Desc>
        <Energ_Kcal>717</Energ_Kcal>
        <Protein_g>0.85</Protein_g>
        <Lipid_Tot_g>81.11</Lipid_Tot_g>
        <Carbohydrt_g>0.06</Carbohydrt_g>
        <Sodium_mg>714</Sodium_mg>
        <Cholestrl_mg>215</Cholestrl_mg>
</Nutrition_Entry>
```

Algorithm Implementation Details

A SOM is a special case of a 2 layer artificial neural network. Where the input vector is equal in length to the number of features selected, and the output is equal to the size of the map. For that reason, a SOM works by using a specific modification scheme on the weights between the neurons. In a 'normal' artificial neural network, the weights are adjusted after seeing the difference between what was expected, 'the target', and what was obtained, 'the output'. In a SOM, there are no expected targets; instead the input becomes the expected target.

When a training example is applied to the inputs, all the neurons are processed, and the outputs are calculated as a normal neural network. That is where the similarities end. The output with the highest value is then selected and declared the winner. The weight adjustments occur based on 3 separate calculations which are then combined to determine the weights net values.

The first part is a simulated annealing process for finding an appropriate learning rate through epochs. The original code was designed for a much smaller data set, and so the original annealing process, based on iterations, was inappropriate when considering our 7,907 items. More on why a switch from iterations to epochs will be detailed in the Problems Experienced section. The simulated annealing process guarantees that as the training progresses the rate of change in the map will lessen, this is critical in order to allow the winning neurons to slow their movement around the map.

The second part is an inclusion of a neighborhood factor in the calculations. The neighborhood factor lessens the effect of weight adjustments on neighboring neurons the further they are away from the winning neuron. This results in a localized potency of weight alteration, with farther neurons' weights being changed less than closer neurons' weights, and ensures a smoother gradient around the winner.

The third part is the lynchpin of the calculation. The value of the input feature is subtracted from the weight connecting the input neuron to the output neuron. This difference is then multiplied against both the annealing factor and the neighborhood factor. When factored together, the result is a smoothing of the maps output topology.

The winning neuron is determined by finding the neuron which produces the shortest distance as calculated using the Euclidean distance formula, with the two input vectors being the input features, and the weights connected to each of the output neurons. In other words, the winner is the neuron with whose weights are most similar is value to their respective input features values. Error is then calculated based on the cumulative amount of corrections, to the weights divided by the size of the map, for each epoch.

Problems Experienced

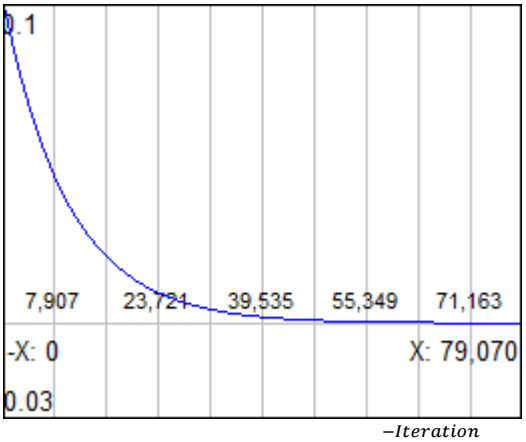

Fig. 1 – Original Annealing Rate: $e^{\frac{-Iteration}{10000}} \times 0.1$

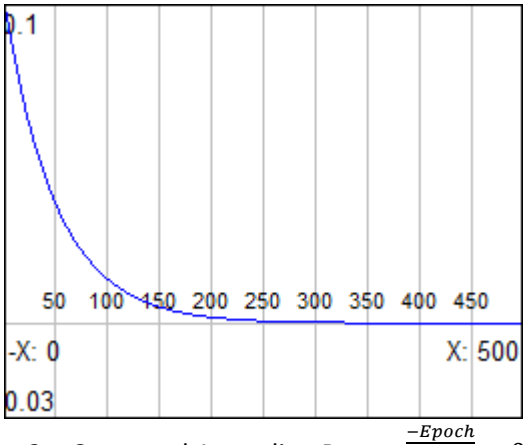

Fig. 2 – Corrected Annealing Rate: $e^{\frac{-Epoch}{50}} \times 0.1$

The original annealing rate (Fig 1.) used 10,000 as a base value per iteration. What this meant is that after each training datum the iteration would increase by 1, and at the conclusion of the first epoch the last datum would have received half the adjustment that the first datum in the same epoch would

have received. This effectively disabled the ability to make appropriate adjustments equally throughout the epoch, which in turn resulted in the map being unable to converge to a substantially low error. The solution was found by making two changes to the annealing rate. The first is that the annealing rate would only be adjusted after each epoch, thereby ensuring that each datum during an epoch receives the same rate of adjustment. The second is to reduce the base value from 10,000 iterations to 50 Epoch, which actually resulted in more datum training. This change resulted in faster convergence, the original learning rate often failed to result in an error of less than 500 within 1 hour, however, the modified algorithm consistently resulted in an error of less than 50 within 1 hour. Convergence to a suitable maximal error of 0.0000001 took a little over 36 hours.

