# A Borda Count Based Initialization Method for Self-Organizing Maps

S. ŞENOL[1] and F. İSMAİLOĞLU[1]

[1] Sivas Cumhuriyet University, Sivas/Turkey, sumeyrasenol43@gmail.com
[1] Sivas Cumhuriyet University, Sivas/Turkey, fismailoglu@cumhuriyet.edu.tr

***Abstract* - In data science, data visualization has long been of interest since it is necessary to explore the patterns and underlying structure of data. Over the years, various data visualization methods have been developed. Among them, Self-Organizing Maps (SOM) is one of the most widely used methods due to its simplicity. However, the robustness of a SOM largely depends on the initial choice of prototype vectors in it. On the other hand, this problem has not been extensively studied in the literature. In this study, we introduce a novel method for initializing the prototype vectors of SOM, which we call BCI and which has its roots in Borda Count (BC), a ranked voting system. A large set of experiments confirmed the effectiveness of BCI and showed that it outperforms both the random initialization and PCI, the state-of-the-art initialization method, in terms of quantization error, which indicates how well the SOM reflects the original feature space.**

***Keywords* – Data Mining, Data Visualization, Self-Organizing Maps, Borda Count.**

## I. INTRODUCTION

Regardless of the type of data we are interested in, e.g., images, audio, sensors, etc., the data often has a high dimension. On the other hand, we humans cannot see more than three dimensions. This makes it difficult to understand the underlying structure of the data. For this reason, more and more data visualization methods have been developed not only in data science but also in various other fields [1,2].

Among all data visualization methods developed so far, Self-Organizing Maps occupy a special place [2]. Since its introduction by Kohonen in 1990, SOM has found various and numerous applications [3]. The main reason for the success of SOM is the fact that SOM is a simple yet powerful visualization method and can preserve the topology of data. Although it is inherently a heuristic method, the theoretical aspects of SOM have also been addressed so that SOM comes with theoretical guarantees [4].

The working principle of SOM can be summarized as follows. A SOM consists of a series of adjacent cells, generally arranged in a 2D map, with each cell associated with a prototype (weight) vector of the same dimension as the original data. Training a SOM involves learning these vectors. Once the training process is complete, each instance is mapped to one of the cells in the map based on its distances from the prototype vectors. Here, the prototype vectors are learned using an iterative algorithm where the initial values for the vectors are randomly selected [2]. Such a random process not only increases the learning time, but also leads to different visualization results.

To illustrate the above random problem, we provide a small dataset in Table 1. This dataset represents seven G7 countries based on five features: population (millions), life expectancy (years),Co2 emissions (MTCo2e), rule of law and GDP per capita (Dollars). With this data, we construct two SOMs of dimensions 3 by 3 and initialize the prototype vectors differently. Even after one thousand iterations in both applications, the underlying algorithm does not converge, resulting in two different SOMs: SOM1 and SOM2. See Figure 1. Here, a natural question to ask is which is the correct one?

Table 1: G7 Countries (Population (millions), life expectancy (years), Co2 emissions (MTCo2e), rule of law and GDP per capita (Dollars)) (Source: World Economics 2021)

| Country | Pop. | Life Exp. | Co2 | Law | GDP |
|---|---|---|---|---|---|
| Canada | 38.1 | 82 | 576.7 | 89.5 | 53509 |
| France | 65.4 | 83 | 323.6 | 81.4 | 52721 |
| Germany | 83.9 | 81 | 702 | 87.1 | 58283 |
| Italy | 60.4 | 84 | 337.1 | 54.6 | 50764 |
| Japan | 126.1 | 85 | 1106.7 | 86.3 | 44744 |
| UK | 68.2 | 81 | 369.9 | 85.6 | 49858 |
| USA | 332.9 | 79 | 5284.7 | 82.4 | 67651 |

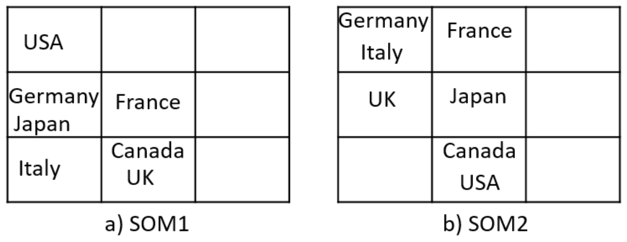

Figure 1: Two SOMs using the G7 data in Table 1 with different initializations

Although random selection of the prototype vectors is a serious problem and leads to quite different SOM results, there are few studies addressing this issue. One notable study is [5], which is based on using PCA to reduce the original dimensionality to two, and then finding the nearest cell for each datum based on the (2D) coordinates of the cells. However, this method requires an eigendecomposition, which is costly, especially for high dimensions.

