UNIVERSITY OF THESSALY

NEURO-FUZZY COMPUTING

ECE447

1st Problem Set

Alexandra Gianni   Nikos Stylianou

ID: 3382   ID: 2917

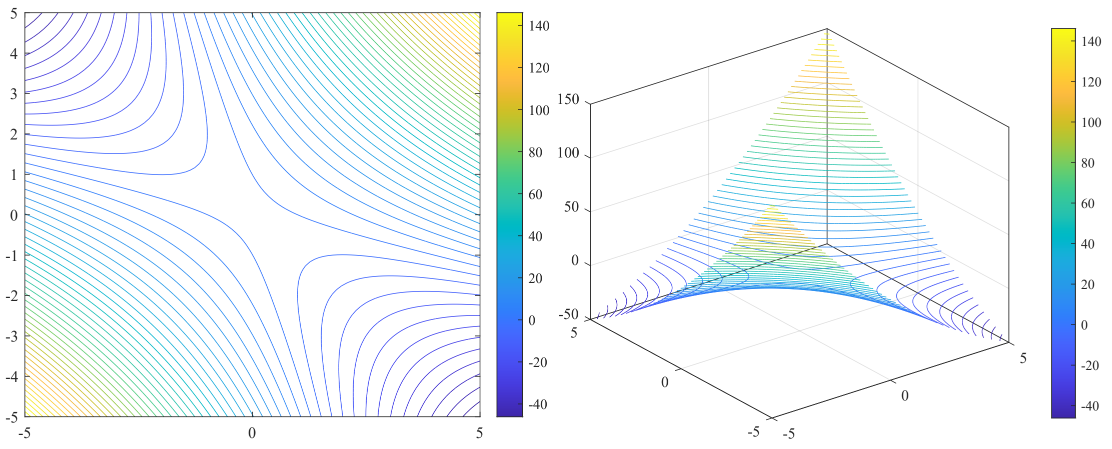
December 18, 2023

Problem 1

The contour lines of $f(x, y)$ are plotted with the following MATLAB code and are presented in figure 1.

```
function [Z] = plot_contour(start_num, end_num)

        x = linspace(start_num, end_num, 100);
        y = x;
        [X, Y] = meshgrid(x, y);
        Z = X.^2 + 4*X.*Y + Y.^2;
        contour(X, Y, Z, 40);
        xlabel('X');
        ylabel('Y');
end
```



(a) 2D Plot (b) 3D Plot

Figure 1: Contour lines of $f(x, y)$

A general formula of a quadratic equation is $f(x, y) = ax^2 + 2bxy + cy^2$. Writing our formula in the previous form, we find that $a = 1,\ b = 2,\ c = 1$. Calculation of the discriminant can help us calculate the location of the function's local minimum/maximum.

$$
\begin{gathered}
D = \begin{bmatrix} f_{xx} & f_{xy} \\ f_{yx} & f_{yy} \end{bmatrix} = f_{xx} f_{yy} - f_{xy}^2 = 2 \times 2 - 4^2 = -12 < 0, \quad \text{where} \\
f_{xx} = \frac{\partial^2 f}{\partial x^2} = 2, \quad f_{yy} = \frac{\partial^2 f}{\partial y^2} = 2, \quad f_{xy} = \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}\right) = 4D = 2 \times 2 - 4^2 = -12 < 0.
\end{gathered} \tag{1}
$$

So, we only have to find the point at which $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ are equal to 0. This point will be a saddle point at which the gradients in each orthogonal direction are 0, but this point is neither a local minimum nor a maximum. More precisely:

$$
\begin{cases} \dfrac{\partial f}{\partial x} = 2x + 4y = 0 \\ \dfrac{\partial f}{\partial y} = 4x + 2y = 0 \end{cases} \Rightarrow \begin{cases} x = 0 \\ y = 0 \end{cases} \tag{2}
$$

Thus, the point $(x, y) = (0, 0)$ is the saddle point mentioned for the given function and this can be justified using the plotted contour lines.

Problem 2

Gradient Descent is an optimization algorithm used to minimize a function by iteratively moving in the direction of steepest descent defined by the negative value of the gradient. It is mostly used to update the parameters of models.
In this exercise we need to execure two iterations of the Gradient Descent to the function

$$f(x_1, x_2) = (x_1 + 2 \cdot x_2 - 7)^2 + (2 \cdot x_1 + x_2 - 5)^2$$

with initial point $x_0 = (-9.5, 9.5)$.
The steps that we must use each time are specific.

