ESCOLA NAVAL

talanto e bio fazer

Edson Giovanni Gonçalves da Cunha Moreira Bastos

Study of the magnification effect on self-organizing maps

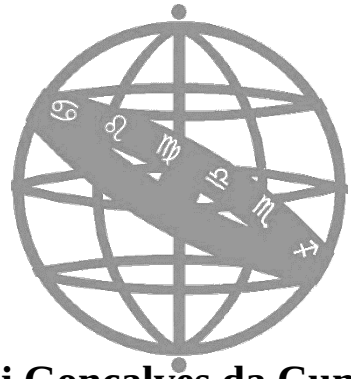
Dissertação para obtenção do Grau de Mestre em Ciências Militares Navais, na especialidade de Engenharia Naval Ramo de Armas e Eletrónica

Alfeite

2021

ESCOLA NAVAL

talanto bien faire



Edson Giovanni Gonçalves da Cunha Moreira Bastos

Study of the magnification effect on self-organizing maps

Dissertação para obtenção do Grau de Mestre em Ciências Militares Navais, na especialidade de Engenharia Naval Ramo de Armas e Eletrónica

Orientação de: Professor Doutor Victor Sousa Lobo

Coorientação de: CFR M Lourenço Gorricha

O Aluno Mestrando,

Moreira Bastos

Edson Bastos

O Orientador,

Victor José A. S. Lobo

Victor Lobo

Alfeite

2021

“Do not go where the path may lead, go instead where there is no path and leave a trail.”

Ralph Waldo Emerson

I dedicate this dissertation to my family and friends who have always supported me in this long journey.

Acknowledgements

First of all, I would like to thank God for giving me the grace of life.

Subsequently, I would like to express my most profound admiration to my master thesis supervisor, Professor Victor Lobo, who spared no effort to help me walking on this beautiful journey. And Lieutenant Commander Lourenço Gorricha for all the attention and help given to my person whenever I needed it.

I want to thank my father, Professor Moreira Bastos, and my mother, Silvana da Cunha, for all the dedication and knowledge transferred from my childhood to adulthood that has contributed to becoming what I am. My thanks to my siblings, Eliana, Érica and Pedro. Thank you to my dear girlfriend Elizângela Kafina for all love and support.

I also want to express my gratitude to all my classmates, especially to the Class of Naval Engineers – Weapons and Electronics, for the friendship, camaraderie, and spirit of mutual help that we nurtured during the five years in the Naval School. Without forgetting my friends from Angola.

I want to thank all my classmates from the Portuguese-speaking African countries community for all the good times we have spent together throughout the course. Furthermore, I would like to thank the Portuguese Army's Coronel Bruno Brito and my English teacher Isabel Caetano at the Portuguese Army's Academy and all the other professors. For treating me like a son, showing the right path to follow whenever they could, and despite being far away, always keeping an eye on my school achievement and giving me advice on diverse matters.

Thank you to all the professors at the Portuguese Naval School, especially to the Department of Science and Technology, for the knowledge given to me and for having prepared me to have a self-critical spirit that was undoubtedly extremely useful to me during school and my personal life. And thank you to everyone who, at some point in my life, helped me become a better person. This thesis' is yours too!

Thank you very much!

Abstract

Self-Organizing Maps (SOM), are a type of neuronal network (Kohonen, 1982b) that has been used mainly in data clustering problems, using unsupervised learning. Among the multiple areas of application, SOM has been used in various problems of direct interest to the Navy (V. J. Lobo, 2009), including route planning and the location of critical infrastructures. The SOM has also been used to sample large databases. In this sort of application, they have a behaviour called the magnification effect (Bauer & Der, 1996), which causes areas of the attribute space of data with less density to be over-represented or magnified.

This dissertation uses an experimental approach to mitigate the lack of theoretical explanation for this effect except for one-dimensional and quite simple cases. From experimental evidence obtained for carefully designed problems we infer a relationship between input data densities and output neuron densities that can be applied universally, or at least in a broad set of situations. A large number of experiments were conducted using one-dimensional to one-dimensional mappings followed by 2D to 2D, 3D to 1, 2 and 3D. We derived an empirical relationship whereby the density in the output space d_{out} is equal to a constant times the density of the input space d_{in} raised to the power of α (alpha) which although depending on a number of factors can be approximated by the root index *n* of 2/3 where *n* is the input space dimension, $d_{out} = K * d_{in}^{\sqrt[n]{\frac{2}{3}}}$.

The correlation that we found in our experiments, for both the well-known 1-dimensional case and for more general 2 to 3-dimensional cases is a useful guide to predict the magnification effect in practical situations.Therefore, in chapter 4 we produce a populational cartogram of Angola and we prove that our relation can be used to correct the magnification effect on 2-dimensional cases.

Keywords: Self-organized maps, neural networks, magnification effect, data science.

Resumo

Os mapas auto-organizados ou SOM (*Self Organizing Maps*), são um tipo de rede neuronal (Kohonen, 1982) que tem sido utilizada sobretudo em problemas agrupamento de dados (*clustering*), usando aprendizagem não supervisionada. Entre as múltiplas áreas de aplicação, os SOM têm sido usados em vários problemas com interesse direto para a Marinha (Lobo, 2009), incluindo o planeamento de rotas e a localização de infraestruturas críticas. Os SOM também têm sido usados para fazer amostragem de grandes bases de dados, e nesse tipo de aplicações têm um comportamento, denominado efeito de magnificação (Bauer & R. Der, 1996), que faz com que zonas do espaço de atributos dos dados com menor densidade sejam sobre representadas, ou seja magnificadas.

Esta dissertação traz uma abordagem experimental para mitigar a falta de explicação teórica para este efeito, com exceção de casos unidimensionais e bastante simples. A partir de provas experimentais obtidas para problemas cuidadosamente concebidos, inferimos uma relação entre densidades de dados de entrada e densidades de neuronios à saída que podem ser aplicadas universalmente, ou pelo menos num conjunto alargado de situações. Foram realizadas um grande numero de experiências usando mapeamentos unidimensionais para mapeamentos unidimensionais seguidos por 2D para 2D, 3D para 1, 2 e 3D. Derivamos uma relação empírica em que a densidade no espaço de saída d_{out} é igual a uma constante vezes a densidade do espaço de entrada d_{in} elevada a α (alpha) que, embora dependendo de uma série de fatores, pode ser aproximado pela raiz de índice *n* de 2/3 onde *n* é a dimensão do espaço de entrada, $d_{out} = K * d_{in}\,\sqrt[n]{\frac{2}{3}}$.

A correlação que encontramos nas nossas experiências, tanto para o caso unidimensional bem como para casos mais gerais de 2 a 3 dimensões é um guia útil para prever o efeito de magnificação em situações práticas. No capítulo 4 produzimos um cartograma populacional de Angola e provamos que a nossa relação pode ser usada para corrigir o efeito de magnificação em casos bidimensionais.

Palavras-chave: Mapas auto-organizados, Redes neuronais, Efeito de magnificação, Ciência de dados.

Contents

List of Figures

List of Tables

Glossary

SOM **S**elf-**O**rganizing **M**ap

ANN **A**rtificial **N**eural **N**etwork

AI **A**rtificial **I**ntelligence

BMU **B**est **M**atching **U**nit

PDF **P**robability **D**ensity **F**unction

VQ **V**ector **Q**uantization

IBM **I**nternational **B**usiness **M**achines Corporation

INE **I**nstituto **N**acional de **E**statística

IOGP **I**nternational Association of **O**il and **G**as **P**roducers

GIS **G**eographic **I**nformation **S**ystem

Chapter 1

1. Introduction

A Self-Organising Map (SOM) is an algorithm or mathematical tool created by Professor Teuvo Kohonen and first presented in 1982 (Kohonen, 1982b). It is generally used in applications such as cluster detection and defining partitions in feature spaces, where it can be a faithful substitute for *k*-means. We can also use it to visualise multidimensional data, which is particularly useful when you have many data and dimensions (*e.g.*, (Gorricha, 2015)).

SOM was conceived from an analogy with the way a human cerebral cortex works. In the beginning of the '80s, researchers discovered that this part of the brain selects specific regions for specific activities. For a given brain activation, the degree of activation of the neurons decreased as the distance from the initial activation region increased (Kohonen, 2001). After introducing SOM, specialists from different areas such as computer science, neurobiology, physics, geography, among others, began to adapt SOM in relevant applications (*e.g.*, (Bação, Lobo, & Painho, 2008)).

SOM represents the fruit of a lot of work and research done by Professor Kohonen (Kohonen, 1974, 1982a, 1982b) in the '80s, and his main inspiration was neurobiological systems. SOM is a type of neuronal network with several computational devices, called *units*, or neurons, arranged in a hexagonal or rectangular grid shape. Each of these units receives input data and does specific data processing, producing an output value. It has an *N*-dimensional input space, *i.e.*, the original space of the data, which is the same space defined by the synaptic weights of the neurons. We can adjust the training algorithm according to some parameters, then apply to the input data, in the training phase, and get information in the output layer. SOM uses unsupervised learning, as we are not looking to classify data. But typically, we want to look at it and understand

what type of distribution it has or visualise it in a two-dimensional space (although other visualization can be made) (e.g.,(Gorricha, 2009)).

1.1. Motivation

Over the past few years, neuronal networks have been used to perform various tasks, because they:

- Can extract the meaning of complex or even inaccurate data.(Calitoiu, Oommen, Nussbaum, & Cybernetics, 2007)
- Model or detect patterns that are too complex for humans or conventional methods of computational analysis.(Krakovsky & Forgac, 2011)
- Have adaptive learning.
- Their operation can be done in real-time.
- Solve complex mathematical problems.(Li & Li, 2013)
- Perform complex tasks such as stock market prediction, climate behaviour study, etc.(Moghaddam, Moghaddam, Esfandyari, & Science, 2016; Poff, Tokar, Johnson, & Oceanography, 1996)
- Can be used in many different applications, such as facial recognition, voice recognition, route planning, decision-making assistance for unmanned autonomous vehicle systems, or even writing tools. (Khashman, 2009) (Ma'Sum et al., 2013)
- Can be used in a variety of military and naval applications, such as traffic monitoring, outlier detection for cybersecurity intrusion detection, etc. (Choraś & Pawlicki, 2020)

The SOM algorithm is an efficient and easy-to-apply tool that has some advantages over conventional neuronal networks, especially for the following tasks:

- Dimensionality reduction.(Campoy, 2009)
- Multidimensional data visualization.(Nikkilä et al., 2002)
- Cluster detection, outlier detection.
- Data ordering (since it preserves topological relations).

1.2. The SOM Algorithm

The main idea of Kohonen's Self-Organising algorithm is to map data patterns, in the input layer $X \in R^n$ into an N-dimensional grid in the output space. The mapping attempts to preserve the neurons topological relationships, *i.e.*, data points that are nearby at the input will be mapped to nearby units in the output space.

In this thesis, while recognising that the computational units (or neurons) of a SOM are usually referred to as "units", thus stressing that they are computational constructs and not imitations of biological neurons, we will refer to them as "neurons". We choose to do this because when we compute densities and do experimental work, we will use the term "unit" very often, usually referring to "unit length" or "unit area". Thus, we avoid possible confusion by using "neuron" to refer to the SOM units.

The SOM is based on two principles, competitive learning and cooperative learning. On competitive learning, the neuron vector most similar to a data vector is modified to be even more similar to it. This way the map learns the position of the data cloud. On the other hand, cooperative learning means that the most similar neuron vector and its neighbours are moved towards the data vector.

Generally, the neurons are in a grid shape lattice, but they can be configured with a hexagonal shape. First, all neurons are connected to the input pattern and receive information from that layer. Then the algorithm calculates the Euclidean distance between the neurons and the input stimuli. The neuron that is closest to the input stimuli is called the "Best Matching Unit".

After finding the nearest neuron to the input all the neighbour neurons are updated according to a neighbourhood function usually with a gaussian-type distribution centred in the selected neuron.

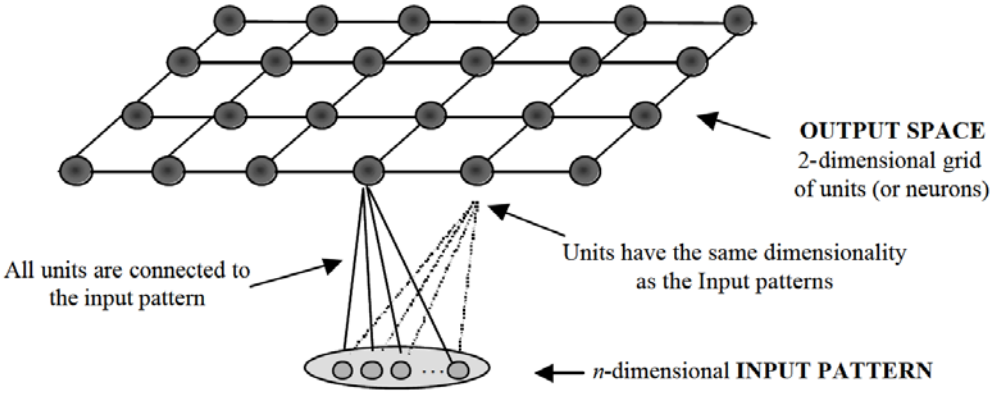


Figure 1 Basic Structure of a Self-Organising Map (SOM) (Lobo, 2002)

The algorithm has three main phases, the calculation phase, the voting phase, and updating phase. For each input signal, the algorithm will,

1. Calculate the distance between the input signal, x_i, X ∈ R^n, and all SOM units r_i.
2. Select the nearest neuron as the winning neuron (or unit) or best matching unit (BMU).
3. Update each neuron according to the update function.
4. Repeat all the steps and update the learning parameters until a stop criterion is met.

1.2.1. Variables

Let the input space be $\boldsymbol{X} \in R^n$; The weight vectors of the neurons $\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots, \boldsymbol{w_M}$ are initialized to random values, where $\boldsymbol{w_i}$ is the weight vector associated to neuron *i* and is a vector of the same dimension as the input. *M* is the total number of neurons and let $\boldsymbol{r_i}$ be the location vector of neuron *i* in the grid. The SOM training algorithm repeats the steps shown in Algorithm 1, where $\eta(v, k, t)$ is the neighbourhood function, and Ω is the set of neuron indexes.

Algorithm 1 Self-Organizing Map algorithm

repeat

1. At each time t, present an input $\mathbf{x}(t)$, and select the winner,

$$\nu(t) = \arg\min_{k \in \Omega} \parallel \mathbf{x}(t) - \mathbf{w}_k(t) \parallel \quad (9)$$

2. Update the weights of the winner and its neighbours,

$$\Delta \mathbf{w}_k(t) = \alpha(t)\eta(\nu, k, t)[\mathbf{x}(t) - \mathbf{w}_\nu(t)] \quad (10)$$

until the map converges

Figure 2 The SOM Algorithm (Yin, 2008)

1.3. Why study the magnification effect?

Like all Artificial Intelligence tools, self-organised maps are helpful. For example, SOM is often used when analysing large volumes of multidimensional data (what we call "Big Data"). Thus, all the progress made in understanding these tools contributes directly to improvements in society.

Studying the magnification effect is essential to understand how we can control it in the general case (*i.e.*, with any dimension of the input and output space and any data distribution). The magnification effect is a problem that influences the reliability of SOM essentially for the multidimensional case. Given that the world has seen exponential growth in data generation lately, a better understanding of the magnification effect on SOM may lead to better future implementations of this tool.

1.4. Purpose of the dissertation

The study of the magnification effect is a task that several authors have long done. It has been possible to obtain good results for the one-dimensional case. Until now, it has been concluded that the density of neurons in the output layer is proportional to the density of the data in the input layer raised to 2/3 (Ritter & Schulten, 1986).

The purpose of the dissertation will be the formulation, of a mathematical law that can help to describe with reasonable precision the variation of density in the entrance space and the exit space of SOM, as well as a consolidation of theoretical concepts acquired during the naval engineering course – Weapons and Electronics.

Based on the study's success, it is expected to be able to contribute to improving the quality of future SOM implementations, which is an advance for science and for its use in matters relevant to the Navy.

1.5. Structure of the dissertation

The organization of the document seeks to reflect all phases of the development of the work carried out. After the introduction, an analysis of the State of the Art (chapter 2) studies focused on Self-Organized Maps are elaborated, mainly on the magnification effect. In this chapter, we still define concepts relating to Kohonen maps.

We explain the design of experiments in chapter 3 by how the MATLAB experiments were carried out, based on the SOM toolbox, made available with a free-use license by the University of Helsinki. Later in chapter 4, we use SOM to produce a cartogram and prove the occurrence of the magnification effect. Consequently, we compensate it basing on the results of the investigation done in chapter 3.

Finally, in chapter 5, we present the conclusions of the project and suggestions for future studies in the area and a future implementation.

Chapter 2

2. State of the Art

In his articles, Teuvo Kohonen first presented the Self-Organising Maps in 1981 (Kohonen, 1981) (Kohonen, 1982). The self-organising feature of this unsupervised learning algorithm drew the attention of many researchers, who began to support its development. This neuronal network model could recognize patterns in the data without suffering external influences from potentially biased agents. Various researchers have contributed to the advance of SOM, with several studies being done on the algorithm, solving many of its problems, and improving the basic algorithm. However, there are still some unsolved problems and cross-cutting issues not only to SOM but also to various other types of similar neuronal network models.

2.1. SOM Applications

Since its inception, SOM has shown encouraging results in various applications, from route planning and study of critical infrastructures to Blockchain monitoring (Lobo, 2009), (Chawathe, 2018). It can also be used in data mining and data compression (*e.g.*, (Ong & Abidi, 1999)). In addition to the applications mentioned above, the SOM can also be used for numerous other purposes (V. Lobo, 2009) (Affonso, 2011).

An example of the application of Kohonen's self-organized maps, illustrated in the figure below, is the analysis of poverty in the world, using as input socioeconomic indicators provided by the United Nations (Kaski, 1997).

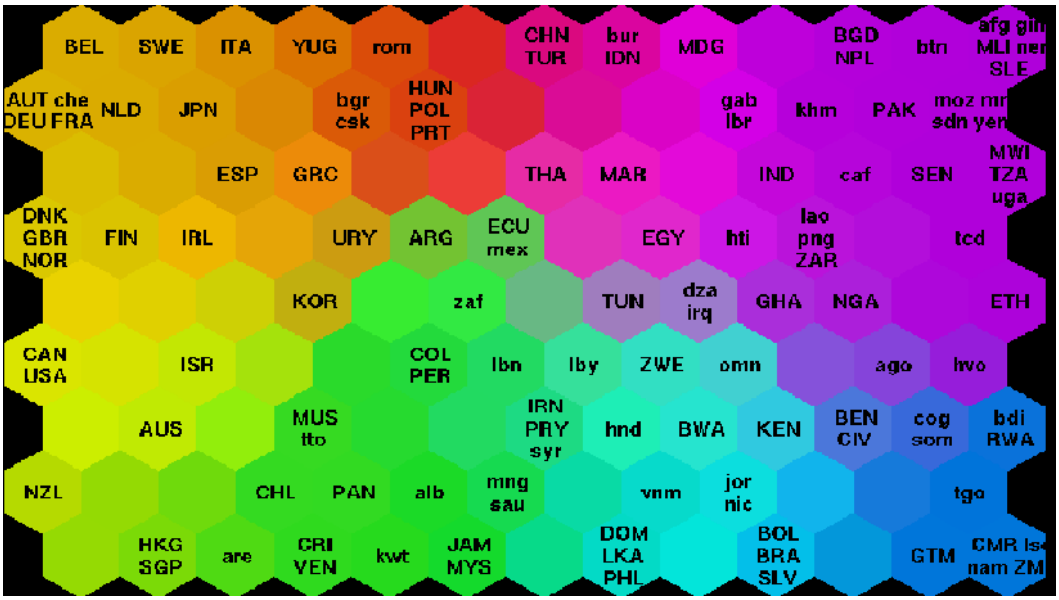


Figure 3 World Poverty Map (Kaski, 1997)

In the figure above we can see a SOM mapping of different countries (identified by a three-letter county code) where the relative position and color of each country depends on correlations in statistical data of those countries. The data consisted of World Bank statistics of countries in 1992. Overall, 39 indicators describing various quality-of-life factors, such as state of health, nutrition, etc., were used.

“Countries that had similar values of the indicators found a place near each other on the map. The different clusters on the map were automatically encoded with different bright colors, so that colors change smoothly on the map display.” (Kaski,1997)

As a result of this process, each country was in fact automatically assigned a color describing its poverty type in relation to other countries.

“The poverty structures of the world can then be visualized in a straightforward manner: each country on the geographic map has been colored according to its poverty type.” (Kaski,1997)

Another example of the application of SOM is depicted in the figure below, where a SOM was used to analyze seismic data (Roden, Smith, & Sacrey, 2015).

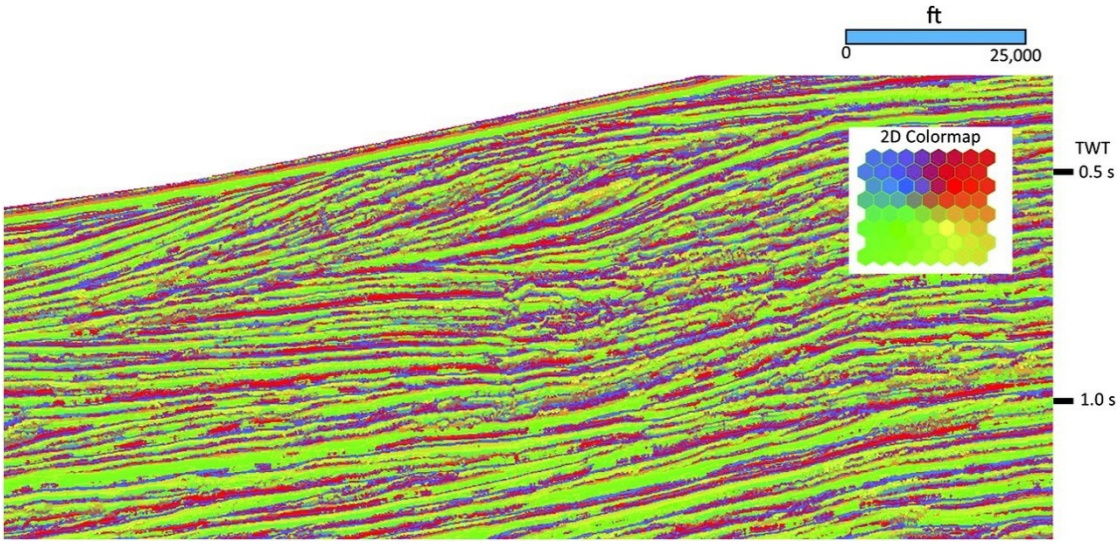


Figure 4 Offshore West Africa 2D seismic line processed by SOM analysis (Roden et al., 2015)

In the figure, each neuron is shown as a unique colour in the 2D colour map. After training, each multi-attribute seismic sample was classified by finding the neuron closest to the sample using the Euclidean distance. The colour of the neuron relays on the seismic sample in the display. According to the authors of the research, SOM neurons yielded a detailed geologic classification.