Real time monitoring of the training process also resulted in substantially slower processing and was disabled. A filter system was then designed which allowed the map to be trained in a pure calculation mode. Whereby after training, a datum filter, would then apply selected data to the map, and display their resulting locations on the map. The resulted in a substantial speed increase in the training system.

Results and Evaluation

Evaluation is the final step of solving a problem with machine learning, assuming that the performance is evaluated favorably. In an unsupervised learning environment, the performance can be hard to gauge because there are no right or wrong answers. In an unsupervised learning problem, previously-unknown patterns are being sought. So, in order to evaluate the SOM implementation, we decided to implement the previously-mentioned application, a case study on baby food.

To understand the baby foods experiment, we first had to learn where the hot points in the map were located. The hot points, are locations where a given features highest values, and the lowest values can be found. The darker areas represent concentrations of higher values; while the lighter colors represent areas of lower amounts, Table 1 shows the actual values being represented.

	Low Regions	High Regions
Calories	<30.0 K-cals	>600.0 K-cals
Carbohydrates	<0.1 g	>90.0 g
Cholesterol	<1.0 mg	>800.0 mg
Lipids	<0.005 g	>80.0 g
Protein	<0.5 g	>50.0 g
Sodium	<1.0 mg	>3000.0 mg

Table 1 – Values of hot point locations for Fig. 3

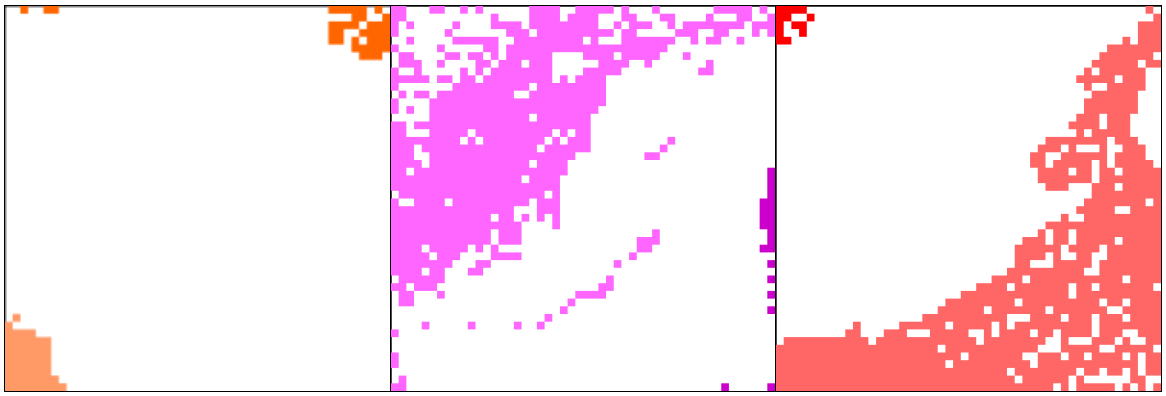

Fig. 3.1 - Calories | Fig. 3.2 - Carbohydrates | Fig. 3.3 - Cholesterol