FIRST ITERATION k = 0

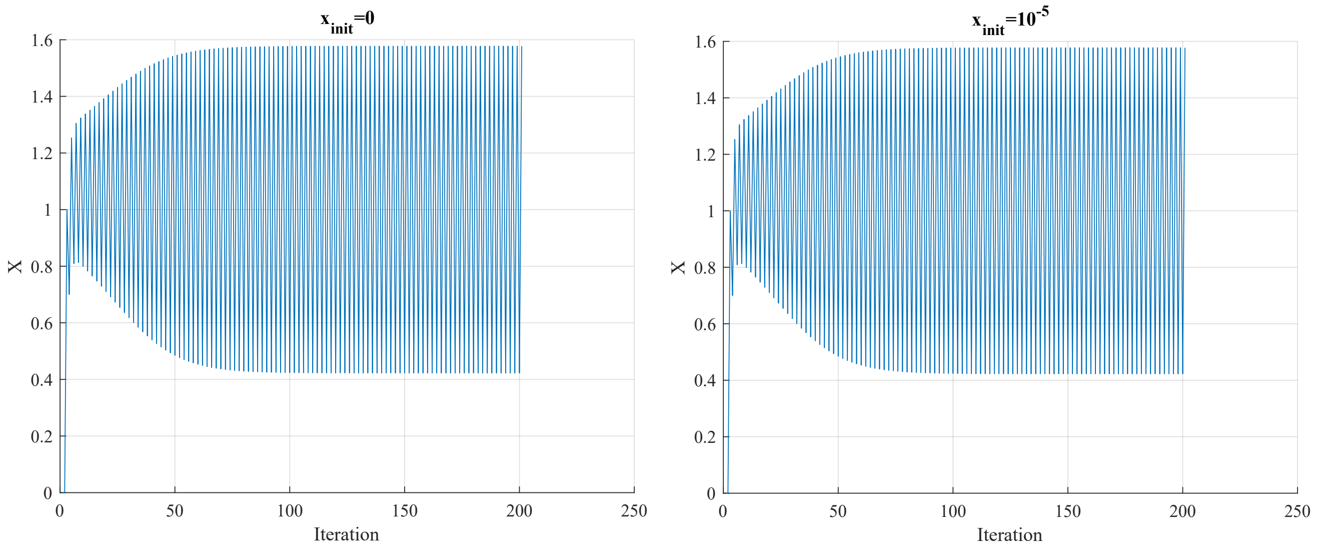
Step1: Calculate the Gradient

Problem 3

Henon map is a dynamic system described by the recursive equation

$$x_{k+1} = 1 - ax_k^2 + bx_{k-1}$$

A lot of dynamic systems transition into chaos as gain or control is increased to a certain point and this system falls into this category. In this problem, we will plot the trajectories of sequences $x_0...x_i$ and describe it. The first parameters are (a,b) = (0.3, 0.4) and the trajectories of the sequences are presented in figure 2.



(a) (b)

Figure 2: Trajectories of Henon map's sequence with (a,b) = (0.3, 0.4)

From the produced plots, this system with parameters (0.3, 0.4) is periodic after converging.

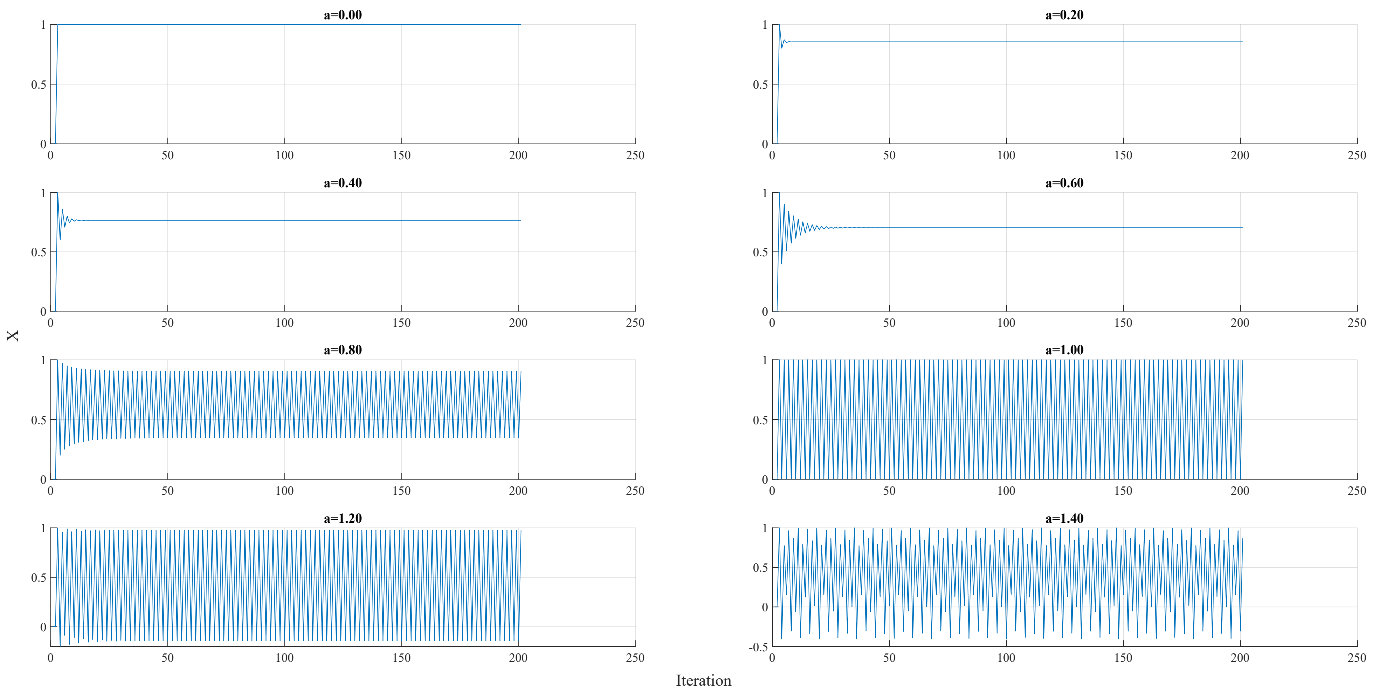


Figure 3: Sequence output for different a and $b = 0$

(a) Multiple a values

Figure 3 shows output of the sequence for different values of a and $b = 0$. When $a \leq 0.6$, the output swings for a bit and then stabilizes to a fixed number, with that number decreasing while a approaches 0.6. When $a \geq 0.8$, output starts to oscillate. As a approaches 1, the oscillation's amplitude is getting bigger until it reaches value 1. After a surpasses 1, the oscillation starts to break down, as shown in the last two sub-figures. The greater a, the greater the disturbance on the oscillation thus chaos is created.

