SELF-ORGANIZING MAP TEAMS UP WITH THE BUDDY SYSTEM

Dillon McTernan, Computer Science and Mathematics Department Southeastern Louisiana University, La 70402 (985)222-0805 W0254691@selu.edu

ABSTRACT

The Self-Organizing Map is a vital part of today's research towards unsupervised learning in Neural Networking. There are many commercial and scholarly applications for Self-Organizing Maps within data analysis, topological visualization, and clustering. In this paper we address the functionality of the Self-Organizing Map in its tendency towards greediness. It is found that the common solution to this issue is through an anti-greedy function called the Block Party. This anti-greedy function was unable to satisfy my desired requirements; thus the objective of creating a new anti-greedy function produced the Buddy System function. This paper compares these two anti-greedy functions and the possible advantages for this new Buddy System function.

INTRODUCTION

Artificial Neural Networking is a considerably new and highly advanced field of study where biological neural networks are reconstructed into computational and mathematical models using supervised, unsupervised, and reinforced learning. The Self-Organizing map(SOM) uses unsupervised learning to position weighted neurons in the vicinity of input data vectors as means to model the input data space. This placement is based upon matching the closest weight vector to the vector taken from the input data space. These best matching neurons are called winning neurons (WN). The new location of the WN is based upon a formula that analyzes the relationship between the WN and the input data vector. The idea of greediness occurs in the SOM when any neuron is not activated as a WN. This conflict will not allow an accurate distribution of the inactivated neuron, therefore producing a poor representation of the sample input data. Anti-greedy functions such as the Block Party were established to minimize this error [4]. Their general solution to this problem is to establish connections between the WN and its surrounding neurons. Thus when the WN moves closer to the input data vector, its neuron neighbors will also move in the same direction though with lesser intensity. The Block Party establishes the WN neighbors by initializing the neurons in a grid pattern where upon it chooses the neighbors with a block formation. A problem arose when desired requirements called for randomized placement for the initialization of the neurons. This lead to the creation of an anti-greedy function called the Buddy System. This new anti-greedy function uses Euclidean distance to find a WN's neighbors and therefore negates the necessity for a pre-designed pattern for the initialization of neurons.

The next section, BACKGROUND, will overview a brief history of beginning Neural Networks leading up to the Buddy System. Following will be a section on functionality where upon we will discuss the internal structure of the SOM, Block Party, and Buddy System as well as the problems and solutions of the Buddy System and its practical applications. The performance of the Buddy System will be evaluated in RESULTS and will be followed with the conclusion and references. Since these findings of the Buddy System were established in two dimensional space, we will be continuing the description of the SOM, Block Party, and Buddy System, in its entirety, in the two dimensional context.

BACKGROUND
Artificial Neural Networks became part of public awareness in the 1960's under the term perceptrons. In 1969 Minsky and Papert published a book that described the attributes of single layer perceptrons. These findings on the single layer perceptrons showed its characteristics to have drastic limitations in basic pattern recognition [5]. These findings where a devastating blow to the field of Artificial Neural Networking and created a loss of interest in this field [5]. It wasn't until the 1980's that this field became popular again as the discovery of trained multilevel perceptrons proved to accomplish the problematic pattern recognitions of single layer perceptrons [5]. The trained multilevel perceptrons created the idea of supervised learning, and in conjunction, it brought up the quandary of unsupervised learning. It was in the early 1980's when Tevo Kohonen helped to establish the idea of unsupervised learning with the invention of the Self-Organizing Map(SOM) [3]. One of the major dilemmas with his SOM was an issue of greediness within his algorithm, where upon the creation the block party function was established to minimalize this issue [4]. Though the Block Party function lends adequate assistance to the SOM, there lacked a sense of randomness in the initialization of the neurons. Thus, a new anti-greedy function, called the Buddy system, was created and established a randomized initialization of neurons for the SOM.

FUNCTIONALITY
The SOM starts with gathering a rectangular boundary around the input coordinates. It then starts to initialize the neurons with x,y coordinates from inside the boundary. After the initialization of the neurons, the SOM selects the first input data point in a list and finds the closest neuron to that data point, where upon it assigns it as the WN. It then uses the formula:

WN[new_location] = WN[old_location] + ((DataPoint-WN[old_location])*epsilon)

to establish the new location of the WN.[2] Note that epsilon is a monotonically decreasing learning coefficient. This procedure continues down the list for every data point in our sample. During this procedure of repeatedly going over the list of data points, the SOM compares the neurons' current location to its previous location. If they have all moved less than the “margin of error,” then the SOM stops; otherwise it continues until such requirements have been met.

