****

Abstract

Designing Neural Networks for pattern Recognition and Self Organized Maps in Optical Lens Classification

Design and Application of ai

Intelligent Systems EEM7010

MSc in Mechatronic Systems Engineering and Engineering Management

Simone Perfetto

Contents

[**Neural Network Structure** 3](#_Toc92588291)

[**Transfer functions** 4](#_Toc92588292)

[**Inputs and Outputs** 5](#_Toc92588293)

[**Neural Network Learning Algorithms** 5](#_Toc92588294)

[Linear Regression 6](#_Toc92588295)

[Perceptron algorithm 6](#_Toc92588297)

[Hebb’s Rule 7](#_Toc92588298)

[Backpropagation 8](#_Toc92588299)

[**Procedure of learning using the toolbox in Matlab** 9](#_Toc92588300)

[Pattern Recognition – Supervised Learning 9](#_Toc92588301)

[**PLOT Results for Pattern Recognition Supervised Learning** 9](#_Toc92588303)

[**Performance** 10](#_Toc92588304)

[**Training State** 10](#_Toc92588305)

[**Error Histogram** 11](#_Toc92588306)

[**Confusion Matrix** 12](#_Toc92588307)

[**ROC Curve** 13](#_Toc92588308)

[Clustering Classification, SOM – Unsupervised Learning 13](#_Toc92588310)

[**PLOT Results for Unsupervised Learning** 14](#_Toc92588311)

[**Conclusion** 16](#_Toc92588312)

[**APPENDIX** 17](#_Toc92588313)

[**My Pattern Recognition Neural Network (SIMULINK)** 17](#_Toc92588314)

[**My Clustering Neural Network (SIMULINK)** 18](#_Toc92588318)

[Academic Books 19](#_Toc92588319)

[Academic Journals 19](#_Toc92588325)

Table of Figures

[Figure 1) Artificial Intelligence's neural network 3](#_Toc92588370)

[Figure 2) Multiple-layer neural network (non-abbreviated notation) 3](#_Toc92588371)

[Figure 3) Multiple-layer neural network (Abbreviated notation) 4](#_Toc92588372)

[Figure 4) Transfer functions commonly used in Neural Networks 4](#_Toc92588373)

[Figure 5) Simple Linear Regression 6](#_Toc92588374)

[Figure 6) Decision Boundary example in Perceptron algorithm 6](#_Toc92588375)

[Figure 7) Anatomy of a Synapse correspondence with machine learning. 7](#_Toc92588376)

[Figure 8) Backpropagation Summary 8](#_Toc92588377)

[Figure 9) Neural Network for Pattern Recognition (SIMULINK network diagrams are in APPENDIX). 9](#_Toc92588378)

[Figure 10) Neural Network for Pattern Recognition: Cross-Entropy Performance analysis. 10](#_Toc92588379)

[Figure 11) Neural Network for Pattern Recognition: Training State. 10](#_Toc92588380)

[Figure 12) Neural Network for Pattern Recognition: Error Histogram 11](#_Toc92588381)

[Figure 13) Neural Network for Pattern Recognition: Confusion Matrices. 12](#_Toc92588382)

[Figure 14) Neural Network for Pattern Recognition: ROC Curves. 13](#_Toc92588383)

[Figure 15) SOM Plots: A) Neighbour Weight Distance B) Weight Planes C) Weight Positions D) Sample Hits (10x10). 14](#_Toc92588384)

[Figure 16) SOM Plots: A) Neighbour Weight Distance B) Weight Planes C) Weight Positions D) Sample Hits (20x20). 15](#_Toc92588385)

[Figure 17) SOM Neural Network Diagrams for default settings (A) and adjusted settings (B). 16](#_Toc92588386)

[Figure 18) SIMULINK Recognition Neural Network 17](#_Toc92588387)

[Figure 19) SIMULINK Recognition Neural Network: Layer 1 17](#_Toc92588388)

[Figure 20) SIMULINK Recognition Neural Network: Layer 1 DELAYS 17](#_Toc92588389)

[Figure 21) SIMULINK Recognition Neural Network: Layer 1 **P**, **W**, f and **a** 17](#_Toc92588390)

[Figure 22) SIMULINK Recognition Neural Network: Layer 2 17](#_Toc92588391)

[Figure 23) SIMULINK Recognition Neural Network: Layer 2 DELAYS 17 Figure 24) SIMULINK Recognition Neural Network: **a1, W2,** softmax**, a2** 17](#_Toc92588392)

[Figure 25) SIMULINK Clustering Neural Network 18](#_Toc92588393)

[Figure 26) SIMULINK Clustering Neural Network: Layer 1 18](#_Toc92588394)

[Figure 27) SIMULINK Clustering Neural Network: Process Input 1 18](#_Toc92588395)

[Figure 28) SIMULINK Clustering Neural Network: Process Output 1**References** 18](#_Toc92588396)

# **Neural Network Structure**

**Neural networks algorithms** are used in machine learning to help model nonlinear and complex problems and patterns of the real world. They are often used in pattern recognition as they are able to generalise and model a high volume of data with high volatility. A standard neural network structure comprises of **inputs, hidden layers (optional), and outputs.**



Figure ) Artificial Intelligence's neural network

**Inputs** are the source of information to pass on to the **neuron** in the **hidden** layers, connected to them via a **weight** matrix. The neuron is connected also to a “**bias**” parameter which sums to the product between weight and input through a **summer**, creating the **net** **input**. This delivers the information to a **transfer** **function**, or activation function. The latter builds a mathematical **model** from the input to find a corresponding **output**. In a **single-input** neuron network, the output will be a **scalar**.