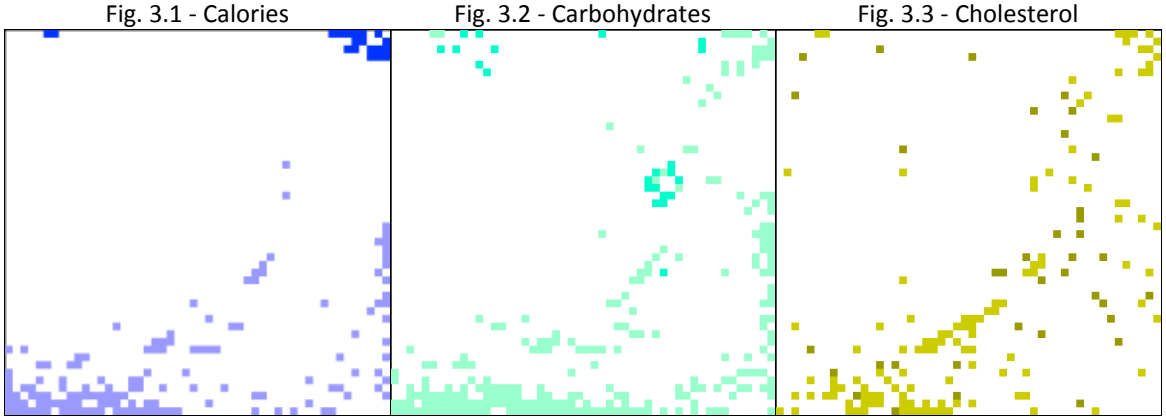

Fig. 3.4 – Lipids | Fig. 3.5 - Protein | Fig. 3.6 Sodium

Fig. 3 – Feature Map Hot Points

Reviewing the hot points, calorie(Fig. 3.1) and lipid(Fig. 3.4) low values have been mapped in the lower left, and increasing in value to the upper right corner. Carbohydrate(Fig. 3.2) high values sweep from the right side to the low values in the upper left. Cholesterol(Fig. 3.3) high value begin in the upper left corner, and fan out to the right and lower sides. Protein(Fig. 3.5) high values begin right of center radiating out to the top, right and lower sides. Sodium(Fig. 3.6) high points appear scattered throughout the map, with a possible congregation in to lower right quadrant, and a similar scattering throughout the map for low values.

With an understanding of how the hot points for each of our features are mapped, understanding the contours is now necessary. Figure 4 illustrates the division of the map according to the features. The values selected to divide the features were chosen to provide the most information about the contours, and have no mathematical meaning. Various mathematically derived values, such as the global mean, modes, and medians were tried, but did not yield any map patterns containing relevant information. Again, high values are in dark, and are equal to greater than the attributed value. For examples, all food items in the database with Calories greater than, or equal to 150 K-Cals are colored using dark orange, while those food items with strictly less than 150 K-Cals are colored in light orange. This method is then repeated for the remaining features.

