CAP 5615 Introduction to Neural Networks

2021 Summer [Solutions]

Homework 4 (**17+3 pts, Due: June 22 2021, Late Penalty: -2/day**)

[If two homework submissions are found to be similar to each other, both submissions will receive 0 grade]

- Homework solutions must be submitted through Canvas. No email submission is accepted.
- Please try to put all results in one file (e.g., one pdf or word file).
- If you have multiple files, please upload them separately (only **pdf**, **word**, and **html** files are allowed).

You can always update your submissions. Only the latest version will be graded.]

Question 1 [2 points: 0.25/each]: Please use your own language to briefly explain the following concepts (Must use your own language. No credit if descriptions are copied from external sources):

- Stochastic Gradient Descent Learning:

Stochastic Gradient Descent Learning is a special case of the gradient descent learning to update the weight values of a network by using gradient with respect to the weight values. According to gradient descent learning, the weight updating of the neural network is to follow the opposite direction of the steepest direction of the change of the network errors (i.g, the objective function)

$W(k+1)=W(k)-\eta\cdot$ (Gradient of the objective function to weight W)

Stochastic gradient descent weight updating is based on the gradient calculated using a single instance, instead of using all training instances (like gradient descent learning does). The main purpose of the stochastic gradient descent learning, compared to gradient descent learning, is to improve training speed by using a single instance, instead of using a batch or the whole training set, to calculate the weight updating.

- Neural network learning rate:

When adjusting the weight values of neural networks, learning rate is applied to adjust the magnitude of the movement as follows

$$W(k+1)= W(k)+ \eta\cdot\Delta W$$

Where η is the learning rate, and ΔW is the delta weight values (change of the weights). For a large learning rate, the weight update will become more noticeable and the algorithm can converge quicker, but the learning may overstep optimal solutions. For a small learning rate, the algorithm will have a slower convergence, but may have a better chance to find optimal solutions.

- Multi-Layer Feed Froward Neural Network:

A multi-layer feedforward neural network is an artificial neural network consisting of multiple layers, including an input layer, one or more hidden layers, or an output layer. The neuros of the networks are connected in a forward manner where input layer points to hidden layers, and then from hidden layers point to the output layer. The connections between neuros do not form a cycle.

- Hidden Layer in Neural Network:

Hidden layer in a neural network refer to a layer of neuros which are between input layer and output layer. A hidden layer consists of one or multiple hidden neurons which connects to the same (or similar) input, and the hidden neurons in the same layer do not direct connect to each other.

- Output Layer in Neural Network:

Output layer in neural network refer to the last layer of the neural network whose neuros' output can be directly measures/observed from the outside of the network. In a classification task, each output node in the outpour layer corresponds to one binary label. E.g., when classifying digits (0 -9), the output layer consists of 10 output nodes, each corresponding to one digit.

- Backpropagation Rule:

Backpropagation rule (BP) is a common approach used to update weight values in multi-layer neural network. BP rule utilizes the gradient descent principle, and intends to update the network weight values in order to minimize the squared loss (or squared errors) of the whole network. BP rule for weight updating consists of two phases: Forward phase and backward phase. During the forward phase, instances are feed to the network to calculate squared loss, and the loss are propagated backward (during the backward phase) to adjust the network weight values.

- Objective function of Multi-layer Neural Network:

The objective function of a multi-layer neural network is to minimize the total mean squared effort of the network on a given set of training samples (or total squared error). Given a training dataset with N instances (x(n), d(n)), the objective function is defined as follows, where $O_j(n)$ is the output of the jth output node with respect to the instance x(n).

$$E(W) = \frac{1}{2N}\sum_n \sum_j (d_j(n) - o_j(n))^2$$

- Momentum term in neural network weight updating:

The momentum term in neural network is to take the weight updating of the previous round into consideration, to update the weight values of the current round. Assume $\Delta w(K-1)$ denotes the weight updating amount of the previous round, $\eta\delta(n)x(n)$ denotes the amount of calculated weight updating, the final weight updating of the current round is given as follows, where $0 \leq \alpha < 1$ defines the momentum constraint.

Question 2 [1 pt] Figure 1 shows a single layer neural network with three weight values (including bias). Please use gradient descent learning to derive weight updating rule for **w_0.** Please define the objective function [0.5 pt] and the derivations [0.5 pt]

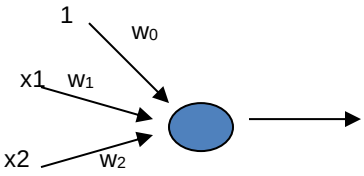


Figure 1: Single layer neural network

Given a training sample (x(n), d(n)) where x(n)=[x_1(n), x_2(n)] denotes features and d(n) denotes label, the objective function is given as follows

$$f(W) = f(w_0, w_1, w_2) = \tfrac{1}{2} \times [d(n) - w_0 - w_1 x_1(n) - w_2 x_2(n)]^2$$

