Winter 2023 COEN 6331 Assignment Report (II)

Jun Huang, Student ID: 40168167
Electrical and Computer Engineering, Concordia University
Email: jun.huang@concordia.ca

***Abstract*—This report presents the experiments and results of building a counter-propagation neural network (CPNN) to observe its performance compared with the multilayer perception (MLP) network. The network is trained for the same binary classification problem as we did in report one. Besides the counter-propagation neural network (CPNN), the report also presents the building and training of the self-organizing map (SOM) solely to understand the counter-propagation neural network better. In this assignment, we examine the effects of different structures, learning rates, and annealing methods for learning rates. Furthermore, finally, we compare the performance of SOM, CPNNs, and MLP. The result shows that SOM and CPNNs could be better in performance. They are good enough for the target classification problem to get a reliable prediction with fewer training effects.**

I. Problem Description

Same as the report one, the problem for this report is to predict if a data point (x_1, x_2) belongs to one of the two classes (C_1 and C_2) in the following coordinate system as Fig. 1 shown. The target is to build a counter-propagation neuron network (CPNN) for such classification problem.

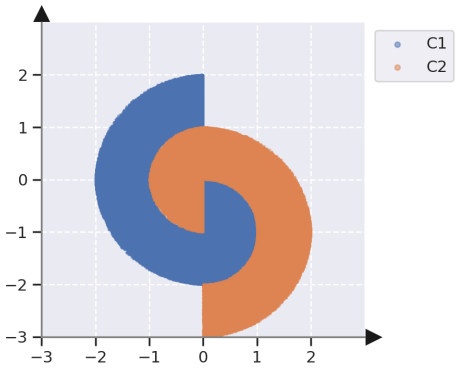

Fig. 1. Non-convex decision regions of the problem.

II. Self-Organizing Map

In order to get a better understanding of the counter-propagation neural network, we first walk through the self-organizing map (also known as Kohonen network). Particularly, we build the self-organizing map from scratch with and train the map to predict our target problem.

A. Strucure of the Map

The first step to building a SOM is to determine the shape of the map. The idea is to map the features on top of a topological structure. In this report, we choose the triangle topology by leveraging the two-dimensional array as Fig. 2 shown. For the selection neuron number in the map, we refer to this work [1] and choose the heuristic number as Eq. 1 presented, where N is the number of total samples. Hence the length of the square map l is $ceil(\sqrt{M})$.

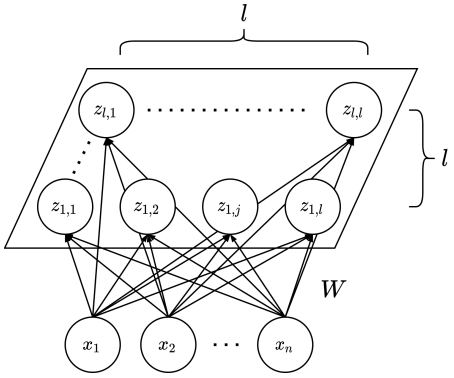

Fig. 2. A $l \times l$ Self-organizing map topology to map a data set with n samples. W denotes the weight matrix between the map neurons and the input samples.

$$M = 5\sqrt{N} \tag{1}$$

B. Training Process

To train the SOM network, we perform the Algorithm 1, where f is the number of input features, l is the length of the map, σ_0 is the initial sigma for the neighbourhood function, η_0 is the initial learning rate, d is the decay factor for both learning rate and the sigma, and I is the total number of the training iteration.

Focue on the Algorithm 1, line five determine the index of the winner neuron. We choose the **euclidean distance** to select the winner by calculating the l_2 norm of the result of the difference between the input vector and the weight matrix as shown in Eq. 2.

$$I(input) = argmin(\|input - W\|_2) \tag{2}$$

This is extremely efficient compared with using two nested for loops to calculate the euclidean distance between the input vector and the neuron weight vector one by one (almost 60 times faster in the experiment enverionment).

For the decay function $decay(v_0, i, I, d)$, we explore different annealing methods [2] including the asymptotic, staircase, and sigmoid decay with different hyperparameters as shown in Fig. 3, 4, and 5.

