Kieran Cosgrove
Student #: 20226842

CISC/CMPE 452/COGS 400

Assignment 3

Theoretical Part

1. The well-known XOR (Exclusive OR) problem is the simplest example of a two-class classification problem where the pattern vectors are not linearly separable. In the XOR problem, there are four two-dimensional input vectors (patterns) (0,0), (0,1), (1,1), and (1,0). The first and third pattern vector belong to the class 1, while the second and the fourth one belong to the class 2. Solve the XOR problem by using a RBF network with two Gaussian basis functions centered at $c_1 = (0,1)^T$ and $c_2 = (1,0)^T$.

2. The input-output relationship of a Gaussian based RBF networks is defined by

$$y(i) = \sum_{j=1}^{K} w_j(n) \exp(-\frac{1}{2\sigma^2(n)} \|\vec{x}(i) - \vec{\mu}_j(n)\|^2), \quad i = 1, 2, \cdots, n$$

   where $\vec{\mu}_j(n)$ is the center point of the jth Gaussian unit, the width $\sigma(n)$ is common to all the K units, and $w_j(n)$ is the linear weight assigned to the output of the jth unit; all these parameters are measured at time n. The cost function used to train the network is defined by

$$E = \frac{1}{2} \sum_{i=1}^{n} e^2(i), \quad e(i) = d(i) - y(i)$$

   (a) Evaluate the partial derivative of the cost function with respect to each of the network parameters, $w_j(n)$, $\vec{\mu}_j(n)$, and $\sigma(n)$ for all i.

   (b) Use the gradient obtained in (a) to express the update formulas for all network parameters, assuming the learning rate parameters η_w, η_μ and η_σ, for the adjustable parameters of the network, respectively.

   (c) The gradient vector $\frac{\partial E}{\partial \vec{\mu}_j(n)}$ has an effect on the input data that is similar to clustering. Justify your answer

3. **5.b–** Given the following input vectors
   $\mathbf{x}_1 = [0,0]^t$, $\mathbf{x}_2 = [1,1]^t$, $\mathbf{x}_3 = [-1,-1]^t$, $\mathbf{x}_4 = [-2,2]^t$, and $\mathbf{x}_5 = [2,-2]^t$.
   i– Calculate the mean, $\mathbf{m}_x$, and covariance matrix, $\mathbf{C}_x$.
   ii– Calculate the eigenvectors and eigenvalues of the presented data.

4. Assume there is a number of C clusters. Consider the following competitive learning algorithm for a clustering network. First, the network is initialized to random weights. A counter, $n_j, j = 1, \cdots, C$ is associated with each neuron indicating the number of patterns assigned to the cluster represented by this neuron. Then, the input patterns, $\{\mathbf{x}^m\}, m = 1, \cdots, M$,are presented in sequence. The neuron which wins the competition for the first time will set its weight vector equal to the presented input pattern $\mathbf{x}^m$,

$$\mathbf{w}_j^1 = \mathbf{x}^m, \quad \text{and} \quad n_j = 1$$

If, during the course of training, neuron j wins again then the following rules are used to update the weights

$$\begin{aligned} n_j &= n_j + 1 \\ \mathbf{w}_j^{\text{new}} &= \mathbf{w}_j^{\text{old}} + \frac{1}{n_j}(\mathbf{x}^p - \mathbf{w}_j^{\text{old}}) \end{aligned}$$

where $\mathbf{x}^p$ is the current input. The weights of losing neurons will be left unchanged. Show that after the presentation of all training set the weight vector associated with neuron j becomes

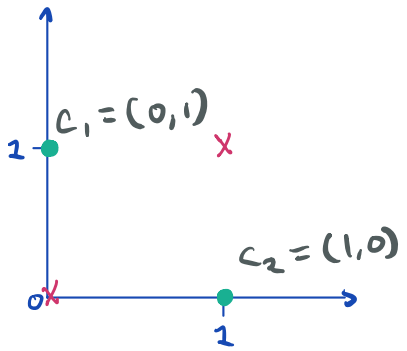
$$\mathbf{w}_j^{\text{final}} = \frac{1}{n_j} \sum_{m=1}^{n_j} \mathbf{x}^m$$

Assignment 3

Kieran Cosgrove
Student #: 20226841

Question 1

x_1	x_2	XOR(x_1, x_2)
0	0	0
0	1	1
1	0	1
1	1	0



X = class 1
O = class 2

Givens:

- The gaussian basis functions are at $c_1 = (0,1)^T$ and $c_2 = (1,0)^T$
- we assume the spread parameter σ is the same for both basis functions and is chosen appropriately (commonly as the squared distance between the centres.
- XOR input patterns are $(0,0), (0,1), (1,1),$ and $(1,0)$.