In this paper, we introduce a novel initialization method for SOM. In particular, our method addresses initialization of the prototype vectors in SOM so that the SOM algorithm can converge faster and end up with the same map for each run. The proposed method has its roots in the Borda count, a voting method that has been used for centuries to determine the ranking of candidates that best fits voters' preferences [6]. In short, we rank the cells of a SOM using Borda count from the most central to the least central; and we rank the data instances from the most central to the most distant. We then match these two rankings and assign the instances to the cells. Finally, the mean of the instances that are assigned to a particular cell becomes the initial prototype vector of that cell. This method offers several advantages, including *i)* ease of implementation *ii)* faster convergence *iii)* robustness.

In the remainder of the paper, we first review the literature on data visualization methods and SOM in particular, and then provide background information on SOM. We then proceed to the proposed method and conduct experiments to test its effectiveness. Finally, we summarize the study and point out possible future developments.

## II. RELATED WORK

Data visualization refers to the visual representation of data, and this representation can consist of charts, plots, graphs, or even animations. With these tools, one can effortlessly gain useful insights from the data at hand. Common data analysis tasks such as outlier detection, correlation analysis, and highlighting trends can all be performed with ease by visualizing the data. Because of these advantages, data visualization has gained an increasing popularity and is being studied by researchers with diverse backgrounds, from statisticians to psychologists [7].

Millions of data are obtained during the day from the internet searches we make every day, the records kept when we go to the hospital, the receipts of the products we buy while shopping, the transactions we make at the bank and many other areas of our daily life. Various data visualization methods are used to understand what these data mean.

As emphasized above, visualizing data is of crucial importance to get to know the data. Nevertheless, data often has a high dimension, which makes it nontrivial to visualize. In this sense, it is common to reduce the dimension of the data to two or three so that we can easily plot it. One of the most common methods for this purpose is PCA, which is based on projecting the data into the space spanned by the eigenvectors of the covariance matrix of the data. Although it offers the advantage of projecting the data into an uncorrelated space, it involves an eigendecomposition problem and is therefore computationally expensive.

Another popular method for data visualization is t-SNE, which, unlike PCA, provides nonlinear dimensionality reduction [8]. In fact, t-SNE is commonly used as the de facto standard for visualizing data in 2D spaces. Generally speaking, t-SNE creates two probability distributions, one for the original high-dimensional data and one for the reduced 2D data. It then tries to minimize the KL divergence between these two probability distributions, where the minimization imposes a non-convex optimization problem. The resulting problem is usually solved using a gradient descent with random initialization. Therefore, t-SNE may get stuck at local minima and not guarantee a global solution [8].

Self-Organizing Maps (SOM) have long served as the primary data visualization method. The reason for its popularity lies in its simplicity and the fact that it allows for clustering of data, in addition to being a visualization method [2, 3].

At a high level, SOM maps instances in a 2D map consisting of a series of adjacent cells of square shape. See Figure 1. The rule that governs the mapping is to assign each instance to the cell whose associated prototype vector is nearest to the vector. Learning the prototype vectors is equivalent to building a SOM. We will detail this process in the next section.

Considering the fact that SOM has been widely used for a long time, it is not surprising that there are several variants of it. One of the best known variants is the growing SOM (GSOM), which (usually) starts with a SOM of size 2x2 and then grows it from all cells based on a heuristic criterion [9]. In contrast to the conventional case, here, there is no fixed shape for the map, in fact it is tailored to the data at hand. Another important variant is the hexagonal SOM, in which the cells have a hexagonal shape. The distinguishing element of this variant is that each cell has six neighbors, not four as is the case with square cells. Finally, Kohonen, the inventor of SOM, proposed a batch version of SOM [10], which proposes to update the prototype vectors only at the end of each iteration of the training, whereas in the conventional case they are updated after the presentation of each instance. The main advantage of this version is that the learning of the prototype vectors is independent of the order of the instances.

As mentioned earlier, Kohonen proposed to randomize the initial values for the prototype vectors. In practice, however, this results in different final values for the vectors, which in turn results in different maps each run of the SOM. Therefore, it is necessary to initialize the vectors deterministically. For this purpose, the most commonly used initialization method is principal component initialization (PCI) [5]. The working principle of PCI is as follows. Let $e_1$ and $e_2$ be the first two eigen vectors of the data; and $a$ and $b$ are two real numbers such that $a < b$. Suppose that we have a SOM with $m$ cells in the vertical axis and $n$ cells in the horizontal axis and that $u = (a, a + \frac{b-a}{m}, ..., b)$ and $v = (a, a + \frac{b-a}{n}, ..., b)$ are two sequences. PCI determines the initial prototype vector of the cell with coordinates $(i, j)$ as $u(i)e_1 + v(j)e_2$, where $u(i)$ denotes the $i$ th element of the sequence $u$.