The Block Party function works along side the SOM, so the chance of a neuron not being set as a WN, or even moved at all, will be minuscule. This function informs the SOM to initialize the neurons in a grid pattern within the input boundary. The function then assigns the neurons that form a block around each neuron as its neighbors. So that when a neuron becomes a WN it has a block of neighbors that follow it . The formula in which the neighbors use to move is the same as the WN except that the epsilon is significantly less, thus moving with a lesser intensity.

The Buddy System works with the SOM on similar principles as the Block Party, except that it informs the SOM to initialize the neurons to random coordinates within the sample boundary. The way that the Buddy System creates a neighborhood of neurons is by first establishing how many neighbors will belong to a WN and then finding that number of the closest neurons to set as neighbors. Like the Block Party, the Buddy system creates the neighborhoods before the SOM starts processing the sample data, keeping the neighborhoods static. Again the neighbors of the WN will use the same formula as the WN except that the epsilon will be significantly less.

Problems and solutions

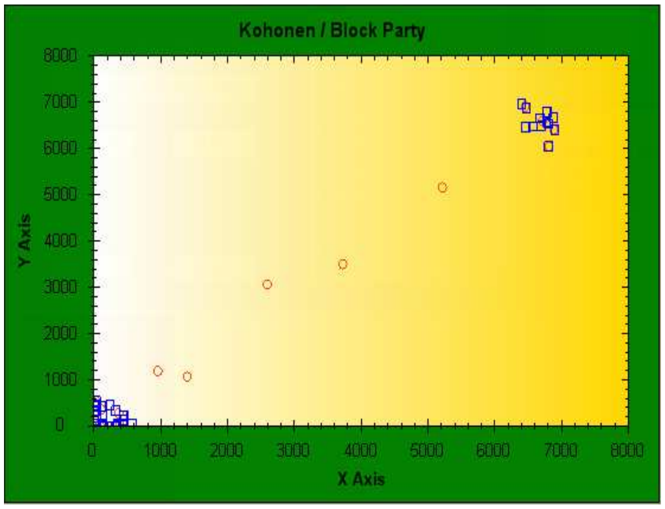
When creating the Buddy System it was found that on occasional examples there were neurons that did not move during the completed process. It was after many examples that brought the attention of a major difference between the Block Party and the Buddy System. Due to the grid formation of the Buddy system it was guaranteed that each WN would have a neighbor and that each neuron would be part of a neighborhood. [1] Even though each WN had a set amount of neighbors in the Buddy System, it was not guaranteed that each neuron was part of a neighborhood. The solution to this problem was to set the amount of neighbors to be one less than the amount of total neurons, thus guaranteeing every neuron to function as a neighbor or expanding a neuron's neighborhood to incorporate this lost neuron. The later solution builds on the idea of allowing these neighborhoods to become dynamic if necessary. The idea is that if the Buddy System function notices a lost neuron, it will add it to its closest moving neighbor's neighborhood, thus dynamically expanding this neighbor's neighborhood to incorporate a lost neuron.

Application

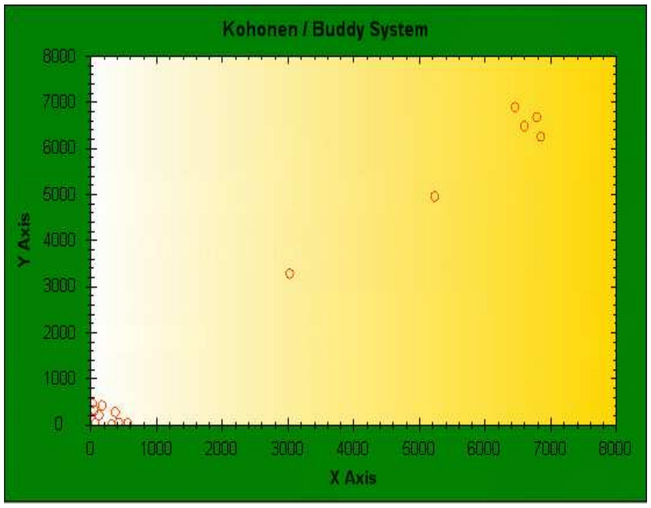
One of the greatest applications of the Buddy System is that if one is able to randomly initialize neurons on a coordinate plane, then one can initialize the neurons to any desired location. The reason that this would be beneficial is if one would want to take note of each neuron's distance from its beginning location to its final location. A practical application of this would be a telecommunications company looking to update the location of their cell towers to meet the needs of their client base; consider the neurons to be cell phone towers and the sample input data to be an updated population survey. The telecommunications company can use the Buddy System along with a SOM to retrieve the cell tower's current location and its potential location that would best suit this new population survey. This will allow them to accurately analyze data for essential documentation such as a cost versus value report or a feasibility report.