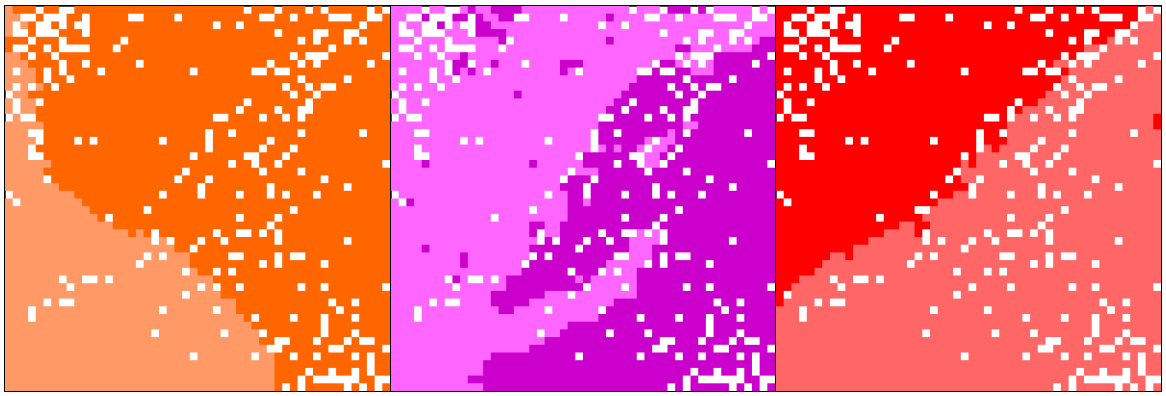

Fig. 3.1 – Calories – 150.0 K-Cals | Fig. 3.2 – Carbohydrates – 10.0 g | Fig. 3.3 – Cholesterol – 50.0 mg

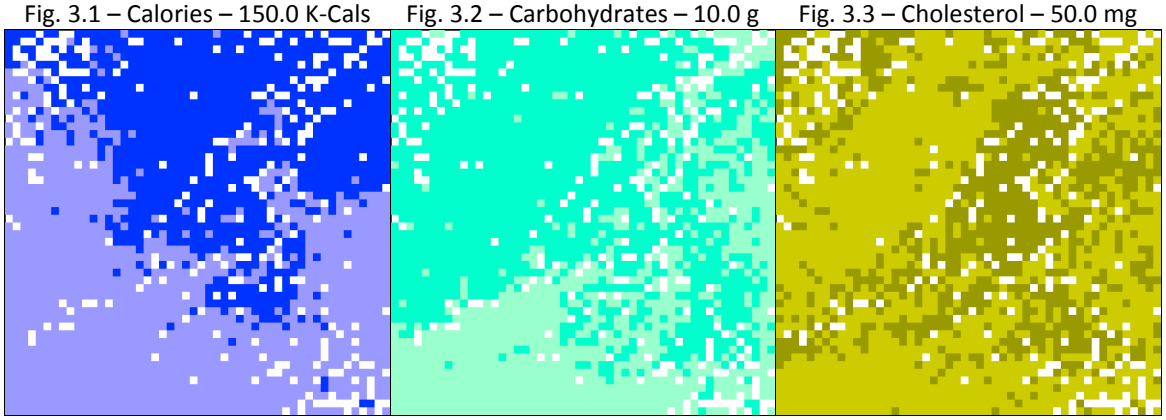

Fig. 3.4 – Lipids – 7.50 g | Fig. 3.5 – Protein – 5.0 g | Fig. 3.6 - Sodium – 85.0 mg

Fig. 4 – Feature Map Contours

Now that we had an understanding of final map structure we wanted to know how our target market fit onto this map. We restricted the map to display only the high valued hot points, strictly as land marks on the map, and then activated all food items with “Babyfood” in the description. Babyfood items appear in bright lime green. (See Fig. 5)

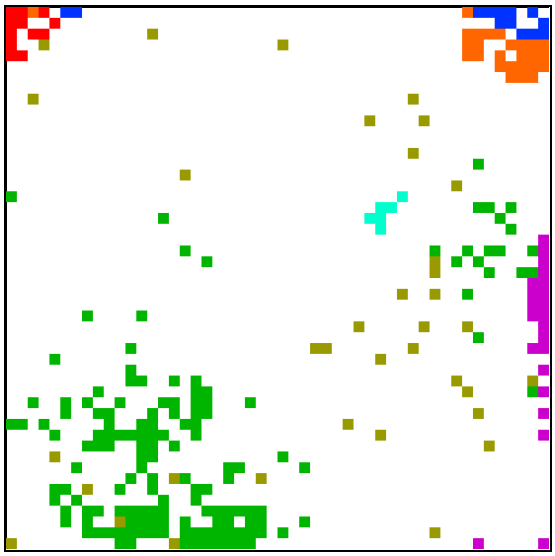

Fig. 5 – Babyfood items super imposed on feature high hotspots

It is notable that most of the baby food items are removed from the high feature centers, probably implying that most baby food is carefully designed to not contain extremely high concentrations of any of our chosen features. High-calories are orange; high-lipids are blue; high-carbohydrates are purple; high-proteins are light blue; high-cholesterols are red. It's also interesting to note that there are 2 distinct clusters of baby food, and then some outliers tending toward higher cholesterol. The two clusters are located in the lower left large blob of green points and on the right side near high carbohydrates. Most of the lower cluster was jar baby food, the typical jar of pureed vegetables, and such baby food found in a jar. The right edge cluster was mostly baby crackers.