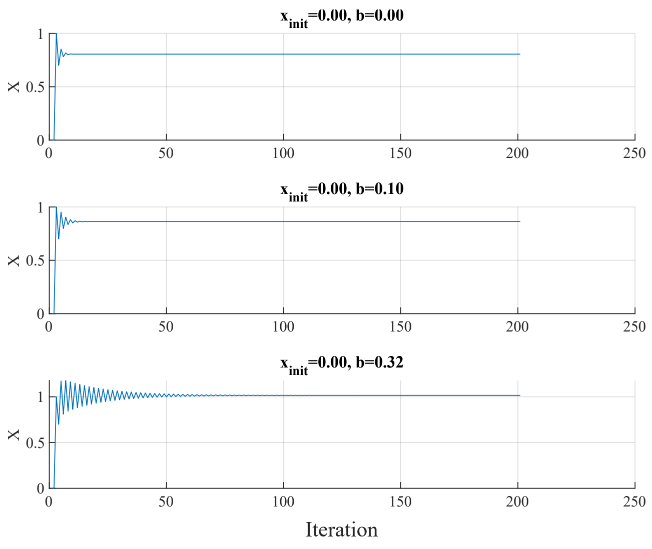
(b) Multiple initial x values

Figure 4 shows the trajectory of sequences with different x_{init} and b, with $a = c$, where $c = 0.3$ from previous question. Inspecting this figure, we can understand some things about those parameters and what they affect on the system, as even with $b = 0$, the system performs a damped oscillation for the first 50 (at most) x_i. For a given x_{init} (ie. 0), increasing b increases the amplitude of oscillation as well as the overshoot percentage (if we see it as an system response).

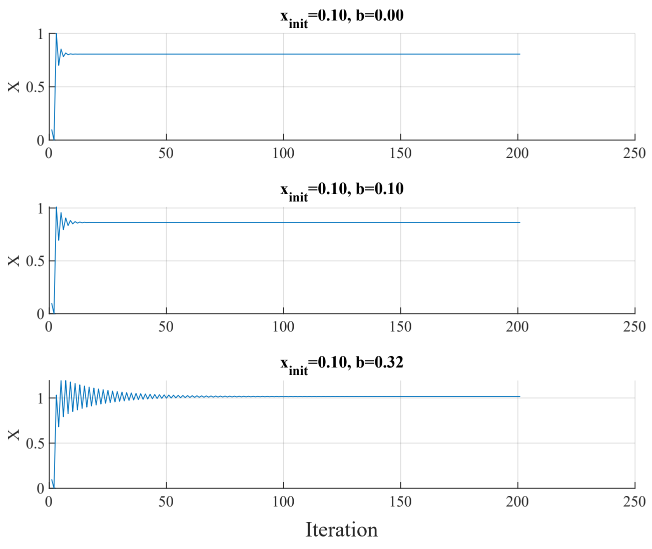
PS: Overshoot % is the percentage of system's maximum value that surpasses its steady-state value over the latter.

(c) (a,b) = (0.3675, 0.3)

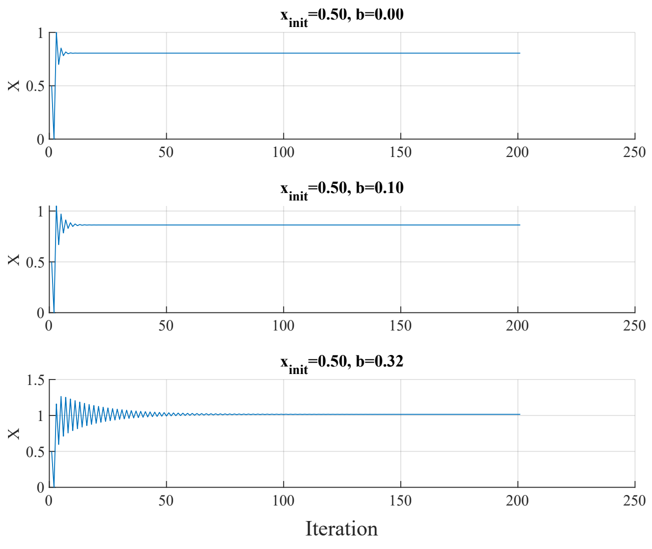
By setting these values to the system in figure 5, we observe that the sequence starts to oscillate with a very high frequency. As the sequence progresses, the observed oscillation's amplitude dampens and settles around a fixed number, whilst still oscillating.



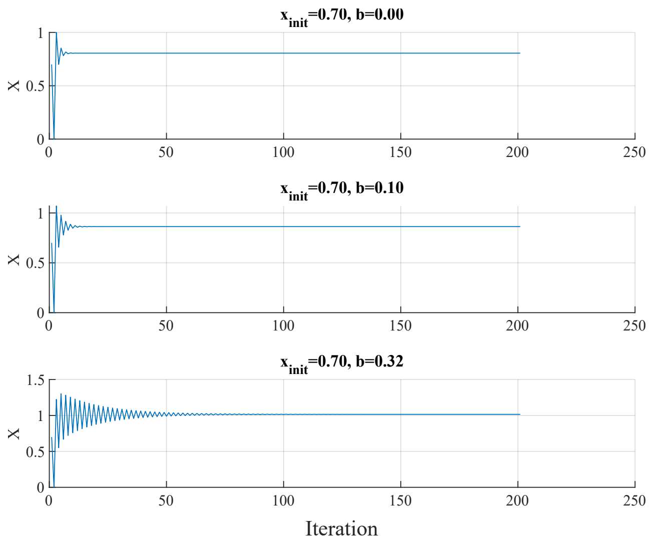
(a)