2.1.1. Studies on the Magnification Effect of SOM

It is a well-known fact that when SOM maps input data, the density of the SOM neurons in the output space is not proportional to the density of the input patterns. Thus, while areas with higher input data density lead to higher density of SOM neurons, the neurons tend to underrepresent high-density areas and overrepresent lower density areas. This is called the "Magnification Effect". In some cases, it is a desirable characteristic, while it is a drawback in other cases.

(Ritter & Schulten, 1986) was the first relevant study about the magnification effect on Kohonen's Self-Organizing Maps. Ritter and Schulten found that the density of neurons is proportional to a power of the input density and that power is a value lower than 1, as seen in the equation below. This power is known as the magnification factor, and Ritter determined that under certain theoretical conditions, that magnification factor α (alpha) is equal to $\frac{2}{3}$.

$$Output\ density \propto Input\ density^{\alpha}$$ Equation 1

It was believed to be an excellent one-dimensional approximation for the relationship between the density of input data and the density of neurons or output data. This statement was supported by an analytical study of the maps stationary state equation, for one dimension, and by a simple simulation.

However, Ritter and Schulten found no general local expression regarding the probability density for the two-dimensional case (Ritter & Schulten, 1986).

Later, the analytical results presented by (Der & Herrmann, 1992) for the "exponents of magnification" were studied in (Bauer & Der, 1996), concluding that these exponents would have to be modified to include the effects of the learning rate control. Bauer & Der intended to control the magnification properties of the self-organized maps, by minimization of the mean squared error, only adjusting the local learning rate, while all other parts of the algorithm remained unchanged. To this end, they took advantage of node-dependent adaptabilities proposed by (Der & Herrmann, 1992) and replaced them in the equation for updating the synaptic weights. The results of this approach were important, bringing a great analytical contribution to the understanding of the magnification effect. However, since they studied a modified Self-Organizing Map which allowed them to control the magnification factor of the output map, they did not contribute to improve the understanding of the magnification effect of the “standard SOM” algorithm.

In 2006, Fort developed a mathematical point of view of SOM’s problems that until then had not been solved (Fort, 2006). Fort studied the magnification effect, but states that the problem could not be completely solved, and concludes his investigation presenting an expression to connect the input and output densities based on combining the results obtained by (Ritter & Schulten, 1986) and (Ritter, 1991).

In 2007, the validity of the self-organizing map (SOM) magnification control scheme introduced by Bauer and Der was analysed (Merényi, Jain, & Villmann, 2007). The study had a specific focus on analysing the magnification effect on data for which Bauer theory does not guarantee success, for instance data that are 2-dimensional. Using Bauer’s method, a magnification factor was found, but it was different from earlier assumptions. Merényi made several experiments and concluded that the magnification factor has an increasing offset on the 2-dimensional SOM.

Chapter 3

3. Design of experiments

In this chapter, we describe the development of the experiments with the help of the SOM toolbox (made available free of charge by the University of Helsinki).

First, we show the hypothesis to be tested and present the fundamental ideas behind each experiment. Lastly, chapter 4 discusses the results of these experiments. For the experiments, we chose MATLAB over other software due to the ease of programming, which prevents from wasting time and effort performing programming tasks that are not directly related to the problem's solution.

3.1. The one-dimensional case

We want to experimentally test the claims (Ritter & Schulten, 1986) that the Kohonen's map output is proportional to the input raised to a power of $\frac{2}{3}$.

We generated different sets of input data with distinct probability distribution functions for the experiments to observe the magnification effect from various perspectives. Thus, from regression, we try to extract an exponent that relates the data input space to the output space of the SOMs.

We performed tests for a uniform distribution and a Heaviside (or step) distribution.

3.1.1. Uniform probability distribution function

In the first experiment, we generated an array of 1000 random values between 0 and 1 with a uniform probability distribution function. Thus, the average density is 1000 points per unit length. We then trained a SOM with 1000 units, using the batch algorithm, a final radius of 0, and 1000 epochs.

The expected average density of the neurons in the SOM will also be 1000 neurons per unit length. In Figure 5, we show the results obtained. This figure shows that slight random variation in the input density generates equivalent but attenuated variations in the output density. There is a subtle "boundary effect" on the edges of the SOM.

Due to the random variations of the uniform distribution, the standard deviation amongst the ten bins of data is 0.2941, while the standard deviation amongst the bins of neurons is 0.2855, which is evidence of the smoothing effect of SOM, even when the number of neurons is equal to the number of points.

Figure 5 Input vs Output distributions with a random uniform distribution of 1000 data points in the interval [0,1].

The SOM also has one thousand neurons, and thus the same average density Figure 5. For comparison, we also generated a rigorously uniform distribution

(*i.e.*, uniformly spaced points) and trained a SOM with those points. As can be seen in the figure, if the input data points are rigorously equally spaced, the SOM units are also similarly spaced, save for the "boundary effect" at the borders of the SOM (Kohonen, 1982b), and a few rounding effects at the bin's edges. We could reduce the "boundary effect" by setting the map shape as toroid or cylinder, as seen in Figure 7. The rounding effects at the bins borders is because if by chance the coordinates of a neuron coincide with the edge, it will be included in the previous (if we use "ceiling") or next (if we use "floor") bin. This rounding effect will produce in certain places an increase or decrease of 1 unit, whatever the density or width of the bins.

Figure 6 Input vs. Output

Figure 7 Neuron map initialized with toroid shape.

In this part, we have experimentally shown that if the data points have a uniform distribution, the neurons will also have an approximately uniform distribution, disturbed only by the “boundary effect” and by minor rounding effects due to the discreet nature of the data. Furthermore, we showed that SOM reflects in the output density of the neurons slight variations made in the input.

3.1.2. Heaviside probability distribution function

In this experiment, we generate an array of *1000* random values in the interval]0,100[with Heaviside[1] distribution (or step distribution), which has a discontinuity in the middle separating two equal-width zones, one with lower and the other with high data density. For the experiment, we used *100* bins. The first *50* bins had a lower density value, and the other bins had a high-density value.

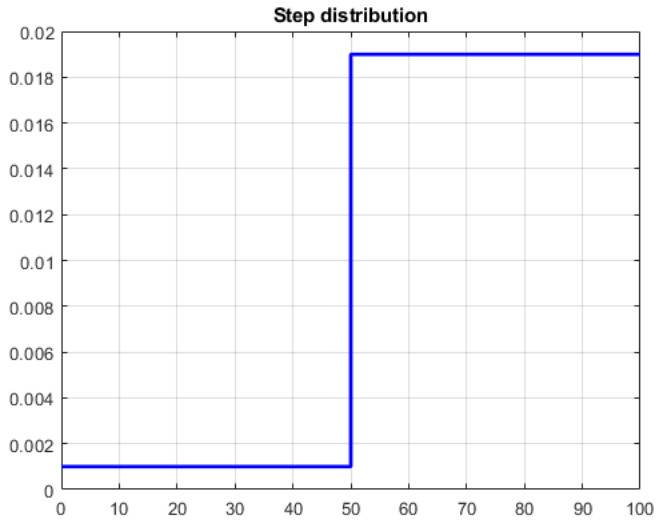


Figure 8 Heaviside or step distribution

The lower density zone has *50* data inputs (*i.e.*, 1 point per unit area), and the high-density one has *950* data inputs *(i.e.*, 19 points per unit area).

The density is expressed in the number of data points in a given length, divided by that length:

$$Density = \frac{Number\ of\ data\ points\ in\ the\ area}{length\ of\ the\ area}$$

The lower density zone has a theoretical value of *1*, and the high-density zone has a value of *19*.

[1] A traditional Heaviside Function has values of 0 in one region and 1 in the other we admit it here in other values.

3.1.2.1. Increasing the training epochs

To see how many training epochs the SOM need to achieve a stable map, we trained several maps from *1* to *1000* training epochs with the number of training epochs unitarily increasing. We expect that SOM approaches its equilibrium state long before reaching one thousand training epochs.

Therefore, the map has linear initialization, *1000* neurons and a Gaussian type of neighbour function with an initial radius of 1/5 of SOM neurons, *i.e.*, on the first iteration, SOM updates 200 neighbour neurons, and that number decreases to *0* at the end of the training epochs. After the experiment, we collected density datasets and used them to find relationships between the two spaces. We show the results on the graphs below,

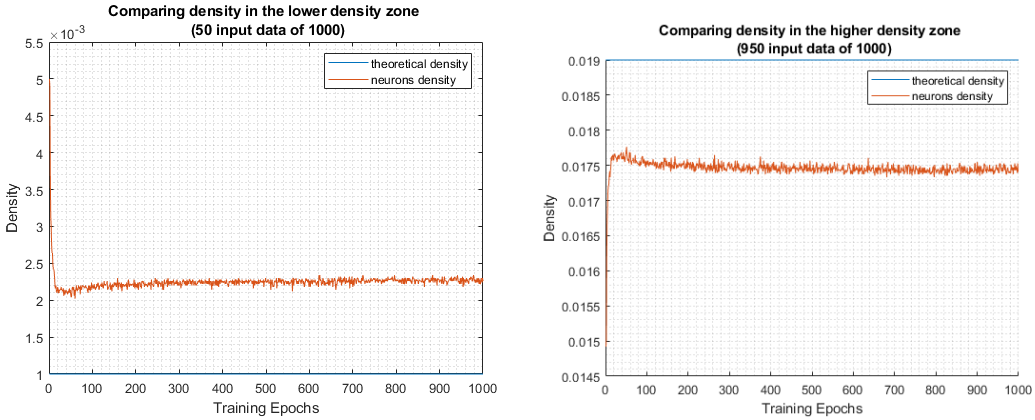


Figure 9 Density of neurons in each of the areas, for different initializations, each with a different (and increasing) number of epochs. In the vertical axis, the density values presented should be multiplied by 1000 to obtain the true value.

The results vary widely for a few epochs, but for more than 100 epochs, the results are stable, and thus we used the average of all tests with more than 100 epochs to compute the final densities. It is utterly reasonable that SOM with fewer training epochs does not have the necessary “time” to correctly map all the input data. The output density presented lower values on the higher density zone and higher on the least dense zone as expected and discussed hereinafter.

3.1.2.2. Variations in data and neuron densities.

To establish relationships between input data density and output neuron density, we analysed the results obtained in the previous subchapter and compared them with those obtained with different densities. In particular, while still using Heaviside distributions, we used input densities of 1 versus 19, then 2 versus 18, and finally 7 versus 13. We show the results obtained for these three experiments in tables 1 to 3.

Table 1 Density of data and neurons in the experiments with a Heaviside distribution of data with 50 points in one area and 950 in the other (1 and 19 data points/unit length). The values presented are averages over 2000 training runs.

	Original Data Density Probability Function (Low density)	**Neurons Density Probability Function (Low density)**	**Original Data Density Probability Function (High density)**	**Neurons Density Probability Function (High density)**
Min	1	1.4	19	17.3
Mean/Std	1	2.0 ± 0.1871	19	17.8 ± 0.1835
Max	1	2.4	19	18.4

Since previous experiments established the convergence properties, for the remaining experiments we trained only 400 SOMs, with training epochs increasing from 100 to 500.

Table 2 Density of data and neurons in the experiments with a Heaviside distribution of data with 100 points in one area and 900 in the other (2 and 18 data points/unit length). The values presented are averages over 400 training runs.

	Original Data Density Probability Function (Low density)	**Neurons Density Probability Function (Low density)**	**Original Data Density Probability Function (High density)**	**Neurons Density Probability Function (High density)**
Min	2	3.3	18	16.1
Mean	2	3.5 ± 0.1574	18	16.3 ± 0.1545
Max	2	3.6	18	16.5

Table 3 Density of data and neurons in the experiments with a Heaviside distribution of data with 350 points in one area and 650 in the other (7 and 13 data points/unit length). The values presented are averages over 400 training runs.

	Original Data Density Probability Function (Low density)	**Neurons Density Probability Function (Low density)**	**Original Data Density Probability Function (High density)**	**Neurons Density Probability Function (High density)**
Min	7	7.8	13	11.9
Mean	7	7.86 ± 0.0712	13	12.1 ± 0.0799
Max	7	7.9	13	12.2

As we can see, the standard deviation is relatively low, and the extreme values are not significantly different. Therefore, we will use only the mean value of density obtained in further calculations.

If Ritter's relation (Ritter & Schulten, 1986) holds, then we would have in all cases:

$$Output\ Density = K * Input\ Density^{\alpha} \qquad \text{Equation 2}$$

If for each case we solve for K and α, we obtain the results presented in table 4, where we can see that Alpha varies between 0.67 and 0.69, which is relatively close to Ritter's value of 0.667.

Table 4 Average magnification constants and exponents for each SOM aforementioned

Experiments with Heaviside Distribution	**INPUT (Low density)**	**INPUT (High density)**	**OUTPUT (Low density)**	**OUTPUT (High density)**	K	α
50 points in one area and 950 in the other	1	19	2	17.8	2.411	0.6759
100 points in one area and 900 in the other	2	18	3.5	16.3	2.336	0.6733
350 points in one area and 650 in the other	7	13	7.86	12.1	2.073	0.6913

To investigate the influence of the difference between the zone's densities of the Heaviside distribution, we calculated the magnification factor and magnification constant from several experiments. First, starting with a step distribution with a ratio of 1/50 in the first zone and 950/50 in the other, bringing the two densities unitarily closer on each run until reaching the limit case, which is equivalent to the uniform distribution, *i.e.*, 499/50 in the first zone and 500/50 in the second. Subsequently, we plotted the relationship between the difference of the two density zones versus the α and versus K,

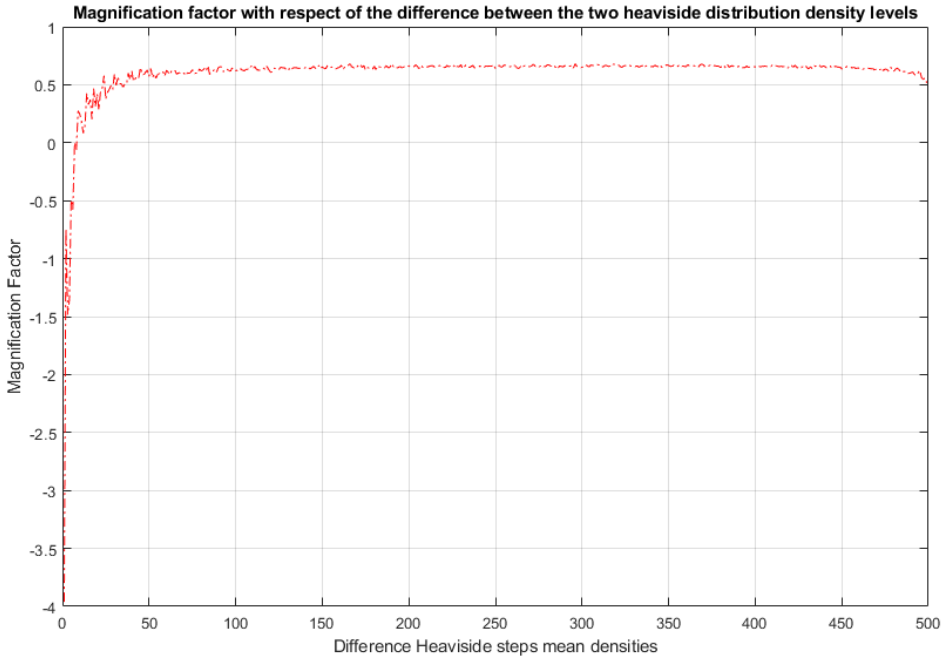


Figure 10 Behaviour of magnification factor for different Heaviside input amplitudes

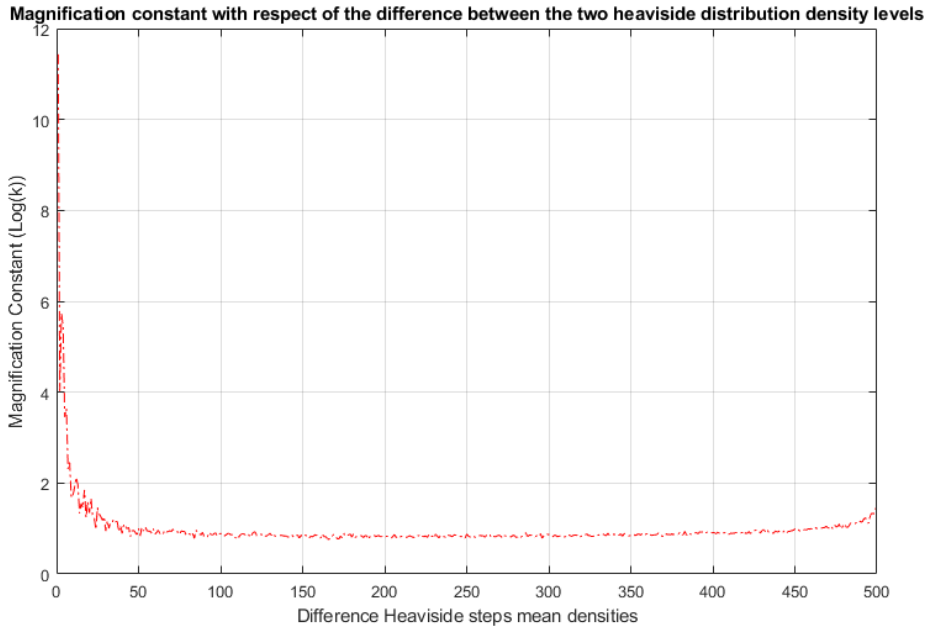


Figure 11 Behaviour of magnification constant for different Heaviside input amplitudes

In the figure above it is important to note that the difference between density zones is represented by the quantity of data in the given length, and we make it unitarily vary on each experiment, also the Y-axis is in logarithmic scale, *i.e., log(K)*, due to extremely high values when the Heaviside is closer to the Uniform distribution.

As we can see, the average Alpha was about 0.6468 ± 0.0229 and the average $K = 2.4353 \pm 0.2374$. These values were obtained not considering the fifty first experiments because the distribution is quite similar to the uniform distribution, and the relationship does not hold. It happens because the neuron density zones tend to flip on the output when the density zones are quite the same, causing very often for the low-density zone to turn onto high-density or vice-versa. It disrupts the equation modelled for the cases when the magnification effect does not turn low-density zones more accentuated than the higher ones on the output and vice-versa.

From these experiments, we conclude that the relation of densities stated in Equation 1 will only produce reliable results when the ratio between the densities is between 1.4 to 17 times higher or lower than the other in comparison.

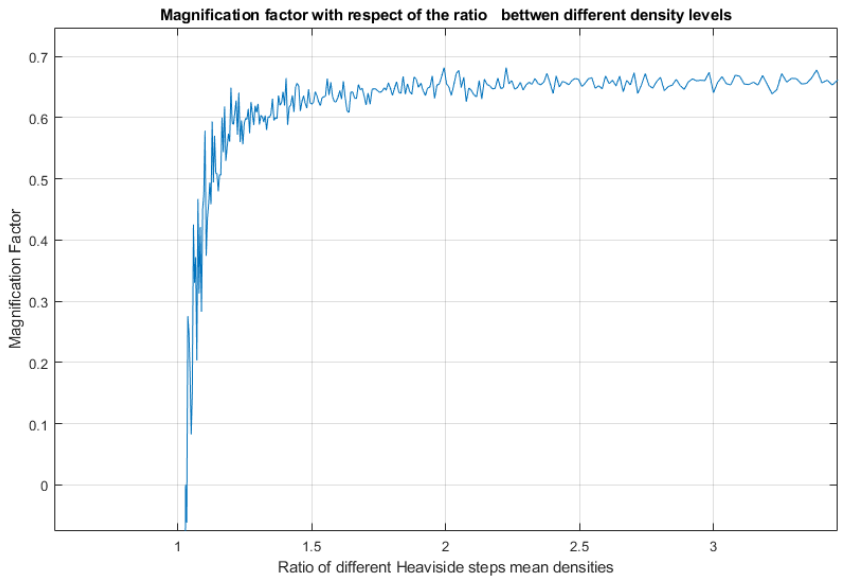


Figure 12 Magnification factor with respect of the ratio between different density levels

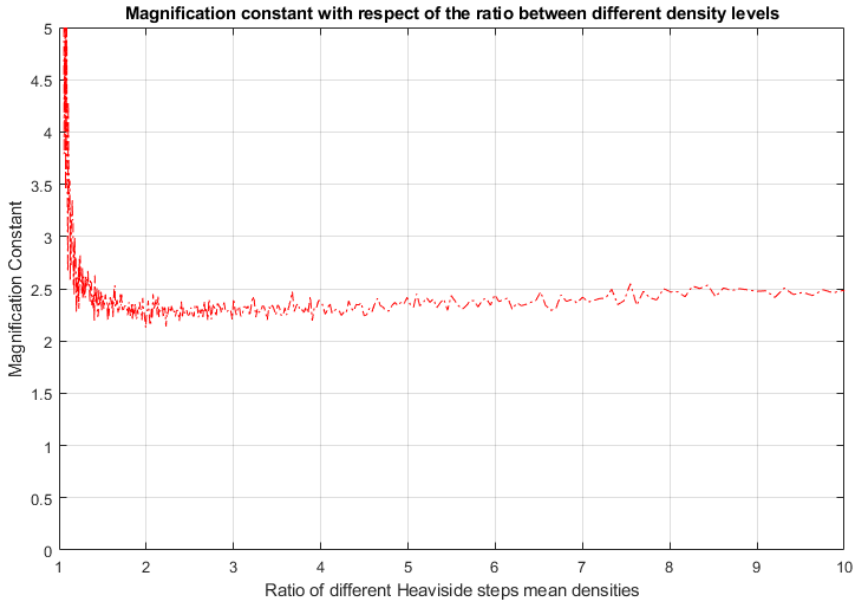


Figure 13 Magnification constant with respect of the ratio between different density levels

Following the investigation, we now calculate the initial radius.