Gradient descent intends to minimize f(W), by using gradient descent learning rule, which equivalent to the inverse direction of the gradient of the objective function. For w_0, its partial derivative to f(W) is equivalent to

$$\begin{aligned}\frac{\partial f(\mathbf{W})}{\partial w_0} &= \frac{\partial f(w_0,w_1,w_2)}{\partial w_0} = \frac{\frac{1}{2}[d(n)-w_0-w_1x_1(n)-w_2x_2(n)]^2}{\partial w_0} \\ &= [d(n) - w_0 - w_1 x_1(n) - w_2 x_2(n)] \times (-1)) \\ &= -[d(n) - w_0 - w_1 x_1(n) - w_2 x_2(n)]\end{aligned}$$

Therefore, the weight updating for w_2 is as follows, where η is the learning rate.

$$\Delta w_0 = -\eta \frac{\partial f(\mathbf{W})}{w_0} = [d(n) - w_0 - w_1 x_1(n) - w_2 x_2(n)]$$

Question 3 [1 pt] Figure 2 shows three neural network structures. (a) is a single layer neural network, (b) is a one hidden layer neural network, and (c) is a two hidden layer neural network, where each rectangle box denotes a one hidden layer network similar to (b). Please draw decision regions for each network, respectively (0.5 pt), and explain how the networks learn to classify samples into different categories (0.5 pt)

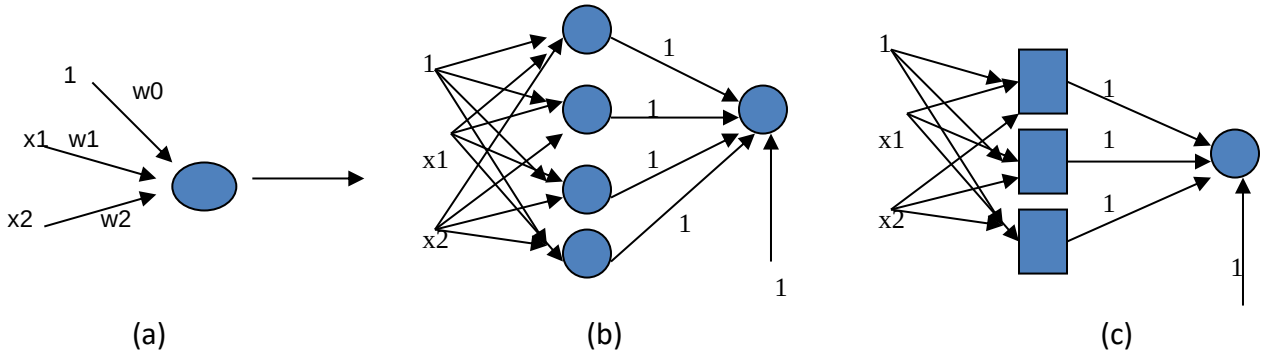
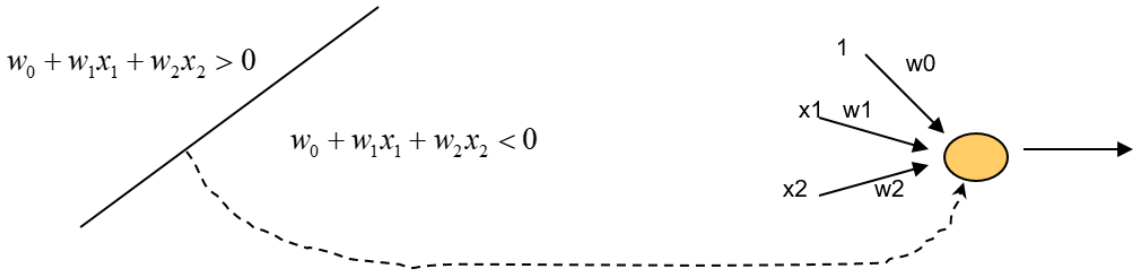
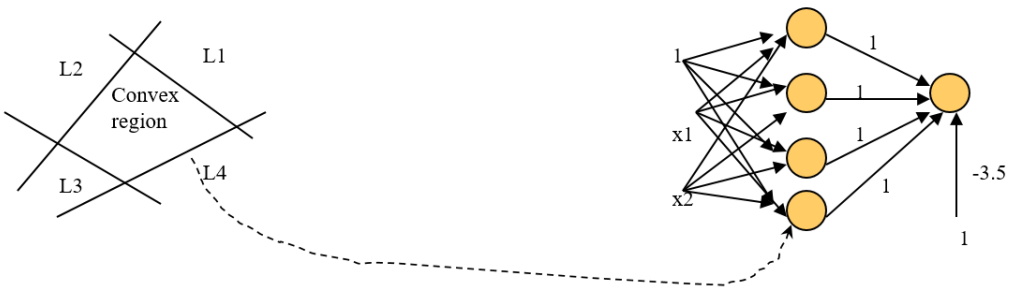