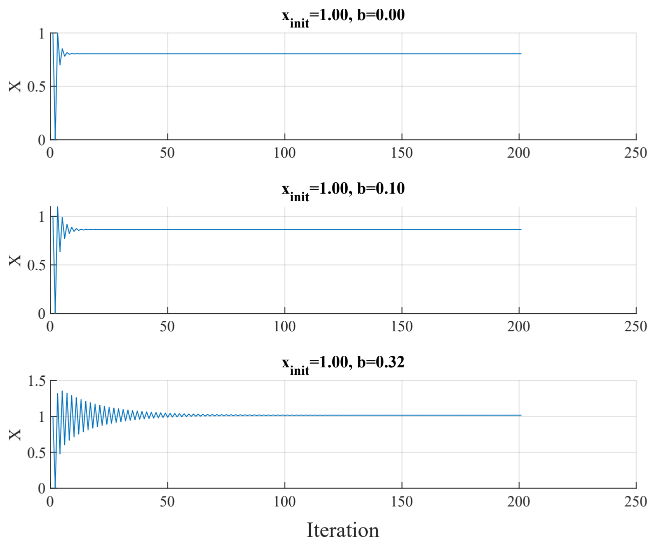
(b)



(c)

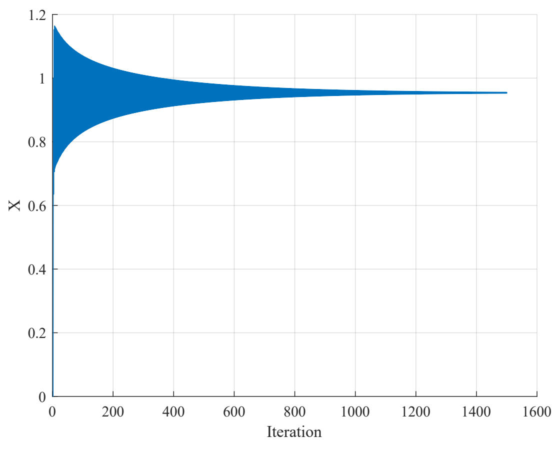


(d)



(e)

Figure 4


(a) Normal trajectory

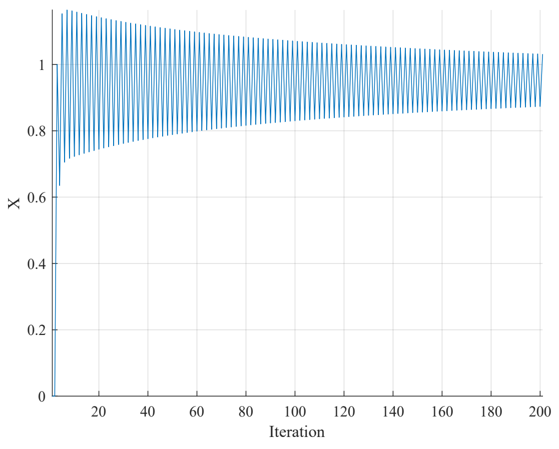
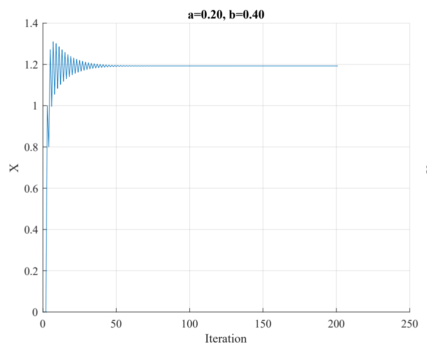

(b) Zoomed in

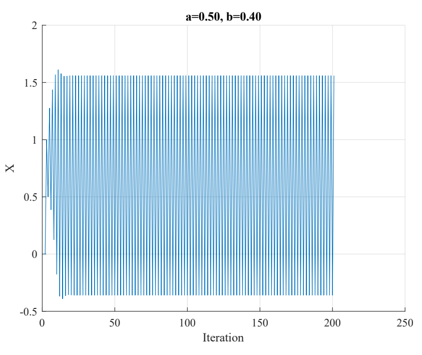
Figure 5: Sequence's trajectory with (0.3675, 0.3)

(d) Multiple a values

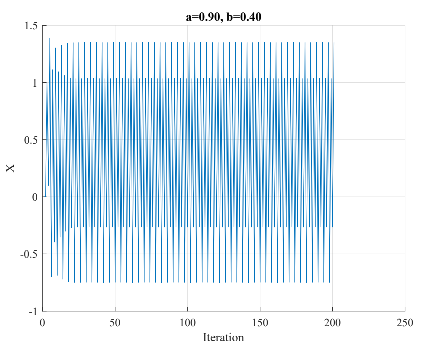
By changing a parameter, we can spot and characterize system's output. Observing figure 6, we understand that increasing the parameter's value increases oscillation duration. At first, (when a is smaller to b), the system oscillates for a bit and then dampens around a fixed number. But as a approaches b, oscillation duration gets bigger until the moment where $a = b$. From this value and over, oscillation lasts for all x_i. As a increases beyond b, the oscillations begin to deteriorate, and beyond a certain point of a, chaos ensues.



(a)



(b)



(c)

Figure 6

Problem 4

In this problem, we will express the derivatives in respect to the matching activation function.

(a) LogSig

This activation function is expressed as

$$S(x) = \frac{1}{1 + e^{-x}}$$

Multiplying itself with $(1+e^{-x})$ gives

$$\left(1+e^{-x}\right)\ S(x)=1 \Leftrightarrow e^{-x}=\frac{1}{S(x)}-1$$

So, activation function's derivative will be

$$\begin{aligned}\frac{dS}{dx}&=\frac{d\left(\left(1+e^{-x}\right)^{-1}\right)}{dx}=\left(1+e^{-x}\right)^{-2}\ e^{-x}=S^2(x)\ \left(\frac{1}{S(x)}-1\right)\\&=S(x)-S^2(x)=S(x)\ \left(1-S(x)\right)\end{aligned}$$

