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 $\mathbf{x}^m$,

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

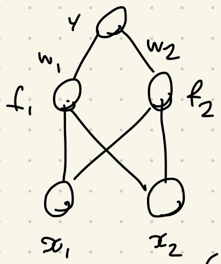
If, during the course of training, neuron j wins again then the following rules are used to update the weights

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

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

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

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 $c_2 = (1,0)^T$.



x_1	0	0	1
x_2	0	1	0
x_3	1	0	0
x_4	1	1	1

$$f_1(x_i) = \exp\left\{-\frac{\|x_i - c_1\|^2}{\sigma_1^2}\right\} = e^{-\|x_i - c_1\|^2} \quad (\sigma_1^2 = 1) \quad c_1 = (0,1)$$

$$f_2(x_i) = \exp\left\{-\frac{\|x_i - c_2\|^2}{\sigma_2^2}\right\} = e^{-\|x_i - c_2\|^2} \quad (\sigma_2^2 = 1) \quad c_2 = (1,0)$$

$$f_1(x_1) = e^{-((0-1)^2+(0-0)^2)} = e^{-1} \; ; \; f_1(x_2) = 1 \; ; \; f_1(x_3) = e^{-2}$$

$$f_2(x_1) = e^{-1} \; ; \; f_2(x_2) = e^{-2} \; ; \; f_2(x_3) = 1$$

$$f_1(x_4) = e^{-1}$$

$$f_2(x_4) = e^{-1}$$

x_1: $f_1 + f_2 = 2e^{-1} = 0.73$

x_2: $1 + e^{-2} = 1.13$

x_3: $1 + e^{-2} = 1.13$

x_4: $2e^{-1} = 0.73$

Let $b = 1$, $w = [w_1, w_2] = [0, 0]$

Then $y(x_i) = \begin{cases} 1 & f_1(x_i) + f_2(x_i) > 1 \\ 0 & f_1(x_i) + f_2(x_i) < 1 \end{cases}$

$y(x_i) = 1 \iff$ Class 2

$y(x_i) = 0 \iff$ class 1

$\square$

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

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

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

(a) Let $\varphi_j^{(i)} = \exp\left(-\frac{1}{2\sigma^2(n)} \|\vec{x}(i) - \vec{\mu}_j(n)\|^2\right)$

$$\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial e} \frac{\partial e}{\partial y} \frac{\partial y}{\partial w_j} = \sum_{i=1}^{n} e(i) \cdot (-1) \cdot \varphi_j^{(i)}$$

$$\frac{\partial E}{\partial \vec{\mu}_j} = \frac{\partial E}{\partial e} \frac{\partial e}{\partial y} \frac{\partial y}{\partial \varphi_j^{(i)}} \frac{\partial \varphi_j^{(i)}}{\partial \vec{\mu}_j} = -\sum_{i=1}^{n} e(i) w_j(n) \varphi_j^{(i)} \frac{\vec{x}(i) - \vec{\mu}_j(n)}{\sigma^2(n)}$$

$$\frac{\partial E}{\partial \sigma} = \frac{\partial E}{\partial e} \frac{\partial e}{\partial y} \frac{\partial y}{\partial \varphi_j^{(i)}} \frac{\partial \varphi_j^{(i)}}{\partial \sigma} = \sum_{i=1}^{n} e(i) w_j(n) \varphi_j^{(i)} \frac{\|\vec{x}(i) - \vec{\mu}_j(n)\|^2}{\sigma^3(n)}$$

(b) $$w_j(n+1) = w_j(n) - \eta_w \frac{\partial E}{\partial w_j}(n) = w_j(n) + \eta_w \sum_{i=1}^{n} e(i) \varphi_j^{(i)}$$

$$\vec{\mu}_j(n+1) = \vec{\mu}_j(n) - \eta_\mu \frac{\partial E}{\partial \vec{\mu}_j} = \vec{\mu}_j(n) + \eta_\mu \sum_{i=1}^{n} e(i) w_j(n) \varphi_j^{(i)} \frac{\vec{x}(i) - \vec{\mu}_j(n)}{\sigma^2(n)}$$

$$\sigma(n+1) = \sigma(n) - \eta_\sigma \sum_{i=1}^{n} e(i) w_j(n) \varphi_j^{(i)} \frac{\|\vec{x}(i) - \vec{\mu}_j(n)\|^2}{\sigma^3(n)}$$

(c) This is because it points in the direction of the mean of the data points the most incorrectly predicted by the j-th Gaussian unit. When adjusting $\vec{\mu}_j(n)$ in the direction of this gradient, the center of the Gaussian unit moves towards the center of its assigned cluster in the input space, which happens in algorithms like k-means.

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.