Figure 2: Feedforward neural networks

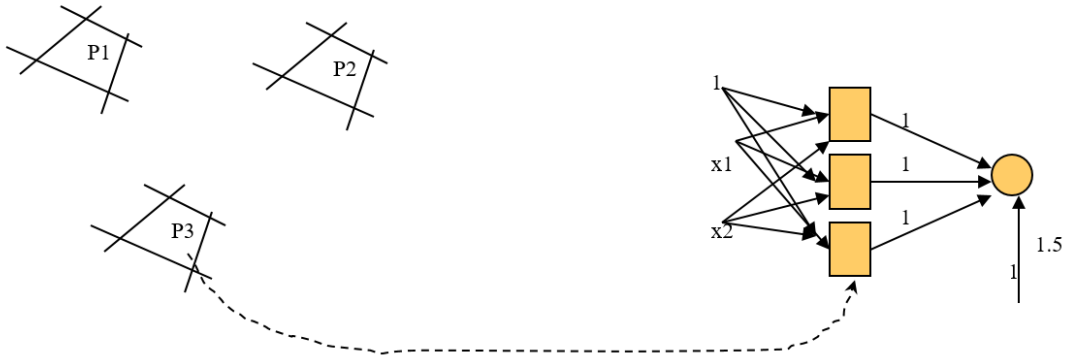
(a) Decision surface is a line (or hyper-plane). The classification is based on whether a point is above or below the line (or hyper-plane).



(b): Decision surface is an open or closed convex region. The classification is based on whether a point is inside the convex region or not.



(c): Decision surface is the union of three convex regions. The classification is based on whether the point is inside any of the convex region.



Question 4 [1 pt]: Figure 3 shows some samples in a two-dimensional feature space (X_1, X_2). Each symbol (circle, triangle, plus, diamond) denotes a sample, and each shape denote one type of samples. Please design a neural network architecture which can learn to classify samples in Figure 3 into correct types. Show your network input dimension, number of hidden nodes (layers), and output node(s) [0.75 pt]. Explain why you use this network architecture [0.25 pt]

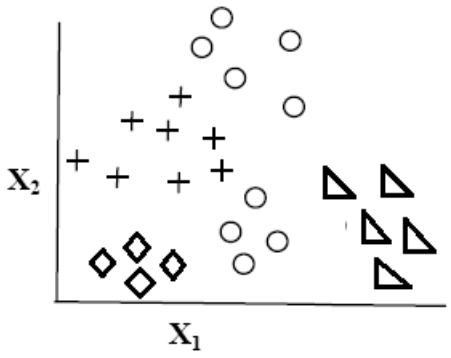
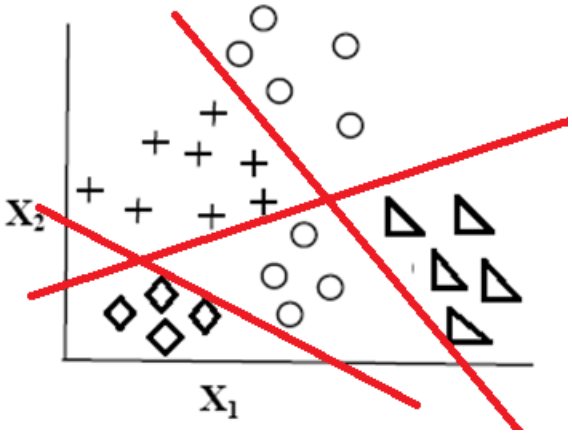
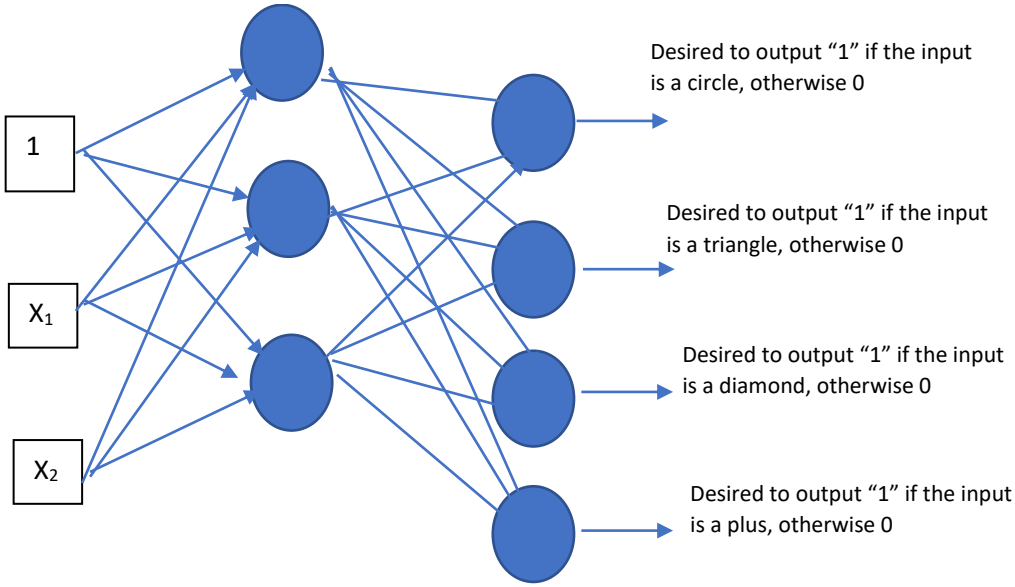


Figure 3:

The input has two-dimensional features, so the network should have three inputs (including bias). There are four types of samples (circle, plus, triangle, diamond), therefore, the network should four output nodes (each corresponding to one type of samples).