Activation functions:

Let the RBF at c_1 be $a_1(x) = e^{\frac{-\|x-c_1\|^2}{2\sigma_1^2}}$

Let the RBF at c_2 be $a_2(x) = e^{\frac{-\|x-c_2\|^2}{2}}$

The next step is to calculate the activation of the RBFs. Let $\sigma = 1$.

For $c_1 = (0, 1)^T$

$$\phi_1(0,0) = e^{\frac{\|(0,0)-(0,1)\|^2}{2(1^2)}} = e^{-0.5} \approx 0.61$$

$$\phi_2(0,1) = e^{\frac{\|(0,1)-(0,1)\|^2}{2(1^2)}} = e^{0} = 1$$

$$\phi_3(1,1) = e^{\frac{\|(1,1)-(0,1)\|^2}{2(1^2)}} = e^{-0.5} \approx 0.61$$

$$\phi_3(1,0) = e^{\frac{\|(1,0)-(0,1)\|^2}{2(1^2)}} = e^{-1} \approx 0.37$$

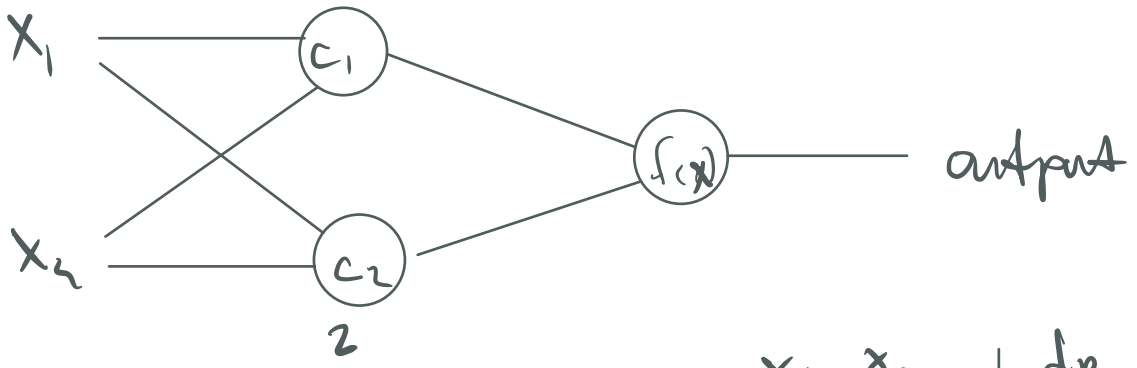
For $c_2 = (1, 0)^T$:

$$\phi_1(0,0) = e^{\frac{\|(0,0)-(1,0)\|^2}{2(1^2)}} = e^{-0.5} \approx 0.61$$

$$\phi_2(0,1) = e^{\frac{\|(0,1)-(1,0)\|^2}{2(1^2)}} = e^{-1} \approx 0.37$$

$$\phi_3(1,1) = e^{\frac{\|(1,1)-(1,0)\|^2}{2(1^2)}} = e^{-0.5} \approx 0.61$$

$$\phi_3(1,0) = e^{\frac{\|(1,0)-(1,0)\|^2}{2(1,0)}} = e^{0} = 1$$



2

$$f(x) = \sum_{p=1} d_p \, \phi(\|x - c_p\|)$$

ϕ: weighted (desired output)

x_1	x_2	d_p
0	0	1
0	1	2
1	0	2
1	1	1

$$f(0,0) = (1(\phi_1(0,0)) + 1(\phi_2(0,0))$$

$$f(0,0) = (1(e^{-0.5}) + 1(e^{-0.5})) = 1.21$$

$$f(0,1) = (2(1) + 2(e^{-1})) = 2.74$$

$$f(1,0) = (2(e^{-1}) + 2(e^{0})) = 2.74$$

$$f(1,1) = (1(e^{-0.5}) + 1(e^{-0.5})) = 1.21$$

Based on the outputs, it is evident that the network will properly classify each point. The model could be further refined so that the classification is closer to the exact class value, but for this XOR example, the values are accurate enough as the closest centroid will be chosen.