In **multiple-input** neuron networks (or even a **layer** of neurons in parallel) the output will be a **vector** of elements corresponding to each neuron’s output.

In **multiple-layer** networks the result will be different layers of outputs corresponding to the layers of neurons. For **layer 1** the input **a0** will result in an output **a1**. Whereas the input for layer 2 equals the output for layer 1 and so on. Thus, each output in each layer becomes the input for the successive layer.

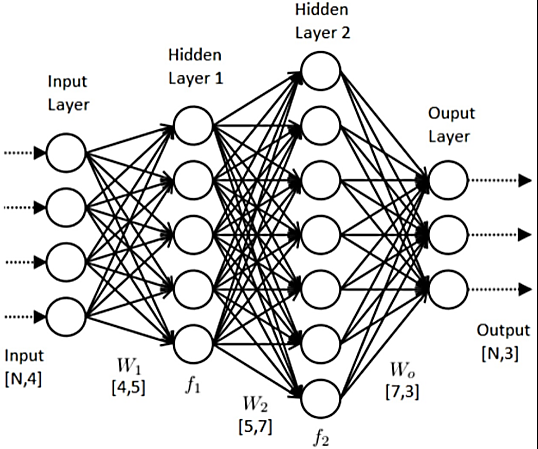


Figure ) Multiple-layer neural network (non-abbreviated notation)

Notations for the last cases can be messy therefore abbreviated notations are often used for clarity.

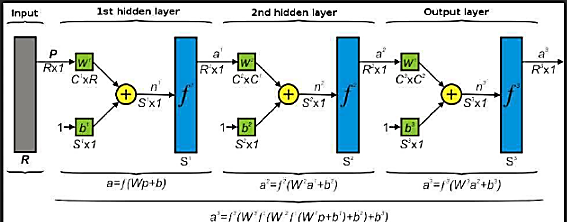


Figure ) Multiple-layer neural network (Abbreviated notation)

# **Transfer functions**

Transfer functions create mathematical models for the net input to derive and adjust the corresponding output. They are chosen to satisfy specific complex problems and therefore may differ. The most common ones are: **hard limit, log-sigmoid and pure linear**. Each transfer function is characterised by a unique symbol in every neural network diagram representing the relationship between the input and the output of that layer in the model.

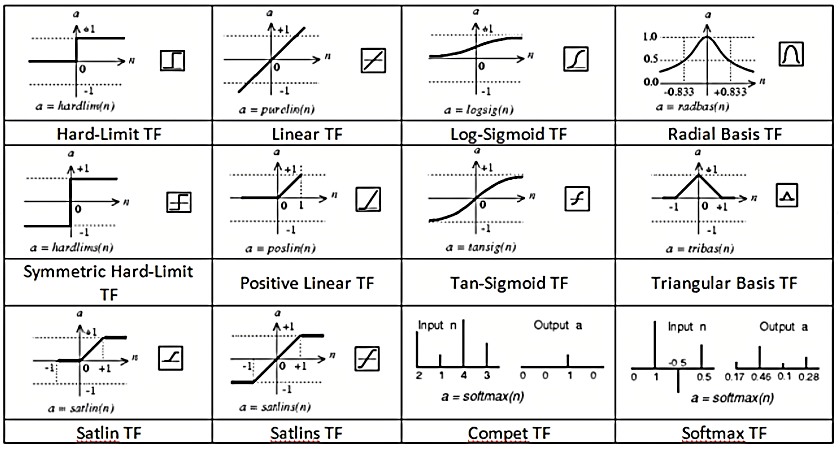


Figure ) Transfer functions commonly used in Neural Networks

The **hard limit transfer function** changes a net input value that is greater than or equal to zero to an output of 1, and net input values that are less than zero to an output of 0. This algorithm is usually applied to classify inputs in two separate categories. A common variation of this algorithm is the symmetrical hard limit transfer function, which turns inputs lower than 0 into an output equal to -1, while the rule for inputs that are greater than or equal to zero stay the same.

In the **pure linear transfer function,** the output is equivalent to the input of the network. The symbol for pure linear is a straight line for symmetrical linear increase/decrease of the change of outputs which are equal to the inputs: .

The **log sigmoid transfer function** is used in multilayer neural networks that are trained with backpropagation and allow inputs from minus to plus Infinity. They limit the output to range between zero and one only. A great advantage of this algorithm is that it is differentiable. The relationship between input and output:

# **Inputs and Outputs**

The outputs are very commonly outnumbered by the inputs if we have a vast set of data. The number of neurons can also differ and the more inputs we have the more weights we need to connect each input to each neuron in the layer. Indices are important notations for diagrams: subscript notations of capital R abbreviate the input elements; weights can have 2 subscripts where S and R represent which neuron and input it is connecting respectively.

For a multiple layer network, a superscript represents the exact layer of neurons that is being referenced.

# **Neural Network Learning Algorithms**

**Supervised Learning** requires both inputs and target outputs to train the model for a strong prediction accuracy and power.