(b) TanSig

Activation function is

$$S(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}}$$

Its derivative is

$$\begin{aligned}\frac{dS}{dx}&=\frac{\left(e^x+e^{-x}\right)\left(e^x+e^{-x}\right)-\left(e^x-e^{-x}\right)\left(e^x-e^{-x}\right)}{\left(e^x+e^{-x}\right)^2}=\\&=\frac{\left(e^x+e^{-x}\right)^2-\left(e^x-e^{-x}\right)^2}{\left(e^x+e^{-x}\right)^2}=1-\left(\frac{e^x-e^{-x}}{e^x+e^{-x}}\right)^2=1-S^2(x)\end{aligned}\tag{3}$$

(c) Swish

Activation function is

$$S(x)=\frac{x}{1+e^{-x}}$$

The derivative in respect to x is

$$\frac{dS}{dx}=\frac{1+e^{-x}+xe^{-x}}{\left(1+e^{-x}\right)^2}=\frac{1+e^{-x}}{\left(1+e^{-x}\right)^2}+\frac{xe^{-x}}{\left(1+e^{-x}\right)^2}=\frac{1}{1+e^{-x}}+x\frac{e^{-x}}{\left(1+e^{-x}\right)^2}$$

Rewriting the function gives us

$$\frac{S(x)}{x}=\frac{1}{1+e^{-x}}\quad\text{and}\quad e^{-x}=\frac{x-S(x)}{S(x)}$$

So, continuing with the derivative:

$$\begin{aligned}\frac{dS}{dx}&=\frac{1}{1+e^{-x}}+x\frac{e^{-x}}{\left(1+e^{-x}\right)^2}=\frac{S(x)}{x}+\frac{e^{-x}}{1+e^{-x}}\frac{x}{1+e^{-x}}=\frac{S(x)}{x}+S(x)\frac{x}{1+e^{-x}}=\\&=\frac{S(x)}{x}+S(x)\frac{S(x)}{x}e^{-x}=\frac{S(x)}{x}\left(1+S(x)e^{-x}\right)=\frac{S(x)}{x}\left(1+S(x)\frac{x-S(x)}{S(x)}\right)=\\&=\frac{S(x)}{x}\left(1+x-S(x)\right)=\frac{S(x)}{x}+S(x)-\frac{S^2(x)}{x}\end{aligned}$$

Let $\sigma=\frac{1}{1+e^{-x}}$, thus

$$\frac{dS}{dx}=\frac{S(x)}{x}+S(x)-\frac{S^2(x)}{x}=\sigma+S(x)-\sigma S(x)=S(x)+\sigma\left(1-S(x)\right)\tag{4}$$

(d) Custom tanh

Activation function is

$$S(x) = x \ \tanh\left(\ln\left(1+e^x\right)\right) = x \ \tanh\left(g(x)\right), \ \text{where } g(x) = \ln\left(1+e^x\right).$$

By calculating the derivative of this function as is, we get

$$\frac{dS}{dx} = \tanh(g(x)) + x \ \tanh^{'}(g(x)) \ \frac{dg}{x} = \tanh(g(x)) + x \ \tanh^{'}(g(x)) \ \frac{1}{1+e^{-x}} =$$
$$= \tanh(g(x)) + \tanh^{'}(g(x)) \ x \ \frac{\text{Swish}(x)}{x} = \frac{S(x)}{x} + \tanh^{'}(g(x)) \ \text{Swish}(x) =$$
$$= \frac{S(x)}{x} + (1 - \tanh^2(g(x))) \ \text{Swish}(x) = \frac{S(x)}{x} + \text{Swish}(x) - \tanh^2(g(x)) \ \text{Swish}(x)$$

So, $\phi(x, S)$ is

$$\frac{S(x)}{x} + \text{Swish}(x) - \frac{S^2(x)}{x^2} \ \text{Swish}(x), \quad \text{where} \tag{5}$$

$S(x)$ is the activation function and Swish(x) is the Swish activation function from before

Problem 5

The given neural network consists of two layers and of three neurons. On the first one, activation function is `logsig` or `swish` and on the second one is `purelin`. On the left side of figure 7 we see the sketches for all outputs when the activation function is `logsig` and on the right side all outputs with activation function being `swish`.