The instances can be separated using three lines as follow. The four lines separate the total space into six regions. Therefore, they can be implemented using one hidden layer neural network,





Alternatively, one can also implement a two hidden layer neural networks, but the network must have three inputs, and four outputs, to match to the give task.

Question 5 [1 pt]: Figure 4 shows a quadratic function y=x^2. Assume we are at the point x=2, and is searching for the next movement to find the minimum value of the quadratic function using gradient descent (the learning rate is 0.1). What is the gradient at point **x=2?** [0.5 pt] Following gradient decent principle, find the next movement towards the global minimum [0.5 pt]

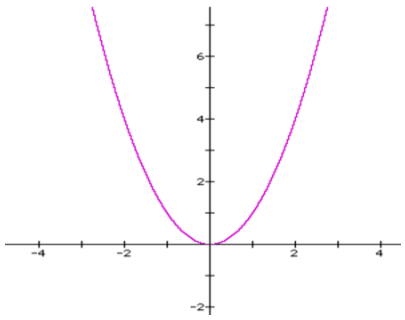


Figure 4: Quadratic function y=x^2

Solutions:

The gradient is the first derivative of the function y=x^2, which is 2x. Therefore, the gradient at point x=2 is 2*2=4.

Following gradient descent learning principle, the next movement is determined as follows:

x(k+1)=x(k)+ η·(- gradient).

Using learning rate 0.1, we have

x(k+1)=x(k)+ 0.1×(- gradient)

which is

x(k+1)=x(k)+ 0.1×(-4)=2.0-0.4=1.6

Therefore, the next x value is 1.6

Question 6 [2.5 pts]: Given feedforward neural network in Figure 5 with one hidden layer and one output layer, assuming the network initial weights are

$\begin{bmatrix} w_{0a} \\ w_{1a} \\ w_{2a} \end{bmatrix} = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \end{bmatrix}$; $\begin{bmatrix} w_{0b} \\ w_{1b} \\ w_{2b} \end{bmatrix} = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \end{bmatrix}$; $\begin{bmatrix} w_{0c} \\ w_{ac} \\ w_{bc} \end{bmatrix} = \begin{bmatrix} \mathbf{1.0} \\ \mathbf{0.1} \\ \mathbf{0.1} \end{bmatrix}$. All nodes use a sigmoid activation function with value a=1.0, and the learning rate η is set to 0.2. The expected output is *1* if an instance's class label is "True", otherwise, the expected output is *0*. Given the following three instances I_1, I_2, I_3 which are labeled as "**False**", "**True**", and "**True**", respectively.

- Calculate their actual outputs from the network? [**0.5 pt**]
- What is the mean squared error of the network over the three instances? [**0.5 pt**]

- Assuming instance I_1 is fed to the network to update the weight, calculate local gradient of each node [0.5 pt]
- please update the network weight (using backpropogation rule) by using this instance (list major steps and results). [1 **pt**]

[Show your calculations]

$$I_1 = \begin{bmatrix} 1.0 \\ 1.0 \\ 0.5 \end{bmatrix}; \qquad I_2 = \begin{bmatrix} 1.0 \\ 0.0 \\ 1.0 \end{bmatrix}; \qquad I_3 = \begin{bmatrix} 1.0 \\ 0.5 \\ 0.5 \end{bmatrix}$$

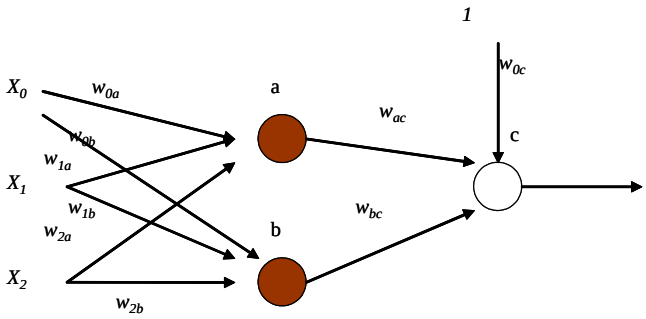


Figure 5

Solution:

Calculate their actual outputs from the network? [**0.5 pt**]

I1:

$$o_a(I_1) = \phi(1 + 1 + 0.5) = \frac{1}{1+e^{-1\times 2.5}} = 0.9241; \qquad o_b(I_1) = \phi(1.0 + 1.0 + 0.5) = \frac{1}{1+e^{-1\times 2.5}} = 0.9241$$

$$o_c(I_1) = \phi(1.0 + 0.1o_a(I_1) + 0.1o_b(I_1)) = \phi(1.1848) = 0.7658$$

I2:

$$o_a(I_2) = \phi(1 + 0 + 1) = \frac{1}{1+e^{-1\times 2}} = 0.8808; \qquad o_b(I_2) = \phi(1.0 + 0.0 + 1.0) = \frac{1}{1+e^{-1\times 2}} = 0.8808$$

$$o_c(I_2) = \phi(1.0 + 0.1o_a(I_2) + 0.1o_b(I_2)) = \phi(1.1762) = 0.7643$$

I3:

$$o_a(I_3) = \phi(1.0 + 0.5 + 0.5) = \frac{1}{1+e^{-1\times 2}} = 0.8808; \qquad o_b(I_3) = \phi(1.0 + 0.5 + 0.5) = \frac{1}{1+e^{-1\times 2}} = 0.8808$$

$$o_c(I_3) = \phi(1.0 + 0.1o_a(I_3) + 0.1o_b(I_3)) = \phi(1.1762) = 0.7643$$

What is the mean squared error of the network over the three instances? [**0.5 pt**]