## III. SELF-ORGANIZING MAPS

Let $X$ be a given data matrix consisting of $k$ instances with dimension $d$, thus $X \in \Re^{k \times d}$. Also let $X[a:b]$ be a submatrix formed by the rows of $X$ from the $a$ th row to $b$ th row. Our task is to map such instances to a SOM, a grid of cells (units) organized as a 2D map. Here, each cell is associated with a prototype vector (weight vector) of the dimension $d$, i.e. that of the original input space. Training a SOM means learning these vectors, and after the training is complete, each instance is matched to a cell with the prototype vector closest to it. In the literature, these cells are called as Best Matching Units (BMU).

Conventionally, the prototype vectors are randomly initialized and then iteratively updated. In each iteration, an instance is picked randomly and its BMU is calculated. Based on the BMU, all prototype vectors are updated. For the update of the $j$ th prototype vector at iteration $(t+1)$ is as follows:

$$w_j(t+1) = w_j(t) + \alpha(t) h_{jc(x)}(x - w_j(t)), \tag{1}$$

where $\alpha(t)$ denotes the learning rate and $h_{jc(x)}$ the magnitude of the closeness between the $j$ th prototype vector and $c(x)$, the BMU of the randomly picked instance $x$. In particular, $\alpha$ is chosen as a decreasing function of time, and typically specified as:

$$\alpha(t) = \alpha_0 \exp(-t \times \eta) \tag{2}$$

where $\alpha_0$ is the initial learning rate and $\eta > 0$ is the decay rate specified by the user. For $h_{jc(x)}$, it is typically to consider the Gaussian neighborhood function:

$$h_{jc(x)} = \exp\left( \frac{-\left\| r_j - r_{c(x)} \right\|}{2\sigma(t)^2} \right) \tag{3}$$

where $r_j$ and $r_{c(x)}$ are the coordinates of the $j$ -th prototype vector and the BMU respectively. Here $\sigma(t)$ corresponds to the width of the neighborhood and decreases over time, and the decrease can be calculated using (2) ensuring that $\alpha_0$ is replaced by $\sigma_0$, the initial value for width. The rationale behind decreasing $\alpha$ and $\sigma$ over time is to is to reduce the effect of updating of prototype vectors over time so that the learning process converges. We also note that the underlying idea of the update rule given (1) is to draw the prototype vectors in proportion to their closeness to the instance of interest.

## IV. PROPOSED METHOD

As mentioned earlier, the initial prototype vectors, i.e. $w_j(0)$, $(j \in \{1,..., m \times n\})$, are determined randomly, which leads to instability and slow convergence when learning the prototype vectors. To address this problem, we propose the following method based on Borda Count (BC). We thus name this method as BCI.

Our method consists of three phases. In the first phase, we rank the cells of SOM using BC. Specifically, for a SOM with $m \times n$ cells, we construct a ranking vector for each cell which is a permutation of integers from 1 to $m \times n$. For the $i$ -th cell, this vector is denoted as $p_i^{cell}$ whose $l$ -th element is the rank of $l$ -th cell for $i$ -th cell, where the ranking is based on the closeness to the $i$ -th cell. In the case of a tie, that is, when more than one cell is equidistant from the cell of interest, each cell is assigned the average rank. For example, for a SOM of size $3 \times 3$, as shown in Figure 2, the ranking vector for the first cell is $p_1^{cell} = (1, 2.5, 5.5, 2.5, 4, 7.5, 5.5, 7.5, 9)$.

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

Figure 2: A 3x3 SOM with indices

BC receives these ranking vectors from all cells as input and then outputs the final ranking by summing all vectors and sorting the cells based on this sum. For the example above, the final ranking of the SOM cells is $cell\_rank = (5,2,4,6,8,1,3,7,9)$. This result reflects the original layout of the SOM, in fact the fifth cell is the most central cell and therefore comes first in the final ranking. Also, the cells with indices 1, 3, 7, and 9 are located in the corners of the SOM hence they share the last places in the ranking.