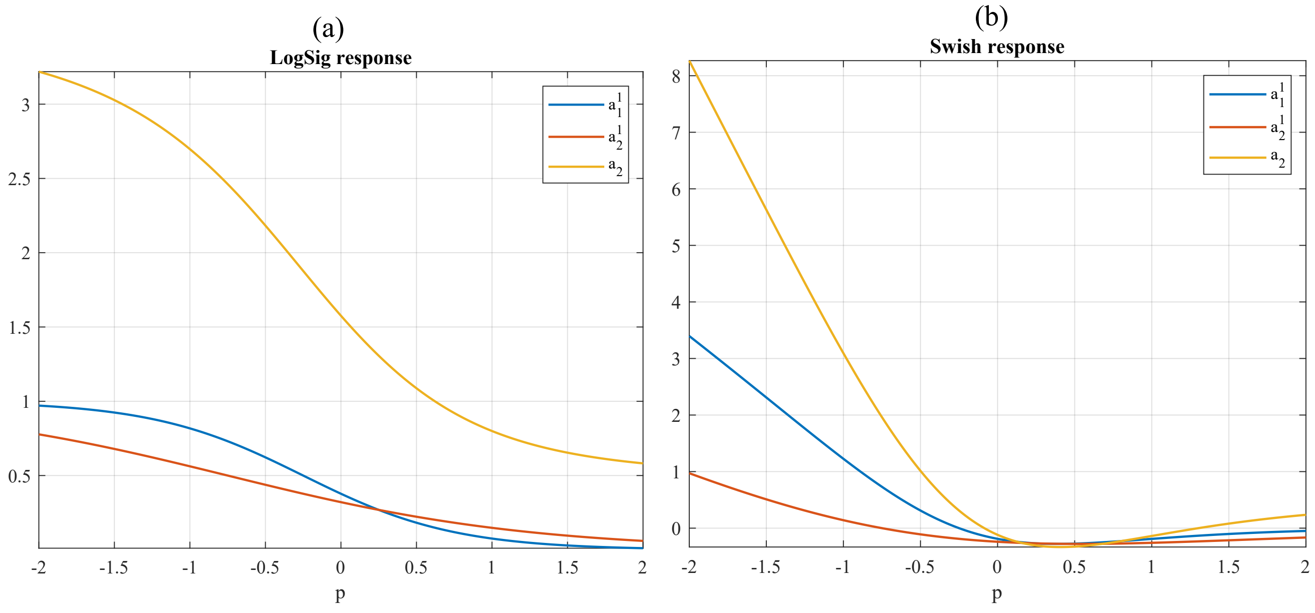


Figure 7: Responses of different outputs of the neural network

Problem 7

(a) Question A

Let an activation function be

$$S(x) = \begin{cases} x^k, & x > L \\ x^k \dfrac{(L+x)^m}{(L+x)^m + (L-x)^m}, & |x| \le L \\ 0, & x < -L \end{cases} \tag{6}$$

This function has some parameters k, L, m called hyperparameters. Using them, this function can approximate others.

(a).1 Swish

In order for our activation function to approximate Swish, we have to see the latter first. Swish's function is

$$S(x) = \frac{x}{1+e^{-x}} = x\frac{e^x}{1+e^x}$$

and is plotted in figure 8. In order to determine the parameters, we have to take a look in the plotted function. When $x > 0$, Swish seems to be almost $y = x$, thus $k = 1$. If we look at $x < 0$, we can se that Swish curves until it settles back to 0. That point x_1 where $Swish(x_1) = 0$ is parameter L. After trying multiple values, we opted with $2e$ as this gives the smaller overall error in that region. After filling k, L, only parameter m is left to be found. In general, x^e is a good approximation of e^x, but an approximation only. Just trying $m = e$ and plotting the created function gets us the plot in figure 8 which is really close to Swish. Finally, the activation function can be written

$$S(x) = \begin{cases} x, & x > 2e \\ x \dfrac{(2e+x)^e}{(2e+x)^e + (2e-x)^e}, & |x| \le 2e \\ 0, & x < -2e \end{cases}$$

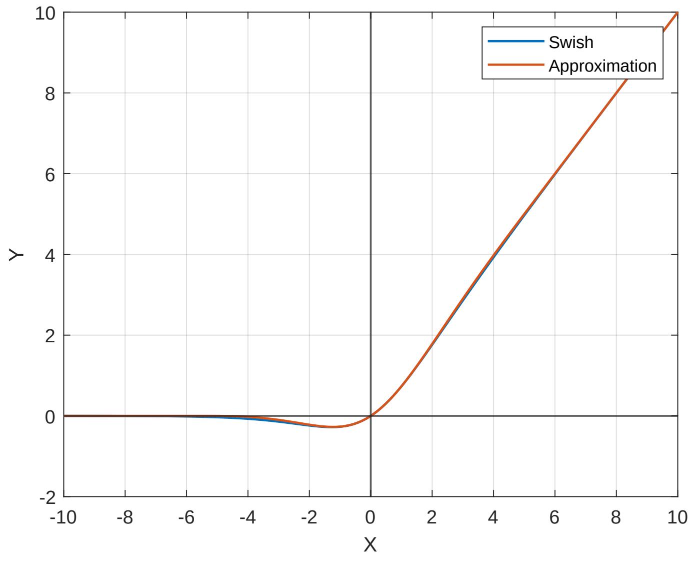


Figure 8: Swish activation function and its approximation.

(a).2 Sigmoid

Sigmoid's function is

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}$$

Just from this type, we understand that is $\frac{Swish(x)}{x}$. So, selecting parameters is easier this time as m and L will not change. Because of the previous equation, parameter k must be decreased by 1, thus $k = 0$. Figure 9 shows both the sigmoid and its approximation.

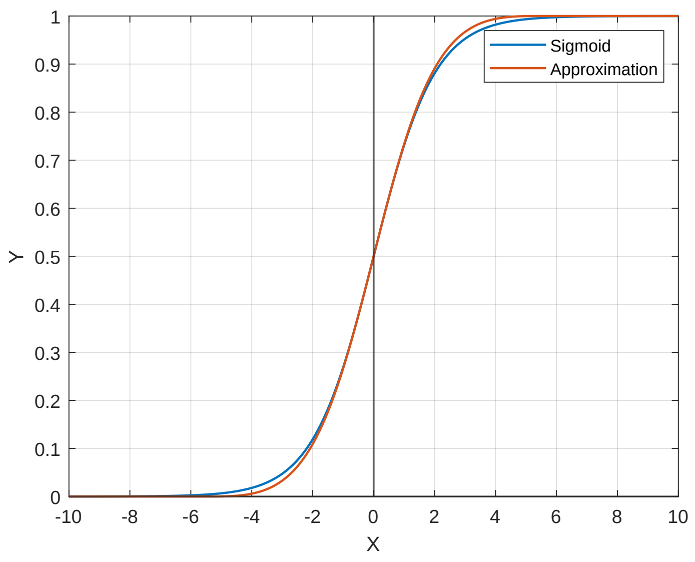


Figure 9: Sigmoid activation function and its approximation.

