Assignment 3
Computational Intelligence
Dr. S. Hajipour

Mohammad Hossein Shafizadegan
99104781

November 25, 2023

Contents

1 Question 1

We have assumed that we have a simple RBF network. Also we note that when we are increasing the number of training inputs we don't change and design the network again. This means that the number of hidden neurons won't change as we increase the training data.

a

Average Training Error:

Regarding what we said above, when we increase the training data, the mean training error will increase as we need more hidden neurons but we have fixed the their number.

Mean Validation Error:

Adding more training data may **decrease** the mean validation error as the model processes more examples and learns to generalize better.

b

As we increase the training data, the mean training error and mean validation error **may converge or get closer**. This is because a well-regularized model, when exposed to sufficient representative data, is more likely to generalize well to both the training and validation sets. This may tends to stabilize or reduce the difference between mean training error and mean validation error.

c

As we increase the number of training data to infinity, the mean training error and mean validation error get closer and closer and finally converge to a specific value. This value can be defined as the **True error** as its independent of the training and validation data.

2 Question 2

In this question the error or cost function is considered as sum of squared error terms.

Updating rule for weights between the hidden layer and output layer :

$$\nabla_{W_u} e_u^{(l)} = \frac{\partial e_u^{(l)}}{\partial W_u} = -2(o_u^{(l)} - out_u^{(l)})\, in_u^{(l)} \quad \Rightarrow \quad \Delta W_u^{(l)} = -\frac{\eta}{2}\nabla_{W_u} e_u^{(l)} = \eta(o_u^{(l)} - out_u^{(l)})\, in_u^{(l)}$$

Updating rule for weights between the input layer and hidden layer :

$$\nabla_{W_v} e^{(l)} = \frac{\partial e^{(l)}}{\partial W_v} = -2 \sum_{s \in secc(v)} (o_s^{(l)} - out_s^{(l)})\, w_{sv} \frac{\partial out_v^{(l)}}{\partial net_v^{(l)}} \frac{\partial net_v^{(l)}}{\partial w_v} \quad , \quad \frac{\partial out_v^{(l)}}{\partial net_v^{(l)}} = \begin{cases} 0 & \text{if } net > \sigma \\ -\frac{1}{\sigma} & \text{O.W.} \end{cases}$$

$$d(x, w) = \left(\sum_{i=1}^{n} |x - w|^3 \right)^{\frac{1}{3}} \quad \Rightarrow \quad \frac{\partial net_v^{(l)}}{\partial w_v} = \frac{1}{3} \left(\sum_{i=1}^{n} |x - w|^3 \right)^{-\frac{2}{3}} |x - w|^2 \times \frac{|x - w|}{x - w}$$

Updating rule for σ :

$$\frac{\partial e^{(l)}}{\partial \sigma_v} = -2 \sum_{s \in secc(v)} (o_s^{(l)} - out_s^{(l)})\, w_{sv} \frac{\partial out_v^{(l)}}{\partial \sigma_v} \quad , \quad \frac{\partial out_v^{(l)}}{\partial \sigma_v} = \begin{cases} 0 & \text{if } net > \sigma \\ \frac{net}{\sigma^2} & \text{O.W.} \end{cases} \quad , \quad \Delta \sigma_v = -\frac{\eta}{2} \frac{\partial e^{(l)}}{\partial \sigma_v}$$

3 Question 3

Firs we discuss the case of $\sigma = 2$

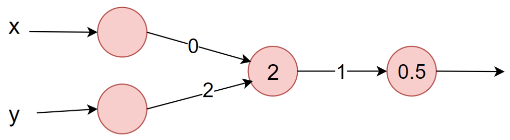
$$f_{act} = \begin{cases} 0 & \text{if } net > 2 \\ 1 - \frac{net}{2} & \text{O.W.} \end{cases}$$

First we map the given outputs to the range of 0 to 1 by defining the threshold used in the output neuron

$$\theta = 0.5 \Rightarrow \begin{cases} z'_1 = 0 & \Rightarrow & net \geq 2 \\ z'_2 = 0 & \Rightarrow & net \geq 2 \\ z'_3 = 0.5 & \Rightarrow & net = 1 \\ z'_4 = 1 & \Rightarrow & net = 0 \end{cases}$$

In this case, we can use the infinity norm as distance function and set the center of the catchment area at $(0, 2)$, therefor:

$$d(w, x) = \max_i |w_i - x_i| \qquad , \qquad w = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$