In the second phase of the proposed method, we instead rank the instances, following much the same procedure used for the cells. We thus omit the details of ranking instances with BC. As an example, if we rank the G7 countries with BC based on the data matrix in Table 1, then the final ranking is France, Canada, Italy, UK, Germany, Japan and USA. We shall denote the rank associated with the instances as $inst\_rank$.

In the final step, we assign the instances to the cells based on the (final) ranking of the cells and the instances obtained in the previous two phases. For the G7 data, this assignment is done as follows:

| Ranking of the instances | | Ranking of the SOM cells |
|---|---|---|
| France | → | 5 |
| Canada | → | 2 |
| Italy | → | 4 |
| UK | → | 6 |
| Germany | → | 8 |
| Japan | → | 1 |
| USA | → | 3 |
| | | 7 |
| | | 9 |

Finally, the initial prototype vector of a cell is considered to be the instance assigned to that cell. For example, the initial prototype vector of the fifth cell is [65.4, 83,323.6, 81.4, 52721] data vector corresponding to France. If the number of instances is less than the number of cells, as in the above example, some cells are not matched with any instances. If this is the case, we can initialize the prototype vectors corresponding to these cells randomly. Normally, however, the number of instances is almost always greater than the number of cells. Below we explain how to use the proposed initialization method if this is the case.

Suppose we have $k$ instances and $m \times n$ cells, where $k > m \times n$. In this case we rank the instances and the cells using BC, as described above, and then split the instances into $k/(m \times n)$ chunks, starting with the first instance in the ranked list. Each chunk of the instance is then assigned to a cell, starting with the first cell in the ranked list of cells. Finally, the mean of the instances in the same chunk is considered the initial prototype vector for the cell to which the chunk is assigned.

Here, we provide a pseudocode for our proposed initialization method for SOM. We also note that the notation $\lceil\ \rceil$ acts like a ceil operator, returning the smallest integer greater than its parameter.

```
Algorithm 1: BCI: A Borda Count Based Initialization Method for Self-Organizing Maps
Input: Data matrix: $X \in \Re^{k \times d}$, SOM with size $m \times n$
Output: Initial values for the prototype vectors of the SOM: $w_i(0) \in \Re^d \ (i \in \{1,..., m \times n\})$
Phase -1: Rank the cells
1: for i =1: $m \times n$ do
2:     calculate $p_i^{cell}$
3: end for
4: Using $p_i^{cell}$, rank the cells using BC, get $cell\_rank$
Phase -2: Rank the instances
5: for i =1: $k$ do
6:     calculate $p_i^{inst}$
7: end for
8: Using $p_i^{inst}$ rank the instances using BC, get $inst\_rank$
9: Based on $inst\_rank$ replace (shuffle) the rows of $X$
Phase -3: The initialization
11: $c = \lceil k / m \times n \rceil$          (chunk size)
10: $t = 1$
11: for i in $cell\_rank$:
12:     $w_i(0) = mean(X[(t-1) \times c + 1 : t \times c])$
13      $t += 1$
```

## V. EXPERIMENTS

In this section we present the results of a range of experiments to verify the efficiency of the proposed method. In this sense, we first give the statistics for the datasets used, then explain the evaluation criterion, and finally provide the results and the related discussion.

### A. *The Datasets Used*

We conducted experiments using different types of UCI datasets. The statistics for these datasets can be found in Table 2.

Table 2: Statistics of the Used UCI Datasets

| Dataset | Number of instances | Number of features |
|---|---|---|
| Abalone | 4177 | 8 |
| Breast Cancer | 569 | 32 |
| Ecoli | 336 | 7 |
| Glass | 214 | 9 |
| Iris | 150 | 4 |
| Yeast | 1484 | 8 |
| Zoo | 101 | 17 |

### B. *EVALUATION CRITERION*

Conceptually, Self-Organizing Maps are examples of unsupervised learning, where there is no actual truth, i.e., classes. Therefore, we cannot simply compare the predictions and the actual truths to evaluate the performance of the proposed method. Instead, to assess the quality of a SOM, the quantization error (QE) is often used. QE measures how well the SOM reflects the original feature space, which is basically the average of the distances between instances and the prototype vectors of the cells to which they are assigned. This is calculated as:

$$QE = \frac{1}{k} \sum_{i=1}^{k} \| X_i - m(X_i) \|_2 \qquad (4)$$

where $X_i$ denotes the $i$-th instance of , i.e., the $i$-th row and $m(X_i)$ is the prototype vector of its BMU in the SOM.