Squared Error:

I1: 0.5 (da-oa)^2= 0.5 (**0**-0.7658)^2=**0.2932**

I2: 0.5 (da-oa)^2= 0.5 (**1**-0.7643)^2=**0.02779**

I3: 0.5 (da-oa)^2= 0.5 (**1**-0.7643)^2=**0.02779**

Total Squared Error E(W)= 0.34878

Mean Squared Error E(W)/3 = 0.11626

Assuming instance I_1 is fed to the network to update the weight, calculate local gradient of each node [0.5 pt]

$\delta_c = o_c (1-o_c)*(d_c-o_c)$ = **-0.13734**

$\delta_b = o_b (1-o_b)*\Sigma(w_{bc} * \delta c)$ = **-0.0009628**

$\delta_a = o_a (1-o_a)*\Sigma(w_{ac} * \delta c)$ = **-0.0009628**

please update the network weight (using backpropogation rule) by using this instance (list major steps and results). [1**.0 pts**]

Δwc =[Δw0c, Δwac, Δwbc] = η * δj * x ij		=	-0.02746862	-0.0253849	-0.0253849
w0c	1	+	-0.02746862	=	**0.9725**
wac	0.1	+	-0.0253849	=	**0.0746**
wbc	0.1	+	-0.0253849	=	**0.0746**

For B	Δw = η * δj * x ij	=	-0.00019257	-0.00019257	-9.6E-05
w0b	1	+	-0.00019257	=	**0.9998**
w1b	1	+	-0.00019257	=	**0.9998**
w2b	1	+	-9.6E-05	=	**0.9999**

For A	Δw = η * δj * x ij	=	-0.00019257	-0.00019257	-9.6E-05
w0a	1	+	-0.00019257	=	**0.9998**
w1a	1	+	-0.00019257	=	**0.9998**
w2a	1	+	-9.6283E-05	=	**0.9999**

Question 7 [2 pts]: Table 1 shows a toy dataset with six instances (including two features and one class label), please use t_1=(-1, 0.5) and t_2=(1, -0.5) as the two centers, and use the following RBF kernels to convert the six instances into a new space:

- Please use Gaussian RBF kernel to (σ=1) convert the six instances into new space. Report the new feature values for each instance.
- Please use Multiquadratics RBF kernel to (c=1) convert the six instances into new space. Report the new feature values for each instance.
- Please use Inverse Multiquadratics kernel to (c=1) convert the six instances into new space. Report the new feature values for each instance.
- Please use hyperspheric RBF kernel to (c=1.5) convert the six instances into new space. Report the new feature values for each instance.

Table 1:

Instance Index	F_1	F_2	Label
1	1	1	1
2	1	-1	1
3	0	-1	1
4	-1	-1	0
5	-1	1	0
6	0	1	0

Solutions:

- Please use Gaussian RBF kernel to (σ=1) convert the six instances into new space. Report the new feature values for each instance.

Instance Index	$\varphi_1()$	$\varphi_2()$	Label
1	$r=\|\|X_1-t_1\|\|=2.062$ $e^{(-\frac{r^2}{2\sigma^2})}=0.119$	$r=\|\|X_1-t_2\|\|=1.5$ $e^{(-\frac{r^2}{2\sigma^2})}=0.325$	1
2	$r=\|\|X_2-t_1\|\|=2.5$ $e^{(-\frac{r^2}{2\sigma^2})}=0.044$	$r=\|\|X_2-t_1\|\|=0.5$ $e^{(-\frac{r^2}{2\sigma^2})}=0.882$	1
3	$r=\|\|X_3-t_1\|\|=1.803$ $e^{(-\frac{r^2}{2\sigma^2})}=0.197$	$r=\|\|X_3-t_1\|\|=1.118$ $e^{(-\frac{r^2}{2\sigma^2})}=0.535$	1

4	r=$\|\|X_4-t_1\|\|$=1.5 $e^{(-\frac{r^2}{2\sigma^2})}$ =0.325	r=$\|\|X_4-t_1\|\|$=2.062 $e^{(-\frac{r^2}{2\sigma^2})}$ =0.119	0
5	r=$\|\|X_5-t_1\|\|$=0.5 $e^{(-\frac{r^2}{2\sigma^2})}$ =0.882	r=$\|\|X_5-t_1\|\|$=2.5 $e^{(-\frac{r^2}{2\sigma^2})}$ =0.044	0
6	r=$\|\|X_6-t_1\|\|$=1.118 $e^{(-\frac{r^2}{2\sigma^2})}$ =0.535	r=$\|\|X_6-t_1\|\|$=1.803 $e^{(-\frac{r^2}{2\sigma^2})}$ =0.197	0

- Please use Multiquadratics RBF kernel to (c=1) convert the six instances into new space. Report the new feature values for each instance.