Now for the case of $\sigma = 4$

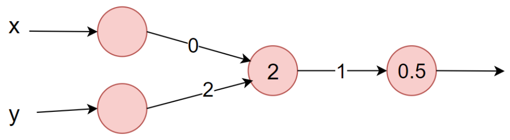
$$f_{act} = \begin{cases} 0 & \text{if } net > 4 \\ 1 - \frac{net}{4} & \text{O.W.} \end{cases}$$

Now we map the given outputs to the range of 0 to 1 by defining the threshold used in the output neuron

$$\theta = 0.5 \Rightarrow \begin{cases} z'_1 = 0.5 & \Rightarrow & net = 2 \\ z'_2 = 0 & \Rightarrow & net \geq 4 \\ z'_3 = 0.5 & \Rightarrow & net = 2 \\ z'_4 = 1 & \Rightarrow & net = 0 \end{cases}$$

In this case, we can use the Manhattan distance as network function and set the center of the catchment area at $(0, 2)$, therefor:

$$d(w, x) = \sum_{i=1}^{2} |w_i - x_i| \qquad , \qquad w = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$



4 Question 4

The RBF network consists of 5 neurons in the hidden layer. 4 neurons utilize Euclidean distance as network function and one uses Manhattan distance as network function.

$$\text{neuron } 1: w = \begin{bmatrix}2\\4\end{bmatrix} \quad , \quad f_{act} = \begin{cases}0 & \text{if } net > \sqrt{2}\\ 1 & \text{O.W.}\end{cases} \quad , \quad f_{net} = \sqrt{(w_1 - x_1)^2 + (w_2 - x_2)^2}$$

$$\text{neuron } 2: w = \begin{bmatrix}4\\4\end{bmatrix} \quad , \quad f_{act} = \begin{cases}0 & \text{if } net > \sqrt{2}\\ 1 & \text{O.W.}\end{cases} \quad , \quad f_{net} = \sqrt{(w_1 - x_1)^2 + (w_2 - x_2)^2}$$

$$\text{neuron } 3: w = \begin{bmatrix}2\\2\end{bmatrix} \quad , \quad f_{act} = \begin{cases}0 & \text{if } net > \sqrt{2}\\ 1 & \text{O.W.}\end{cases} \quad , \quad f_{net} = \sqrt{(w_1 - x_1)^2 + (w_2 - x_2)^2}$$

$$\text{neuron } 4: w = \begin{bmatrix}4\\2\end{bmatrix} \quad , \quad f_{act} = \begin{cases}0 & \text{if } net > \sqrt{2}\\ 1 & \text{O.W.}\end{cases} \quad , \quad f_{net} = \sqrt{(w_1 - x_1)^2 + (w_2 - x_2)^2}$$

$$\text{neuron } 5: w = \begin{bmatrix}3\\3\end{bmatrix} \quad , \quad f_{act} = \begin{cases}1 & \text{if } net > \sqrt{2}\\ 0 & \text{O.W.}\end{cases} \quad , \quad f_{net} = |w_1 - x_1| + |w_2 - x_2|$$

Since the fact the only one neuron out of the first 4 first neurons has an output equal to 1 for an input, the only thing we have to do in the output layer is to add the output of all these 4 neurons and subtract them from the output of the 5th neuron. Therefor the specifications of the output neuron is as follows:

$$f_net = -out_5 + \sum_{i=1}^{4} out_i \quad , \quad f_{act} = net - 0$$

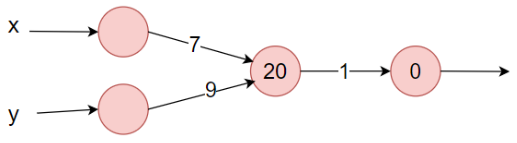
5 Question 5

Regarding the two conditions discussed in the question about the movement criteria of the robots, the boundary that they can serve the tables is where the addition of the distance to two points are equal to a constant value (10 in this case). Therefor, it can be inferred that the boundary is an oval and the robots can serve the tables inside an oval which its bigger diameter equals to 10 and its two focal points are the robot stations.