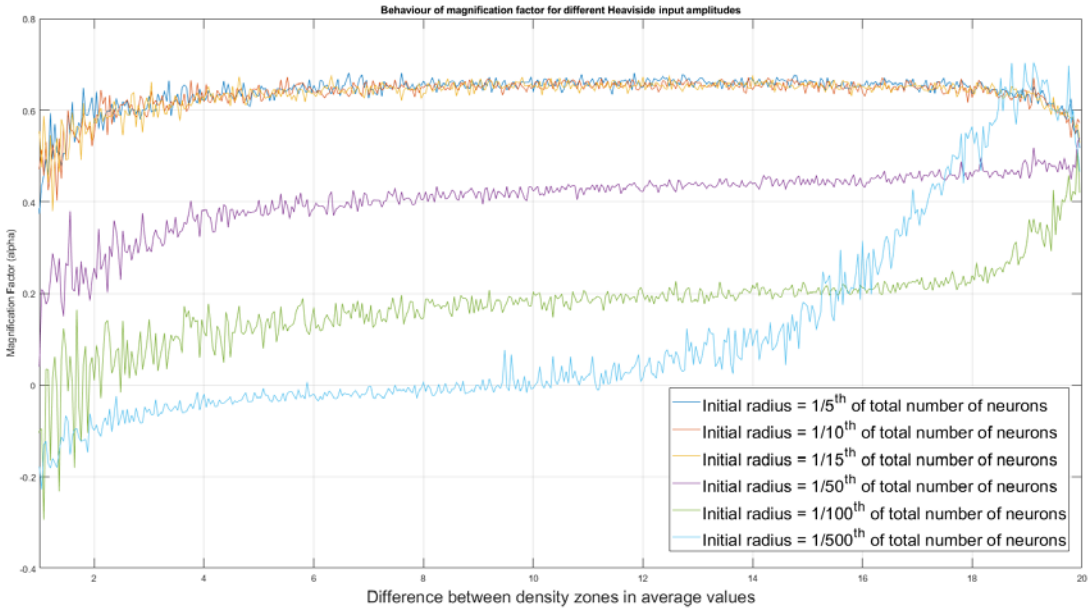


Figure 14 Several experiments each with different initial radius. Analysing the behaviour of magnification factor for different Heaviside input amplitudes.

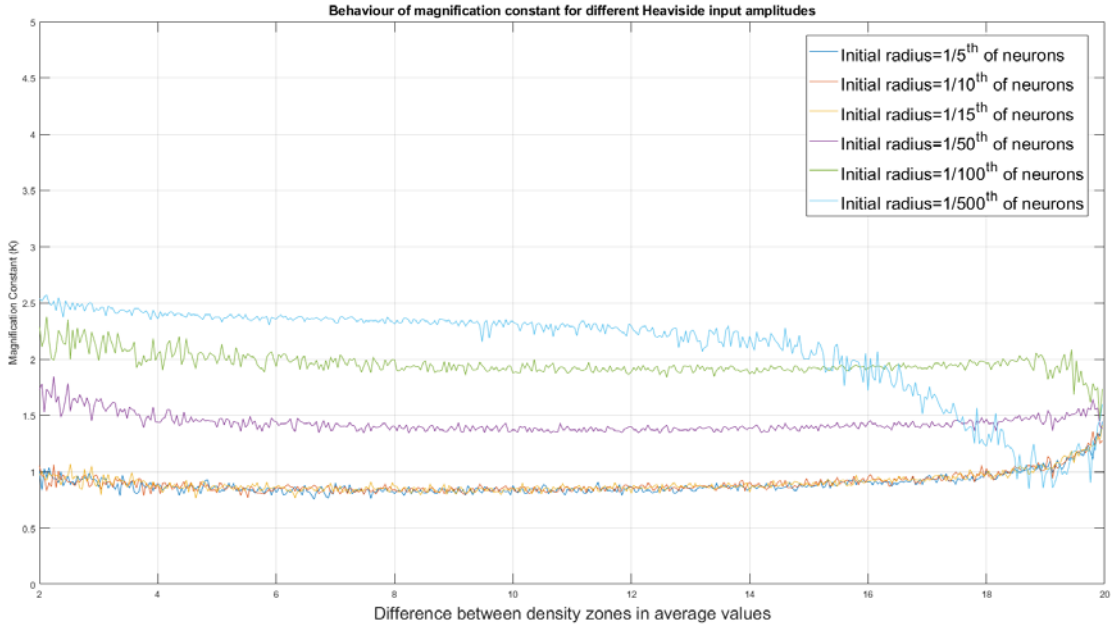


Figure 15 Several experiments each with different initial radius. Behaviour of magnification constant for different Heaviside input amplitudes

From the experiments, we can confirm that α varies along with the initial radius. It is possible to perceive that the more significant the difference between the amplitudes of the zones of the Heaviside distribution, then the more considerable influence the radius will have on the α regardless of the other parameters, and for the cases in which

the initial radius is too small the effect is powerful causing it to converge by force. We suspect that this may occur because when the initial radius is too small and the difference between densities is high on the lower density zone it may do not even have a single neuron or very few. However, we opt to let this for further investigation.

For the extreme case in which the initial radius is equal to 1/500 of the total number of neurons, we have a neighbourhood of only two neurons (in a total of 1000 neurons), so, naturally, SOM cannot converge (or it takes a long time). On the other hand, we verify that α is almost independent of the initial radius when we consider large values, *i.e.*, equal or greater than 1/5 of the total number of neurons.

3.1.2.3. Increasing the neurons

The next experiment was to find a relationship between the number of neurons and the density variation between the input layer and the output fixing all the parameters and varying the total number of neurons. For this experiment, we used an array of 1000 random values in $]0,100[$, linear initialization, an initial radius of 1/5 of neurons, final radius 0 and 1000 epochs. We increased the number of neurons linearly from 1 to 2000 with a step of 1. Then, we computed the density of neurons in both areas of the Heaviside function.

We show the results in the graphs below,

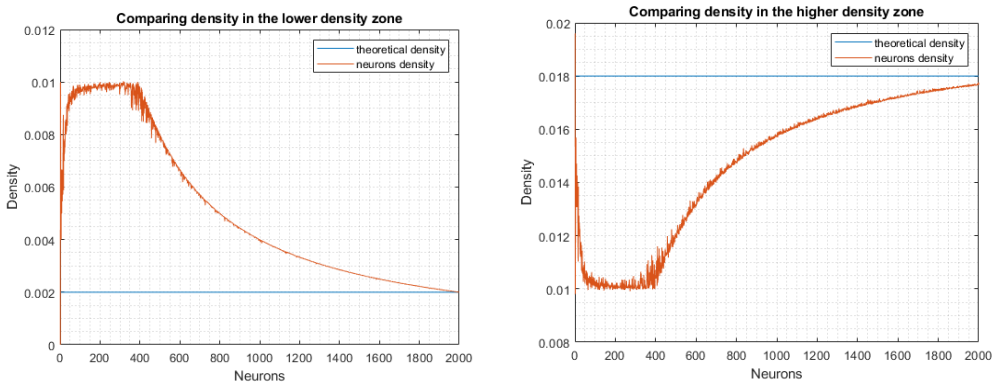


Figure 16 Neuron density (measured in relative frequency) vs. number of neurons, in both areas of the Heaviside function.

From the figures above, it is clear that there are three distinct behaviours:

1) If the number of neurons is less than 100 (10% of the data), the results are unreliable. The “winner takes (almost) all” characteristic of the SOM leads to the higher density area having almost all the neurons.

2) If the number of neurons is more than 100 (10% of the data), but less than 400 (40% of the data), the density of neurons in all input space is more or less uniform, irrespective of the input data density.

3) If the number of neurons is higher than 400 (40% of the data), the neuron density converges rapidly (and exponentially) to the values predicted in our approximations.

Although increasing the number of neurons makes SOM converge to the theoretical density, it does seem that the magnification factor is inversely proportional to the number of neurons. However, we chose not to explore this relationship. Also, the output density function appears to converge exponentially, but a more rigorous analysis would be necessary to confirm this hypothesis.

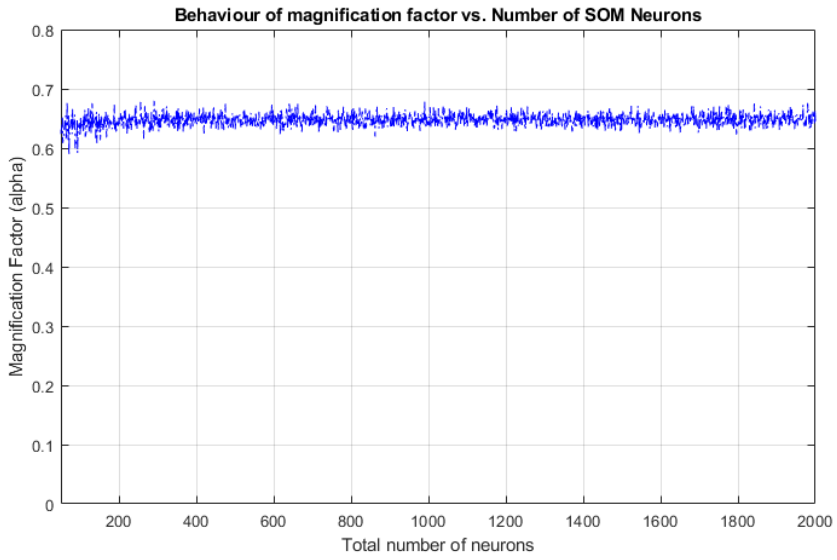


Figure 17 Magnification factor vs. number of SOM Neurons, the magnification factor did not vary too much with the increase of neurons, and since this we can assume that it is independent of the quantity of neurons for values greater than 100.

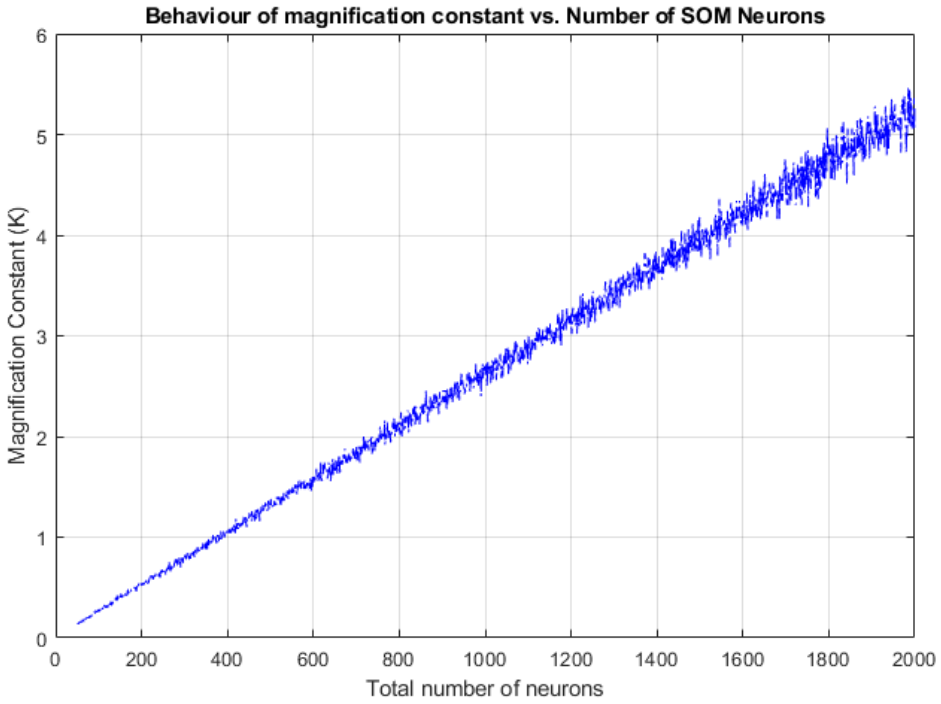


Figure 18 Magnification constant vs. Number of SOM Neurons

From the experiments above, it is possible to verify that the increasing of neurons did not influence the magnification factor. This fact can bring light to a possible non-relation between these two parameters.

Also, we could experimentally verify that *K* and the total number of neurons have a linear relationship, as expected.

$$K = total\ number\ of\ neurons * 0.0026.$$

In this case, 0.0026 stands for the derivative (or the slope) of *K* with respect to the number of neurons. In the figure below, we can see the linear function that best fits the data.

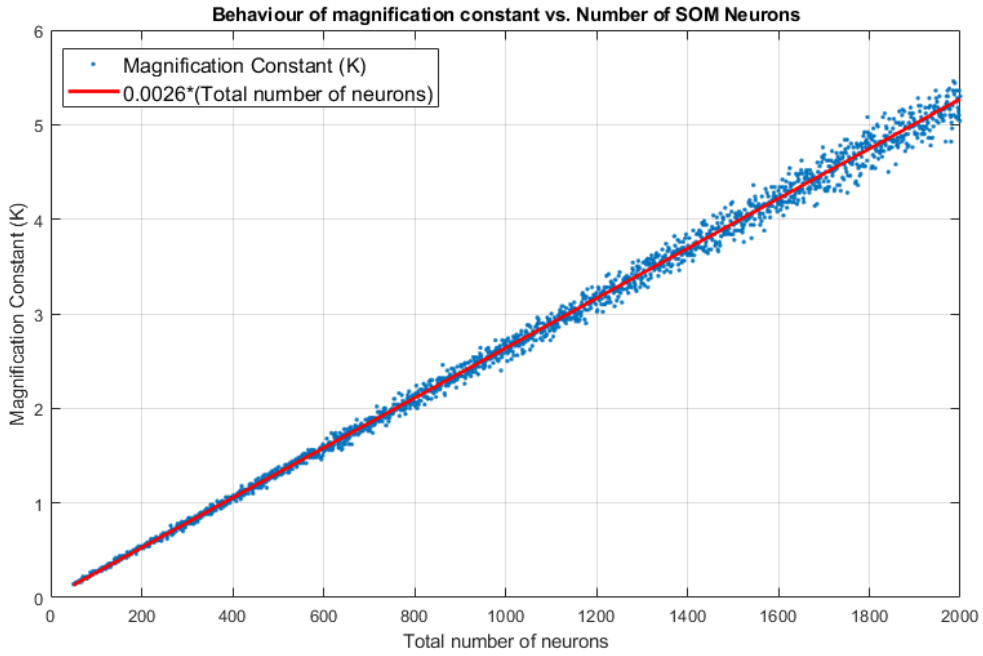


Figure 19 Magnification constant vs. Number of SOM Neurons

We show below, in a table, the alpha and *K* values for the different amounts of neurons.

Table 5 α and K Values

Experiments with Heaviside Distribution	**Average K and α values**	
	K	α
From 50 to 500 neurons	0.726	0.6449
From 500 to 1000 neurons	1.9739	0.6482
From 1000 to 1500 neurons	3.291	0.6483
From 1500 to 2000 neurons	4.6022	0.6485

3.1.3. Multiple Step probability distribution function

Very analogous to the previous subchapter, here we want to estimate the value for the magnification factor, but for the case where there are multiple areas with different densities. It is reasonable to expect that the results do not differ much from those previously obtained. For this, in this experiment, we generate an array of *1000* random values in the interval $]0,100[$ with multiple-step distribution, which has discontinuities separating four equal-width zones, ones with lower and the others with high data density. For the experiment, we used *100* bins; we divide the space into four zones and randomly distribute the lower density values and the high-density ones.

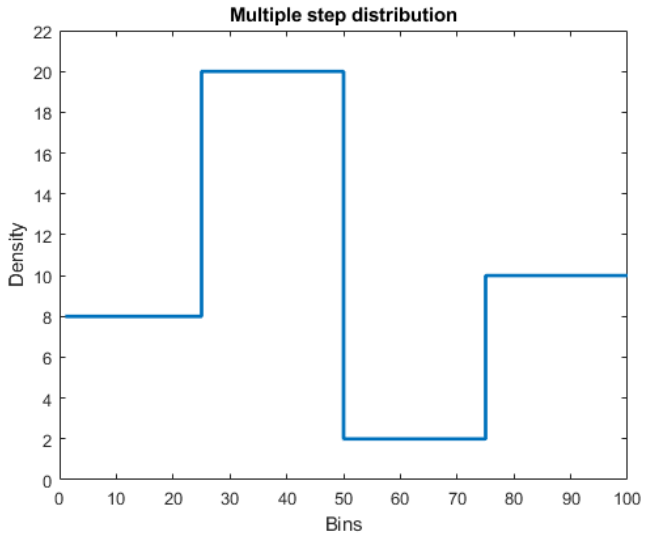


Figure 20 Multiple Step Distribution

We express the density as the number of data points in each length, divided by that length as follows the equation:

$$Density = \frac{Number\ of\ data\ points\ in\ the\ area}{length\ of\ the\ area}$$

3.1.3.1. Increasing the training epochs

For further experiments, let us consider the following system of equations,

$$\begin{cases} d_1 = K * {w_1}^{\alpha} \\ d_2 = K * {w_2}^{\alpha} \\ d_3 = K * {w_3}^{\alpha} \\ d_4 = K * {w_4}^{\alpha} \end{cases}$$

Where d_i is the output density on area i, $K$is the magnification constant and w_i the input density on area i. After solving the equations for α and K, we get the following expressions:

$$\begin{cases} \alpha_1 = \dfrac{\log\left(\frac{d_1}{d_2}\right)}{\log\left(\frac{w_1}{w_2}\right)} \vee \alpha_2 = \dfrac{\log\left(\frac{d_1}{d_3}\right)}{\log\left(\frac{w_1}{w_3}\right)} \vee \alpha_3 = \dfrac{\log\left(\frac{d_1}{d_4}\right)}{\log\left(\frac{w_1}{w_4}\right)} \vee \alpha_4 = \dfrac{\log\left(\frac{d_2}{d_3}\right)}{\log\left(\frac{w_2}{w_3}\right)} \vee \alpha_5 = \dfrac{\log\left(\frac{d_2}{d_4}\right)}{\log\left(\frac{w_2}{w_4}\right)} \vee \alpha_6 = \dfrac{\log\left(\frac{d_3}{d_4}\right)}{\log\left(\frac{w_3}{w_4}\right)} \\ \alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \alpha_5 = \alpha_6 \\ K_i = \dfrac{d_i}{{w_i}^{\alpha}} \end{cases}$$

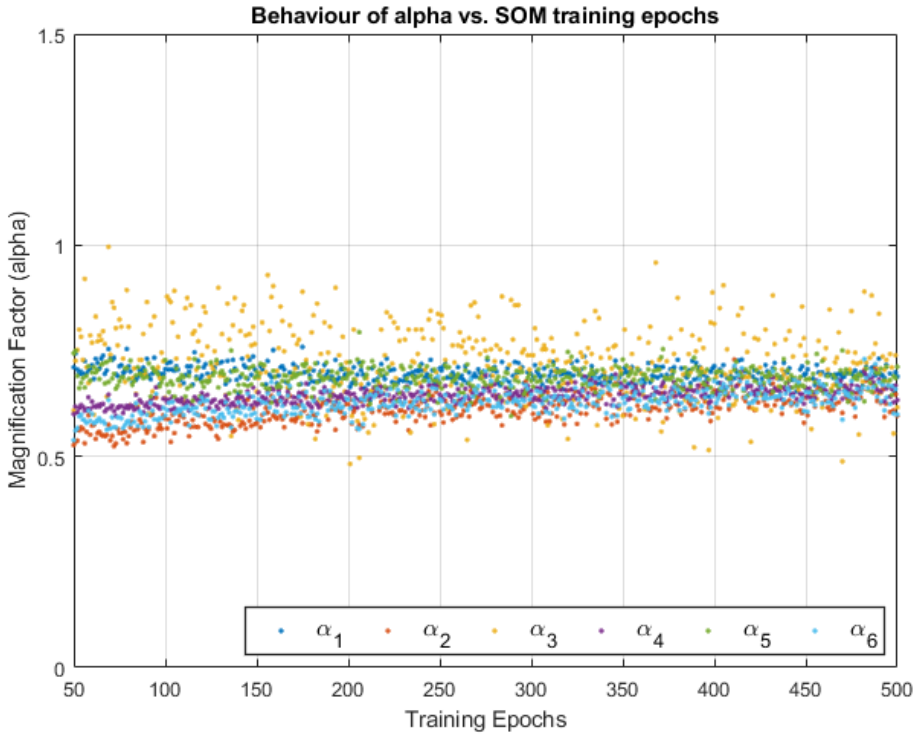


Figure 21 Magnification factors for multiple input step distribution considering the increase of training epochs.

From the experiment above, it is possible to perceive that the values of α_3 (represented in yellow) have a large dispersion; this happens since we calculate the magnification factor considering two similar average densities (d_1 *and* d_4). Also,

according to the resulting graph, we can say that alpha does not vary with the increase of training epochs since all α_i approximately converge to the same value.