Between the clusters however is a fairly large gap. Since a review of the contours, and the high and low hot points don't indicate that this gap would contain extreme values of any of the selected features, it was decided to explore this region of the map. All data points were then activated, as can be seen in Fig. 6, as grey points. And we began to explore the data points. Two points selected were revealed to represent Tacos, Burgers, Lemon Chicken and Pizza.

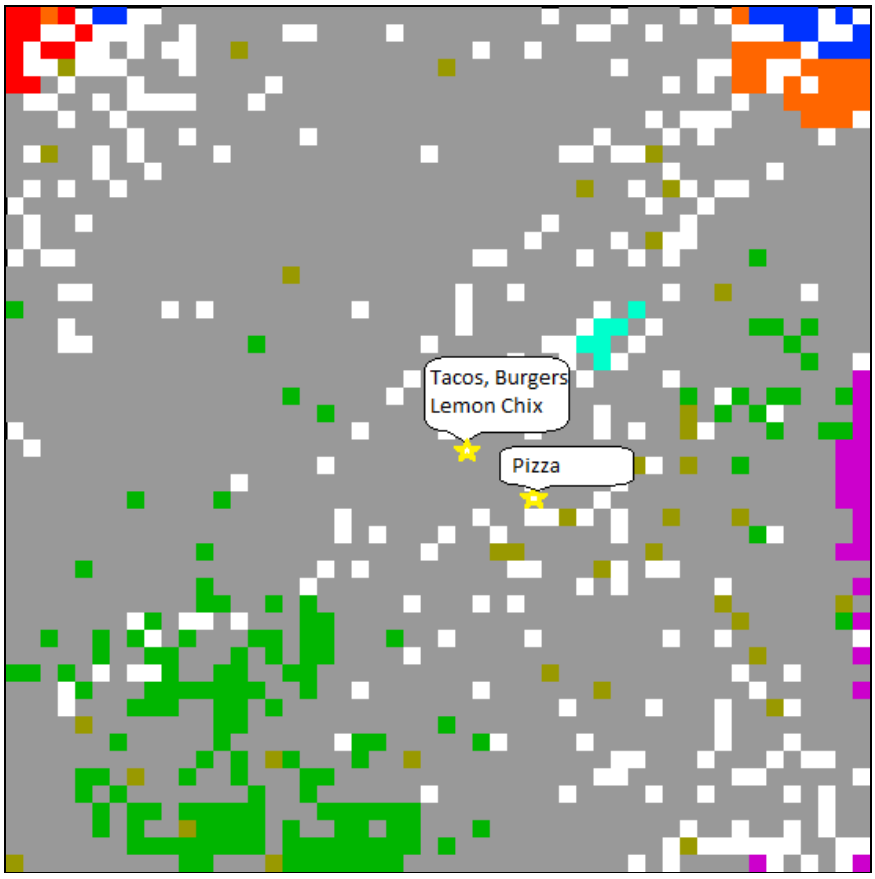

Fig. 6 – Implementation of all data points in grey, and selection of 2 previously unexploited points.

This makes the interesting suggestion that some combinations of the features found in tacos, burgers, etc. are underrepresented in baby food. As stated earlier, this evaluation is not suggesting that babies consuming this food are malnourished by any metric; the actual observation should not be taken as something pragmatically significant. Instead, it is an experiment in finding patterns in large amounts

of data. Because we were able to find a hole in the distribution, we are confident that such an algorithm could be used in other situations where similar pattern discovery and identification are needed.

Further exploration of these unused food groups resulted in a wide distribution of those food items. (See Fig 7.) As can be seen by the new items in light yellow, there is a wide array of food items in this category covering a variety of selectable feature combinations. This would indicate an exploitable area of the baby food market.

