Applications of Self Organized Maps (SOM)

Jacob Epifano
Machine Learning 1
22 December 2017

***Abstract*—A Self Organized Map (SOM) is a type of unsupervised neural network. It is a type of artificial neural network where each 'neuron' is linked to create a map of the input space. By clustering a high dimensional input, a 2-D representation of the input space can be created by the relationship of the linked 'neurons'. [1]**

Since SOMs are designed to analyze the relationships of high dimensional data, a perfect data set for them is genome data. A paper by Todsanai Chumwatana used SOMs to cluster and classify genome sequences. Using a different approach, I attempted to recreate his results. I used the Batch Kohonen training method to train my model. I was able to classify the genome sequences with an accuracy of 75% at the first taxonomic level.

I. Introduction and Objectives

Self Organized Maps (SOM) is very similar to the popular clustering algorithm, K-means. Instead of using centroids to mark the center of each cluster, SOM uses linked 'neurons' to map the input space. In K-means the number of centroids is determined by the number of clusters. In SOMs, the number of neurons is much greater than the number of clusters. The more neurons you have, the more accurate of a representation of the input space you will generate. The figure below accurately represents what the algorithm is doing. [1]

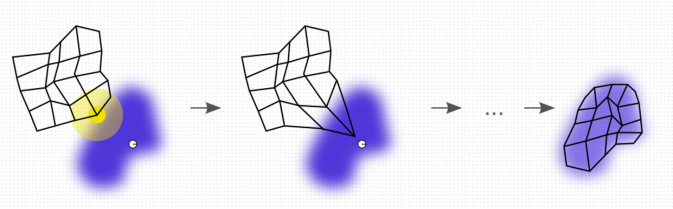

Fig. 1. SOM Mapping Diagram [1]

In figure 1, the blue field is a N-dimensional input space and the grid is the untrained SOM. As the SOM trains, it begins to wrap itself around that input space. The right most image in figure 1 shows the trained model. The grid is now a 2-D representation of the input.

My objective was to re-create the work of Todsanai Chumwatana by analyzing a set of genetic sequences and use SOM to classify the sequences. In the article by Chumwatana, he globally aligned sequences and identified frequent sub-strings in each sequences to classify them. The results for his algorithm are below.

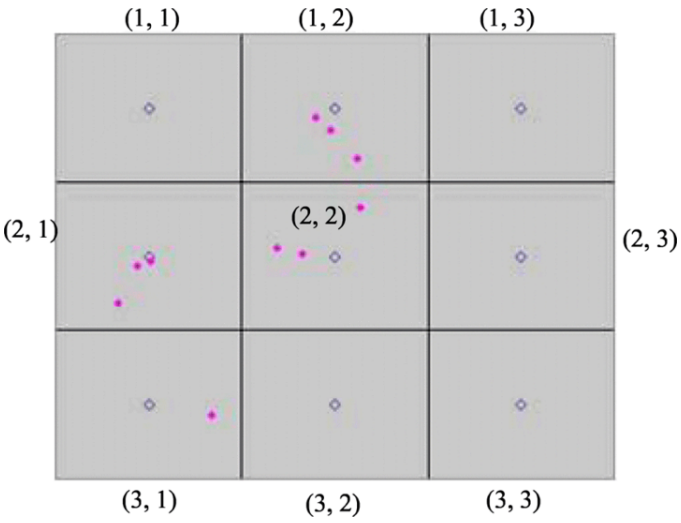

Fig. 2. Chumwatana's Classification Results [2]

In figure 2, each cell represents a different neuron. Each pink dot represents an input to the algorithm. You can see that many of them are clustered together. In the paper by Chumwatana they determined one classification to be mammal and another to be marsupial classification. I will be taking a slightly different approach to achieve the same outcome. [2]

II. Background and Relevant Theory

A. Dataset

I first attempted to use the same dataset that Chumwatana used in his paper. He used full length DNA from different animals. Most of these sequences were close to one million nucleotides in length. When trying to globally align these sequences I ran out of RAM, thus I had to use a different dataset. I ended up using the RDP dataset, which is a set of ten thousand "quality-controlled, aligned and annotated Bacterial and Archaeal 16S rRNA sequences." [3] These samples are much shorter in length and are pre-aligned making them much easier to work with. The data that was used as the input to the training algorithm was the K-mers of the RDP dataset.

B. K-mers

K-mers are a count of the occurrences of every combination of sub-string of length k. Since gnomic sequences are made up of nucleotides with four possible combinations; 'A','T','C','G', a K-mer would be a list of length 4^k. The below table shows a 2-mer or K = 2. There are 16 possible combinations of nucleotide sub-strings.

TABLE I
2-MER OF SAMPLE 1

Sub-string	Count
GG	168
CG	120
GC	118
TG	115
GT	109
GA	106
AG	98
AC	90
CC	86
AA	85
CA	78
CT	73
TA	63
TC	63
AT	59
TT	49

Since the count of each combination are 'unique' to each sequence as we increase the value of K, greater separation in each sample is shown. Since the sequences in the RDP dataset vary in length the data must be normalized. Before passing the data into the algorithm, the number of occurrences for each sub-string was summed and each sub-string was then divided by that sum.