(a).3 ReLU

ReLU is defined as

$$ReLU(x) = \max(0,x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

From this, we can already determine parameter k to be $k = 1$, as we want $x^k = x \Rightarrow k = 1$. Setting $L = 0$ ensures that the transition point between the linear and non-linear parts of the function occurs at $x = 0$, which is consistent with the behavior of ReLU. Also, setting $m = 0$ means that the denominator term becomes 1 for both the linear and non-linear parts of the function, simplifying the expression. The plotted functions are presented in figure 10

With parameters those mentioned above, $S(x)$ is the following

$$S(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

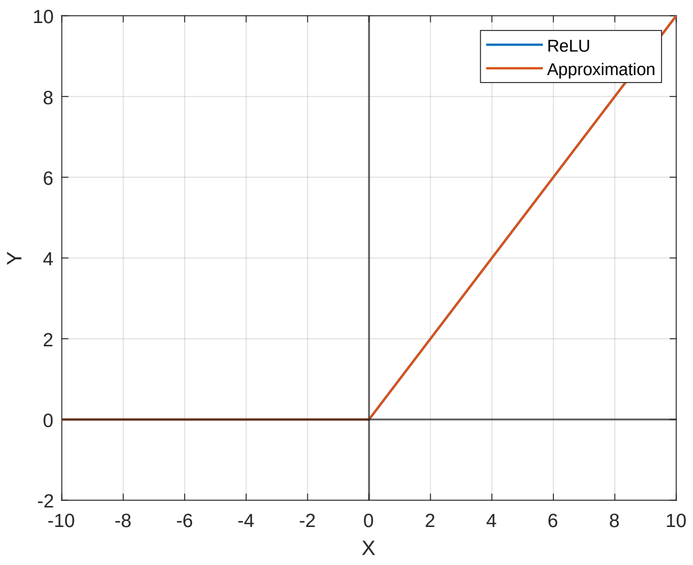


Figure 10: ReLU activation function and its approximation.

(b) Question B

(b).1 Derivative with respect to x

$$\frac{dS}{dx} = kx^{k-1} + x^k \frac{m\left(L+x\right)^m\left(\left(L+x\right)^{m-1} + \left(L-x\right)^{m-1}\right) - \left(L+x\right)^m\left(m\left(L+x\right)^{m-1} + m\left(L-x\right)^{m-1}\right)}{\left(\left(L+x\right)^m + \left(L-x\right)^m\right)^2} =$$
$$= kx^{k-1}$$

(b).2 Derivative with respect to k

We know from algebra that $\frac{d}{dx}\left(a^x\right) = a^x \ln a$. So

$$\frac{dS}{dk} = x^k \ln(k) \frac{\left(L+x\right)^m}{\left(L+x\right)^m + \left(L-x\right)^m}$$

Problem 8

ADALINE is a single-layer artificial neural network that can learn and adapt to non-linear relationships between inputs and outputs.
It consists of a single neuron with a linear activation function. Each input of the neuron has a corresponding weight,which is adapted during training to minimize the error between the network's output and the desired output. To adjust the weights the algorithm uses the learning rule α.

Suppose that we have the following three reference patterns and their targets:

$$\left\{p_1 = \begin{bmatrix} 2 \\ 4 \end{bmatrix}, t_1 = [26]\right\} \quad \left\{p_2 = \begin{bmatrix} 4 \\ 2 \end{bmatrix}, t_2 = [26]\right\} \quad \left\{p_3 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, t_3 = [-26]\right\}$$

The probability of vector p1 is $P_1 = 0.20$, the probability of vector p2 is $P_2 = 0.70$, and the probability of vector p3 is $P_3 = 0.10$.

(a) Question a

The number of inputs to an ADALINE network for each neural network is determined by the dimensionality of our data,not by the number of patterns we have. In our case, each pattern is a 2-dimensional vector. Therefore, our ADALINE network has two inputs, one for each dimension.
In an ADALINE network, the number of weights is equal to the number of inputs. In our case, we have two inputs. Thus, our neural network has two weights, one for each dimension of the input.
From theory, the output of an ADALINE network is a = purelin(W_p+b).

So, the network diagram for the given ADALINE network with no bias that will be trained with these patterns is shown in figure 11.

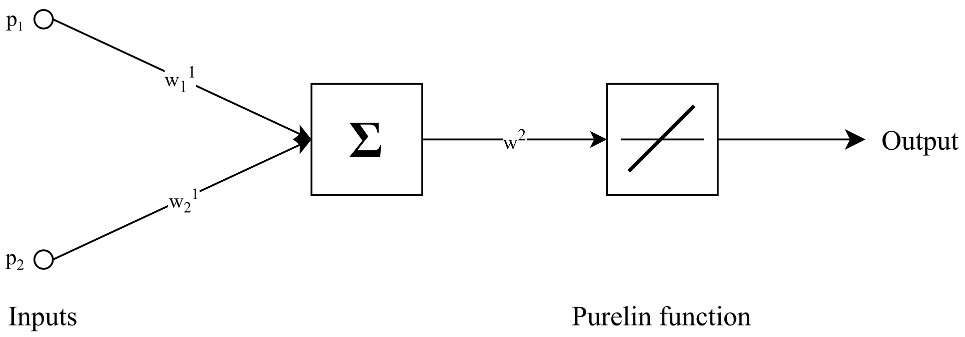


Figure 11: ADALINE neural network architecture

(b) Question b

In order to sketch the contour plot of the mean square error performance index, we first must calculate the various terms of the quadratic function.
Recall that, $F(x) = c - 2 \cdot x^T \cdot h + x^T \cdot R \cdot x$ where,

- c: A scalar constant term. It shifts the function up or down along the y-axis.
- R: Correlation matrix of the input data. It determines the curvature of the function
- h: The cross-correlation between the input data and its associated target. It determines the slope of the function.
- x: The vector of variables (or weights).