Question 2

a) The partial derivations of the cost function:

$$\frac{\partial E}{\partial w_{j(n)}} = \frac{\partial E}{\partial e(i)} \times \frac{\partial e(i)}{\partial y(i)} \times \frac{\partial y(i)}{\partial w_{j(n)}}$$

$$= \left(\frac{1}{2} \sum_{i=1}^{n} 2e(i)\right) \frac{\partial e(i)}{\partial e(i)} \cdot \frac{\partial y(i)}{\partial w_{j(n)}}$$

$$= \left(\sum_{i=1}^{n} e(i)\right)(-1)\left(e^{\left(-\frac{1}{2\sigma^2_{(n)}} \| x(i) - w_{i(n)} \|^2\right)}\right)$$

$$= -\sum_{i=1}^{n} e(i) \left(e^{\left(-\| x(i) - w_{i(n)} \|^2\right)}\right)$$

$$\frac{\partial E}{\partial w_{i(n)}} = \frac{\partial E}{\partial e(i)} \times \frac{\partial e(i)}{\partial y(i)} \cdot \frac{\partial y(i)}{\partial w_{j(n)}}$$

$$= \frac{w_{j(n)} e \left(\frac{-\| x(i) - w_{j(n)} \|^2}{2\sigma^2_{(n)}}\right)(x(i) - w_{j(n)})}{\sigma^2_{(n)}}$$

$$\frac{\partial E}{\partial w_{j(n)}} = -\sum_{i=1}^{n} e(i) w_{j(n)} e^{\left(\frac{-\| x(i) - w_{j(n)} \|^2}{2\sigma^2_{(n)}}\right)} \frac{(x(i) - w_{j(n)})}{\sigma^2_{(n)}}$$

$$\frac{\partial E}{\partial \sigma(n)} = \frac{\partial E}{\partial e(i)} \cdot \frac{\partial e(i)}{\partial y(i)} \cdot \frac{\partial y(i)}{\partial \sigma(n)}$$

$$\frac{\partial E}{\partial \sigma(n)} = w_j(n) e^{\left(\frac{-\|x(i) - u_i(n)\|^2}{2\sigma^2(n)}\right)} \left(\sigma(n) \left(\frac{\|x(i) - u_i(n)\|^2}{2}\right)\right)$$

$$= -\sum_{i=1}^{n} e(i) w_j(n) e^{\left(\frac{-\|x(i) - u_j(n)\|^2}{2\sigma^2(n)}\right)} \left(\sigma(n) \left(\|x(i) - u_j(n)\|^2\right)\right)$$

b) For $w_j(n)$:

$$w_j(n+1) = w_j(n) - \eta \frac{\partial E}{\partial w_j(n)}$$

For $\sigma(n)$:

$$\sigma(n+1) = \sigma(n) - \eta \frac{\partial E}{\partial \sigma(n)}$$

For $M_i(n)$:

$$M_i(n+1) = M_j(n) - \eta \frac{\partial E}{\partial M_i(n)}$$

c) The gradient vector $\frac{\partial E}{\partial \vec{u}_j(n)}$ describes how the cost function E changes with respect to small changes in the centres of the Gaussian units, $u_j(n)$. In RBF networks, each Gaussian unit has a centre u_j that determines where the basis function is centred in the output space.

When the gradient $\frac{\partial E}{\partial \mu_{ji}}$ is calculated, we're looking at how E changes if we move the centre of the Gaussian unit slightly. This gradient tells us the direction in which we should adjust μ_j to decrease the error. In other words, if we have an input that is close to the centre μ_j and the network's prediction is wrong, then adjusting μ_j according to the gradient will move μ_j closer to that output.

This is similar to clustering because, in clustering, we also adjust the centres of clusters to minimize the distance between the data points and the clusters centres. The goal of RBF networks and clustering is to arrange the centres so that they are positioned in regions with high data density.