C. Batch-Kohonen Training

The algorithm iterates over the whole dataset and finds the distance from each neuron to each input vector. The neuron with the closest input vector is the 'winner' or Best Matching Unit (BMU). After all the winners are determined, the neurons move towards the mean of all the points they were the BMU for according to the following update rule: [4]

$$w_{t+1} = w_t + L_t(V_t - w_t) \tag{1}$$

w_{t+1}: new weight position, w_t: old weight position, L_t: learning rate, $V_t - w_t$: distance from input vector to weight.

III. RESULTS AND DISCUSSION

A. Mapping Results

I collected the results using MATLAB's built in SOM function. The algorithm was tested using four different values of K. The results are as follows:

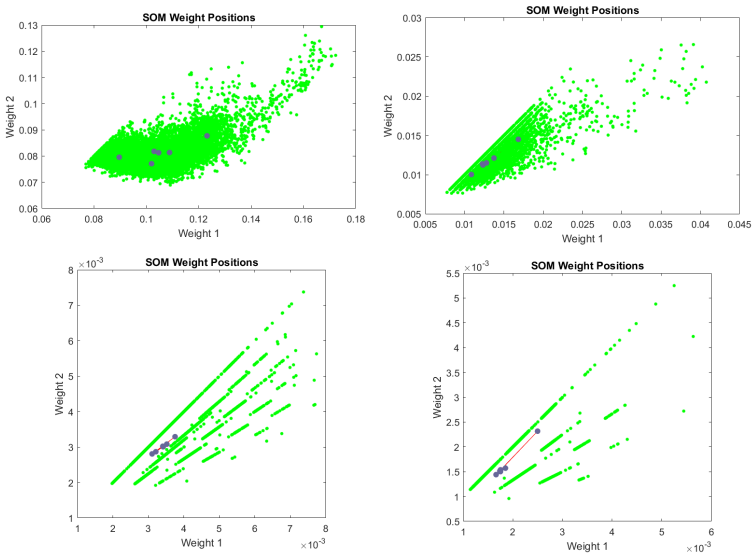

Fig. 3. SOM mapping for each value of K

The algorithm use a 40x40 grid of neurons to map the dataset. This gives me an overall neuron count of 1600. The mapping performance of the algorithm decreases as the value of K increases. Over 99% of the neurons seem to settle on the same point on the graph. My hypothesis is as, K increases, the amount of data 'decreases'. What I mean by this is that there are less sub-string combinations that exist in each dataset. In table I, at K = 2, each sub-string has many occurrences in each sequence. When K is increased to 8, 99% of the counts are 0 for each sequence. While this provides more separation in the samples, this gives the algorithm less overall data to build the map. The majority of the neurons are settling where that data lies.

B. Classification Results

Lowering the neuron count down to two, let me use the SOM to classify at the first taxonomic level. The two classes presented in this set of data are Bacteria and Archaea. Repeating the classification for K = 2-8, I found the following results.

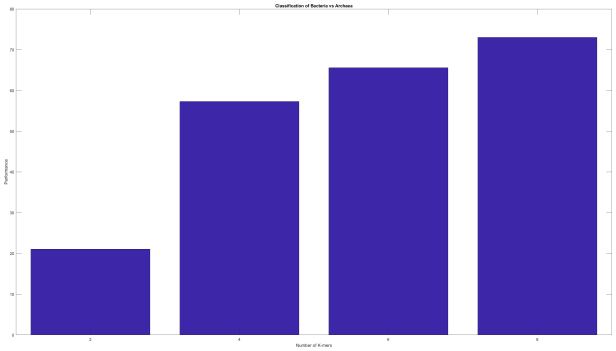

Fig. 4. Performance for each value of K:[2,4,6,8]

The performance from figure 4 shows that as I increase the value of K, the performance of classification also increases. While there are only two classes, I believe this shows the SOM has the ability to work as a classification algorithm. If I had more computational power to run higher neuron counts at higher K values, I would expect that I could achieve similar results at deeper taxonomic depths.

IV. Conclusions

The objective of this experiment was to replicate the classification results of the paper by Chumwatana. Using K-mers as a means of directly comparing sequences, the batch Kohonoen training algorithm proved effective at classifying at the first taxonomic level. With more computation power, I would be able to replicate this results at deeper taxonomic levels with higher levels of K.

References

[1] En.wikipedia.org. (2017). Self-organizing map. [online] Available at: https://en.wikipedia.org/wiki/Self-organizing-map [Accessed 2 Dec. 2017].

[2] Chumwatana, T. (2013). Genome sequence clustering using hybrid method: Self-organizing map and frequent max substring techniques. 2013 International Conference on Machine Learning and Cybernetics.

[3] Rdp.cme.msu.edu. (2017). RDP Release 11 – Sequence Analysis Tools. [online] Available at: https://rdp.cme.msu.edu/ [Accessed 2 Dec. 2017].

[4] Blog.yhat.com. (2017). Cite a Website - Cite This For Me. [online] Available at: http://blog.yhat.com/posts/self-organizing-maps-2.html [Accessed 2 Dec. 2017].

APPENDIX

All code and results can be found in the following github repository and path:
https://github.com/jrepifano/Machine-Learning
Machine-Learning $\Rightarrow$ GeneSequencingSOM $\Rightarrow$ V2