$$\frac{(x-x_0)^2}{a^2} + \frac{(y-y_0)^2}{b^2} = 1 \Rightarrow b^2(x-x_0)^2 + a^2(y-y_0)^2 = (ab)^2 \quad , \quad (x_0, y_0) = \frac{1}{2}\left(\begin{bmatrix}4\\9\end{bmatrix} + \begin{bmatrix}10\\9\end{bmatrix}\right) = \begin{bmatrix}7\\9\end{bmatrix}$$

$$d = \sqrt{b^2(x-x_0)^2 + a^2(y-y_0)^2} \quad , \quad 2a = \text{bigger diameter} = 10 \Rightarrow a = 5$$

$$b = \sqrt{a^2 - c^2} \quad , \quad 2c = \text{distance for the focal points} = 6 \Rightarrow c = 3 \quad \Rightarrow \quad b = 4 \quad \Rightarrow \quad d = ab = 20$$

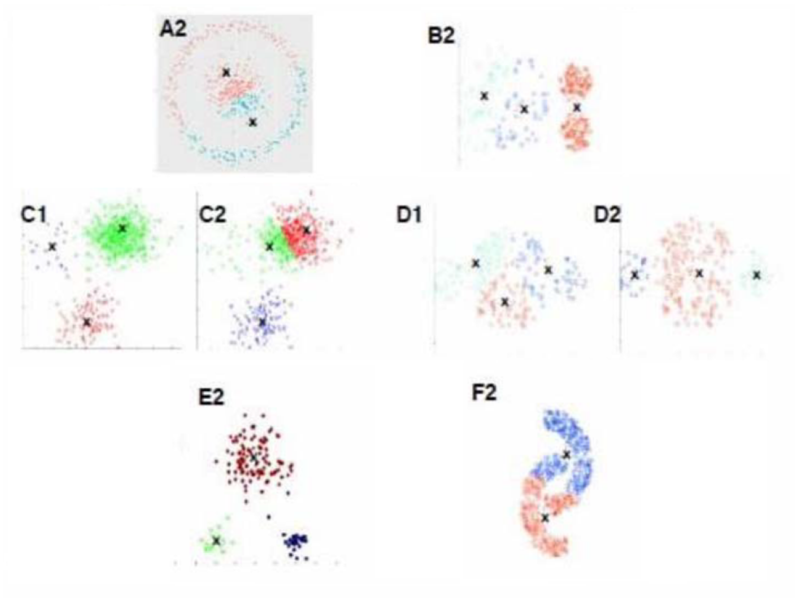


We note that the output neuron has a linear activation function with $\theta = 0$

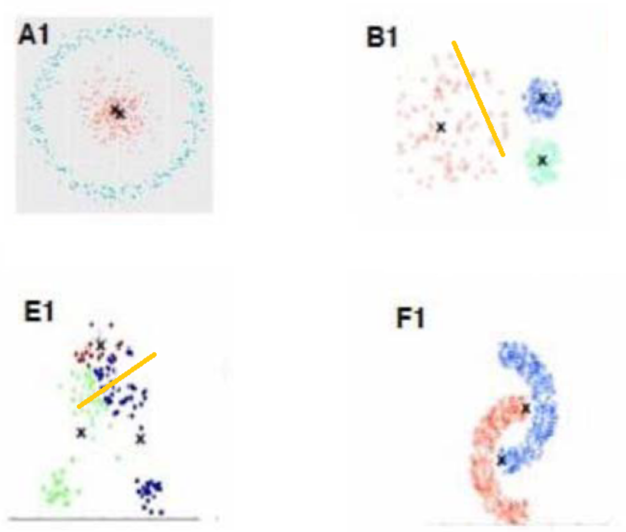
6 Question 6

K-means organizes a dataset into k distinct groups based on similarity. The algorithm begins by randomly selecting k initial cluster centers, assigning each data point to the cluster whose center is closest, and then updating the cluster centers to be the mean of the data points in each cluster. This assignment and update process iterates until convergence, aiming to minimize the sum of squared distances between data points and their respective cluster centers.

Regarding what we mentioned above, the following figures and results are clustered using the kmeans algorithm as in the final result, each data point is assigned to its closet centroid.



The following figures contain clusters that are not resulted from kmeans algorithm as there are some data point that are not assigned to their closest centroid. For figure **B1** and **E1**, the orange line demonstrates that some data point are not belong to their proper centroid.



7 Question 7

a

First we randomly choose 20 out of 500 samples we have. In order to select the hidden neurons more wisely we divide the whole domain into 20 sectors and randomly choose one sample from each section. This way, the weight between the input layer and the hidden layer which indeed the coordinate of this points are defined.