Table 6 α values

Experiments with multiple step Distribution	
$\boldsymbol{\alpha_1}$	0.6955±0.0194
$\boldsymbol{\alpha_2}$	0.6123±0.0352
$\boldsymbol{\alpha_3}$	0.7256±0.0858
$\boldsymbol{\alpha_4}$	0.6454±0.0197
$\boldsymbol{\alpha_5}$	0.6858±0.02439
$\boldsymbol{\alpha_6}$	0.6208±0.0297

Considering the equations above, we calculated the alpha in each of the one thousand experiments generated for the one-dimensional case. Therefore, we conclude that the value for the magnification factor is $\alpha = \frac{2}{3}$, or more precisely 0.6654 ± 0.0357, which is the same as Ritter's assumptions for 1D SOM.

3.1.3.2. Increasing the number of neurons

In this subchapter, we want to explore the influence in the magnification factor by increasing the total number of SOM neurons and extract the average magnification factor that is a solution to the equations stated in the previous subchapter. For this experiment, we fix all parameters, we set the training epochs at 500, and on each run, we increase the number of SOM neurons by one. Further, we analyse the behaviour of α with respect to the number of neurons. We expect a similar behaviour as in the earlier subchapter.

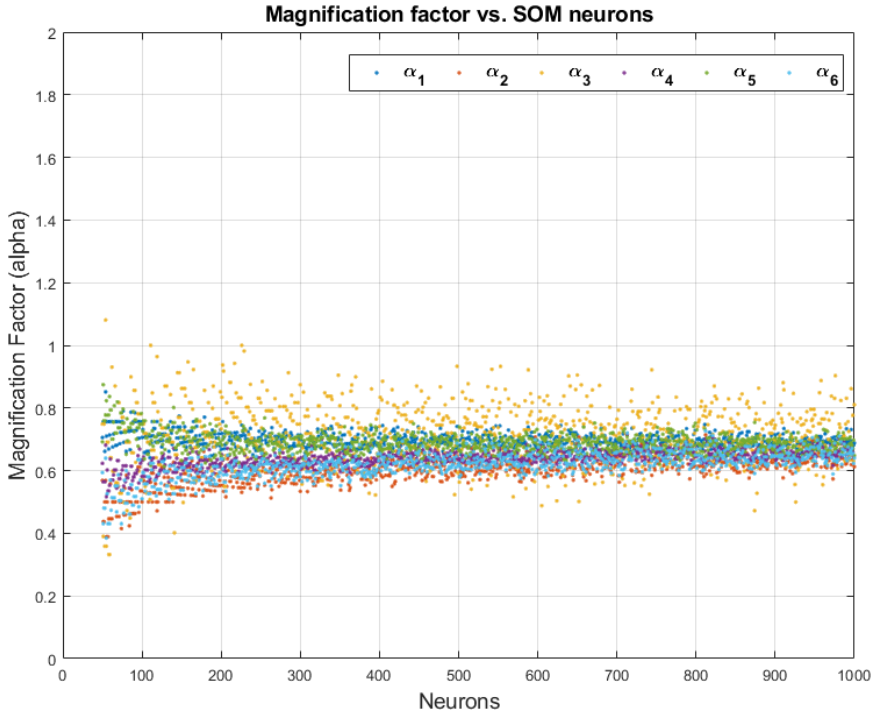


Figure 22 Magnification factors for each experiment vs. the increase of neurons

In the figure, it is possible to note that α_3(the yellow dots) has high dispersion, when related to the other magnification factors; this happens because we calculate this parameter considering two similar average densities (d_1 and d_4).

As it is possible to see, the magnification factor tends to converge to the same value aforementioned 0.6654 ± 0.0357. As a result, this set of experiments confirmed that for the one-dimensional case, the magnification factor is in fact equal to $\frac{2}{3}$.

3.2. Two-dimensional input data

In subchapter 3.1 we experimentally showed the magnification effect on 1-D data. We want to verify a similar behaviour for two-dimensional data using a similar method; we initialise SOM and train it, insert the input data, and correlate the density of input data with the SOM neurons density to generate an "exponent" magnification factor α. In this experiment, we additionally expect to find a relationship between unidimensional and two-dimensional magnification factors.

We generate two sets of experiments, one with two different density areas and the other with four distinct density areas. Once again, we fixed most of the parameters and analysed them one case at a time.

3.2.1. Two Different Density Areas

We opt to generate a rectangular input data map with two equally-width areas, each with a different input data density. The first zone (on the left) has 85 blue dots and the second zone with 15 dots in magenta, totalling 100 input data points, as illustrated in Figure 23.

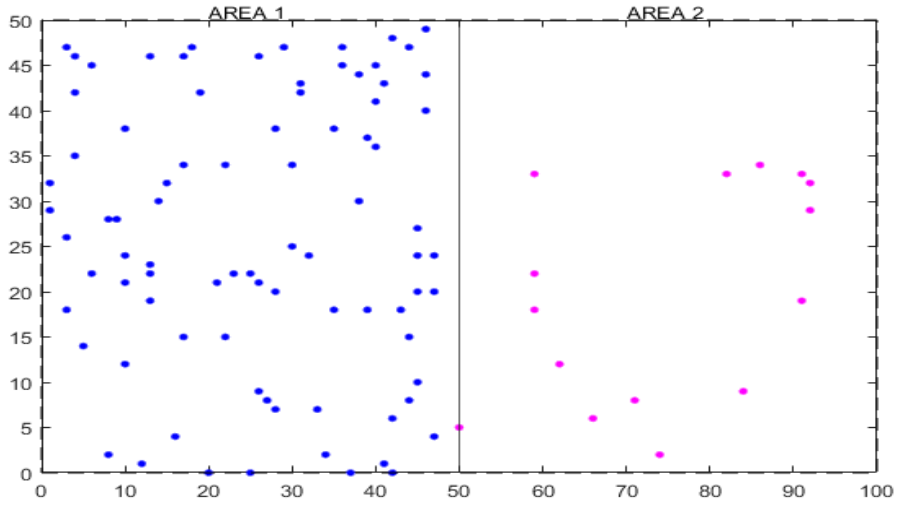


Figure 23 2D Input data

We calculated the density values according to the following expression $\frac{Number\ of\ points\ in\ the\ area}{Area\ (squared\ unit\ area)}$, and we present in the next table,

Table 7 2-D Experiment Average Densities

Average Density		
Data	**Area 1**	**Area 2**
Input	0.034	0.006
Output (neurons)	0.0316	0.0084

We initialise SOM on sequential training with 100 neurons in a $10 * 10$ lattice.

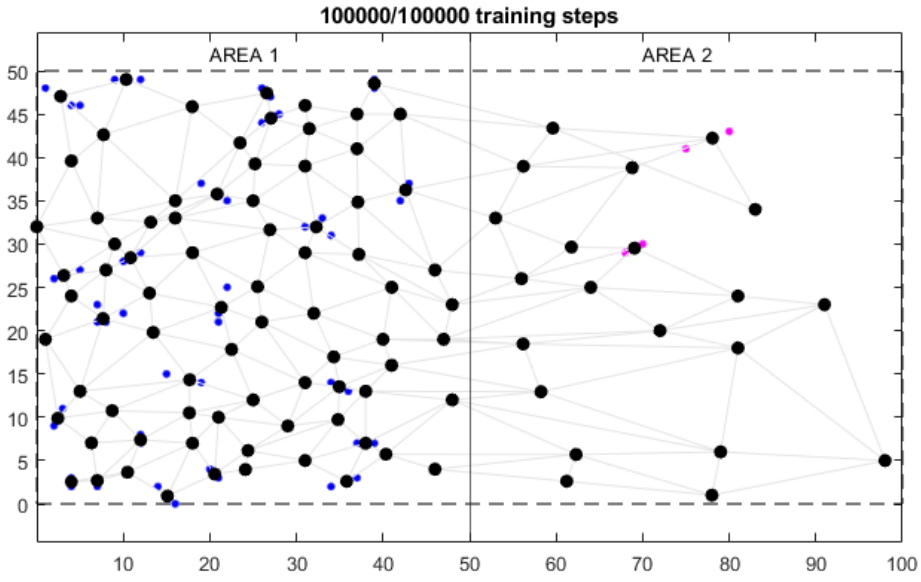


Figure 24 SOM neurons (represented in black dots) adjusting to input data. The black dots show positions of map units, and the grey lines show connections between neighbouring map units.

In this experiment, we realised that the SOM magnification factor on 2-D behaves slightly differently compared to the one-dimensional case.

We show in the figure below the variations of density on the input to the output.,

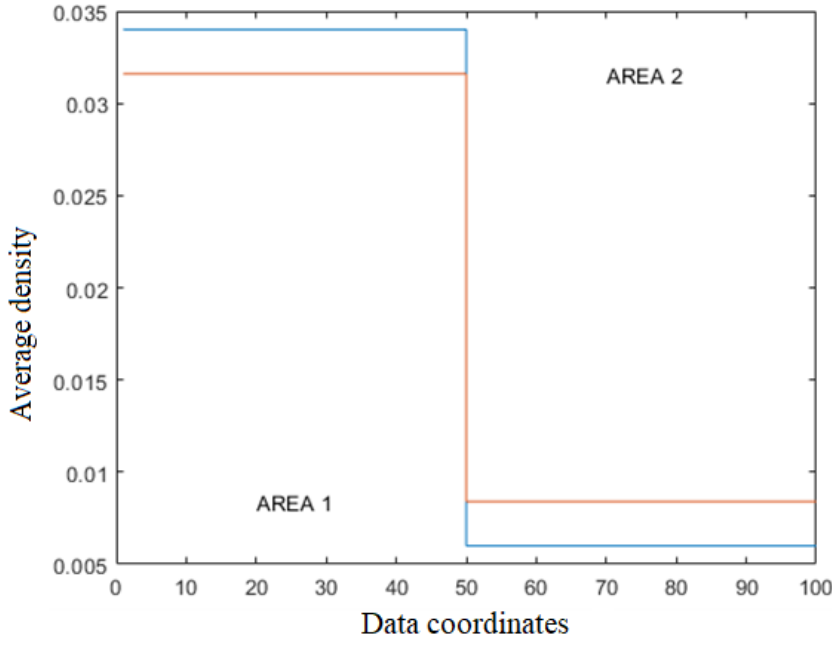


Figure 25 Variations on average density on the input (blue line) to the output (orange line).

As shown in the figure above, the magnification effect is still present in the 2-D case (magnifying the low-density areas and "demagnifying" high-density areas), the same as in the 1-D case. We follow the method used in the previous chapter to calculate the magnification factor, *i.e.*, calculating the solutions for the equations mentioned above to find K and α in each of a total of one thousand experiments.

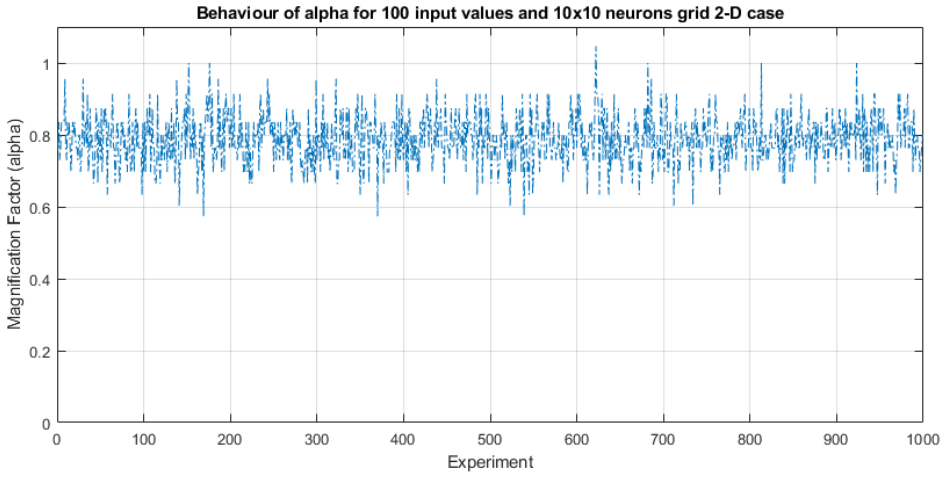


Figure 26 Magnification factors for 2-D (10x10 neurons grid)

Table 8 Average α and K

Average α and K from the experiment	
α	K
0.7875	0.4724

However, we believe that these results may suffer from the curse of dimensionality (Bellman, 2015). Due to this, we understand that having more input data and neurons will bring better results.

To avoid the dimensionality curse, we increase the input data points and initialise SOM with a 25x40 neurons grid. We trained SOM with the same parameters abovementioned and depict the results in the figure below,

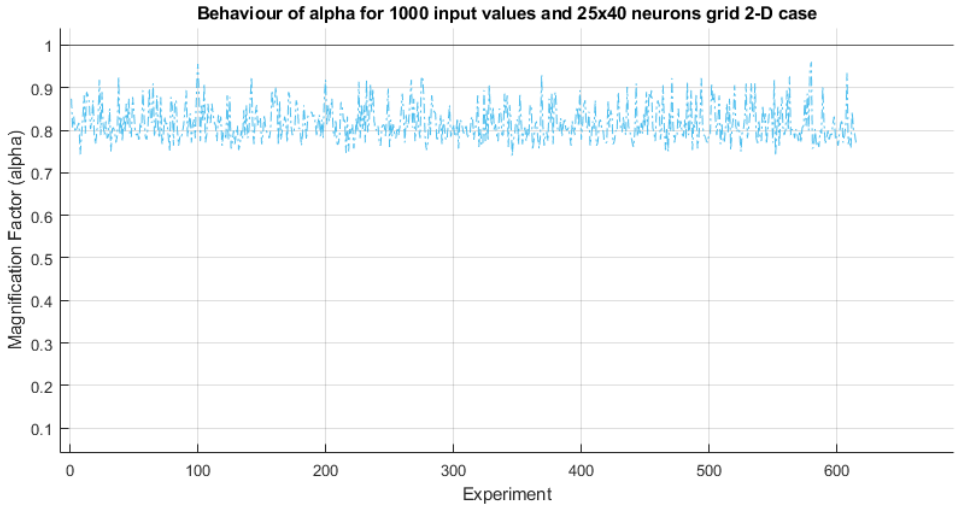


Figure 27 Magnification factors for 2-D SOM (25x40 neurons grid)

The figure above shows that the magnification factor rounds around 0.8166 with a standard deviation of 0.043. We notice that the standard deviation is lower for this case, a variation of 5.26% in the average value found for the magnification factor.

The magnification constant were as follows,

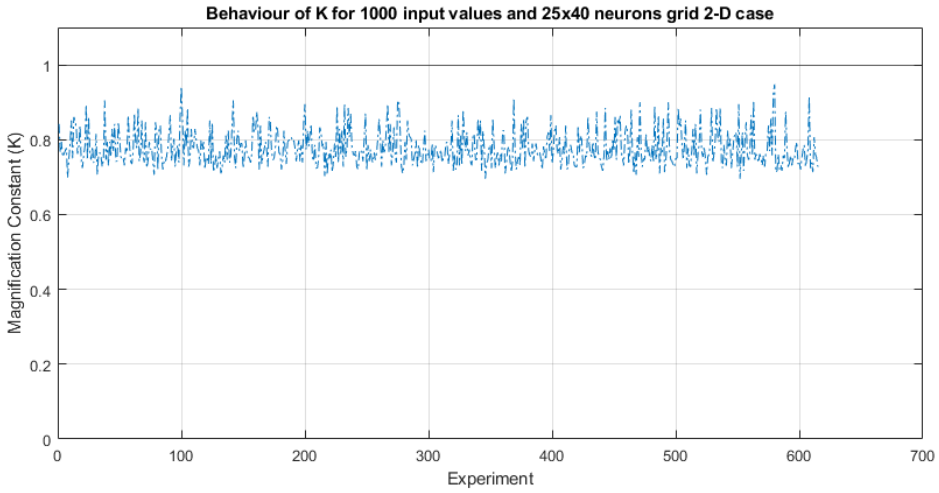


Figure 28 Magnification constant on each experiment.

Resulting on an average $K = 0.778$ with standard deviation of 0.0483, representing 6% of average K value variation.

With these results we can say that for this specific case the magnification factor is around 0.8166 ± 0.043 which is equal to $\sqrt{\frac{2}{3}}$.

3.2.1.1. Increasing the amount of neurons

We want to realize whether increasing the neurons on the two-dimensional SOM will influence the convergence of the original density value, comparable to the earlier chapter. To do this, we took advantage of the two equally-width areas data distribution generated in the previous subchapter. We fixed all remaining values, except for the number of neurons on the SOM grid.

We used a rectangular-shaped grid with NxM neurons; initially, we generated a grid with thirty-five neurons, i.e., 7 x 5, then we increased the number of neurons in each next experiment. In this set of experiments, we expect SOM to converge rapidly.

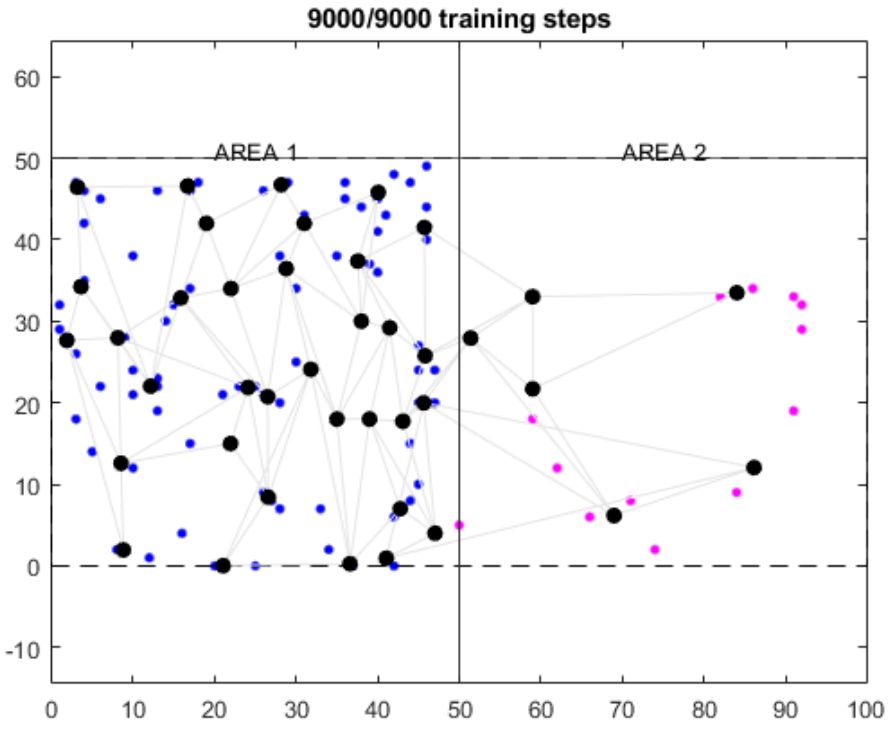


Figure 29 First run, with 35 neurons and 100 data points.

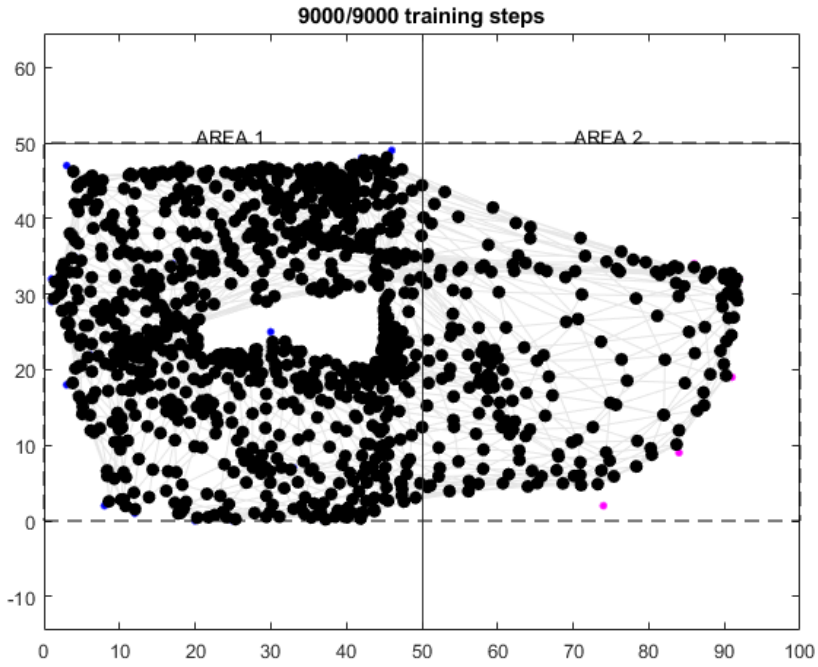


Figure 30 Last experiment, with 1024 neurons.

Looking at the resulting data, we find that the magnification factor is around 0.8833 ± 0.057, slightly higher than the one found for 1-D. Although this value is not

precise, and it is a mean value from all the sets of experiments with the number of neurons increasing, we observed the influence of increasing neurons. From these results, we can verify that the SOM does not converge asymptotically with the increase of neurons as 1-D does.

Surprisingly, the density values at the output are very discrepant each time the number of neurons increases, which can mean that at 2-D, the number of neurons will take considerable influence on magnification factor or (SOM's output representation), and it is more significant than on 1-D.