### C. *RESULTS*

We compare the results of QE, obtained with the proposed BCI, with those obtained with random initialization and PCI. When it comes to determine size of the SOM for each dataset of interest, we set it to $10 \times 10$, resulting in a SOM with 100 cells, which is less than the number of instances in all the datasets used. Table 3 shows the QE results. The best results are highlighted for each dataset. We also note that the QE values obtained with the random initialization are the average of 100 runs for each dataset.

Table 3: The QE Results Obtained with the Random Initialization, PCI and BCI

| Dataset | Random Init. | PCI [4] | BCI (Ours) |
|---|---|---|---|
| Abalone | 0.47 | **0.24** | 0.27 |
| Breast Cancer | 1.79 | 0.52 | **0.47** |

| | | | |
|---|---|---|---|
| Ecoli | 0.47 | 0.36 | **0.17** |
| Glass | 0.62 | 0.36 | 0.**25** |
| Iris | 0.24 | 0.31 | **0.04** |
| Yeast | 0.56 | 0.26 | **0.21** |
| Zoo | 1.61 | 1.17 | **0.19** |

### *D. DISCUSSIONS*

The immediate conclusion that can be drawn from Table 3 is that one should not randomly initialize the prototype vectors in a SOM. The difference between the QE values from the random initialization and from PCI and BCI is significant. In fact, for Zoo dataset the QE value is about 8 times higher than the value obtained with the proposed method BCI. This means that random initialization of prototype vectors in SOM is risky and can lead to slow convergence, since it does not yield ideal initial values.

Comparing the state-of-the-art initialization method, PCI, and the proposed BCI, it is evident that BCI outperforms PCI all datasets except Abalone. The difference between the QE scores ranges from 0.05 to 0.98 in favor of BCI. Relying on these results, we conclude that the proposed method BCI is a powerful initialization method for prototype vectors of SOM and can be used for datasets with different number of instances and different number of features.

## VI. CONCLUSION

Data visualization has long been of interest not only to data scientists, but also to researchers in other fields, because it allows for obtaining useful insights from the data at hand. In this sense, a range of data visualization methods have been developed. Of them, Self-Organizing Maps (SOM) is one of the most commonly used methods due to its simplicity and efficiency. However, the conventional use of SOM suggests initializing the prototype vectors randomly, which may ultimately lead to instability and low convergence.

In this study, we propose a novel method for initializing prototype vectors based on Borda Count (BC). Thus, we call the proposed method BCI. Basically, BCI first ranks cells of SOM and instances separately with BC and then assigns the instances to the cells based on these two rankings. Finally, BCI calculates the mean of the instances assigned to the same cell and then considers this mean value as the initial value for the prototype vector of the cell. The experiments conducted on seven different UCI datasets confirm the efficiency of BCI and show that BCI outperforms both the random initialization and PCI, the state-of-the-art initialization method based on PCA. Future research will focus on the application of PCI to different types of SOM, such as growing SOM (GSOM).

## REFERENCES

[1] M. Sadiku, A. Shadare, S. M. Musa, C.M. Akujuobi and R. Perry, R. "Data visualization". International Journal of Engineering Research And Advanced Technology (IJERAT), vol: 2(12) pp. 11-16, 2016.

[2] T. Kohonen. "The self-organizing map". *Proceedings of the IEEE*, vol: 78(9), pp: 1464-1480, 1990.

[3] T. Kohonen. "Essentials of the self-organizing map." *Neural networks*, vol: 37, pp: 52-65, 2013.

[4] M. Cottrell, M. Olteanu, F. Rossi and N. Vialaneix. "Theoretical and applied aspects of the self-organizing maps". WSOM 2016.

[5] A. Akinduko, A. E. Mirkes. and A. N. Gorban. "SOM: Stochastic initialization versus principal components". *Information Sciences*, vol: 364, pp: 213-221, 2016.

[6] P. Emerson. "The original Borda count and partial voting". *Social Choice and Welfare*, vol:40(2), pp:353-358.2013.

[7] K. Healy. "Data visualization: a practical introduction" *Princeton University Press*. 2018.

[8] S. Arora, W. Hu and P.K. Kothari. "An analysis of the t-sne algorithm for data visualization". *In Conference On Learning Theory*, pp. 1455-1462, PMLR, 2018.

[9] D. Alahakoon, S. K. Halgamuge, and B. Srinivasan. "Dynamic self-organizing maps with controlled growth for knowledge discovery". *IEEE Transactions on neural networks*, vol: 11(3), pp: 601-614, 2000.

[10] T. Kohonen. "Things you haven't heard about the Self-Organizing Map" *In IEEE international conference on neural networks*. pp:1147- 1156. 1993.