**Reinforced Learning** uses GRADES or SCORES as target outputs which gauge how accurately the model is performing with the provided inputs. Less popular than supervised learning technique and mostly applied in control systems.

**Unsupervised Learning** requires no external and no target outputs, relying solely on the inputs to train the designed model. It groups, interprets and finds hidden patterns inside the data in exploratory data analysis (examples of applications for this algorithm are bioinformatics, data mining and medical imaging). A popular technique in unsupervised learning is clustering, subdivided into two categories:

**Soft clustering**: each input data point falls under only one cluster. Examples of such algorithms are: K-Means, K-Medoids, Hierarchical clustering, Self-organizing Map.

**Hard clustering**: each cluster can contain multiple data points that share similarities. Examples of such algorithms are: K-Means, K-Medoids, Hierarchical clustering, Self-organizing Map.

## Linear Regression

Gradient m and intercept c are unknown parameters and are found using an error function which returns the two variables in the matrix form: where is the pseudoinverse of

## 

Figure ) Simple Linear Regression

## Perceptron algorithm

the matrix containing all the input data points.

Perceptron network is a Supervised learning algorithm used for classification where the equation for net input set to zero, creates a decision boundary and classifies the corresponding outputs as either outside the decision boundary or inside it (shaded area).

A single-input perceptron is limited for linearly inseparable categories, whereas a multi-layered perceptron is not and is useful in Backpropagation algorithms.

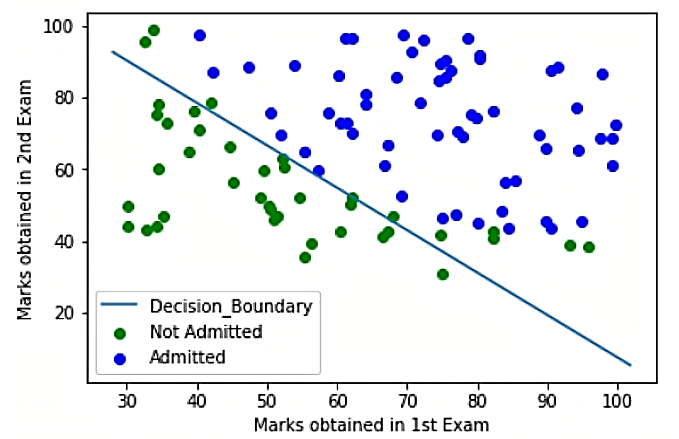


Figure ) Decision Boundary example in Perceptron algorithm

## Hebb’s Rule

Uses the analogy where two neurons of a synapse activated simultaneously increases efficiency of the synapse.

**In computational**, the connecting **weight** between **input (PRESYNAPTIC SIGNAL)** and **output (POSTSYNAPTIC SIGNAL)** becomes greater.

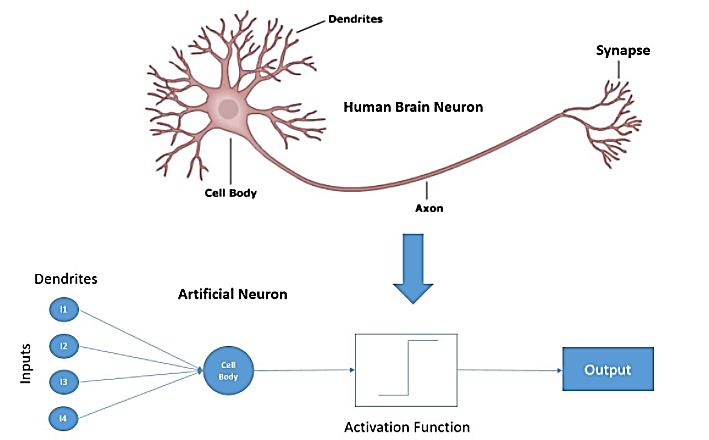
****

Figure ) Anatomy of a Synapse correspondence with machine learning.

The equation for Hebb Rule in **UNSUPERVISED LEARNING** is:

Weight **INCREASES** when p and a are both positive or both negative but **DECREASES** when they are opposite sign.

The equation for Hebb Rule in **SUPERVISED LEARNING** is:

If the inputs are either not normalised or not orthogonal, Hebb’s rule would derive a wrong result. In which case, PSEUDOINVERSE RULE is applied to find the right output.

The equation for Pseudoinverse Rule is:

## Backpropagation

This algorithm is a Supervised generalisation of the least mean squared (LMS) algorithm and uses mean square error as performance index for any mathematic model. This algorithm is commonly characterised by multi-layered Perceptrons and applies error correction learning through gradient descent. Thus, its purpose is to improve the efficiency of the model by minimizing the mean square error.

The network’s outputs are compared with the targets and the parameters are adjusted accordingly. The weights and bias, arbitrary values from 0 to 1, are updated by taking the chain rule of the partial derivatives of the Loss functions derived with Taylor Series, to find the corresponding functions for weight and bias in terms of the partial derivatives of Transfer functions and errors.

In the multi-layer Network, every output becomes the next layer’s input until the network’s final output is reached.