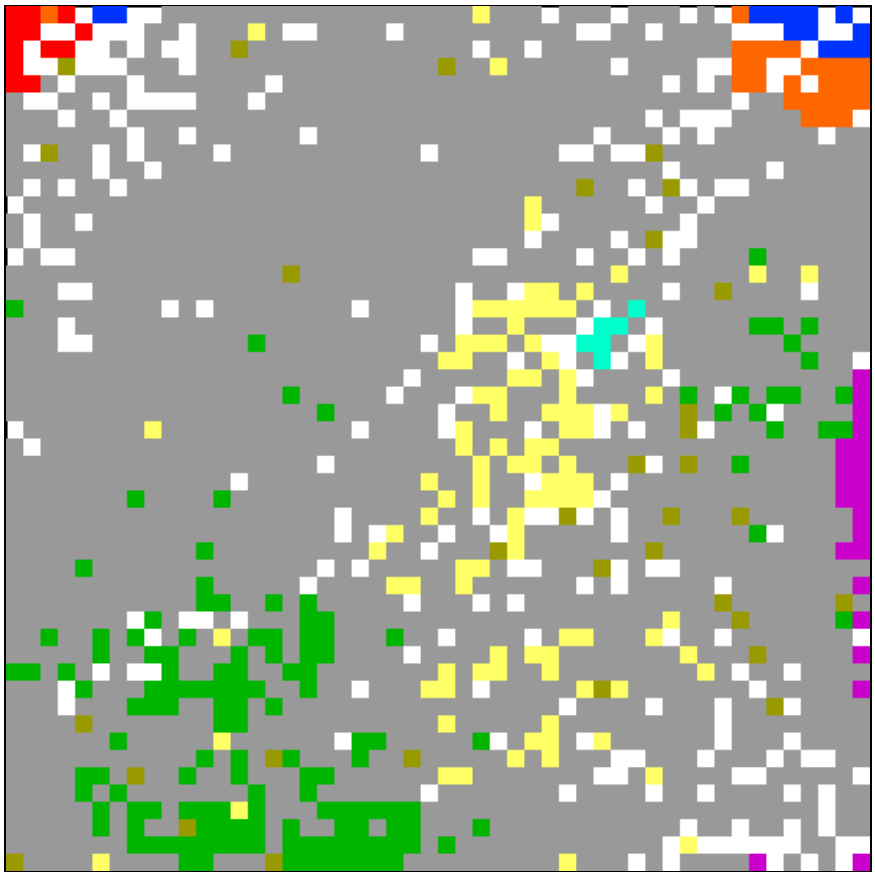

Fig. 7 – Distribution of Tacos, Pizza and Burgers

Future work with this specific implementation could involve using it to map other dietary needs. For example, if babies really do need other nutritional components found in burgers and pizza, then information could become available through this type of data exploration. It would be interesting to see how well these patterns map to vitamin and other nutrients. A business could use it to fulfill a market need, or an individual may use it to fill a personal health need. It is also of very high intrigue that this general algorithm produces results that are so intuitive to human observers.

References

Kohonen, Teuvo. *Self-organizing Maps, 3rd Ed.* Berlin, New York: Springer, 2001.

Vicente P Guerrero Botea, Félix de Moya Anegónb, Victor Herrero Solanab.
Document Organization using Kohonen's Algorithm.

Guerrero Bote, Vicente P., Félix de Moya Anegón, Victor Herrero Solana.
Document organization using Kohonen's algorithm. Information Processing & Management, Volume 38, Issue 1, January 2002, Pages 79-89, ISSN 0306-4573, 10.1016/S0306-4573(00)00066-2.
(http://www.sciencedirect.com/science/article/pii/S0306457300000662).

USDA National Nutrient Database for Standard Reference.
(http://www.ars.usda.gov/Services/docs.htm?docid=8964)

Wakefield, John. *C# Self Organising Map (SOM).*
(http://dynamicnotions.blogspot.com/2008_11_01_archive.html)