For the neighbourhood function, we choose the gaussian

Algorithm 1: Self-Organizing Map Training

input: X, f, l, σ_0, η_0, d, I
output: W

$W \leftarrow$ initialize(l, f) // initialize weight matrix
for $i \leftarrow 0$ *to* I **do**
  /* competition */
  $t \leftarrow i \ \%$ length$(X) - 1$
  $x \leftarrow X[t]$ // pick up a sample
  $winner\_idx \leftarrow$ winner(x, W)
  /* get decaied learning rate and sigma */
  $\eta \leftarrow$ decay(η_0, i, I, d)
  $\sigma \leftarrow$ decay(σ_0, i, I, d)
  /* neighbourhood function */
  $\theta \leftarrow$ neighbourhood$(l, winner_idx, \sigma)$
  /* update */
  $W \leftarrow W + \eta * \theta * (x - W)$

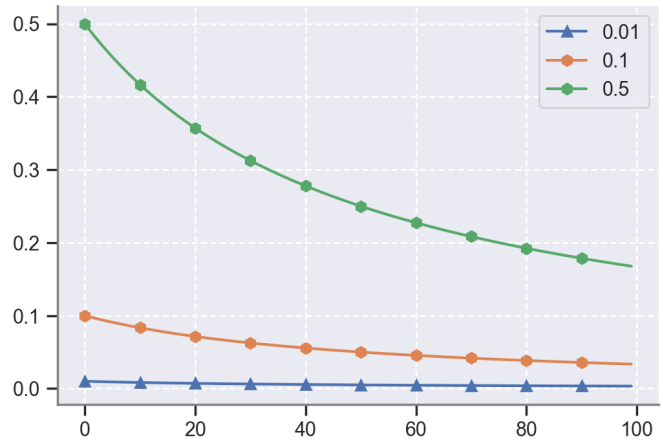

Fig. 3. Plotting of the asymptotic decay function with initial value settings: 0.01, 0.1, and 0.5.

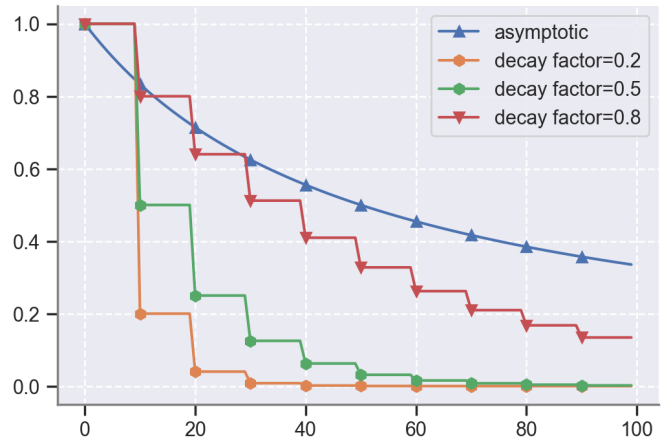

Fig. 4. Plotting of the staircase decay function with an initial value set to 1.0 and the decay factor settings: 0.2, 0.5, and 0.8 compared with the asymptotic decay.

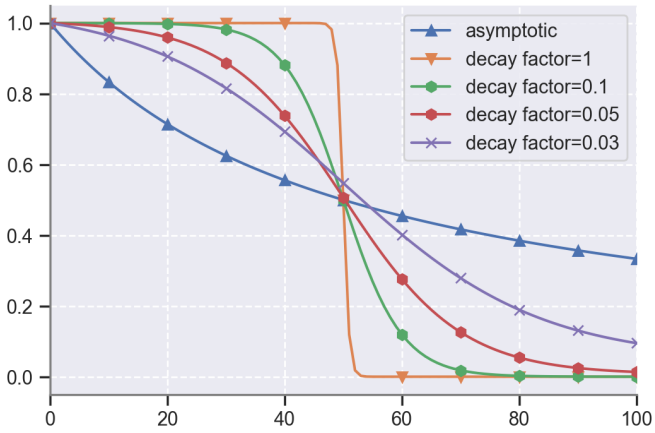

Fig. 5. Plotting of the sigmoid decay function with an initial value set to 1.0 and the decay factor settings: 1.0, 0.1, 0.05, and 0.03 compared with the asymptotic decay.