Important equations to consider are:

1. **where**: L = loss function, e = error, k = iteration

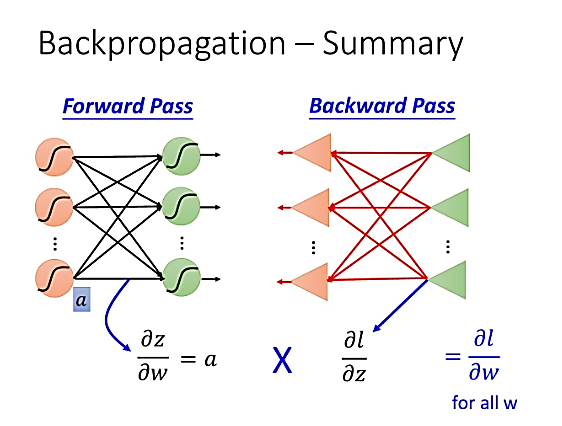


Figure ) Backpropagation Summary

# **Procedure of learning using the toolbox in Matlab**

## Pattern Recognition – Supervised Learning

Matlab Deep Learning Pattern Recognition toolbox uses supervised learning which requires to set up both inputs and target matrices which are imported in Matlab workspace and selected when setting up the parameters for the toolbox (automatically set up with the necessary code lines for the algorithm to work, which is available at the end of the process if the user wants to learn and modify the algorithm). The toolbox sets a default number of ten neurons to use in the Hidden layer, as well as separating number of samples which are used for training, validation and testing of the algorithm.

**Training**: this often taken most of the available data so that the software builds an algorithm to train it for pattern recognition and make predictions over successive new sets of data.

**Validation**: new data sets are infused in the algorithm during its training process to validate how well the model performs in predicting correct outputs for data that it had no access to originally.

**Testing**: this set of data is to finalise the algorithm accuracy and prediction power after the model is built by testing it with another new set of data. This process provides a final version of the real-world model to verify if it works effectively in predicting results for other unseen data.

These 3 categories get plotted into performance curves, training states, error histograms and confusion matrices. The latter are used to plot Receiver Operating Characteristic (ROC) curves, achieved by plotting the True Positive Rate against the False Positive Rate.

In the Optical Lens problem, the inputs were typed out in columns inside Notebook on Windows 10 and the Targets were typed in a separate file before importing both datasets in the Matlab workspace, using the New Line (or blank space) as delimiter for the software to identify the different elements as individual values.

# 

Figure ) Neural Network for Pattern Recognition (SIMULINK network diagrams are in APPENDIX).

# **PLOT Results for Pattern Recognition Supervised Learning**

The plots my model has measured to analyse performance and efficiency are the following:

* Performance.
* Training State.
* Error Histogram.
* Confusion Matrix.
* Receiver Operating Characteristic (ROC) curve.

## **Performance**

The toolbox used a cross-entropy performance to return results which heavily penalize predicted outputs that are extremely inaccurate, or extremely far from the target outputs. Good classifiers are characterised by minimal cross-entropy (Mathworks). My model recorded a performance of 0.00597 which makes be a very efficient classifier.

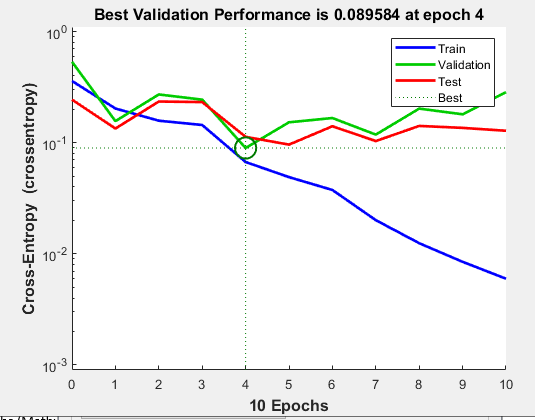


Figure ) Neural Network for Pattern Recognition: Cross-Entropy Performance analysis.

## **Training State**

My model used a Scaled Conjugate Gradient training network that adjusts the weights and bias parameters using backpropagation algorithm and does the same to measure derivates of performance with respect to the weights and bias variables and stops when the performance has been minimalised to the goal (0) or has reached its maximum amounts of repetitions, also known as epochs (Mathworks).

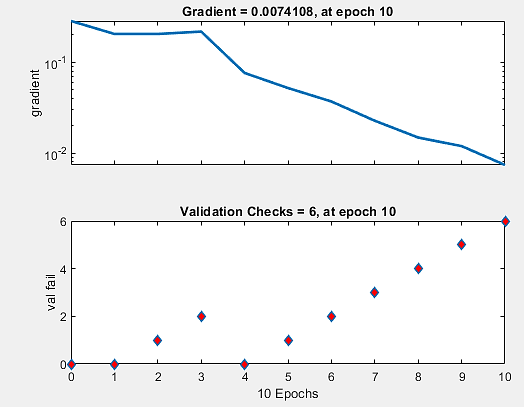


Figure ) Neural Network for Pattern Recognition: Training State.

## **Error Histogram**

The model plotted a histogram presenting instances where most of the repetitions have a minimal error where the it ranges from bins 0 to 0.229. The most amount of error performed by the model is in the bins -0.5849 and 0.6982. These are relatively high errors performed by the final testing part od the process which mean the model could need some retraining but since the instances showing these results are lowest in the histogram it is an acceptable result. The majority of the instances show little errors, whose extremes range from 0.04 to 0.02, to no error at all.