RESULTS

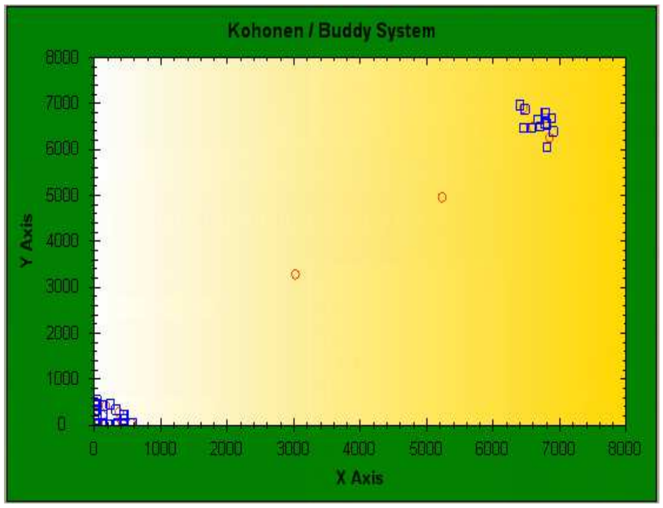
A SOM was implemented in C# to analyze both the Block Party and the Buddy System. The data results from figures (1a) and (1b) show the Buddy System while figure(2a) and (2b) show the Block Party. The input data sample, number of neurons (15), and margin of error (.001), were the same in both models. Take note that the blue squares are the input data points and the red circles are the neurons. Figures (1b) and (2b) had their data points removed for visibility purposes. Times were taken for both models in which the Buddy System finished processing in 4.04 seconds, and the Block Party finished processing in 8.39 seconds, but due to the



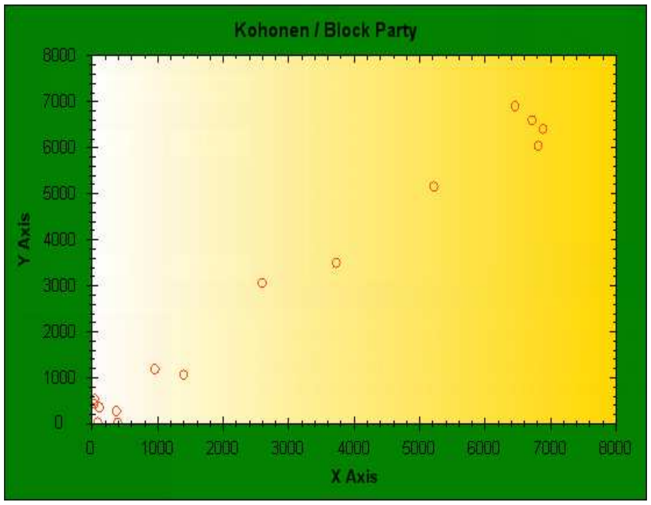
(1a)



(1b)



(2a)



(2b)

possibility of non-efficient coding, recorded times are only an estimated reference. From this sample data of two disjoint regions, there shows similar representations between both anti-greedy functions. This clearly shows that the Buddy System function can produce a reasonably accurate depiction of a SOM with the Block Party function and still be capable of randomizing the initialized neurons.

CONCLUSION

In today's market there are many applications for SOMs and as the need grows so does the diversity for desired specifications. With the new Buddy System function, the SOM will now be able to expand its potential in satisfying client requirements. Its ability to implement random neurons before creating neighborhoods sets this anti-greedy function apart from the common Block Party. With the competitive accuracy of the Block Party function and the potential for a user declaration of neuron initialization location, the Buddy System proves to be a powerful tool when teamed up with the Kohonen Self-Organizing Map.

REFERENCES

1. Heaton,J.,*Introduction to Neural Networks for Java, 2nd Edition*, St. Louis: Heaton Research Inc. July 2007.
2. Honkela, T., *Self-organizing maps in natural language processing*, Doctoral Dissertation, Helsinki University of Technology, Espoo, Finland, 1997. Available at: http://www.cis.hut.fi/~tho/thesis/
3. Kohonen, T. Self-organizing formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59-69. 1982.
4. Koutsougeras,C., Liu, Y., Zheng,R,Event-driven sensor deployment using self-organizing maps, *Int. J. Sensor Networks*,3, (3), 142-151,2008.
5. Stergiou, C., Siganos, D., Neural Networks, *Surprise 96 Journal*, Vol. 14, Imperial College of Science, Technology and Medicine, London, 1996. Available at: http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.htm