function to decide the neighbourhood of the winner neuron. The function can be described as follow:

$$h_{j,w}(i) = exp\Big(-\frac{d_{j,w}^2}{2\sigma^2(i)}\Big) \tag{3}$$

where j is the neighbourhood neuron, w is the winner neuron, $d_{j,w}^2$ is the distance between the two neurons, i is the current iteration, and σ is decaied before by time.

C. Experiment Result

To build the SOM network, we collect 51,004 samples and split them into 35,702 (70%) as training set and 15,302 (30%) as testing set. This number is emprical because we found that the SOM can use small amount of training data to get a good performance. Hence our SOM network has $l = 31$ based on Eq. 1. And then, we train the model with different hyperparameters settings to observe the variety of the performance with the result listed in Table. I. After that, we use more iterations to reach the best performance it can to observe the comparison with the multilayer perception network we trained in the previous report. This result is shown in Table. II.

III. Counter-Propagation Neuron Network

With the foundation of the experience, we have better to perform the building and training for the counter-propagation neuron network regarding the winner neuron function, the selection of learning rate annealing methods, and the Kohonen learning process.

The counter-propagation neuron network consist of three layers: (1) input layer, (2) unsupervised Kohonen layer, and (3) the teachable output layer. From another perspective, the network can also be considered as the combination of two models: (1) the in-star model, which consists of the input layer and the Kohonen layer and (2) the out-star model, which consists of the Kohonen layer and the output layer. The in-star model takes input from the data sample and performs **Kohonen learning** on the Kohonen layer as SOM did. And

after the training of the Kohonen layer, the network trains the out-star model to learning the pattern of the Kohonen layer and the output by **Grossberg learning**.

There are two kinds of basic counter-propagation neuron network: full CPNN and the forward-only CPNN. In this report, we will discuss both of them in the follow sections.

A. Full CPNN

Full CPNN indicates that the CPNN train the model with the input-output pairs as input so that the network learning the associated pattern at the same time. In this paper, we build the full CPNN with the structure shown in Fig. 6.

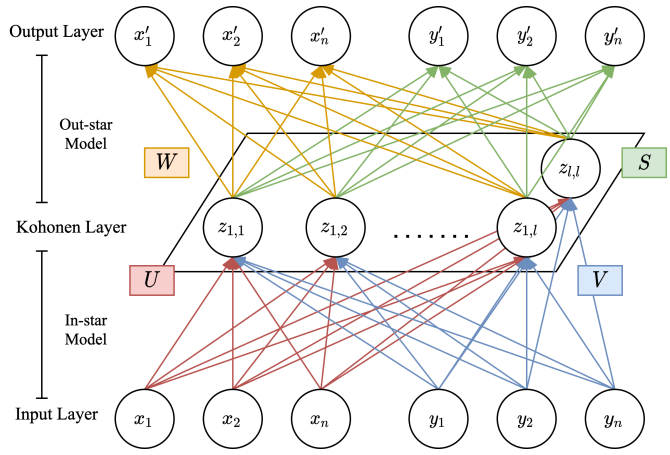

Fig. 6. A full counter-propagation neuron network (CPNN) with a $l \times l$ Kohonen layer to map a data set with n samples. U denotes the weight matrix between the input X and the Kohonen layer. V denotes the weight matrix between the output Y and the Kohonen layer. W denotes the weight matrix between the Kohonen layer and the appriximated input X'. S denotes the weight matrix between the the Kohonen layer and the appriximated output Y'.

The training process is described as Algorithm 2. Unlike the SOM training, the update of the winner neuron does not collaborate with its neighbourhood but updates itself.

B. Forward-Only CPNN

Forward-Only CPNN indicates that the CPNN trains the model with Kohonen learning by input data and Grossberg learning by ground truth output data. In this paper, we build the forward-only CPNN with the structure shown in Fig. 7 and the training process is listed in Algorithm 3.