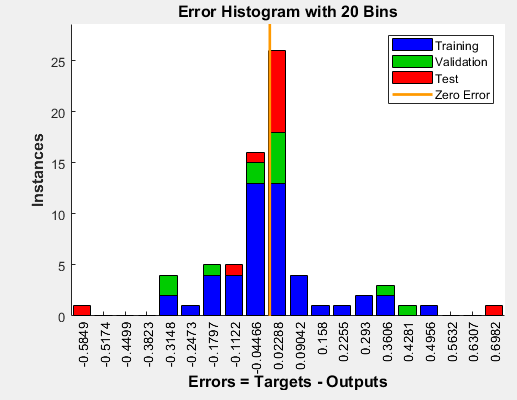


Figure ) Neural Network for Pattern Recognition: Error Histogram

## **Confusion Matrix**

The confusions matrix is a straight forward tabular measurement of how my efficient my model’s testing process performs in predicting and classifying outputs or classes.

The four classes that every confusion matrix measures are:

* True Positive (TP, green): model correctly classifies a class in the positive class.
* True Negative (TN, green): model correctly classifies a class in the negative class.
* False Positive (FP, red): model incorrectly classifies a class in the positive class.
* False Negative (FN, red): model incorrectly classifies a class in the negative class.

The model classified the data set in 3 distinct output and target classes for training, validation and testing, I will measure the categories relatively to the ALL confusion matrix.

For 1st class: TP = 4; TN = 19; FP = 1; FN = 0.

For 2st class: TP = 5; TN = 19; FP = 0; FN = 0.

For 3st class: TP = 14; TN = 9; FP = 0; FN = 1.

For EVERY class: TP = 23; TN = 47; FP = 0; FN = 1.

The TP for every class indicates that the success rate of my model was 98%, a near perfect efficiency value.

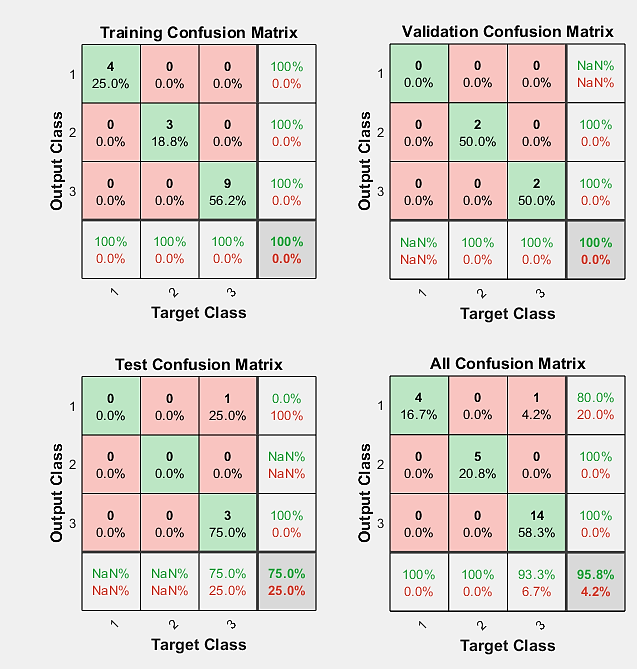
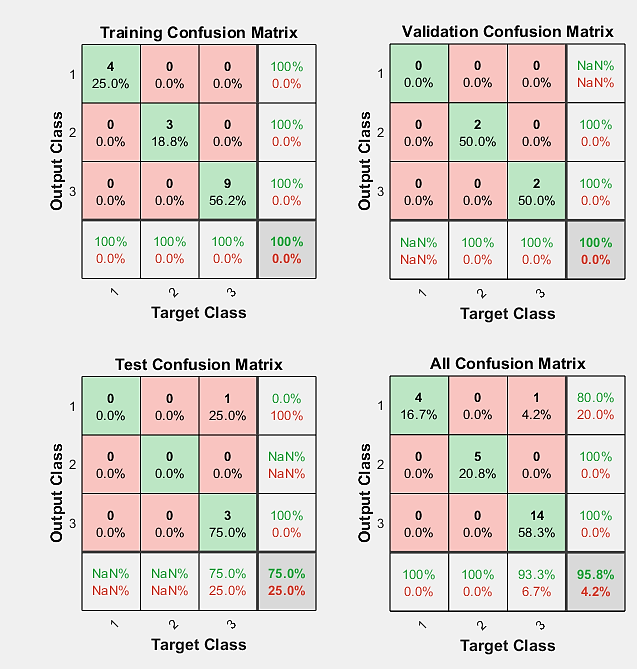
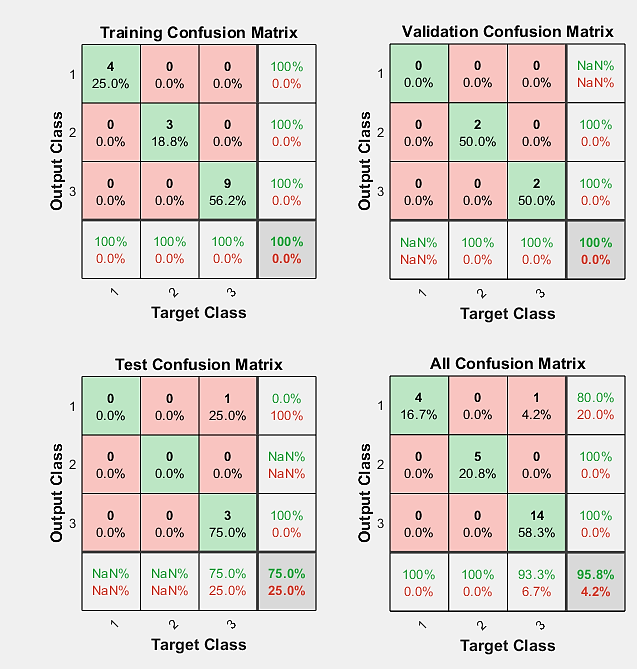
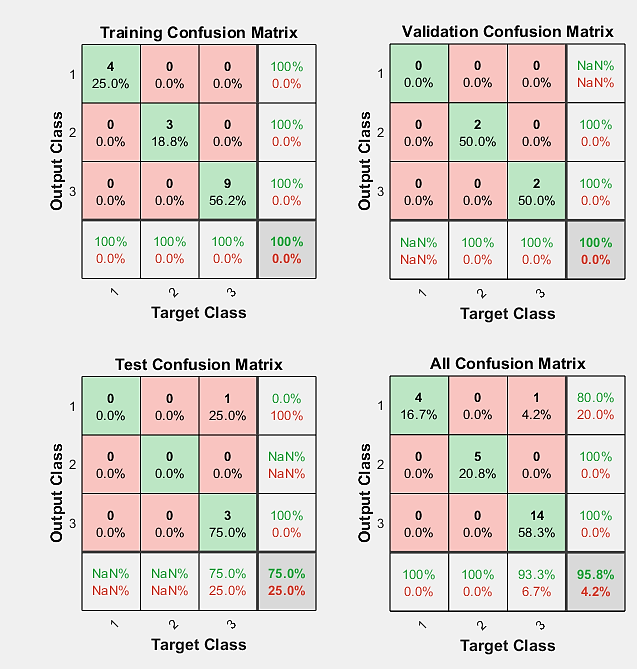