Instance Index	$\varphi_1()$	$\varphi_2()$	Label
1	r=$\|\|X_1-t_1\|\|$=2.062 $\sqrt{r^2+c^2}$ =2.291	r=$\|\|X_1-t_2\|\|$=1.5 $\sqrt{r^2+c^2}$ =1.803	1
2	r=$\|\|X_2-t_1\|\|$=2.5 $\sqrt{r^2+c^2}$ =2.693	r=$\|\|X_2-t_1\|\|$=0.5 $\sqrt{r^2+c^2}$ =1.118	1
3	r=$\|\|X_3-t_1\|\|$=1.803 $\sqrt{r^2+c^2}$ =2.061	r=$\|\|X_3-t_1\|\|$=1.118 $\sqrt{r^2+c^2}$ =1.5	1
4	r=$\|\|X_4-t_1\|\|$=1.5 $\sqrt{r^2+c^2}$ =1.803	r=$\|\|X_4-t_1\|\|$=2.062 $\sqrt{r^2+c^2}$ =2.291	0
5	r=$\|\|X_5-t_1\|\|$=0.5 $\sqrt{r^2+c^2}$ =1.118	r=$\|\|X_5-t_1\|\|$=2.5 $\sqrt{r^2+c^2}$ =2.692	0
6	r=$\|\|X_6-t_1\|\|$=1.118 $\sqrt{r^2+c^2}$ =1.5	r=$\|\|X_6-t_1\|\|$=1.803 $\sqrt{r^2+c^2}$ =2.062	0

- Please use Inverse Multiquadratics kernel to (c=1) convert the six instances into new space. Report the new feature values for each instance.

Instance Index	$\varphi_1()$	$\varphi_2()$	Label
1	r=$\|\|X_1-t_1\|\|$=2.062 $\frac{1}{\sqrt{r^2+c^2}}$=0.436	r=$\|\|X_1-t_2\|\|$=1.5 $\frac{1}{\sqrt{r^2+c^2}}$=0.555	1
2	r=$\|\|X_2-t_1\|\|$=2.5 $\frac{1}{\sqrt{r^2+c^2}}$=0.371	r=$\|\|X_2-t_1\|\|$=0.5 $\frac{1}{\sqrt{r^2+c^2}}$=0.894	1
3	r=$\|\|X_3-t_1\|\|$=1.803 $\frac{1}{\sqrt{r^2+c^2}}$=0.485	r=$\|\|X_3-t_1\|\|$=1.118 $\frac{1}{\sqrt{r^2+c^2}}$=0.667	1
4	r=$\|\|X_4-t_1\|\|$=1.5 $\frac{1}{\sqrt{r^2+c^2}}$=0.555	r=$\|\|X_4-t_1\|\|$=2.062 $\frac{1}{\sqrt{r^2+c^2}}$=0.436	0
5	r=$\|\|X_5-t_1\|\|$=0.5 $\frac{1}{\sqrt{r^2+c^2}}$=0.894	r=$\|\|X_5-t_1\|\|$=2.5 $\frac{1}{\sqrt{r^2+c^2}}$=0.371	0
6	r=$\|\|X_6-t_1\|\|$=1.118 $\frac{1}{\sqrt{r^2+c^2}}$=0.667	r=$\|\|X_6-t_1\|\|$=1.803 $\frac{1}{\sqrt{r^2+c^2}}$=0.485	0

- Please use hyperspheric RBF kernel to (c=1.5) convert the six instances into new space. Report the new feature values for each instance.

Instance Index	$\varphi_1()$	$\varphi_2()$	Label
1	r=$\|\|X_1-t_1\|\|$=2.062 r<= c? $\rightarrow$ 0	r=$\|\|X_1-t_2\|\|$=1.5 r<= c? $\rightarrow$ 1	1
2	r=$\|\|X_2-t_1\|\|$=2.5 r<= c? $\rightarrow$ 0	r=$\|\|X_2-t_1\|\|$=0.5 r<= c? $\rightarrow$ 1	1
3	r=$\|\|X_3-t_1\|\|$=1.803	r=$\|\|X_3-t_1\|\|$=1.118	1

	r<= c? → 0	r<= c? → 1	
4	r=||X_4-t_1||=1.5 r<= c? → 1	r=||X_4-t_1||=2.062 r<= c? → 0	0
5	r=||X_5-t_1||=0.5 r<= c? → 1	r=||X_5-t_1||=2.5 r<= c? → 0	0
6	r=||X_6-t_1||=1.118 r<= c? → 1	r=||X_6-t_1||=1.803 r<= c? → 0	0

Question 8 [**1.5 pts**]: Design an RBF network to solve the XOR problem with input and output given as follows, where X_1 and X_2 are features and Y is the output.