C. Experiment Result

Same as in SOM, we use the 51,004 samples to train full CPNN and forward-only CPNN with small training iterations and a fixed 0.5 learning rate. The result for both CPNN structures are listed in Table. I, and their comparison with the MLP network is in Table. II.

IV. Conclusion

In this report, we explore the training process of the CPNN structure. We also explore the SOM structure as the foundation exploration towards the CPNN structure. Finally, we compare the well-trained models of SOM, CPNNs, and MLP. The result

Algorithm 2: Full CPNN Training

input: X, Y, f, l, η_0, d, I
output: U, V, W, S

/* initialize weight matrix w */
$U \leftarrow$ initialize(l, f)
$V \leftarrow$ initialize(l, f)
$W \leftarrow$ initialize(l, f)
$S \leftarrow$ initialize(l, f)
/* in-star model training */
for $i \leftarrow 0$ *to* I **do**
  /* competition */
  $t \leftarrow i$ % length(dataset) – 1
  $winner\_idx \leftarrow$ winner$(X[t], Y[t], U, V)$
  /* get decaied learning rate */
  $\eta \leftarrow$ decay(η_0, i, I, d)
  /* update */
  $U[winner\_idx] \leftarrow U + \eta * (X[t] - U)$
  $V[winner\_idx] \leftarrow V + \eta * (Y[t] - S)$
/* out-star model training */
for $i \leftarrow 0$ *to* I **do**
  /* competition */
  $t \leftarrow i$ % length(dataset) – 1
  $winner\_idx \leftarrow$ winner$(X[t], Y[t], U, V)$
  /* get decaied learning rate */
  $\eta \leftarrow$ decay(η_0, i, I, d)
  /* update */
  $W[winner\_idx] \leftarrow W + \eta * (X[t] - W)$
  $S[winner\_idx] \leftarrow S + \eta * (Y[t] - S)$

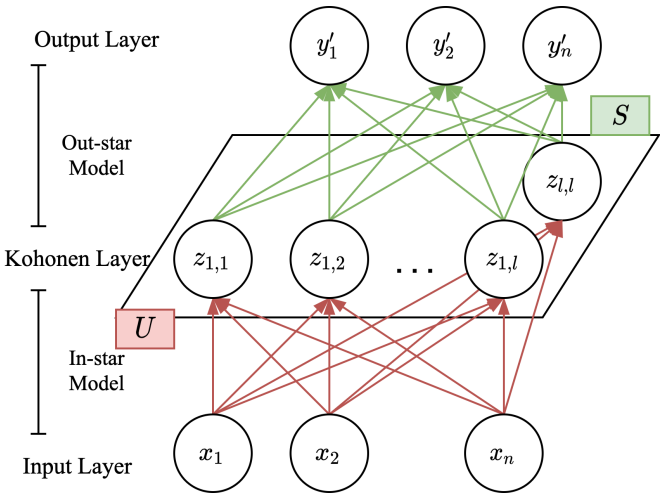

Fig. 7. A forward-only counter-propagation neuron network (CPNN) with a $l \times l$ Kohonen layer to map a data set with n samples. U denotes the weight matrix between the input X and the Kohonen layer. S denotes the weight matrix between the the Kohonen layer and the output Y.

in Table. I shows that, generally, SOM structures are better than CPNN structures. We use a small number of training iterations of 100 for all settings to observe the difference. Also, regarding the annealing methods, the asymptotic decay is better than the staircase decay and the sigmoid decay overall

Algorithm 3: Forward-Only CPNN Training

input: X, Y, f, l, η_0, d, I
output: U, S

/* initialize weight matrix w */
$U \leftarrow$ initialize(l, f)
$S \leftarrow$ initialize(l, f)
/* in-star model training */
for $i \leftarrow 0$ *to* I **do**
  /* competition */
  $t \leftarrow i$ % length(dataset) – 1
  $winner\_idx \leftarrow$ winner$(X[t], U)$
  /* get decaied learning rate */
  $\eta \leftarrow$ decay(η_0, i, I, d)
  /* update */
  $U[winner\_idx] \leftarrow U[winner\_idx] + \eta * (X[t] - U[winner\_idx])$
/* out-star model training */
for $i \leftarrow 0$ *to* I **do**
  /* competition */
  $t \leftarrow i$ % length(dataset) – 1
  $winner\_idx \leftarrow$ winner$(X[t], U)$
  /* get decaied learning rate */
  $\eta \leftarrow$ decay(η_0, i, I, d)
  /* update */
  $S[winner\_idx] \leftarrow S[winner\_idx] + \eta * (Y[t] - S[winner\_idx])$