Figure ) Neural Network for Pattern Recognition: Confusion Matrices.

**Accuracy:** total samples correctly classified by my model over the total samples.

**Misclassification Rate:** fraction of predictions classified incorrectly.

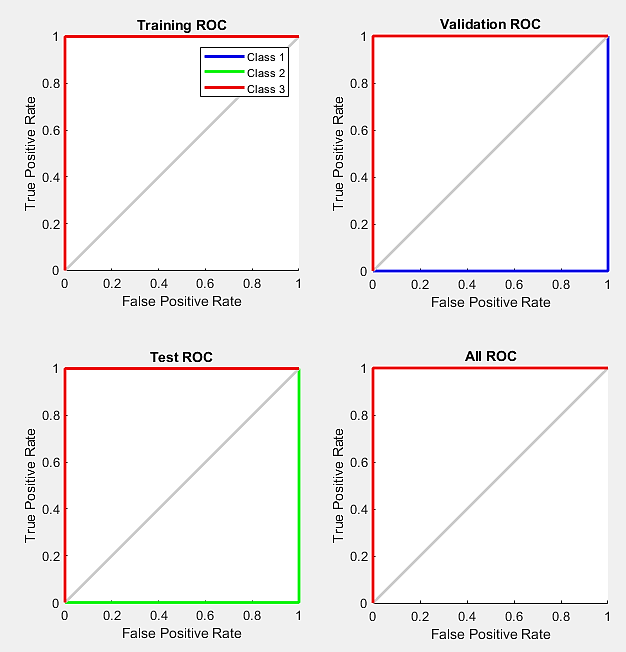
**Precision:** how many positive predictions were correctly classified as positive.

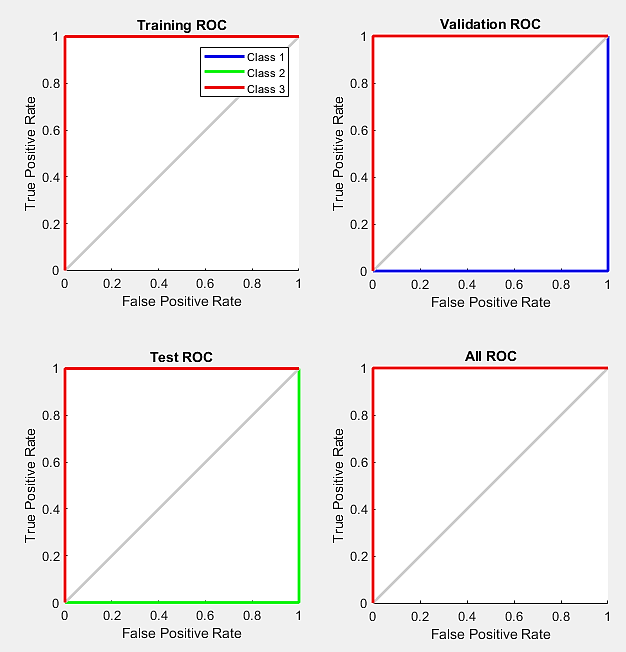
**Recall:** how may of ALL positive samples were predicted correctly by the model. Measurement of True Positive Rate (**TPR**), Sensitivity, Probability of Detection.