Table 9 Average α and K

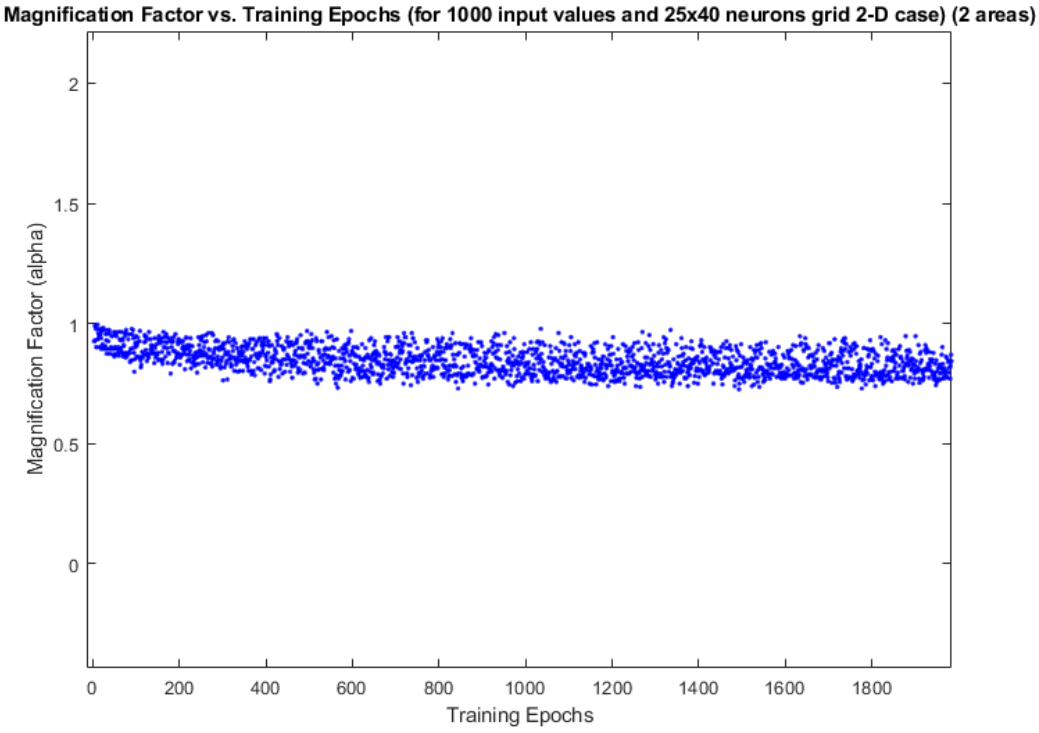
Average α and K from the experiment	
α	K
0.8833	3.6274

3.2.1.2. Increasing the training epochs

In the previous subchapter, we fixed all parameters and changed only the SOM neurons to understand that parameter's influence. Although we used the sequential algorithm to train SOM during these experiments, we did not work with epochs but with single sample iterations. Therefore, to compare with previous results, the number of iterations has to be (a previous number of epochs) * (the number of data points).

Here, we trained the SOM in batch mode; the input data varies on each experiment but not the density. Thus, we increase the number of epochs in each experiment. We present the results afterwards and compare them to the ones analysed before.

We want to verify for how many training epochs the map converges to its equilibrium state. For this, we fixed all the remaining parameters and unitarily increased the training epochs on each experiment. Subsequently, from the experiments, we can extract the magnification factor.



Considering the average value of all of the 1500 resulting magnification factors α calculated, starting from 500 training epochs (which in this experiment we ponder on

being a fair minimum value for this training parameter) to 2000 training epochs, we can verify that $\alpha = 0.8316 \pm 0.05$ which is approximately equal to $\sqrt{\frac{2}{3}}$.

3.2.2. Four Different Density Areas

In this experiment, we generate a rectangular shaped 2-D data point distribution with four different density areas. We opt to generate a rectangular input data map with four equally-width areas, each with a different input data density. In the figure below, we show 20 blue dots in the first area (in the top left corner), the second (in the top right corner) with 210 dots in magenta, the third (in the bottom left corner) with 70 red points and the fourth area (in the bottom right corner) with 700 green points totalling 1000 input data points.

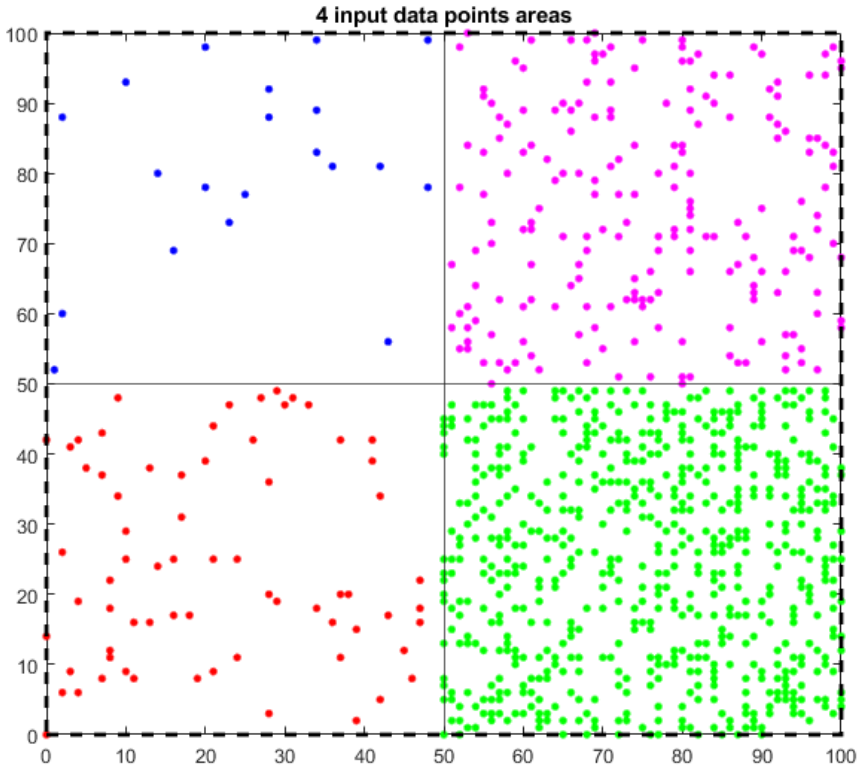


Figure 31 2D Input data distributed on 4 areas.

We calculated the density values according to the following expression $\frac{Number\ of\ points\ in\ the\ area}{Area\ (squared\ unit\ area)}$, and we presented the values in Table 10,

Table 10 Average Density

Area 1	Area 2	Area 3	Area 4
0.008	0.084	0.028	0.28

The SOM is initialised on sequential training with 1000 neurons in a $25 * 40$ lattice.

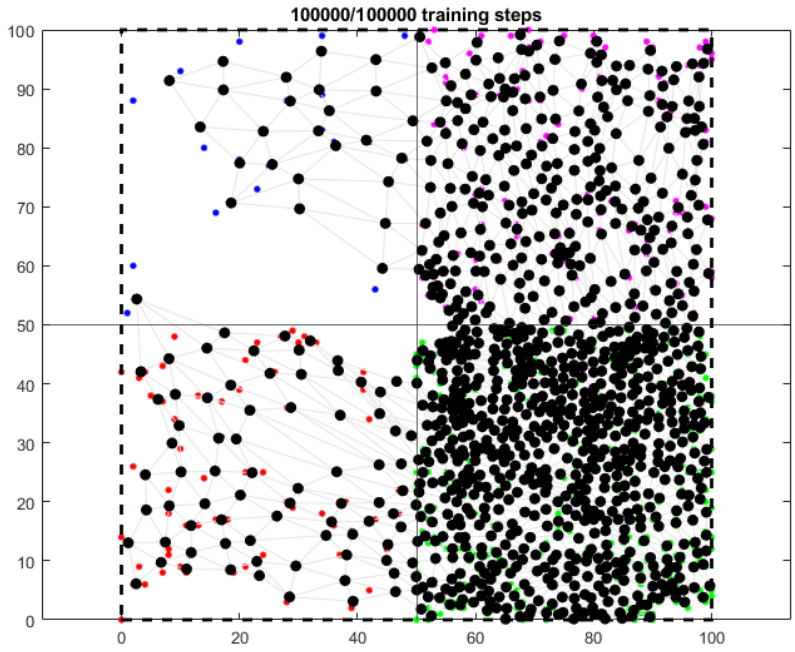


Figure 32 SOM neurons (represented in black dots) adjusting to input data.

The black dots show positions of map neurons, and the grey lines show connections between neighbouring map neurons.

In the figure below we show the variations of density in the input to the output,

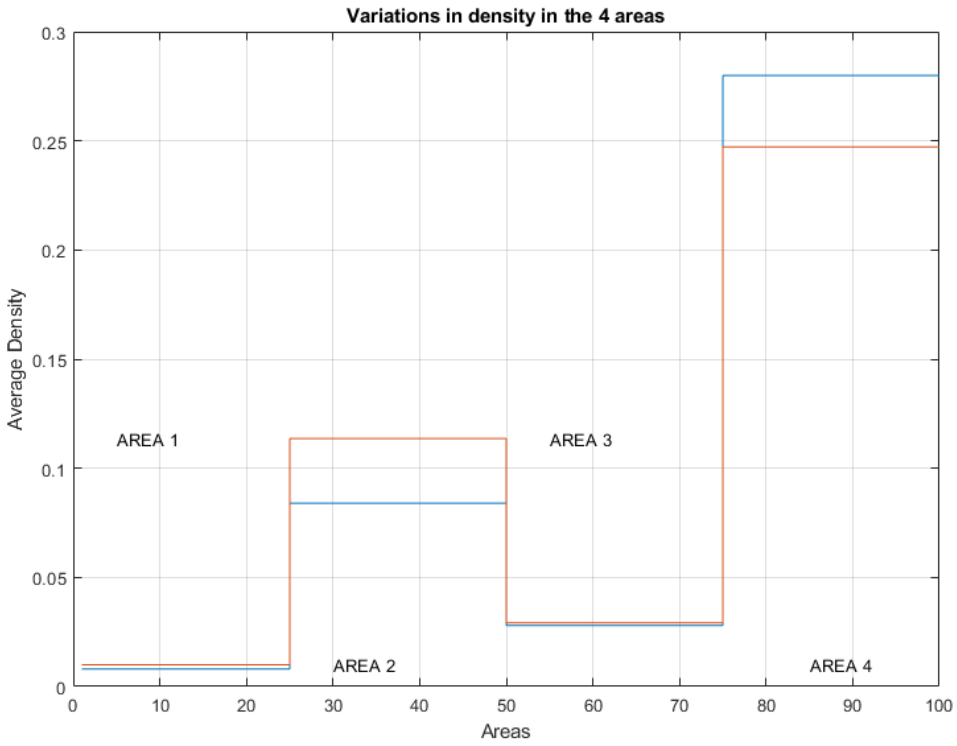


Figure 33 Variations on average density on the input (blue line) to the output (orange line).

Now we write a system of four equations to find, K and α, the solution to each experiment.

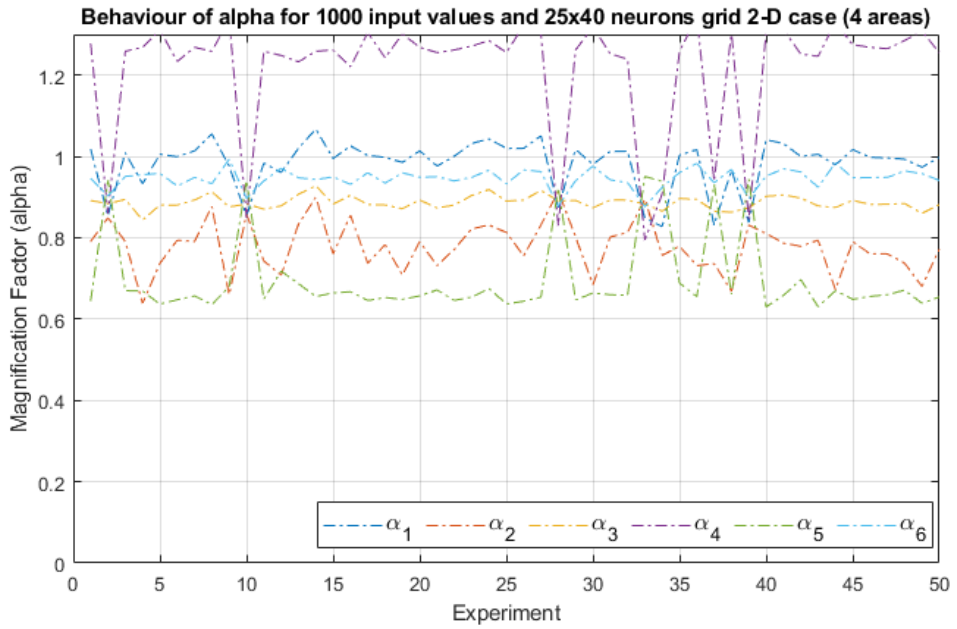


Figure 34 Magnification factors for 2-D SOM (4 areas)

Since we fixed the input data values, we consider that increasing the number of experiments would only prove what we already demonstrated earlier. Furthermore, it is possible to see in Figure 35 that the α_4 had dispersed values; this happens since the two density differences are not significant to verify the effect, and consequently, we discard these values when calculating the magnification factor on this case..

Figure 35 Magnification constants for 2-D SOM (4 areas)

Table 11 Average αand K

Average α and K from the experiment	
α	K
0.827±0.0506	0.9767 ±0.148

We can perceive from the results above that the magnification factor rounds around 0.827 with a standard deviation of 0.0506. We can see that the standard deviation is lower for this case, representing a variation of only 6.11%. The experiment also resulted in an average K=0.97 with a standard deviation of 0.148, representing 15.25% of the average K value variation.

Once more, with these results, we can say that for this specific case, the magnification factor is approximately $\sqrt{\frac{2}{3}}$.

3.3. Three-dimensional input data

In this subchapter we calculate the magnification factor on three different sets of experiments, all with 3-dimensional input data. The input data has a random uniform distribution with different densities in two distinct areas in all three experiments. The output representation of SOM will vary according to the experiment. The first experiments will train a 3D SOM, the second a 2D SOM and the last a 1D SOM. The magnification factor calculations will proceed using the same method as in the previous subchapters.

3.3.1. Calculating the magnification factor

In this set of experiments, we generated a three-dimensional random uniform distribution data with two different area densities. Then, we trained a SOM and calculated the magnification factor according to Equation 1.

In the experiments, we set the input data points to 9950 on the higher density zone vs 50 on the lower density zone. Therefore, we fixed the training epochs at 500 and the neighbourhood radius went from 1/5th of the number of neurons to 0.

We show the results of the set of experiments in the following table,

Table 12 Average α and *K* for the three-dimensional set of experiments

Input dimension/Output dimension	$\boldsymbol{\alpha}$	$\boldsymbol{K}$
3D to 3D	0.9391 ± 0.0253	0.4286 ± 0.0278
3D to 2D	0.8793 ± 0.0279	0.5956 ± 0.0436
3D to 1D	0.8315 ± 0.0157	0.3243 ± 0.0134

We can realise that for the 3D to the 2D case, we have $\alpha \cong \sqrt[3]{\frac{2}{3}}$.

Thus far, we have found evidence that α has a value that is around $\sqrt[f(\cdot)]{\frac{2}{3}}, with\ f(\cdot)$ being a function of the dimension of the SOM grid, the dimension of the input data and output neurons grid. Although $f(\cdot)$ depends on many variables, the most important one is the dimension of the input data n so in many expressions we may approximate $f(\cdot)$ by *n*. Consequently, considering the previous expression, we can present the following law, which is a small adjustment to Ritter's assumptions,

$$Output\ density\ \propto Input\ density^{\sqrt[n]{\frac{2}{3}}}$$

We should note that for 1D data and 1D SOM, $f(\cdot) \approx n = 1$ and the equation is exactly what Ritter proposed.

Notwithstanding, the expression above has shown promising results. Nevertheless, we have proved that alpha varies with the input data dimensionality and varies according to the SOM grid dimension, the number of neurons, the training epochs, and the initial and final neighbourhood radius. Hence, to achieve better results, is important to consider all those abovementioned parameters.

Therefore, $f(\cdot)$ must be a multivariable function which depends on each of the stated parameters.

$$n = f(training\ epochs, rad_{initial}, rad_{final}, num\ neurons, dimm_{in}, dimm_{SOM})$$

We propose three methods to find a solution:

- **First**: To train an ANN to find the function $f(\cdot)$ that interpolate the experiment evidence for all the dimensions, or at least for a substantial number of dimensions.
- **Second**: To calculate the function $f(\cdot)$ using non-linear regression algorithms.
- **Third**: To try to derive analytically (or guess) the right expression for $f(\cdot)$, and in this case, we believe that a set of assumptions will need to be made in order to extract a feasible solution.

However, we will let this problem for a future investigation, and settle with the empirical equation of $\sqrt[n]{\frac{2}{3}}$ that we postulated and verified in our experiments.

Chapter 4

4. Testing the new magnification factor rule in a SOM based Cartogram

To ascertain the validity of the magnification law derived from our experiments, we opt to show an example of SOM implementation that could be corrected if using the magnification factor found.

4.1. Angola Cartogram using Carto-SOM

In this experiment, we generated cartograms based on the Carto-SOM algorithm (R. Henriques, 2009) to produce a 2-D cartogram of Angola's population.

First, we applied the Carto-SOM using the real population density. Afterwards we corrected the population to be used in the Carto-SOM algorithm, using our law, in an attempt to obtain a better cartogram.

We compensate the magnification effect using the magnification law derived earlier in this work. For the experiment, we collected Angola's administrative boundaries data from Data Catalog Website (Admin, 2019). The geodetic data of the administrative boundaries are in a cartesian system (EPSG:32733) (Wikipedia, 2021). The EPSG is a Geodetic Parameter Dataset created by the International Association of Oil and Gas Producers (IOGP). It is a public archive of geodetic datums which is basically used to define coordinate reference systems and transformations for mapping.

For this experiment we generated and randomly distributed uniform data points, proportional to the population in a given region, across all the Angolan map. Since the population numbers in each region are vast, and to spare unnecessary computational effort we use just a percentage of the real population dataset as input data. We know that the number of neurons is by far one of the parameters that most influences the magnification on SOM, as shown in the previous chapters, and thus we concluded that using several neurons equivalent to more than 40% of the total data points should be sufficient to bring satisfactory results.

To see the magnification effect, we will look to the data densities of each region and then, after training the SOM, we will compare input data density to the density of neurons. Since for this case we will work with 2-D input data to 2-D neuron grid, according to our expression found in chapter 3, it is expected to find a magnification factor of $\alpha \cong \sqrt{\frac{2}{3}}$. We will first run some experiments to investigate this assumption and then use the relation that was found earlier to correct the population thus obtaining a better cartogram.

Angola has 18 regions, but as a matter of computational simplification we will consider only 17 of them (we excluded Cabinda because it is an enclave in the Congo and separated from the main territory) plus 3 new regions which we called "South Atlantic, Congo and Zambia" (see Figure 36). These 3 new regions have density that is equal to the country average demographic density. They are important in order to generate a rectangular map and let the original map of Angola be identifiable as usually happens when creating cartograms.

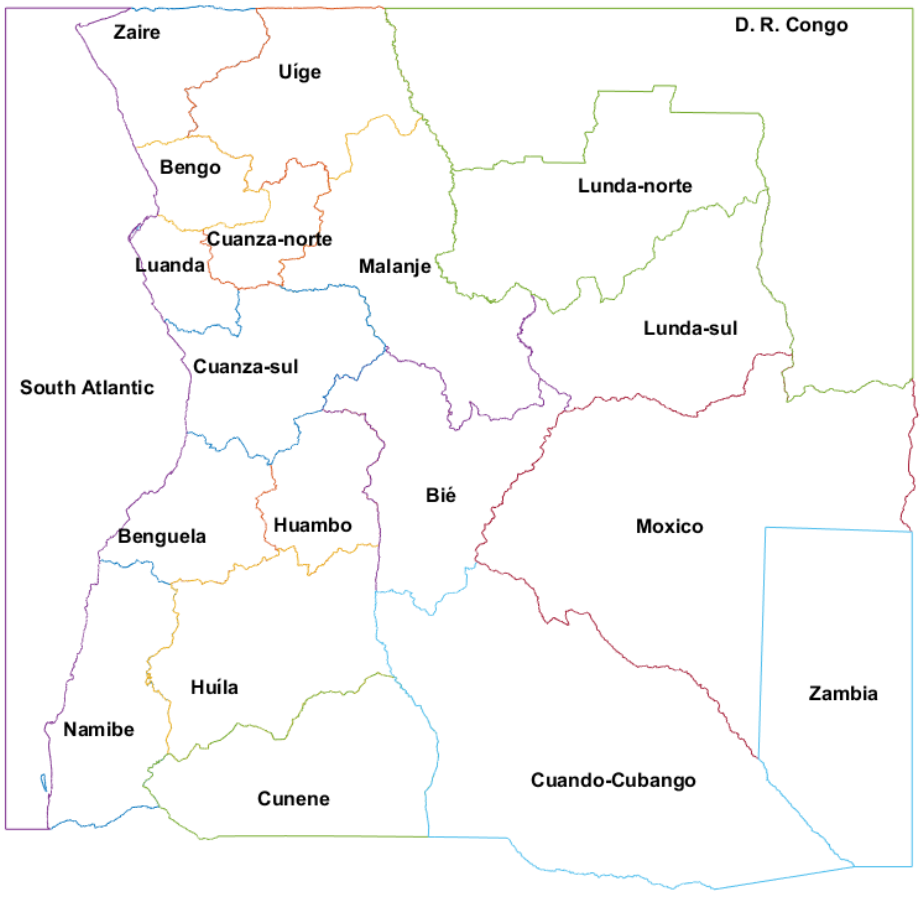


Figure 36 - Map of Angola used to produce the cartogram

Angola's demographic data was collected from the 2014 census published by Instituto Nacional de Estatística (INE) and can be obtained in https://bit.ly/3kBQP8m. The capital Luanda is by far the most densely populated city in Angola so the map will distort, and the city of Luanda will be the most prominent of all. In the table of appendix C we show all Angola regions as well as their population and population density.

Using this data we used the Carto-SOM algorithm and obtained the cartogram showed in Figure 37.