settings. Compared between full and forward-only CPNNs, the full CPNN can capture the input and output pattern well than the forward-only structure. But with more training iterations Finally, comparing the SOM and CPNN with MLP, SOM and CPNN require less time and data set to get a good and stable performance. Due to the complexity of the target problem, SOM and CPNN performed better than the MLP with less training time. We want to observe if this is also true for a more complex classification problem in the future.

REFERENCES

[1] J. Tian, M. H. Azarian, and M. Pecht, "Anomaly detection using self-organizing maps-based k-nearest neighbor algorithm," in *PHM society European conference*, vol. 2, no. 1, 2014.

[2] K. Nakamura, B. Derbel, K.-J. Won, and B.-W. Hong, "Learning-rate annealing methods for deep neural networks," *Electronics*, vol. 10, no. 16, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/16/2029

TABLE I
EVALUATION METIRCS FOR DIFFERENT SETTINGS OF SELF-ORGANIZING MAP NETWORK, FULL, AND FORWARD-ONLY COUNTER-PROPAGATION NEURON NETWORK RANKED BY THE ACCURACY

Model Type	**Setting Label**	***Accuracy***	***Precision***	***F1-Score***	***Recall***	***Top-1 Accuracy***	***Training Time in Seconds***
SOM	lr=0.5	0.983139	0.986330	0.983100	0.979890	0.983139	0.011295
SOM	step decay factor=1	0.955561	0.958596	0.955457	0.952337	0.955561	0.017482
SOM	lr=0.1	0.883937	0.896951	0.884855	0.873081	0.883937	0.014482
FU. CPNN	lr=0.5	0.883675	0.821241	0.875315	0.937013	0.883675	0.081413
SOM	sigmoid decay factor=0.5	0.869821	0.867771	0.868913	0.870058	0.869821	0.011832
SOM	sigmoid decay factor=0.2	0.855117	0.868165	0.856291	0.844737	0.855117	0.010893
SOM	sigmoid decay factor=1	0.823552	0.860016	0.828962	0.800073	0.823552	0.013444
SOM	lr=0.01	0.783754	0.638801	0.746028	0.896514	0.783754	0.013779
SOM	step decay factor=0.5	0.781466	0.669295	0.752809	0.860135	0.781466	0.015748
SOM	step decay factor=0.2	0.771729	0.578996	0.716086	0.938232	0.771729	0.014066
FO. CPNN	lr=0.5	0.756306	0.671793	0.732707	0.805770	0.756306	0.047516

All training are done within 100 iterations.

TABLE II
EVALUATION METIRCS FOR WELL-TRAINED SELF-ORGANIZING MAP NETWORK, FULL, AND FORWARD-ONLY COUNTER-PROPAGATION NEURON NETWORK, MULTILAYER PERCEPTION NETWORK RANKED BY THE ACCURACY

Model Type	**Setting Label**	***Accuracy***	***Precision***	***F1-Score***	***Recall***	***Top-1 Accuracy***	***Training Time in Seconds***
MLP	lr=1e-3,mo=0.93	0.996668	0.996198	0.996671	0.997146	0.996668	32.661339
SOM	lr=1	0.986538	0.991982	0.986536	0.981149	0.986538	0.019647
FO. CPNN	lr=1	0.977977	0.970952	0.977698	0.984540	0.977977	7.273861
FU. CPNN	lr=1	0.944321	0.919558	0.942603	0.966833	0.944321	16.370582

The SOM training only needs 100 iterations. In contrast, the CPNN training needs 10,000 iterations to reach over 95% accuracy.