X_1	X_2	Y
0	0	-1
1	0	1
0	1	1
1	1	-1

Assume the RBF network is showing as follows, and we use Gaussian RBF function: $\varphi(r) = \exp(-\frac{r^2}{2\sigma^2})$ with σ=1, and two centers are t_1=(0.1, 0.1), t_2=(0.9, 0.9), respectively. Please use pseudo-inverse to calculate the weight values W=[w_0, w_1, w_2] of the output node [1 **pt**], and validate your results with respect to the four instances [**0.5 pt**]

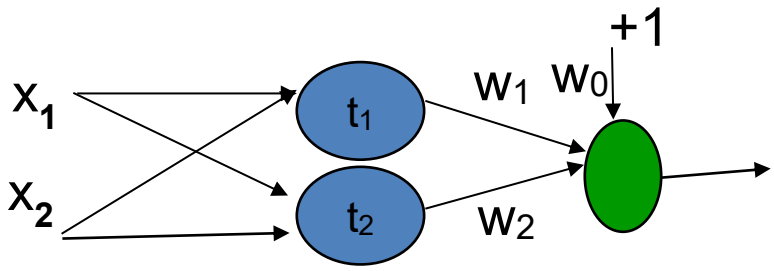


Figure 6: Single layer RBF neural network

Solutions:

$$\varphi_1(\| x - t_1 \|) = e^{-\frac{|x-t_1|^2}{2\sigma}}$$

$$\varphi_2(\| x - t_2 \|) = e^{-\frac{|x-t_2|^2}{2\sigma}}$$

Instances	$\varphi_1(\| x - t_1 \|)$	$\varphi_2(\| x - t_2 \|)$
(0,0)	0.99	0.4449
(1,0)	0.6637	0.6637
(0,1)	0.6637	0.6637
(1,1)	0.4449	0.99

Converted instances:

ϕ			d (desired output)
1	0.99	0.4449	**-1**
1	0.6637	0.6637	**1**
1	0.6637	0.6637	**1**
1	0.4449	0.99	**-1**

Pseudo inverse matrix:

ϕ^+			
-6.174	6.674	6.674	-6.174
5.568	-4.651	-4.651	3.734
3.734	-4.651	-4.651	5.568

Weight W=ϕ^+d:

W=[25.696, -18.604, -18.604]^T

Validate using the calculated weights

Instances	$\phi 1$	$\phi 2$	$Y=w_0+w_1\phi_1+w_2\phi_2$	Class
(0,01)	0.99	0.4449	-1.0001	-1
(1,0)	0.6637	0.6637	0.9999	1
(0,1)	0.6637	0.6637	0.9999	1
(1,1)	0.4449	0.99	-1.0001	-1

Question 9 [**1.5 pts**]: Design an RBF network to solve the XOR problem with input and output given as follows, where X_1 and X_2 are features and Y is the output.

X_1	X_2	Y
0	0	-1
1	0	1
0	1	1
1	1	-1

Assume the RBF network is showing as follows, and we use Multiquadric RBF function: $\varphi(r)=\sqrt{r^2+c^2}$ with c=0.5, and three centers are t_1=(0.1, 0.1), t_2=(0.9, 0.9), t_3=(0.5, 0.5), respectively. Please use pseudo-inverse to calculate the weight values W=[w_0, w_1, w_2, w_3] of the output node [1 **pt**], and validate your results with respect to the four instances [**0.5 pt**]

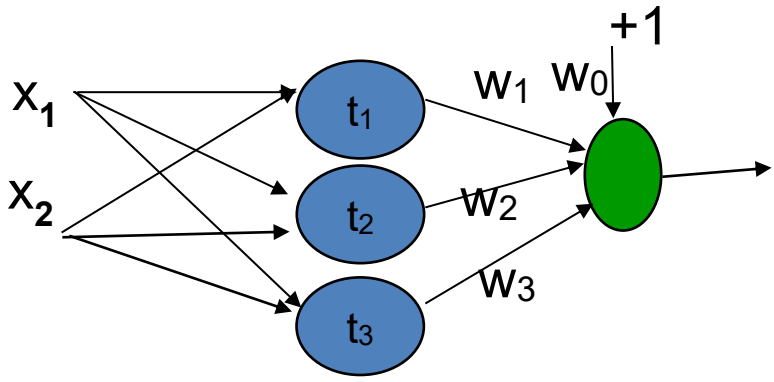


Figure 7: A single layer RBF neural network

Solutions:

$$\varphi_1(\| x - t_1 \|) = \sqrt{\| x - t_1 \|^2 + c^2}$$
$$\varphi_2(\| x - t_2 \|) = \sqrt{\| x - t_2 \|^2 + c^2}$$
$$\varphi_3(\| x - t_3 \|) = \sqrt{\| x - t_3 \|^2 + c^2}$$

Instances	$\varphi_1(\| x - t_1 \|)$	$\varphi_2(\| x - t_2 \|)$	$\varphi_3(\|\|x - t_3\|\|)$
(0,0)	0.5196	1.3675	0.8660
(1,0)	1.0344	1.0344	0.8660
(0,1)	1.0344	1.0344	0.8660
(1,1)	1.3675	0.5196	0.8660

Converted instances:

ϕ				d (desired output)
1	0.5196	1.3675	0.8660	**-1**
1	1.0344	1.0344	0.8660	**1**
1	1.0344	1.0344	0.8660	**1**
1	1.3675	0.5196	0.8660	**-1**