**Specificity:** how many samples are correctly predicted as negative over all the negative samples. Measurement of True Negative Rate (**TNR**).

**F1-Score:** harmonic mean of precision and recall.

## **ROC Curve**

The ROC curve is a useful plot of True Positive Rate against False Positive Rate. Thus, measures the relationship between the sensitivity and specificity of every cut-off and shows how accurately my model can distinguish between True Positives and Negatives. Essentially, the more the curve dominates the upper left corner of the graph, having a greater area under the curve (**AUC**), the greater the efficiency of the model. My model showed good results as in every step of training the curves stay predominantly on the upper left side of the graph. The only concerns is the class 2 samples which show terrible results in the TEST ROC.



## 

Figure ) Neural Network for Pattern Recognition: ROC Curves.

## Clustering Classification, SOM – Unsupervised Learning

A self-organising map is an unsupervised learning method that, differently from backpropagation, applies competitive learning and uses a neighbourhood function within their input space to preserve it. A common example of SOM applied in Neural Networks is Kohonen mapping.

Whilst in a competitive Kohonen network algorithm only one winning weight is rewarded with becoming most similar to the vector input sent in the network in a winner-takes-all process (following measurements of Euclidean distance) in the Kohonen SOFM the weight closest as well as the neighbouring weights are recognised and rewarded with the similarity.

The algorithm follows a decaying rule so that over the course of the epochs the learning rate decreases with the winning node’s neighbourhood. Weight parameters more distant from the winning weight, known as the **Best Matching Unit (BMU)** are awarded less similarity. The procedure is repeated with the input vectors until the number of neurons (arbitrary for the network designer) are all clustered into different classes.

The closer a neuron is to the BMU the more its weights are adjusted to similarity, and less the more distant it is to the winning neuron.

# **PLOT Results for Unsupervised Learning**

The following results and plots were taken using the default size of the 2-D map in Network Clustering Matlab tool: 10.

This designed a 10x10 map, therefore using 100 neurons in the network.

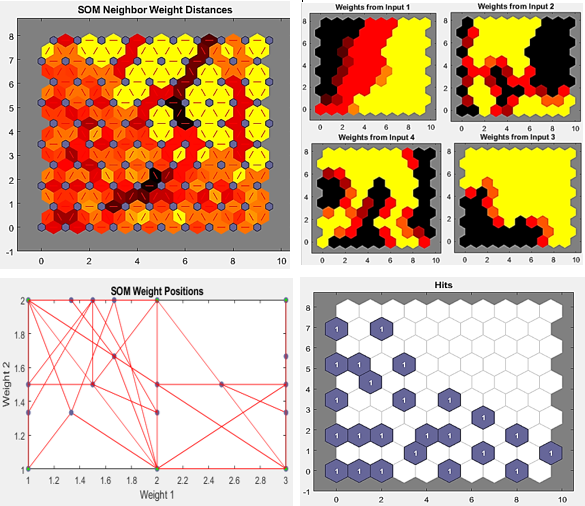
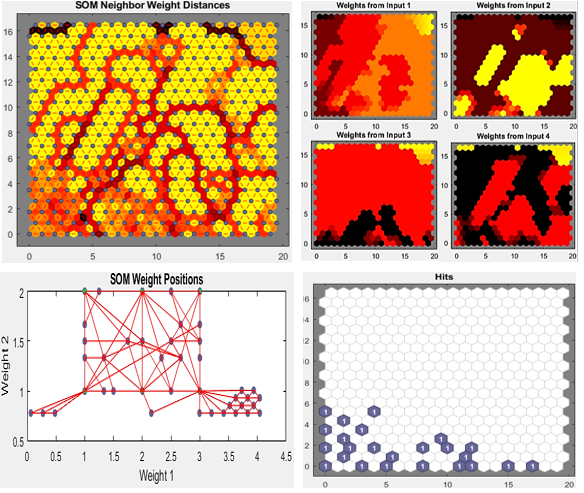


Figure ) SOM Plots: A) Neighbour Weight Distance B) Weight Planes C) Weight Positions D) Sample Hits (10x10).

These results are not bad but they show scarce and very generalised clusters of my data due to the map size probably being small. Figures 7A and 7B present clusters where the darker colours represent clusters with a shorter average distance between neighbourhood neurons with the most similar weights. The colours get lighter as dissimilarity increases within the clusters. The ones with most similarity are mostly represented in the corners of Fig. 7B.

Plot C does not present any clustering. Most nodes are all at long distances therefore need to increase the model’s accuracy. This was achieved by increasing the number of neurons used to 400 with a 20x20 map size.

This step saw an improvement in figure 8C, showing more accurate clusters to the dataset of inputs provided.

Figure ) SOM Plots: A) Neighbour Weight Distance B) Weight Planes C) Weight Positions D) Sample Hits (20x20).

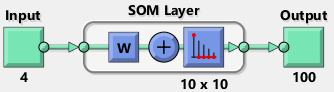
Figures 7A and 7B show ones where the most similarity is represented in the bottom right square of 7B. Sample Hits represents clusters representing how many times the neuron won and only 4 clusters whereas the empty white hexagons represent distance between the rest of the clusters.