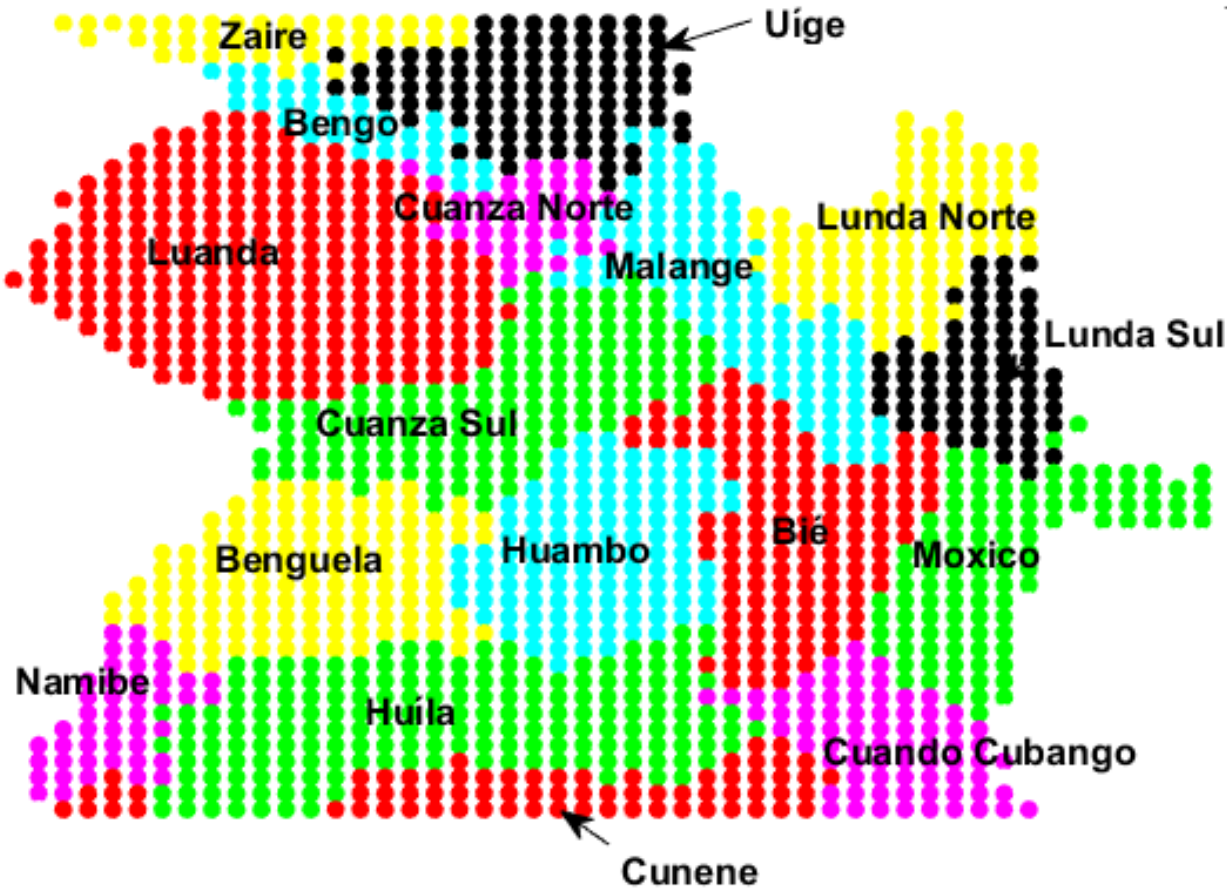


Figure 37 - Angola Cartogram using Self-organizing Map

This cartogram is a reasonable depiction of the Angola's population but there are some inaccuracies. In the figure above we can see that Cunene was split in two. That happens because that province has one of the lowest demographic densities and it was "smacked" by his neighbour Huíla that its far denser. It is also evident for who knows Angola that while Luanda and Benguela have a large area it is not proportional to their true and very large population.

After this first cartogram, we set up our way to compensate the magnification factor and try to build a perfect cartogram of Angola based on the law that we derived. The idea is to correct the provinces original densities so that when the SOM is trained the magnification would help give a more realistic representation of the map. To this end we postulate the following relation,

$$d_{out} = K * {d_{corrected}}^{\alpha} = d_{in} \qquad \text{Equation 3}$$

Where, d_{in} is the province original input density and $d_{corrected}$ is the density desired to compensate the magnification on the map. As we know from our earlier experiments, in this case α should be equal to $\sqrt{\frac{2}{3}}$ and K is approximately 0.91. We manipulate the equation above and calculate the corrected densities for each region,

$$d_{correcred} = \sqrt[\alpha]{\frac{d_{in}}{K}} \qquad \text{Equation 4}$$

Finally, we produce the corrected Angola cartogram as shown in the projection below,

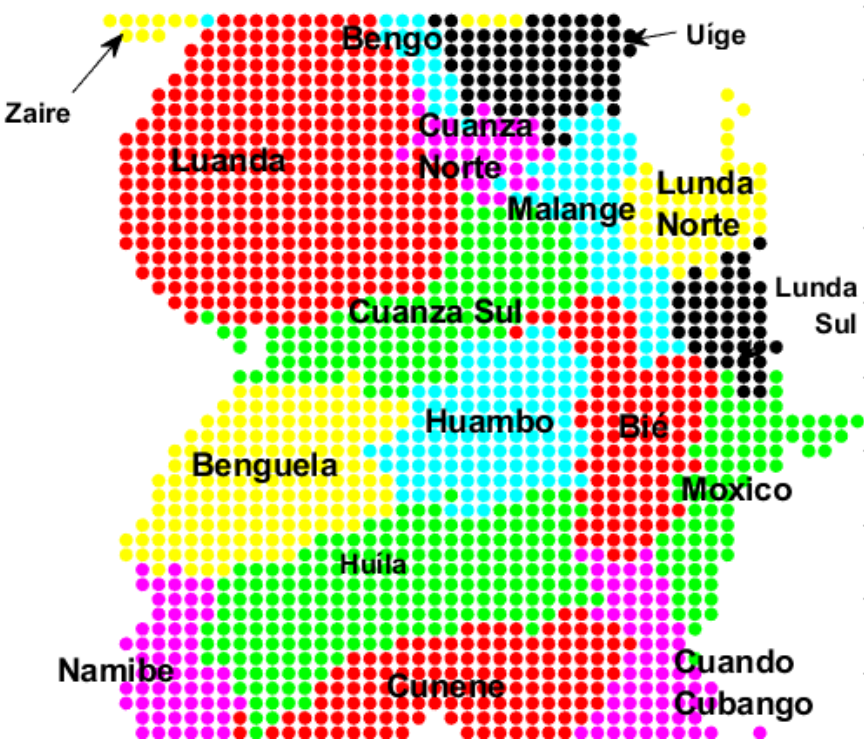


Figure 38 Angola's corrected cartogram using the magnification law

As we can see, the capital city Luanda holds the most space in the map due to its populational density and caused the northern provinces Bengo and Zaire to compress. Also, now despite Huíla being denser than Cunene it is not sufficient to make that region

split. The topographic appearance of the map is perfectly recognizable, but it has a strong distortion due to variations in the true population density. The borders would be smoother and “nicer” if we used a Geographic Information System (GIS) program such as ArcGIS to obtain a better tessellation.

Chapter 5

5. Conclusions

The study of the magnification effect is a very complex matter because of the number of parameters to be analysed simultaneously. Due to this, we chose to fix the maximum number of parameters and observe a given parameter's influence on the magnification factor. It was possible to experimentally verify the value that until now had only been demonstrated analytically by Ritter & Schulten with many approximations and assumptions.

The expression $d_i = K * w^{\alpha}$ proved to be powerful enough to study the magnification effect on one dimension. However, the self-organising feature of the algorithm produced a statistical standard deviation on the set of experiments. Unfortunately, we could not eliminate this statistical variation, but it had small values in practice. Nevertheless, these variations make the exact calculation of the magnification factor impossible (it only has a statistical value).

For small numbers of neurons, the numerical computation of α is very unstable but converges rapidly as the number of neurons increases. Above a specific value (in our case, 100 neurons for the Heaviside distribution), the magnification factor converges and does not vary with increasing the number of neurons.

As expected, we verified that the number of neurons is directly proportional to the constant K. However, some questions are still unexamined as the influence of the increase of neurons to one dimension has a moment where magnification stabilises regardless of the increase in neurons.

Several experiments done and depicted in appendix B evidenced that there must be a relationship between the neighbourhood function initial radius and the number of neurons for the one-dimensional case. Therefore, we conclude that the value for the initial radius should be approximately equal to 20% of neurons for better results.

We consider the study of the magnification effect on one-dimension only didactic since it is better to study the effect on multidimensional cases where a precise analysis

is very needed. The results for 1-D were good to the extent that it was approximately equal to the value found analytically by Ritter.

As expected, the two-dimensional SOM brought more difficulties, due not only to the increase of one dimension but also because of the coding complexity. However, a magnification factor of $\sqrt{\frac{2}{3}}$ was found and tested, giving reliable results for the two-dimensional experiments.

The three-dimensional SOM analysis brought light to a magnification factor convergence law. After various experiments were analysed and several calculations done, α was found to be approximately equal to $\sqrt[f(\cdot)]{\frac{2}{3}}$, *i.e.*, there is evidence that the magnification factor is proportional to $\frac{2}{3}$ raised to the power of $\frac{1}{f(\cdot)}$, where $f(\cdot)$" is the function of multiple parameters such as the number of neurons, training epochs, initial radius, final radius, input data dimension, output representation dimensions, etc. Fortunately, $f(\cdot)$ can be approximated reasonably well by simply the dimension of the input space (n).

For the more useful multidimensional cases, *i.e.*, when the input space is n-dimensional, where no analytical derivation is known, we showed that the magnification rule derived by Ritter still holds, but with a magnification factor that instead of being $\frac{2}{3}$ is approximately the *n*-root of 2/3, where n is the dimension of the input data:

$$Output\ density\ \propto Input\ density^{\sqrt[n]{\frac{2}{3}}}$$

This result holds quite well when the output space has dimension 2 and can be used with relative certainty. If the output space is the same as the input space (in 3D and 4D mappings), there is a slight deviation, and the magnification factor is higher.

In future work we intend to perform more experiments and derive an empirical function that considers factors such as dimensions of both spaces, the number of neurons and training parameters, and asymmetries in the input densities, to derive a magnification law similar to:

$$Output\ density\ \propto Input\ density^{\sqrt[f\left(t_e,r_i,r_f,neurons,dim_{in},dim_{SOM}\right)]{\frac{2}{3}}}$$

In chapter 4 we created a version of the Angola's population cartogram with SOM, and afterwards corrected the cartogram. We showed that the magnification effect can be compensated using the relationship found in this work. As far as we know this is the first SOM based cartogram of Angola, and one of the very few ever made using any algorithm.

We have sent an article based on this discovery to the International Conference on Intelligent Data Engineering and Automated Learning (IDEAL) in Manchester, and we wait for acceptance.

All the code developed as well as the files created are placed in my GitHub repository Edson Bastos (2021). https://github.com/MoreiraBastos/Study-of-the-Magnification-Effect-on-Kohonen-s-Self-Organizing-Map-

With these results, we have shown that the magnification effect exists in all SOM and can be predicted (and thus considered or corrected) in more general practical cases.

Bibliography

Admin, C. (2019). AGO Administrative Boundaries Level 1. Retrieved from http://riskprofilesundrr.org/layers/geonode:ago_administrative_boundaries_level_1_1/metadata_detail

Affonso, G. S. J. A. A. à. u. d. S. P. (2011). Mapas Auto-organizáveis de Kohonen (SOM) aplicados na avaliação dos parâmetros da qualidade da água.

Bação, F., Lobo, V., & Painho, M. J. S.-O. M. a. i. g. i. s. (2008). Applications of different self-organizing map variants to geographical information science problems. 21-44.

Bauer, H.-U., & Der, R. J. N. c. (1996). Controlling the magnification factor of self-organizing feature maps. *8*(4), 757-771.

Bellman, R. E. (2015). *Adaptive control processes: a guided tour*: Princeton university press.

Calitoiu, D., Oommen, B. J., Nussbaum, D. J. I. T. o. S., Man,, & Cybernetics, P. B. (2007). Desynchronizing a chaotic pattern recognition neural network to model inaccurate perception. *37*(3), 692-704.

Campoy, P. (2009). *Dimensionality reduction by self organizing maps that preserve distances in output space.* Paper presented at the 2009 International Joint Conference on Neural Networks.

Chawathe. (2018). Monitoring Blockchains with Self-Organizing Maps. doi:10.1109/TrustCom/BigDataSE.2018.00283

Choraś, M., & Pawlicki, M. (2020). Intrusion detection approach based on optimised artificial neural network.

Der, R., & Herrmann, M. (1992). Attention based partitioning.

Fort, J.-C. (2006). SOM's mathematics. *Neural Networks, 19*. doi:https://doi.org/10.1016/j.neunet.2006.05.025

Gorricha, L. (2009). *Visualization of Clusters in Geo-referenced Data Using Three-dimensional Self-Organizing Maps.* (Master in Statistics and Information Management), Universidade Nova de Lisboa,

Gorricha, L. (2015). *Exploratory data analysis using self-organising maps defined in up to three dimensions.* (Doctor), Universidade Nova de Lisboa,

Kaski, S. (1997, September,1997). Example of application of the SOM: World Poverty Map. *Example of application of the SOM: World Poverty Map.* Retrieved from http://www.cis.hut.fi/research/som-research/worldmap.html

Khashman, A. (2009). Application of an emotional neural network to facial recognition. *18*(4), 309-320.

Kohonen, T. (1974). An adaptive associative memory principle. *100*(4), 444-445.

Kohonen, T. (1982a). Analysis of a simple self-organizing process. *44*(2), 135-140.

Kohonen, T. (1982b). Self-Organized Formation of Topologically Correct Feature Maps. *Biological cybernetics, 43*(1), 59-69. doi:Doi 10.1007/Bf00337288

Kohonen, T. (2001). *Self-Organizing Maps* (3 ed.): Springer-Verlag Berlin Heidelberg.

Krakovsky, R., & Forgac, R. (2011). *Neural network approach to multidimensional data classification via clustering.* Paper presented at the 2011 IEEE 9th International Symposium on Intelligent Systems and Informatics.

Li, S., & Li, Y. J. I. t. o. c. (2013). Nonlinearly activated neural network for solving time-varying complex Sylvester equation. *44*(8), 1397-1407.

Lobo, V. (2002). Ship Noise Classification - A contribution to prototype based classifier design.

Lobo, V. (2009). *Application of Self-Organizing Maps to the Maritime Environment.*

Lobo, V. J. (2009). Application of self-organizing maps to the maritime environment. In *Information Fusion and Geographic Information Systems* (pp. 19-36): Springer.

Ma'Sum, M. A., Arrofi, M. K., Jati, G., Arifin, F., Kurniawan, M. N., Mursanto, P., & Jatmiko, W. (2013). *Simulation of intelligent unmanned aerial vehicle (uav) for military surveillance.* Paper presented at the 2013 international conference on advanced computer science and information systems (ICACSIS).

Merényi, E., Jain, A., & Villmann, T. (2007). Explicit Magnification Control of Self-Organizing Maps for "Forbidden" Data. *IEEE Transactions On Neural Networks, 18*.

Moghaddam, A. H., Moghaddam, M. H., Esfandyari, M. J. J. o. E., Finance, & Science, A. (2016). Stock market index prediction using artificial neural network. *21*(41), 89-93.

Nikkilä, J., Törönen, P., Kaski, S., Venna, J., Castrén, E., & Wong, G. J. N. n. (2002). Analysis and visualization of gene expression data using self-organizing maps. *15*(8-9), 953-966.

Ong, J., & Abidi, S. S. R. (1999). *Data Mining Using Self-Organizing Kohonen Maps: A Technique for Effective Data Clustering & Visualization.* Paper presented at the IC-AI.

Poff, N. L., Tokar, S., Johnson, P. J. L., & Oceanography. (1996). Stream hydrological and ecological responses to climate change assessed with an artificial neural network. *41*(5), 857-863.

R. Henriques, F. B. V. L. (2009). Carto-SOM: cartogram creation using self-organizing maps. *International Journal of Geographical Information Science, 23:4*, 483-511. doi:10.1080/13658810801958885

Ritter, H. (1991). Asymptotic level for a class of vector quantization process. *2*, 173-175. doi:10.1109/72.80310

Ritter, H., & Schulten, K. (1986). On the stationary state of Kohonen's self-organizing sensory mapping. *Biological cybernetics, 54*(2), 99-106.

Roden, R., Smith, T., & Sacrey, D. J. I. (2015). Geologic pattern recognition from seismic attributes: Principal component analysis and self-organizing maps. *3*(4), SAE59-SAE83.

Wikipedia. (2021). EPSG Geodetic Parameter Dataset. In *Wikipedia*. Wikipedia, The Free Encyclopedia: Wikipedia, The Free Encyclopedia.

Yin, H. (2008). The Self-Organizing Maps: Background, Theories, Extensions and Applications.

A. Appendix A – Matlab routines

Here, most of the developed matlab routines, and those that are more important, are presented. All the code developed as well as the files created are placed in my GitHub repository Edson Bastos (2021). https://github.com/MoreiraBastos/Study-of-the-Magnification-Effect-on-Kohonen-s-Self-Organizing-Map-

A.1. One-dimensional SOM algorithm with Uniform distribution

```
    % Initialize all variables %

bins=10; %number of bins
neurons=1000; %SOM neurons
initialradius=neurons/5; %Initial radius
finalradius=0; %Final radius
iterations=1000; %Number of iterations

for j=4:4

z1=10^4; %data points

    for i=1000:iterations

epochs=i; %On each iteration the training epochs will
unitarily increase

%% Generate random data %

% dados=rand(z1,1); %Generate data

dados=0:0.001:0.999; dados=dados+0.0005; dados=dados';

qtd=zeros(bins,1); %initialize a variable to store the
quantity of data in each bin

for x=1:length(dados)
    qtd(ceil(dados(x)*bins)) = qtd(ceil(dados(x)*bins))+1;
%storing the quantity of data in each bin
end

dens=qtd/(1/bins); % copy the density of input data in
each bin to a variable
```

```
%% TRAINING THE SOM %



sMap =
som_lininit(dados,'munits',neurons,'shape','toroid'); %
%Initialize linearly the SOM Map

sMap2=
som_batchtrain(sMap,dados,'radius_ini',initialradius,'radi
us_fin',finalradius,'trainlen',epochs); %Training the SOM

SOM_data= sMap2.codebook; %Copy the neurons to SOM_data

qtdn=zeros(bins,1); %initialize a variable to store the
quantity of neurons in each bin

for x=1:length(sMap2.codebook)
    qtdn(ceil(SOM_data(x)*bins)) =
qtdn(ceil(SOM_data(x)*bins))+1;    %storing the quantity
of neurons in each bin
end

neurons_density=qtdn/(1/bins); % copy the density of
neurons in each bin to a variable

tiledlayout(2,1) % Create a tiled chart layout

nexttile % Top plot

title('Step distribution'); %figure title
grid on;
hold on;

bar(dens); %Show the results in a bars graph
ylim([0 1200]);
xlabel('Bins') , ylabel('Density');
yline(1000,'-','Average Density');
hold off;
title(sprintf('%d input data with uniform
distribution',z1));%figure title

nexttile % Bottom plot

hold on
bar(neurons_density,'r'); %Show the results in a bars
graph
xlabel('Bins') , ylabel('Density');
yline(neurons,'-','Average Unit Density');
```

```
hold off
texto=sprintf('SOM Output after %d training
epochs',epochs);
title(texto);
grid on


    end

save(sprintf('distribuicaouniforme_toroid.mat')); %Saving
all the experiment

end
```

A.2. One-dimensional SOM algorithm with Heaviside distribution

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
%
% Name: Heaviside
%
% Objective:
%
% Input/Output Parameters:
%
% Obs: This matlab routine will generate data
with Heaviside or step
% distribution and subsequently train a Self-
Organizing Map.
%
% V1.0 - Moreira Bastos, Jun 2021
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%

%% Initialize all variables


bins=100; % Number of data bins
neurons=1000; % Number of SOM units or neurons
initialradius=neurons/5; % SOM initial
neighbourhood function radius
finalradius=0; % SOM final neighbourhood function
radius
limiteinferior=50; % Length of first zone
limitesuperior=50; % Displacement of second zone
iterations=1;% How many times SOM is trained with
specific input parameters
totalpoints=1000; % Total input data points
experiments=499; % Total number of experiments

%% Code
for j=1:experiments
```

```
    z1= j; % Data points in the first zone (Low
density)
    % For the variations in data density
experiment let z1=j;
    % It is also possible to compare the
algorithm behaviour
    % in function of data density zones
difference

    z2=totalpoints-z1; % Data points in the
second zone (High density)

    % neurons=j; % For the increasing neurons
experiment uncomment this line
    % epochs=j; % For the increasing epochs
experiment uncomment this line
    % initialradius=neurons/(0.5*j); % For the
increasing initial radius experiment uncomment
this line

        for i=1:iterations % This loop will
specify how many times the routine will train SOM
with a fixed step input data distribution

epochs=500; % Number of training epochs

randomdata1=rand(z1,1)*limiteinferior; %Generate
data in the first zone

randomdata2=(rand(z2,1)*limiteinferior)+limitesup
erior; % Generate data in the second zone

dados=[randomdata1;randomdata2]; % Put all data
in one matrix

qtd=zeros(bins,1); % Initialize a variable to
store the quantity of data in each bin

for x=1:length(dados)
    qtd(ceil(dados(x))) = qtd(ceil(dados(x)))+1;
end

dens=qtd/(sum(qtd)); % Copy the density of input
data in each bin to a variable
```

```
%% TRAINING THE SOM

sMap=som_lininit(dados,'munits',neurons); %
Initialize linearly the SOM Map

sMap2=som_batchtrain(sMap,dados,'radius_ini',init
ialradius,'radius_fin',...
    finalradius,'trainlen',epochs); %TRAINING THE
SOM

SOM_data=sMap2.codebook; % Copy the neurons to
SOM_data

qtdn=zeros(bins,1); % Initialize a variable to
store the number of neurons in each bin

for x=1:length(sMap2.codebook)
     qtdn(ceil(SOM_data(x))) =
qtdn(ceil(SOM_data(x)))+1;
end