Question 3

i) Given the vector inputs, the mean is:

$$m_x = \frac{1}{5}(x_1 + x_2 + x_3 + x_4 + x_5)$$

$$= \frac{1}{5}\left([0,0]^T + [1,1]^T + [-1,-1]^T + [-2,2]^T + [2,-2]^T\right)$$

$$= \frac{1}{5}[0,0]^T = [0,0]^T$$

Therefore the mean m_x of the input vectors is $[0,0]$.

The covariance of the inputs is:

$$C(x,y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

$$= \frac{1}{5-1}\left((0-0)(0-0) + (1-0)(1-0) + (-1-0)(-1-0) + (-2-0)(2-0) + (2-0)(-2-0)\right)$$

$$= \frac{1}{4}\left((0\times 0) + (1)(1) + (-1)(-1) + (-2)(2) + (2)(-2)\right)$$

$$C_x(x,y) = -\frac{2}{4} = -0.5 \qquad C_x(y,x) = -0.5$$

$$Var(x) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2$$

$$= \frac{1}{4}\left((0-0)^2+(1-0)^2+(-1-0)^2+(-2-0)^2+(2-0)^2\right)$$

$$= \frac{1}{4}(0+1+1+4+4)$$

$$\boxed{Var(x) = \frac{10}{4} = 2.5}$$

$$Var(y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2$$

$$= \frac{1}{5-1}\left((0-0)^2+(1-0)^2+(-2-0)^2+(2-0)^2\right)$$

$$= \frac{1}{4}(0+1+1+4+4)$$

$$\boxed{Var(y) = \frac{10}{4} = 2.5}$$

Therefore $C_x = \begin{bmatrix} 2.5 & -0.5 \\ -0.5 & 2.5 \end{bmatrix}$

ii) We must use C_x to calculate the eigenvectors and eigenvalues.

$$\det(A-\lambda I)=0$$

$$\det\left(\begin{bmatrix} 2.5-\lambda & -0.5 \\ -0.5 & 2.5-\lambda \end{bmatrix}\right)=0$$

$$0=(2.5-\lambda)^2-(0.5)^2$$

$$0=6.25-5\lambda+\lambda^2-0.25$$

$$0=\lambda^2-5\lambda+6$$

$$\boxed{\lambda_1=3 \quad \lambda_2=2}$$

$$\Rightarrow \quad \underline{X}(A-\lambda_1 I)=0$$

$$\underline{X}\begin{pmatrix} 2.5-3 & -0.5 \\ -0.5 & 2.5-3 \end{pmatrix}=0$$

$$\underline{X}\begin{pmatrix} -0.5 & -0.5 \\ -0.5 & -0.5 \end{pmatrix}=0$$

Eigenvector for $\lambda_1=3$: $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$

$$\underline{X}(A-\lambda_2 I)=0$$

$$\underline{X}\left(\begin{bmatrix} 2.5-2 & -0.5 \\ -0.5 & 2.5-2 \end{bmatrix}\right)=0$$

$$\underline{X}\begin{pmatrix} 0.5 & -0.5 \\ -0.5 & 0.5 \end{pmatrix}$$

Eigenvector for $\lambda_2=2$: $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$

Question 4

I will approach this problem inductively:

During neuron j's (n_j) first win:

$n_j = 1 \quad w_j^1 = x^m = x_1$

During neuron j's second win:

$n_j = 2 \quad w_j^2 = x_1 + \frac{1}{2}(x_2 - x_1)$

During neuron j's third win:

$n_j = 3 \quad w_j^3 = (x_1 + \frac{1}{2}(x_2 - x_1) + \frac{1}{3}(x_1 + \frac{1}{2}(x_2 - x_1))$

$= \frac{x_1 + x_2 + x_3}{3}$

Now consider neuron j when $n_j = m$:

$n_j = m \quad w_j^m = w_j^{m-1} + \frac{1}{m}(x_m - w_j^{m-1}) = \frac{x_1 + x_2 + x_3 + \cdots x_m}{m}$

It is shown that when $n_j = m$ that x will be the summation of m. Another way of thinking about it is that with each new win, the weight vector of the neuron is essentially the average of all input patterns that cause the neuron to win.

Thus the generalization of our proof can be expressed as:

$$w_j^{final} = \frac{1}{n_j} \sum_{m=1}^{n_j} x^m$$