Pseudo inverse matrix:

ϕ^+			
3.2532	-2.9674	-2.9674	3.2532
-3.3415	2.7518	2.7518	-2.1621
-2.1621	2.7518	2.7518	-3.3415
2.8172	-2.5698	-2.5698	2.8172

Weight W=ϕ^+d:

$W=[-12.441, 11.007, 11.007, -10.774]^T$

Validate using the calculated weights

Instances	$\phi 1$	$\phi 2$	$\phi 3$	$Y=w_0+w_1\phi_1+w_2\phi_2+w_3\phi_3$	Class
(0,0)	0.5196	1.3675	0.8660	-1.000	-1
(1,0)	1.0344	1.0344	0.8660	1.000	1
(0,1)	1.0344	1.0344	0.8660	1.000	1
(1,1)	1.3675	0.5196	0.8660	-1.000	-1

Programming Tasks: For all programming tasks, please submit the Notebook (or Markdown) as html files for grading (**the notebook must include scrips/code and the results of the script**)

Question 10 [1.5 pts] Please revise the **MultiLayer NN Decision Boundary Notebook** in the Canvas to implement following tasks

- Generate a moon shaped dataset with two class lables (including 0.3 noise). Split the data into 60% training and 40% test data. Use 60% training data to train a neural network with 3 hidden ~~layer~~ nodes. Show the training instances on the plot (color node into different colors, depending on the lables). Also show the three lines corresponding to the three hidden nodes in the same plot [0.5 pt]
- Retrain the network using 4 hidden nodes, and 5 hidden nodes, respectively. Show the four lines corresponding to the four hidden nodes, and the five lines corresponding to the five hidden nodes, respectively, on two separated plots (show the training instances on the same plot (color node into different colors, depending on the labels)) [0.5 pt]
- Explain how does the neural network decision surface change, when increasing the number of hidden nodes from 3, to 4, to 5. Explain what are the benefits and risk of increasing number of hidden nodes, respectively [0.5 pt]

Question 11 [2 pts] The **Multi-Layer NN Face Recognition Notebook** in the Canvas shows detailed procedures about how to use Oivetti face dataset (from AT&T) to train Neural Network classifiers for face classification. The dataset in the Canvas also provides "olivetti_faces.npy" and "olivetti_faces_target.npy", which includes 400 faces in 40 classes (40 different person). Please implement following face recognition task using Neural Networks.

- Please show at least one face images for each class in the Olvetti face dataset100. Randomly split the dataset into 60% training and 40% test samples. Train a one-hidden layer neural network with 10 hidden nodes. Report the classification accuracy of the classifier on the test set [1 pt]
- Please use one time 10-fold cross validation to compare the performance of different neural network architectures, including (1) one-hidden layer NN with 10 hidden nodes, (2) one-hidden layer NN with 50 hidden nodes, (3) one-hidden layer NN with 500 hidden nodes, and (4) two-hidden layer NN with 50 hidden nodes (1st layer) and 10 hidden nodes (2n layer). Please report and compare the cross-validation accuracy of the four neural networks, and conclude which classifier has the best performance [1 pt].

Question 12 [Extra Credit: 3 pts]

Please download IMDB50000.csv dataset from Canvas, and use a programming language (Python, R, etc.) to implement tasks below. The IMDB50000.csv file includes 50,000 movie reviews (from IMDB) and the sentiment (positive vs. negative) of the reviewer (there are 25,000 positive reviews and 25,000 negative reviews). Each review (each row) contains two parts. The first column is the reviews (text), and the second column is the sentiment (positive or negative).

- Read reviews from the IMDB50000.csv. Tokenize each review using space key, so each review is represented as a set of tokens (words). Use the top 1,000 most frequent tokens (words) as features to represent each review (so each review is represented as an instance with 1000 features). The value of each feature is 1 if the view has the corresponding token/word, or 0 otherwise. Use 1 as the label of the positive review, and 0 as the label of the negative review (so each review is represented using 1,000 features and a class label). Report the shape of your data frame, and use .head() to show the first several rows of the data frame. [0.5 pt]
- Randomly select 80% instances as training set, and the remaining 20% instances as test set. Create a one hidden layer neural network (with 500 hidden nodes). Train the network using training set, and validate the performance on the test set. Report the accuracy on the test set. [0.5 pt]
- Design one solution to improve the accuracy (which is better than the accuracy than step 2).
    - Explain the motivation of your design (what motivate your design) [0.5 pt], the implementation [0.5]
    - Report the results of your new model. Use a plot to compare new model's accuracy vs. the accuracy from step 2 (must use same training and same test sets) [0.5 pt].
    - Explain any thoughts/changes you may consider to further improving your model performance [0.5 pt]
    - Hint for model design:

- You can consider using preprocessing to find better (more informative) words. E.g., removing function words, such as “a”, “the”, “of”, removing punctations etc.
- You can consider using information gain to find important features.
- You can consider using TF-IDF, instead of using 0/1 feature values
- You can consider changing your network structures, e.g., using two hidden layer NN.