# **Conclusion**

Artificial Intelligence aids for faster and more reliable models. The ones designed for these measurements showed very satisfying results.

The supervised learning method had great performance although it needs more training as some uncertainties and errors were still present.

Unsupervised learning started out not so great with the default map size and neuron numbers that were used for the self-organized mapping. But, adjusting the SOM algorithm is a trial-and -error process and it was great to see much progress over the second attempt already after increasing the map size to double the original size and introducing 4x the number of neurons. Densities of similar clusters were represented clearly although it is not clear which classes are being represented in them. The model has been able to set a clear pattern and cluster the data into similar classes finding correlation between the 4 inputs.



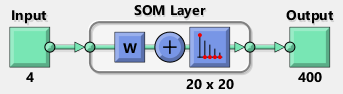


Figure ) SOM Neural Network Diagrams for default settings (A) and adjusted settings (B).

An issue that could follow from increasing the neuron count by that amount could be overfitting.

Disadvantages of pattern-recognition learning: difficult to execute, slow method and requires a big dataset for enhanced and accurate results to show.

Disadvantages of SOM (particularly Kohonen maps): does not provide generative model for the data and if there isn’t enough input for the weights the map will be inaccurate or misinformative and takes a long time to prepare and train the model for large datasets.

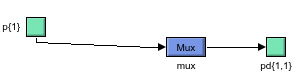
# **APPENDIX**

# **My Pattern Recognition Neural Network (SIMULINK)**

# 

Figure ) SIMULINK Recognition Neural Network

# 

Figure ) SIMULINK Recognition Neural Network: Layer 1

# 

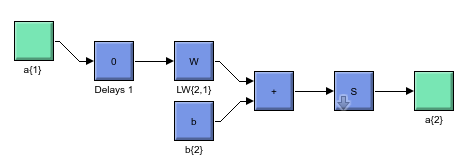
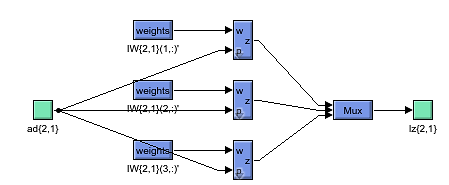
****Figure ) SIMULINK Recognition Neural Network: Layer 1 DELAYS

Figure ) SIMULINK Recognition Neural Network: Layer 1 **P**, **W**, f and **a**



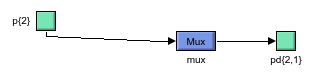
Figure ) SIMULINK Recognition Neural Network: Layer 2

Figure ) SIMULINK Recognition Neural Network: Layer 2 DELAYS Figure ) SIMULINK Recognition Neural Network: **a1, W2,** softmax**, a2**

# **My Clustering Neural Network (SIMULINK)**

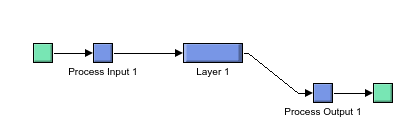


Figure ) SIMULINK Clustering Neural Network

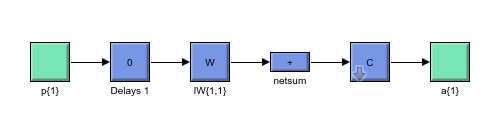


Figure ) SIMULINK Clustering Neural Network: Layer 1

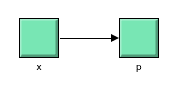


Figure ) SIMULINK Clustering Neural Network: Process Input 1

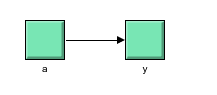


Figure ) SIMULINK Clustering Neural Network: Process Output 1**References**

## Academic Books

## Kosko, B. (1992). *Neural networks and fuzzy systems: A dynamical systems approach to machine intelligence*. Prentice-Hall.

## Masters, T. 1994, *Signal and image processing with neural networks: a C[plus plus] sourcebook,*J. Wiley, New York;Chichester;.

## Johnson, J., Picton, P. & Open University 1995, *Designing intelligent machines,*Butterworth-Heinemann in association with the Open University, Oxford.

## Gurney, K. 1997, *An introduction to neural networks,*UCL Press, London.

## Patterson, D.W. 1996, *Artificial neural networks: theory and applications,*Prentice-Hall, London;Singapore;.

## Academic Journals

## Ehsani, A.H., Quiel, F. & Malekian, A. 2009;2010;, "Effect of SRTM resolution on morphometric feature identification using neural network—self organizing map", *GeoInformatica,*vol. 14, no. 4, pp. 405-424.

## Kohn, N., Eickhoff, S.B., Scheller, M., Laird, A.R., Fox, P.T. & Habel, U. 2014, "Neural network of cognitive emotion regulation — An ALE meta-analysis and MACM analysis", *NeuroImage (Orlando, Fla.),*vol. 87, pp. 345-355.

## Tian, C., Ma, J., Zhang, C. & Zhan, P. 2018, "A Deep Neural Network Model for Short-Term Load Forecast Based on Long Short-Term Memory Network and Convolutional Neural Network", *Energies (Basel),*vol. 11, no. 12, pp. 3493.

## Khashei, M. & Bijari, M. 2010, "An artificial neural network ( p, d, q) model for timeseries forecasting", *Expert systems with applications,*vol. 37, no. 1, pp. 479-489.