Since we use Gaussian activation function the parameter σ is initialized as follows:

$$\sigma_i = \frac{d_{max}}{\sqrt{2m}} \quad , \quad d_{max} = \max_{l_j\,,\,l_k \in L_{fixed}} d(i^{(l_j)}, i^{(l_k)}) \quad , \quad m = 20$$

Then at the section where we assign the initial values for the weights between the hidden layer and the output layer, we have the followings:

$$A = \begin{bmatrix} 1 & out_{v_1}^{l_1} & out_{v_2}^{l_1} & \dots & out_{v_{20}}^{l_1} \\ 1 & out_{v_1}^{l_2} & out_{v_2}^{l_2} & \dots & out_{v_{20}}^{l_2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & out_{v_1}^{l_{500}} & out_{v_2}^{l_{500}} & \dots & out_{v_{20}}^{l_{500}} \end{bmatrix} \quad , \quad w_u = \begin{bmatrix} -\theta_u \\ w_{uv_1} \\ \vdots \\ w_{uv_{20}} \end{bmatrix} \quad , \quad Aw_u = o_u \quad , \quad A^{\dagger} = (A^T A)^{-1} A^T \Rightarrow w_u = A^{\dagger} o_u$$

Finally following the rules below, we iteratively update the parameters:

Updating rule for weights between the hidden layer and output layer :

$$\nabla_{W_u} e_u^{(l)} = \frac{\partial e_u^{(l)}}{\partial W_u} = -2(o_u^{(l)} - out_u^{(l)})\, in_u^{(l)} \quad \Rightarrow \quad \Delta W_u^{(l)} = -\frac{\eta}{2} \nabla_{W_u} e_u^{(l)} = \eta(o_u^{(l)} - out_u^{(l)})\, in_u^{(l)}$$

Updating rule for weights between the input layer and hidden layer :

$$\nabla_{W_v} e^{(l)} = \frac{\partial e^{(l)}}{\partial W_v} = -2 \sum_{s \in secc(v)} (o_s^{(l)} - out_s^{(l)})\, w_{sv}\, \frac{\partial out_v^{(l)}}{\partial net_v^{(l)}} \frac{\partial net_v^{(l)}}{\partial w_v}$$

Gaussian activation function $\Rightarrow \dfrac{\partial out_v^{(l)}}{\partial net_v^{(l)}} = -\dfrac{net_v^{(l)}}{\sigma_v^2}\, e^{-\frac{(net_v^{(l)})^2}{2\sigma_v^2}}$

Updating rule for σ :

$$\frac{\partial e^{(l)}}{\partial \sigma_v} = -2 \sum_{s \in secc(v)} (o_s^{(l)} - out_s^{(l)})\, w_{sv}\, \frac{\partial out_v^{(l)}}{\partial \sigma_v}$$

b

If we want to use the rectangular activation function for the hidden neurons, perhaps we have to find another way to initialize the value of σ.

More importantly in the rules of updating the weights between the input layer and the hidden layer the expression for $\frac{\partial out_v^{(l)}}{\partial net_v^{(l)}}$ will change correspondingly.

c

The initial value for the weights between the input layer and the hidden layer is defined as the coordinates of the hidden neurons are already provided. Following the previous formula for the initial value of σ we have :

$$\sigma_i = \frac{d_{max}}{\sqrt{2m}} \quad , \quad d_{max} = \max_{l_j\,,\, l_k \in L_{fixed}} d(i^{(l_j)}, i^{(l_k)}) = 5\sqrt{2} \quad , \quad m = 4 \quad \Rightarrow \quad \sigma = \frac{5\sqrt{2}}{2\sqrt{2}} = 2.5$$

If these 4 hidden neurons are among that 500 given samples, we simply create the matrix A as explained the section a, and find the initial value for the output layer weights by solving the system of linear equations using pseudo inverse matrix.

Otherwise, as a simple solution, we can define a parameter k and look for the k nearest neighbors to our hidden neurons among the given 500 samples. Then we consider the average of the neighbors output as the output of the hidden neuron. This way we can define the matrix A and we follow the method we discussed earlier.

The algorithm and formulas for updating the parameters is quite the same as before.

d

The idea which can be useful for this question is to first recognize which region has the greater error value, then we try to enhance our estimation by dividing that region into more sections. In a loop, we repeatedly perform the same process. Every time we create new sections we have to add some neurons e.g. 2 ones to create 4 new subsections. Also in each step we need some neurons to determine which region has the greater error value.