%% PLOTTING THE RESULTS

densidade_inputx=[0 50 51 100]; % Data density
zone boundaries
densidade_input=[z1/50 z1/50 z2/50 z2/50]; %
Theoretical density in each zone from (0 to 50)
and (50 to 100)

densidade_outputx=[0 50 51 100]; % Neurons
density zone boundaries
densidade_output=[sum(qtdn(1:51))/50
sum(qtdn(1:51))/50 ...
    sum(qtdn(51:100))/50 sum(qtdn(51:100))/50]; %
Theoretical density in each zone from (0 to 50)
and (50 to 100)

save(sprintf('heaviside%d.mat',j)); % Save all
the experiment

        end

end
```

A.3. One-dimensional SOM algorithm with multiple Heaviside distribution

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Name: degraus
%
% Objective:
%
% Input/Output Parameters:
%
% Obs: This matlab routine will generate data with multiple Heaviside or step
% distribution and subsequently train a Self-Organizing Map.
%
% V1.0 - Moreira Bastos, Jun 2021
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initialize all variables %
z1=200;
z2=500;
z3=50;
z4=250;

bins=100;% Number of data bins
neurons=1000; % Number of SOM units or neurons
initialradius=neurons/5; % SOM initial neighbourhood function radius
finalradius=0; % SOM final neighbourhood function radius
limite1=25; % Displacement of first zone
limite2=50; % Displacement of second zone
limite3=75; % Displacement of third zone
limite4=100; % Displacement of fourth zone

% epochs=500;

    for i=1:500
%% Generate the data
```

```
% epochs=i; % For the increasing epochs
experiment uncomment this line
% neurons=i; % For the increasing neurons
experiment uncomment this line
% initialradius=neurons/(0.5*j); % For the
increasing initial radius experiment uncomment
this line


randomdata1=rand(z1,1)*limite1; %Generate random
uniform data in the first zone

randomdata2=(rand(z2,1)*25+limite1); %Generate
random uniform data in the second zone

randomdata3=(rand(z3,1)*25+limite2); %Generate
random uniform data in the third zone

randomdata4=(rand(z4,1)*25+limite3); %Generate
random uniform data in the fourth zone

dados=[randomdata1;randomdata2;randomdata3;random
data4]; % Put all data in one matrix

qtd=zeros(bins,1); % initialize a variable to
store the quantity of data in each bin

for x=1:length(dados)
    qtd(ceil(dados(x))) = qtd(ceil(dados(x)))+1;
end

dens=[z1/25 z2/25 z3/25 z4/25]; % copy the
density of input data in each bin to a variable
verticd=[z1/25 z1/25 z2/25 z2/25 z3/25 z3/25
z4/25 z4/25]; %calculate zone densities for
plotting
horizd=[1 25 25 50 50 75 75 100]; % different
zones
plot(horizd,verticd); % Plotting the input data
density
grid on, hold on;

%% TRAINING THE SOM
```

```
sMap =
som_lininit(dados,'munits',neurons,'shape','toroi
d'); % %Initialize linearly the SOM Map

sMap2=
som_batchtrain(sMap,dados,'radius_ini',initialrad
ius,'radius_fin',finalradius,'trainlen',epochs,'s
hape','toroid'); %TRAINING THE SOM

SOM_data= sMap2.codebook; %Copy the neurons to
SOM_data

qtdn=zeros(bins,1); %initialize a variable to
store the quantity of neurons in each bin

for x=1:length(sMap2.codebook)
    qtdn(ceil(SOM_data(x))) =
qtdn(ceil(SOM_data(x)))+1;
end

densn=[sum(qtdn(1:25))/25 sum(qtdn(26:50))/25
sum(qtdn(51:75))/25 sum(qtdn(76:100))/25]; % copy
the density of neurons in each bin to a variable
verticn=[sum(qtdn(1:25))/25 sum(qtdn(1:25))/25
sum(qtdn(26:50))/25 sum(qtdn(26:50))/25 ...
    sum(qtdn(51:75))/25 sum(qtdn(51:75))/25
sum(qtdn(76:100))/25 sum(qtdn(76:100))/25];
%calculate zone neuron densities for plotting
horizn=[1 25 25 50 50 75 75 100];
plot(horizn,verticn);

save(sprintf('experiencia_degraus%d',i)); %
Saving all the experiment

end
```

A.4. Two-dimensional SOM algorithm with two different density areas

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Name: main_rectangulo_2areas
%
% Objective:
%
% Input/Output Parameters:
%
% Obs: This matlab routine will generate 2-D
input data with two distinct
% density areas and subsequently train a Self-
Organizing Map.
%
% V1.0 - Moreira Bastos, Jun 2021
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
area1 = 850; % Data points on the first area
area2 = 150; % Data points on the second area

%% Generating random input data
xRandom1=zeros(1,area1); xRandom2=zeros(1,area2);
yRandom1=zeros(1,area1); yRandom2=zeros(1,area2);

for i=1:area1 % Distributing data on the first
area
xRandom1(i) = randi([0 49],1,1);
yRandom1(i) = randi([0 49],1,1);
end
Dados1=[xRandom1;yRandom1]; % saving the data

for i=1:area2 % Distributing data on the second
area
xRandom2(i) = randi([50 100],1,1);
yRandom2(i) = randi([0 49],1,1);
end
Dados2=[xRandom2;yRandom2]; % saving the data

Dados_entrada=[Dados1 Dados2]; % saving all the
data
```

```
%% Uncomment for fixed input values area1=850
points and area2=150 points
%
Dados_entrada=[6,17,26,22,43,40,41,37,13,27,3,24,
48,14,37,25,49,0,45,44,3,32,39,4,46,5,5,25,41,43,
2,47,39,38,28,49,7,42,0,32,44,20,12,21,11,17,2,2,
0,32,18,30,22,37,19,7,22,30,18,14,32,16,21,7,49,4
7,20,12,36,31,46,30,47,46,12,16,43,25,32,1,15,35,
40,14,5,32,32,14,11,15,25,15,16,23,0,8,16,48,38,1
1,24,11,48,40,42,44,36,39,11,40,19,0,6,49,8,32,14
,19,29,13,18,3,40,9,27,38,44,45,43,48,45,41,34,6,
37,22,48,24,44,30,17,4,31,17,17,9,43,20,35,36,49,
14,14,26,48,29,0,40,11,41,34,0,17,35,24,30,21,43,
29,47,32,43,12,39,39,36,19,1,37,36,4,41,23,10,48,
26,2,1,20,44,34,8,25,42,10,9,6,27,3,35,40,21,39,1
2,38,6,41,39,4,4,7,34,21,48,18,21,35,28,15,41,16,
48,44,35,39,26,2,27,29,35,27,46,49,19,3,27,44,26,
17,43,16,10,47,25,35,1,33,4,23,12,30,20,34,4,32,1
2,11,21,35,28,31,44,45,26,20,11,22,11,7,27,46,40,
49,41,29,13,10,12,25,2,21,25,7,42,26,23,2,37,36,3
8,32,38,45,18,33,9,38,46,36,38,1,10,48,15,12,47,3
8,37,24,47,2,45,23,19,3,2,44,47,2,33,33,32,8,9,47
,49,13,30,43,44,28,35,28,13,38,24,12,20,20,5,20,1
6,1,22,44,0,1,8,39,13,33,47,7,18,8,21,35,36,12,17
,44,37,26,7,41,49,49,45,14,26,46,43,42,1,25,46,42
,48,1,23,11,32,4,7,12,26,5,2,13,12,28,49,26,43,41
,44,24,30,37,18,44,20,40,48,49,36,12,46,16,30,43,
38,27,12,36,13,9,49,4,40,21,47,31,25,33,45,20,41,
7,17,40,29,3,3,21,16,5,25,38,41,8,19,0,27,1,10,16
,46,22,14,7,38,44,26,28,1,36,36,24,40,40,7,13,35,
0,8,24,33,23,3,13,47,20,20,37,40,42,14,35,35,47,3
3,16,20,14,35,11,45,10,33,47,28,42,33,47,45,29,41
,17,25,20,17,40,45,2,5,7,14,42,49,6,28,17,31,26,4
9,21,9,19,29,23,47,12,17,40,4,46,37,18,16,25,31,2
,26,31,35,42,18,14,40,12,16,15,10,33,26,1,24,32,2
2,39,0,0,7,31,28,40,8,33,32,24,12,48,14,19,10,31,
3,45,28,20,1,29,34,36,42,33,23,32,17,23,49,18,45,
19,33,1,19,2,0,41,18,47,48,40,3,42,14,40,9,24,40,
1,23,7,37,23,44,1,44,42,8,42,40,23,10,47,42,36,15
,35,20,11,44,7,2,21,9,14,15,29,35,31,11,46,40,21,
14,48,19,11,49,10,45,31,12,30,20,19,46,31,6,22,30
,37,18,20,11,0,18,17,2,20,33,37,48,16,48,13,22,22
,26,13,18,43,38,48,30,36,18,47,34,23,30,38,21,18,
0,7,32,7,10,40,42,47,24,41,17,22,46,45,33,41,7,3,
```

5,7,24,28,17,26,36,40,41,28,26,36,4,46,2,2,48,38,
22,24,47,8,37,21,33,41,3,13,49,22,36,47,7,22,0,18
,10,21,12,16,1,11,41,29,44,30,19,45,28,37,5,35,13
,22,40,20,41,20,19,3,40,20,13,16,17,5,22,7,22,37,
11,5,42,15,8,16,18,6,8,2,6,37,11,43,48,2,12,24,5,
46,37,33,22,31,6,26,16,12,48,29,47,22,42,17,44,48
,6,22,12,36,7,31,47,41,22,43,19,42,25,15,41,48,17
,22,9,42,11,31,41,26,40,44,22,23,22,35,35,7,48,27
,28,28,6,3,23,82,86,77,99,69,87,99,67,90,63,54,91
,53,70,75,72,74,79,60,76,69,82,97,88,64,68,85,92,
85,66,98,52,57,83,76,86,67,86,63,85,72,87,68,54,6
4,93,52,76,95,94,73,73,100,52,50,56,96,82,81,72,8
8,54,97,63,78,50,97,83,83,69,86,51,51,92,72,59,65
,73,95,72,52,54,58,73,99,91,62,78,52,55,73,76,58,
55,65,79,70,94,63,80,99,72,92,81,90,93,100,52,51,
52,86,69,65,97,99,69,65,87,76,55,52,61,79,73,88,9
0,77,94,70,76,93,88,97,50,51,53,85,100,97,79,100,
63,51,59,96,86,65,80,67,66;39,25,4,18,44,17,21,9,
18,44,9,32,46,3,45,8,25,12,9,7,10,34,45,6,49,33,4
9,21,42,12,36,49,35,18,7,39,26,47,45,16,40,25,26,
46,37,21,30,45,2,29,22,14,3,47,47,24,0,10,41,31,4
3,23,6,25,9,46,47,49,29,16,18,24,20,17,35,1,8,20,
38,39,7,24,23,27,32,6,26,1,42,43,41,38,29,49,15,5
,18,42,42,44,5,27,40,32,26,14,49,29,42,7,36,39,39
,23,31,44,42,38,0,14,12,16,29,16,0,15,15,10,1,1,4
2,29,45,27,35,12,17,37,8,25,20,16,41,36,17,31,7,4
5,26,29,48,47,8,2,40,26,41,45,45,16,8,45,34,27,49
,33,33,9,0,30,30,25,2,39,49,23,7,22,12,24,1,40,24
,34,42,3,33,46,24,19,25,18,0,36,29,3,18,41,18,29,
30,49,35,0,3,2,34,37,40,25,16,33,16,42,11,5,11,18
,8,8,35,0,3,3,49,30,41,32,45,30,37,42,28,5,8,41,0
,40,28,0,49,34,24,3,4,11,5,35,30,47,40,45,15,14,2
2,44,46,8,21,30,18,33,17,35,3,28,16,43,0,18,27,47
,13,44,16,29,1,15,10,34,7,29,10,39,15,32,29,42,38
,44,3,33,15,24,10,16,17,40,8,22,15,36,38,3,23,48,
4,6,19,14,24,4,45,5,6,49,4,36,48,28,18,10,24,27,8
,9,5,3,15,3,34,5,19,10,20,26,16,19,30,44,49,48,32
,25,30,9,23,22,20,38,16,27,11,43,1,40,9,43,7,46,4
3,32,43,8,45,39,13,38,39,24,41,24,4,33,40,38,32,1
7,18,10,25,18,4,32,27,42,35,34,16,0,9,17,14,35,46
,44,44,12,12,20,16,13,47,28,49,39,38,28,44,14,9,4
3,12,35,23,15,14,45,8,17,34,48,15,12,43,16,21,30,
4,34,32,22,33,12,42,16,29,21,36,12,31,12,49,31,5,
4,36,47,11,43,21,31,21,49,38,44,25,15,5,12,49,22,

```
24,19,20,48,5,30,21,31,28,11,20,11,6,36,36,7,8,40
,28,5,44,26,27,19,9,21,40,1,19,22,15,41,0,28,46,6
,26,34,16,13,38,41,8,28,24,39,5,46,47,26,49,11,30
,41,34,45,26,4,23,46,0,24,41,49,48,19,15,0,35,21,
11,31,35,38,10,35,1,7,49,17,37,43,23,8,14,40,6,30
,7,35,39,14,39,27,44,4,42,16,20,38,46,49,26,24,8,
33,35,38,28,32,11,14,6,37,19,16,4,36,6,46,33,32,5
,45,22,17,41,38,40,26,2,5,0,40,36,20,14,5,49,29,4
8,31,4,10,27,30,27,31,34,15,20,23,45,47,8,3,42,34
,16,33,16,25,40,27,31,4,13,35,1,6,20,15,37,41,3,4
0,16,39,43,21,45,42,23,12,16,22,49,41,3,23,11,9,1
,20,32,13,47,17,29,35,31,31,31,0,37,0,12,19,36,1,
49,31,10,29,41,3,35,19,23,15,16,48,6,1,37,42,31,7
,11,48,38,39,8,34,38,47,14,24,11,31,31,20,2,39,15
,27,9,43,45,25,36,1,19,43,17,20,42,34,0,0,6,31,7,
43,3,14,20,40,3,45,4,18,49,24,41,47,34,11,1,27,39
,46,18,40,12,13,21,9,8,18,9,10,5,41,42,4,3,29,17,
21,44,23,33,14,31,49,34,32,22,0,29,17,13,23,0,48,
9,2,47,7,28,45,49,26,21,10,14,28,31,44,8,45,45,49
,33,21,32,21,25,46,43,13,32,30,48,48,19,36,18,18,
16,8,1,26,47,32,43,41,18,23,33,46,25,8,11,33,41,4
6,34,19,44,46,3,10,30,3,2,21,32,32,49,31,41,15,8,
25,33,45,35,10,41,35,36,35,30,35,39,10,26,47,23,0
,2,22,40,13,39,1,45,38,15,11,48,34,16,11,2,34,31,
25,17,36,20,12,5,24,3,49,39,46,39,38,1,25,15,0,43
,23,6,20,49,41,12,1,3,40,27,36,22,10,13,40,23,31,
19,11,7,2,34,41,44,19,4,8,18,26,33,26,44,27,30,16
,27,42,30,10,42,42,43,45,7,6,38,33,6,7,14,5,2,39,
11,39,13,28,42,9,48,24,34,49,24,17,22,31,3,42,38,
41,22,44,5,5,19,24,17,49,26,40,42,36,14,42,48,30,
14,35,32,36,43,34,42,2,42,0,27,23,28,17];
%
xRandom1=[6,17,26,22,43,40,41,37,13,27,3,24,48,14
,37,25,49,0,45,44,3,32,39,4,46,5,5,25,41,43,2,47,
39,38,28,49,7,42,0,32,44,20,12,21,11,17,2,2,0,32,
18,30,22,37,19,7,22,30,18,14,32,16,21,7,49,47,20,
12,36,31,46,30,47,46,12,16,43,25,32,1,15,35,40,14
,5,32,32,14,11,15,25,15,16,23,0,8,16,48,38,11,24,
11,48,40,42,44,36,39,11,40,19,0,6,49,8,32,14,19,2
9,13,18,3,40,9,27,38,44,45,43,48,45,41,34,6,37,22
,48,24,44,30,17,4,31,17,17,9,43,20,35,36,49,14,14
,26,48,29,0,40,11,41,34,0,17,35,24,30,21,43,29,47
,32,43,12,39,39,36,19,1,37,36,4,41,23,10,48,26,2,
1,20,44,34,8,25,42,10,9,6,27,3,35,40,21,39,12,38,
```

```
6,41,39,4,4,7,34,21,48,18,21,35,28,15,41,16,48,44
,35,39,26,2,27,29,35,27,46,49,19,3,27,44,26,17,43
,16,10,47,25,35,1,33,4,23,12,30,20,34,4,32,12,11,
21,35,28,31,44,45,26,20,11,22,11,7,27,46,40,49,41
,29,13,10,12,25,2,21,25,7,42,26,23,2,37,36,38,32,
38,45,18,33,9,38,46,36,38,1,10,48,15,12,47,38,37,
24,47,2,45,23,19,3,2,44,47,2,33,33,32,8,9,47,49,1
3,30,43,44,28,35,28,13,38,24,12,20,20,5,20,16,1,2
2,44,0,1,8,39,13,33,47,7,18,8,21,35,36,12,17,44,3
7,26,7,41,49,49,45,14,26,46,43,42,1,25,46,42,48,1
,23,11,32,4,7,12,26,5,2,13,12,28,49,26,43,41,44,2
4,30,37,18,44,20,40,48,49,36,12,46,16,30,43,38,27
,12,36,13,9,49,4,40,21,47,31,25,33,45,20,41,7,17,
40,29,3,3,21,16,5,25,38,41,8,19,0,27,1,10,16,46,2
2,14,7,38,44,26,28,1,36,36,24,40,40,7,13,35,0,8,2
4,33,23,3,13,47,20,20,37,40,42,14,35,35,47,33,16,
20,14,35,11,45,10,33,47,28,42,33,47,45,29,41,17,2
5,20,17,40,45,2,5,7,14,42,49,6,28,17,31,26,49,21,
9,19,29,23,47,12,17,40,4,46,37,18,16,25,31,2,26,3
1,35,42,18,14,40,12,16,15,10,33,26,1,24,32,22,39,
0,0,7,31,28,40,8,33,32,24,12,48,14,19,10,31,3,45,
28,20,1,29,34,36,42,33,23,32,17,23,49,18,45,19,33
,1,19,2,0,41,18,47,48,40,3,42,14,40,9,24,40,1,23,
7,37,23,44,1,44,42,8,42,40,23,10,47,42,36,15,35,2
0,11,44,7,2,21,9,14,15,29,35,31,11,46,40,21,14,48
,19,11,49,10,45,31,12,30,20,19,46,31,6,22,30,37,1
8,20,11,0,18,17,2,20,33,37,48,16,48,13,22,22,26,1
3,18,43,38,48,30,36,18,47,34,23,30,38,21,18,0,7,3
2,7,10,40,42,47,24,41,17,22,46,45,33,41,7,3,5,7,2
4,28,17,26,36,40,41,28,26,36,4,46,2,2,48,38,22,24
,47,8,37,21,33,41,3,13,49,22,36,47,7,22,0,18,10,2
1,12,16,1,11,41,29,44,30,19,45,28,37,5,35,13,22,4
0,20,41,20,19,3,40,20,13,16,17,5,22,7,22,37,11,5,
42,15,8,16,18,6,8,2,6,37,11,43,48,2,12,24,5,46,37
,33,22,31,6,26,16,12,48,29,47,22,42,17,44,48,6,22
,12,36,7,31,47,41,22,43,19,42,25,15,41,48,17,22,9
,42,11,31,41,26,40,44,22,23,22,35,35,7,48,27,28,2
8,6,3,23];
%
xRandom2=[82,86,77,99,69,87,99,67,90,63,54,91,53,
70,75,72,74,79,60,76,69,82,97,88,64,68,85,92,85,6
6,98,52,57,83,76,86,67,86,63,85,72,87,68,54,64,93
,52,76,95,94,73,73,100,52,50,56,96,82,81,72,88,54
,97,63,78,50,97,83,83,69,86,51,51,92,72,59,65,73,
```

```
95,72,52,54,58,73,99,91,62,78,52,55,73,76,58,55,6
5,79,70,94,63,80,99,72,92,81,90,93,100,52,51,52,8
6,69,65,97,99,69,65,87,76,55,52,61,79,73,88,90,77
,94,70,76,93,88,97,50,51,53,85,100,97,79,100,63,5
1,59,96,86,65,80,67,66];
%
yRandom1=[39,25,4,18,44,17,21,9,18,44,9,32,46,3,4
5,8,25,12,9,7,10,34,45,6,49,33,49,21,42,12,36,49,
35,18,7,39,26,47,45,16,40,25,26,46,37,21,30,45,2,
29,22,14,3,47,47,24,0,10,41,31,43,23,6,25,9,46,47
,49,29,16,18,24,20,17,35,1,8,20,38,39,7,24,23,27,
32,6,26,1,42,43,41,38,29,49,15,5,18,42,42,44,5,27
,40,32,26,14,49,29,42,7,36,39,39,23,31,44,42,38,0
,14,12,16,29,16,0,15,15,10,1,1,42,29,45,27,35,12,
17,37,8,25,20,16,41,36,17,31,7,45,26,29,48,47,8,2
,40,26,41,45,45,16,8,45,34,27,49,33,33,9,0,30,30,
25,2,39,49,23,7,22,12,24,1,40,24,34,42,3,33,46,24
,19,25,18,0,36,29,3,18,41,18,29,30,49,35,0,3,2,34
,37,40,25,16,33,16,42,11,5,11,18,8,8,35,0,3,3,49,
30,41,32,45,30,37,42,28,5,8,41,0,40,28,0,49,34,24
,3,4,11,5,35,30,47,40,45,15,14,22,44,46,8,21,30,1
8,33,17,35,3,28,16,43,0,18,27,47,13,44,16,29,1,15
,10,34,7,29,10,39,15,32,29,42,38,44,3,33,15,24,10
,16,17,40,8,22,15,36,38,3,23,48,4,6,19,14,24,4,45
,5,6,49,4,36,48,28,18,10,24,27,8,9,5,3,15,3,34,5,
19,10,20,26,16,19,30,44,49,48,32,25,30,9,23,22,20
,38,16,27,11,43,1,40,9,43,7,46,43,32,43,8,45,39,1
3,38,39,24,41,24,4,33,40,38,32,17,18,10,25,18,4,3
2,27,42,35,34,16,0,9,17,14,35,46,44,44,12,12,20,1
6,13,47,28,49,39,38,28,44,14,9,43,12,35,23,15,14,
45,8,17,34,48,15,12,43,16,21,30,4,34,32,22,33,12,
42,16,29,21,36,12,31,12,49,31,5,4,36,47,11,43,21,
31,21,49,38,44,25,15,5,12,49,22,24,19,20,48,5,30,
21,31,28,11,20,11,6,36,36,7,8,40,28,5,44,26,27,19
,9,21,40,1,19,22,15,41,0,28,46,6,26,34,16,13,38,4
1,8,28,24,39,5,46,47,26,49,11,30,41,34,45,26,4,23
,46,0,24,41,49,48,19,15,0,35,21,11,31,35,38,10,35
,1,7,49,17,37,43,23,8,14,40,6,30,7,35,39,14,39,27
,44,4,42,16,20,38,46,49,26,24,8,33,35,38,28,32,11
,14,6,37,19,16,4,36,6,46,33,32,5,45,22,17,41,38,4
0,26,2,5,0,40,36,20,14,5,49,29,48,31,4,10,27,30,2
7,31,34,15,20,23,45,47,8,3,42,34,16,33,16,25,40,2
7,31,4,13,35,1,6,20,15,37,41,3,40,16,39,43,21,45,
42,23,12,16,22,49,41,3,23,11,9,1,20,32,13,47,17,2
```