(i)

$$m_x = \begin{bmatrix} \frac{1}{5}\cdot(0+1+-1+2-2) \\ \frac{1}{5}(0+1-1+2-2) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$C_x(0,0) = Var(X) = E[X^2] - E[X]^2 = \frac{1}{5-1}[1+1+4+4] - 0^2$$

$$= \frac{10}{4} = 2.5$$

$$C_x(1,1) = Var(Y) = Var(X) = 2.5$$

$$C_x(0,1) = C_x(1,0) = \frac{1}{4}\cdot\sum_{i=1}^{5}(X_i - \underbrace{\bar{X}}_{=0})(Y_i - \underbrace{\bar{Y}}_{=0})$$

$$= \frac{1}{4}\cdot\left(0\cdot0 + 1\cdot1 + (-1)(-1) + (-2)(2) + (2)(-2)\right)$$

$$= -\frac{3}{2}$$

$$\Rightarrow C_x = \begin{bmatrix} 2.5 & -1.5 \\ -1.5 & 2.5 \end{bmatrix}$$

$$(\lambda I - C_x) = \begin{bmatrix} \lambda-2.5 & 1.5 \\ 1.5 & \lambda-2.5 \end{bmatrix}$$

$$\Rightarrow \det|\lambda I - C_x| = (\lambda-2.5)^2 - 1.5^2$$

$$= \lambda^2 - 5\lambda + 2.5^2 - 1.5^2$$

$$= \lambda^2 - 5\lambda + 4$$

$$= (\lambda-4)(\lambda-1)$$

$$\lambda = 4$$

$$\lambda = 1$$

$\underline{\lambda = 4}$: $\ker(\lambda I - C_x) = 0 \iff \begin{bmatrix} 1.5 & 1.5 \\ 1.5 & 1.5 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$$1.5x_1 + 1.5x_2 = 0$$

$$x_1 = -x_2$$

$\underline{\lambda = 1}$: $\ker(\lambda I - C_x) = 0 \iff \begin{bmatrix} -1.5 & 1.5 \\ 1.5 & -1.5 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$$\Rightarrow x_1 = x_2$$

Normalize:

$\underline{\lambda_1 = 4}$: $V_1' = \frac{1}{\|v_1\|}[1 \quad -1] = \frac{1}{\sqrt{2}}[1 \quad -1]$

$$= [0.707 \quad -0.707]$$

$\underline{\lambda_2 = 1}$: $V_2' = [0.707 \quad 0.707]$

$\square$

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

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

When the first pattern x_m is assigned to neuron j, the weight is set to x_m and n_j is set to 1.

$$w_j^{new} = x^m, \quad n_j = 1$$

The next time a pattern, x^p is assigned to neuron j, the weights update rule is applied:

(1) $w_j^{old} = w_j^{new}$

(2) $n_j = n_j + 1$

(3) $w_j^{new} = w_j^{old} + \frac{1}{n_j}(x^p - w_j^{old})$

Since w_j^{old} is the average of all provious assigned patterns to neuron j, adding $\frac{1}{n_j}(x^p - w_j^{old})$ updates this average to include the new pattern x^p. After n_j patterns have been assigned to neuron j, the final weight w_j^{final} is

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

Induction

Base case:

For one element, x^1

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

$$= \frac{1}{1}\sum_{m=1}^{1} x^m$$

$$= x^1$$

This follows with how the algorithm is initialized in α

Inductive step:

Assume that after k patterns, $w_j^{(k)} = \frac{1}{k}\sum_{m=1}^{k} x_m$ now for $w_j^{(k+1)}$:

$$w_j^{(k+1)} = w_j^{(k)} + \frac{1}{k+1}\left(x^{k+1} - w_j^{(k)}\right)$$

$$= \frac{1}{k}\sum_{m=1}^{k} x^m + \frac{1}{k+1}\left(x^{k+1} - \frac{1}{k}\sum_{m=1}^{k} x^m\right)$$

$$= \frac{1}{k+1}\left[\frac{k+1}{k}\sum_{m=1}^{k} x^m - \frac{1}{k}\sum_{m=1}^{k} x^m + x^{k+1}\right)$$

$$= \frac{1}{k+1}\left[\underbrace{\left(\frac{k+\cancel{1}-\cancel{1}}{\cancel{k}}\right)}_{=1}\sum_{m=1}^{k} x^m + x^{k+1}\right)$$

$$= \frac{1}{k+1}\left[\sum_{m=1}^{k} x^m + x^{k+1}\right]$$

$$= \frac{1}{k+1}\sum_{m=1}^{k+1} x^m$$

Hence $w_j^{final} = \frac{1}{n_j}\sum_{m=1}^{n_j} x^m \quad \forall\, n_j$ $\square$