```
9,35,31,31,31,0,37,0,12,19,36,1,49,31,10,29,41,3,
35,19,23,15,16,48,6,1,37,42,31,7,11,48,38,39,8,34
,38,47,14,24,11,31,31,20,2,39,15,27,9,43,45,25,36
,1,19,43,17,20,42,34,0,0,6,31,7,43,3,14,20,40,3,4
5,4,18,49,24,41,47,34,11,1,27,39,46,18,40,12,13,2
1,9,8,18,9,10,5,41,42,4,3,29,17,21,44,23,33,14,31
,49,34,32,22,0,29,17,13,23,0,48,9,2,47,7,28,45,49
,26,21,10,14,28,31,44,8,45,45,49,33,21,32,21,25,4
6,43,13,32,30,48,48,19,36,18,18,16,8,1,26,47,32,4
3,41,18,23,33,46,25,8,11,33,41,46,34,19,44,46,3,1
0,30,3,2,21,32,32,49,31,41,15,8,25,33,45];
%
yRandom2=[35,10,41,35,36,35,30,35,39,10,26,47,23,
0,2,22,40,13,39,1,45,38,15,11,48,34,16,11,2,34,31
,25,17,36,20,12,5,24,3,49,39,46,39,38,1,25,15,0,4
3,23,6,20,49,41,12,1,3,40,27,36,22,10,13,40,23,31
,19,11,7,2,34,41,44,19,4,8,18,26,33,26,44,27,30,1
6,27,42,30,10,42,42,43,45,7,6,38,33,6,7,14,5,2,39
,11,39,13,28,42,9,48,24,34,49,24,17,22,31,3,42,38
,41,22,44,5,5,19,24,17,49,26,40,42,36,14,42,48,30
,14,35,32,36,43,34,42,2,42,0,27,23,28,17];

%% Uncomment to plot the input data distribution
before training SOM
%
% plot(xRandom1,yRandom1,'b.','MarkerSize',12);
hold on
% plot(xRandom2,yRandom2,'m.','MarkerSize',12);
hold on
% rectangle('Position', [0, 0, 100,
50],'LineWidth',0.5,'LineStyle','--');
% xline(50); txt = 'AREA 1'; text(20,51,txt)
% txt = 'AREA 2'; text(70,51,txt)
% pause
% close all;
%
%% TRAINING THE SOM
for u=1:1

    %% Self-Organizing Map input parameters
msize = [25 40];
```

```
sMap =
som_lininit(Dados_entrada','msize',msize,'hexa','
sheet','shape','toroid');

som_grid(sMap,'Coord',sMap.codebook); axis equal;
neurons=msize(1)*msize(2);

npstep=100; vstep=1000; o=ones(npstep,1); r=(1-
(1:vstep)/vstep);

for i=1:vstep
    radius_ini=neurons*(1/125);

    %% Training the Self-Organizing Map
  sMap =
som_seqtrain(sMap,Dados_entrada','tracking',0,...
              'trainlen',npstep,'samples',...

'alpha',0.1*o,'radius',(radius_ini*r(i))*o);

          %% Plot the data and the neurons grid
mapping animation
figure
plot(xRandom1,yRandom1,'b.','MarkerSize',12);
hold on
plot(xRandom2,yRandom2,'m.','MarkerSize',12);
hold on
xline(50); txt = 'AREA 1'; text(20,52,txt)
txt = 'AREA 2'; text(70,52,txt)
rectangle('Position', [0, 0, 100,
50],'LineWidth',0.5,'LineStyle','--');
hold on
title(sprintf('%d/%d training
steps',npstep*i,npstep*vstep)); axis equal;
hold on, som_grid(sMap,'Coord',sMap.codebook);
hold off

drawnow
axis equal;

end

neuron1=sum(floor(sMap.codebook(:,1))<=49); %
number of neurons on the first area
```

```
neuron2=sum(floor(sMap.codebook(:,1))>49); %
number of neurons on the second area

dens=[area1/(50^2) area2/(50^2)]; % Data density
densn=[neuron1/(50^2) neuron2/(50^2)]; % Neurons
density

%% Uncomment to plot average input area densities
vs. neuron area densities
% figure
% plot([1 50 50 100],[area1/(50^2) area1/(50^2)
area2/(50^2) area2/(50^2) ]); hold on;
% plot([1 50 50 100],[neuron1/(50^2)
neuron1/(50^2) neuron2/(50^2) neuron2/(50^2)]);
% txt = 'AREA 1';
% text(20,neuron2/(50^2),txt);
% txt = 'AREA 2';
% text(70,neuron1/(50^2),txt);

save(sprintf('2D_1000dados_1000neuronios_2areas_a
umentodeepocas%d.mat',u));% Saving all the
experiment

end
```

A.5. Two-dimensional SOM algorithm with four different density areas

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
%
% Name: main_rectangulo_4areas
%
% Objective:
%
% Input/Output Parameters:
%
% Obs: This matlab routine will generate 2-D
input data with four distinct
% density areas and subsequently train a Self-
Organizing Map.
%
% V1.0 - Moreira Bastos, Jun 2021
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%

area1 = 20; % Data points on the first area
area2 = 210; % Data points on the second area
area3 = 70; % Data points on the third area
area4 = 700; % Data points on the fourth area

%% Generating random input data
xRandom1=zeros(1,area1); xRandom2=zeros(1,area2);
xRandom3=zeros(1,area3); xRandom4=zeros(1,area4);
yRandom1=zeros(1,area1); yRandom2=zeros(1,area2);
yRandom3=zeros(1,area3); yRandom4=zeros(1,area4);

for i=1:area1 % Data on the first area
xRandom1(i) = randi([0 49],1,1);
yRandom1(i) = randi([50 100],1,1);
end
Dados1=[xRandom1;yRandom1];

for i=1:area2 % Data on the second area
xRandom2(i) = randi([50 100],1,1);
yRandom2(i) = randi([50 100],1,1);
end
Dados2=[xRandom2;yRandom2];

for i=1:area3 % Data on the third area
xRandom3(i) = randi([0 49],1,1);
yRandom3(i) = randi([0 49],1,1);
end
Dados3=[xRandom3;yRandom3];

for i=1:area4 % Data on the fourth area
xRandom4(i) = randi([50 100],1,1);
yRandom4(i) = randi([0 49],1,1);
end
Dados4=[xRandom4;yRandom4];

Dados_entrada=[Dados1 Dados2 Dados3 Dados4]; %
Saving all the data

%% TRAINING THE SOM
for u=1:10

msize = [5 5];
```

```
sMap =
som_lininit(Dados_entrada','msize',msize,'hexa','
sheet','shape','toroid');

som_grid(sMap,'Coord',sMap.codebook)
axis equal;
neurons=msize(1)*msize(2);

npstep=10; vstep=900; o = ones(npstep,1); r=(1-
(1:vstep)/vstep);

for i=1:vstep
    radius_ini=neurons*(1/125);

    %% Training the Self-Organizing Map
  sMap =
som_seqtrain(sMap,Dados_entrada','tracking',0,...
              'trainlen',npstep,'samples',...

'alpha',0.1*o,'radius',(radius_ini*r(i))*o);
    %% Plot the data and the neurons grid mapping
animation

% plot(xRandom1,yRandom1,'b.','MarkerSize',12);
hold on
% plot(xRandom2,yRandom2,'m.','MarkerSize',12);
hold on
% plot(xRandom3,yRandom3,'r.','MarkerSize',12);
hold on
% plot(xRandom4,yRandom4,'g.','MarkerSize',12);
hold on
% rectangle('Position', [0, 0, 100,
100],'LineWidth',2,'LineStyle','--');
% hold on, xline(50), yline(50);
% title(sprintf('%d/%d training
steps',npstep*i,npstep*vstep)); axis equal;
% hold on, som_grid(sMap,'Coord',sMap.codebook);
% hold off
%
% drawnow
% axis equal;

end
```

```
neuron1=0;neuron2=0;neuron3=0;neuron4=0; %
Initializing variables

%% Get the number of neurons on each area
for p=1:neurons
if sMap.codebook(p,1)<50 &&sMap.codebook(p,2)>49
neuron1=neuron1+1;
end
if sMap.codebook(p,1)>49 &&sMap.codebook(p,2)>49
neuron2=neuron2+1;
end
if sMap.codebook(p,1)<50 &&sMap.codebook(p,2)<50
neuron3=neuron3+1;
end
if sMap.codebook(p,1)>49 &&sMap.codebook(p,2)<50
neuron4=neuron4+1;
end
end

dens=[area1/(50^2) area2/(50^2) area3/(50^2)
area4/(50^2)]; % Data density
densn=[neuron1/(50^2) neuron2/(50^2)
neuron3/(50^2) neuron4/(50^2)]; % Neurons density

%% Uncomment to plot average input area densities
vs. neuron area densities
% figure
% plot([1 25 25 50 50 75 75 100],[area1/(50^2)
area1/(50^2) area2/(50^2) area2/(50^2)
area3/(50^2) area3/(50^2) area4/(50^2)
area4/(50^2)]); hold on
% plot([1 25 25 50 50 75 75 100],[neuron1/(50^2)
neuron1/(50^2) neuron2/(50^2) neuron2/(50^2)
neuron3/(50^2) neuron3/(50^2) neuron4/(50^2)
neuron4/(50^2)]);
% txt = 'AREA 1';
% text(5,neuron2/(50^2),txt)
% txt = 'AREA 2';
% text(30,neuron1/(50^2),txt)
% txt = 'AREA 3';
% text(55,neuron2/(50^2),txt)
% txt = 'AREA 4';
% text(85,neuron1/(50^2),txt)
```

```
save(sprintf('experiencia_2D_1000dados_25x40neuro
nios%d.mat',u)); % Save all the experiment

end
```

A.6. Three-dimensional SOM algorithm with two different density areas

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
%
% Name: main_3D
%
% Objective:
%
% Input/Output Parameters:
%
% Obs: This matlab routine will generate 3-D input data
with two distinct
% density areas and subsequently train a Self-Organizing
Map and generate
% Output on 3-D neuron grid
%
% V1.0 - Moreira Bastos, Jun 2021
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%

area1 = 9950; % Data points on the first area
area2 = 50; % Data points on the second area

%% INIITIALIZE INPUT DATA VARIABLES
xRandom1=zeros(1,area1); xRandom2=zeros(1,area2);
yRandom1=zeros(1,area1); yRandom2=zeros(1,area2);
zRandom1=zeros(1,area1); zRandom2=zeros(1,area2);


%% PLOT THE INPUT DATA
% plot3(xRandom1,yRandom1,zRandom1,'b.','MarkerSize',12);
hold on
% plot3(xRandom2,yRandom2,zRandom2,'m.','MarkerSize',12);
hold on


for u=1:30
```

```
%% GENERATE RANDOM INPUT DATA
for i=1:area1 % Distributing data on the first area
xRandom1(i) = randi([0 49],1,1);
yRandom1(i) = randi([0 49],1,1);
zRandom1(i)=randi([0 49],1,1);
end
Dados1=[xRandom1;yRandom1;zRandom1]; % saving the data

for i=1:area2 % Distributing data on the second area
xRandom2(i) = randi([0 49],1,1);
yRandom2(i) = randi([0 49],1,1);
zRandom2(i)=randi([50 100],1,1);
end

Dados2=[xRandom2;yRandom2;zRandom2]; % saving the data

Dados_entrada=[Dados1 Dados2]; % mixing and saving all the
data (lower density vs. higher density)

msize = [10 25 20]; % Initialize the neuron grid setup

%% TRAINING THE SOM

sMap =
som_lininit(Dados_entrada','msize',msize,'rect','sheet');
% SOM linear initialization

neurons=msize(1)*msize(2)*msize(3); % get the number of
neurons

epochs=500; % number of training epochs

sMap=
som_batchtrain(sMap,Dados_entrada','radius_ini',neurons/12
5,'radius_fin',0,'trainlen',epochs); %Training the SOM
using the batch algorithm

neuron1=sum(floor(sMap.codebook(:,3))<=49); % number of
neurons on the first area
neuron2=sum(floor(sMap.codebook(:,3))>49); % number of
neurons on the second area

dens=[area1/(50^3) area2/(50^3)]; % Data density
densn=[neuron1/(50^3) neuron2/(50^3)]; % Neurons density

%% Uncomment to plot average input densities vs. neuron
densities
% figure
%
```

```
% plot([1 50 50 100],[area1/(50^3) area1/(50^3)
area2/(50^3) area2/(50^3) ]); hold on;
% plot([1 50 50 100],[neuron1/(50^3) neuron1/(50^3)
neuron2/(50^3) neuron2/(50^3)]);
% axis equal;
%
% txt = 'AREA 1';
% text(20,neuron2/(50^3),txt);
% txt = 'AREA 2';
% text(70,neuron1/(50^3),txt);

save(sprintf('Magnification
Factor/experiencia_3D_%d.mat',u));% Saving all the
experiment

end
```

A.7. Calculating the magnification factor for a 1-D SOM

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
%
% Name: calculomagnificacao
%
% Objective:
%
% Input/Output Parameters:
%
% Obs: This matlab routine calculates the som
magnification factor as well
% as the magn. constant for a given heaviside 1D input
data density.
%
% V1.0 - Moreira Bastos, Jun 2021
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
%% FORMULAES
% d1=K*d1'^alfa;
% d1=K*d2'^alfa;
% alfa=log(d1/d2)/log(d1'/d2');
% K=d2/d2'^alfa ou K=d1/d1'^alfa;
%
%
format long
expoente_e_constante=zeros(2,2);

for i=1:499
```

```
load(sprintf('heaviside%d.mat',i),'densidade_output','dens
idade_input','neurons');

alfa=log(densidade_output(1)/densidade_output(3))/log(dens
idade_input(1)/densidade_input(3));
K=densidade_output(3)/densidade_input(3)^alfa;

expoente_e_constante(i,1)=alfa;
expoente_e_constante(i,3)=K;

end
save('expoente_e_constante.mat','expoente_e_constante');
```

A.8. Calculating the magnification factor for a 2-D SOM

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
%
% Name: calculodemagnificacao
%
% Objective:
%
% Input/Output Parameters:
%
% Obs: This matlab routine calculates the som
magnification factor as well
% as the magn. constant for a given 2D input data density.
%
% V1.0 - Moreira Bastos, Jun 2021
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
%% FORMULAES
% d1=K*w1^alfa;
% d2=K*w2^alfa;
% alfa=log(d1/d2)/log(w1/w2);
% K=d2/w2^alfa or K=d1/w1^alfa;
%
format long
expoente_e_constante=zeros(2,2);

for i=1:140

load(sprintf('Magnification
Factor/experiencia_2Dpara1D%d.mat',i),'dens','densn');

w=dens; d=densn;
```

```
alfa=log(d(1)/d(2))/log(w(1)/w(2)); %
alfa=log(d1/d2)/log(w1/w2);
K=d(1)/w(1)^alfa; % K=d2/w2^alfa or K=d1/w1^alfa;

expoente_e_constante(i,1)=alfa; % Alfa

expoente_e_constante(i,3)=K; %Constant K

end
save('expoente_e_constante.mat','expoente_e_constante');
```

A.9. Calculating the magnification factor for a 3-D SOM

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
%
% Name: main_3D
%
% Objective:
%
% Input/Output Parameters:
%
% Obs: This matlab routine calculates the som
magnification factor as well
% as the magn. constant for a given 3D input data density
%
% V1.0 - Moreira Bastos, Jun 2021
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
%
%
%
%% FORMULAES
% d1=K*w1^alfa;
% d2=K*w2^alfa;
% alfa=log(d1/d2)/log(w1/w2);
% gamma=d1/n*w1
% K=d2/w2^alfa or K=d1/w1^alfa;
%
%%
% Search for magnification exponent and magnification
constant for each SOM
% experiment done
%
format long
expoente_e_constante=zeros(2,2);

for i=1:30
```

```
load(sprintf('Magnification
Factor/experiencia_3D_%d.mat',i),'dens','densn'); % Load
data from SOM (3D to 3D experiment)
%load(sprintf('Magnification Factor 3D to
2D/experiencia_3D_to_2D%d.mat',i),'dens','densn'); % Load
data from SOM (3D to 2D experiment)
%load(sprintf('Magnification Factor 3D to
1D/experiencia_3D_to_1D%d.mat',i),'dens','densn'); % Load
data from SOM (3D to 1D experiment)

w=dens; % input data densities
d=densn; % neuron densities

alfa=log(d(1)/d(2))/log(w(1)/w(2)); %
alfa=log(d1/d2)/log(w1/w2);
K=d(1)/w(1)^alfa; % K=d2/w2^alfa or K=d1/w1^alfa;

expoente_e_constante(i,1)=alfa; % Alfa
expoente_e_constante(i,3)=K; %Constant K

end

save('expoente_e_constante_3D.mat','expoente_e_constante')
;
%
save('expoente_e_constante_3D_para_2D.mat','expoente_e_con
stante');
%
save('expoente_e_constante_3D_para_1D.mat','expoente_e_con
stante');
```

B. Appendix B – Relationship between the neighbourhood function initial radius and the number of neurons for the one-dimensional case

From the set of experiments, we conclude that the initial radius does not influence the magnification factor. However, since the magnification factor values are stable for lower initial radius, we opt to use 20% of the total number of neurons.

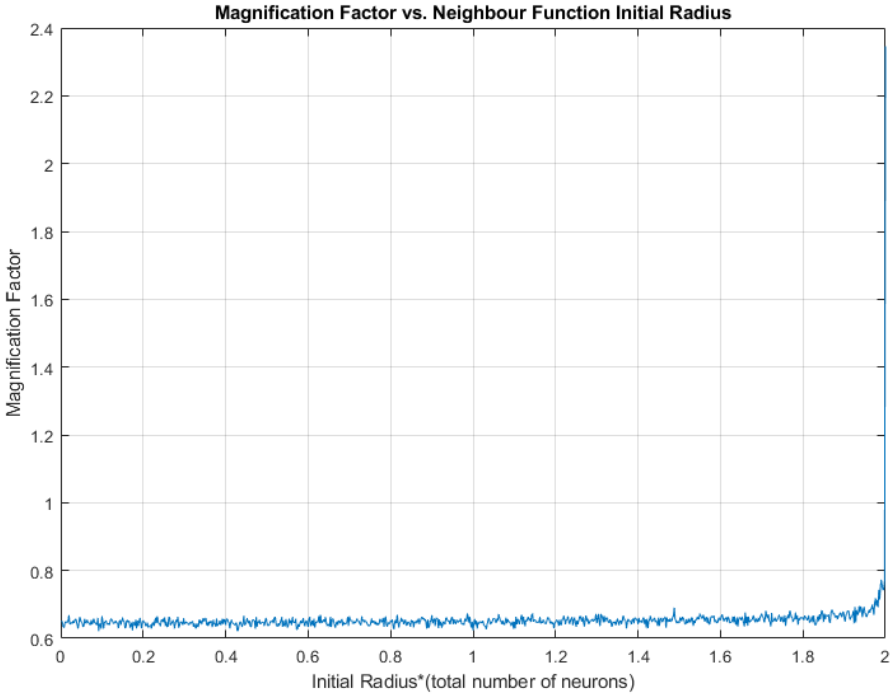


Fig. 1 - Magnification factor vs neighbourhood function initial radius

We believe that the essential matter to analyse is the training epochs since we proved that this parameter influences the magnification factor substantially.

C.Appendix C – Angola’s demographic information

Table 13 - Angola's Demographic data collected from Angolan 2014 Census

Province	**Population**	**Area (km^2)**	**Population density (people/km^2)**
Zaire	594428	37230	16
Uige	1483118	62579	24
Malanje	986363	86805	12
Lunda Sul	537587	82827	7
Lunda Norte	862566	99363	9
Luanda	6945386	18826	369
Benguela	2231385	39105	58
Namibe	495326	57119	9
Huambo	2019555	33296	61
Huíla	2497422	78898	32
Bié	1455255	70746	21
Cunene	990087	77156	13
Cuando Cubango	534002	201247	3
Moxico	758568	203203	4
Cuanza Sul	1881873	55574	34
Cabinda	716076	7235	99
Cuanza Norte	443386	20429	22
Bengo	356641	20298	18