These parameters define the shape of the quadratic function. So,we must calculate c,h,R in relation to

$$x = \left[\begin{array}{cc} W_{11} & W_{12} \end{array} \right]$$

The calculations:

$$c = E[t^2] = t_1^2 \cdot p_1 + t_2^2 \cdot p_2 + t_3^2 \cdot p_3 = 26^2 \cdot 0.2 + 26^2 \cdot 0.7 + (-26)^2 \cdot 0.1$$
$$\rightarrow c = 676$$

$$h = E[t \cdot p] = P_1 \cdot t_1 \cdot p_1 + P_2 \cdot t_2 \cdot p_2 + P_3 \cdot t_3 \cdot p_3 =$$
$$0.2 \cdot 26 \cdot \begin{bmatrix} 2 \\ 4 \end{bmatrix} + 0.7 \cdot 26 \cdot \begin{bmatrix} 4 \\ 2 \end{bmatrix} + 0.1 \cdot (-26) \cdot \begin{bmatrix} -2 \\ -2 \end{bmatrix}$$
$$\rightarrow h = \begin{bmatrix} 88.4 \\ 62.4 \end{bmatrix}$$

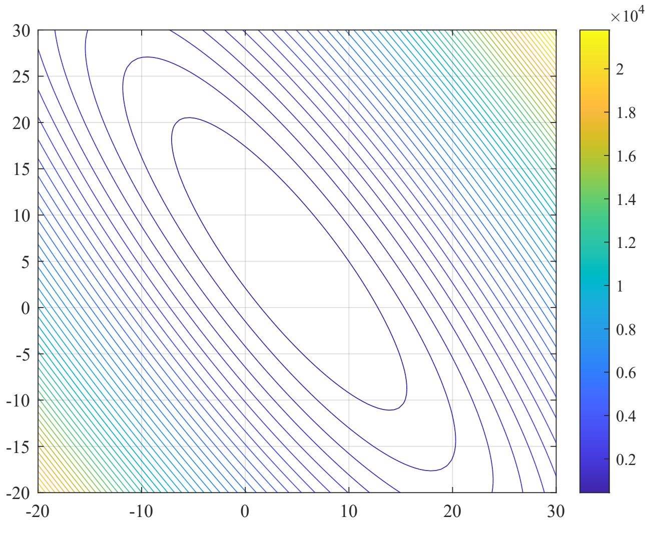
$$R = E[p \cdot p^T] = P_1 \cdot p_1 \cdot p_1^T + P_2 \cdot p_2 \cdot p_2^T + P_3 \cdot p_3 \cdot p_3^T$$
$$= 0.2 \cdot \begin{bmatrix} 2 \\ 4 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 4 \end{bmatrix}^T + 0.7 \cdot \begin{bmatrix} 4 \\ 2 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 2 \end{bmatrix}^T + 0.1 \cdot \begin{bmatrix} -2 \\ -2 \end{bmatrix} \cdot \begin{bmatrix} -2 \\ -2 \end{bmatrix}^T$$
$$\rightarrow R = \begin{bmatrix} 12.4 & 7.6 \\ 7.6 & 6.4 \end{bmatrix}$$

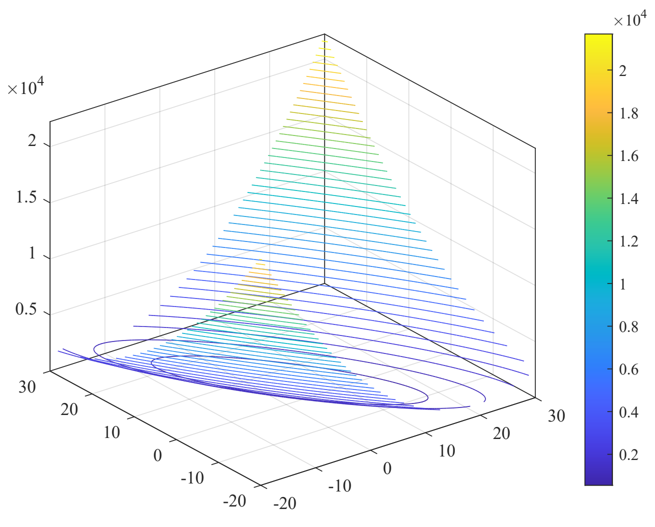
In conclusion the Mean Square Error (MSE) performance index is:

$$F(x) = 676 - 2 \cdot \begin{bmatrix} W_{11} & W_{12} \end{bmatrix} \cdot \begin{bmatrix} 88.4 \\ 62.4 \end{bmatrix} + \begin{bmatrix} W_{11} & W_{12} \end{bmatrix} \cdot \begin{bmatrix} 12.4 & 7.6 \\ 7.6 & 6.4 \end{bmatrix} \cdot \begin{bmatrix} W_{11} \\ W_{12} \end{bmatrix} \rightarrow$$

$$F(x) = 676 - 176.8 \cdot W_{11} - 124.8 \cdot W_{12} + 12.4 \cdot W_{11}^2 + 15.2 \cdot W_{11} \cdot W_{12} + 6.4 \cdot W_{12}^2$$

By inserting this function into Matlab we get the 2D and 3D contour plots of the MSE index as shown in figures 12a and 12b


(a) 2D plot of MSE index


(b) 3D plot of MSE index

(c) Question c

A decision boundary is a hypersurface that separates different classes in a classification problem.
The optimal decision boundary is the one that minimizes a certain transfer function. Here it is a line that can be described by the equation

$$f(x) = W^T \cdot x^*,$$

where $W = \begin{bmatrix} W_{11} & W_{12} \end{bmatrix}$ and x^* the minimum square error
x^* is the strong minimum, a stationary point that is indeed the center of the cycle of Figure 12a.

So, in order to find the optimal decision boundary we follow the mentioned steps:

- $f(x) = W^T \cdot x^*$
- Set f(x) = 0
- Calculate $x^* = R^-1 \cdot h$
- Replace the value in f(x)

By calculating $x^* = R^-1 \cdot h$ we get that $x^* = \begin{bmatrix} 4.2370 \\ 4.7185 \end{bmatrix}$

and the optimal decision boundary is $f(W_{11}, W_{12}) = 4.2370 \cdot W_{11} + 4.7185 \cdot W_{12}$

With the help of Matlab we get the following figure 13

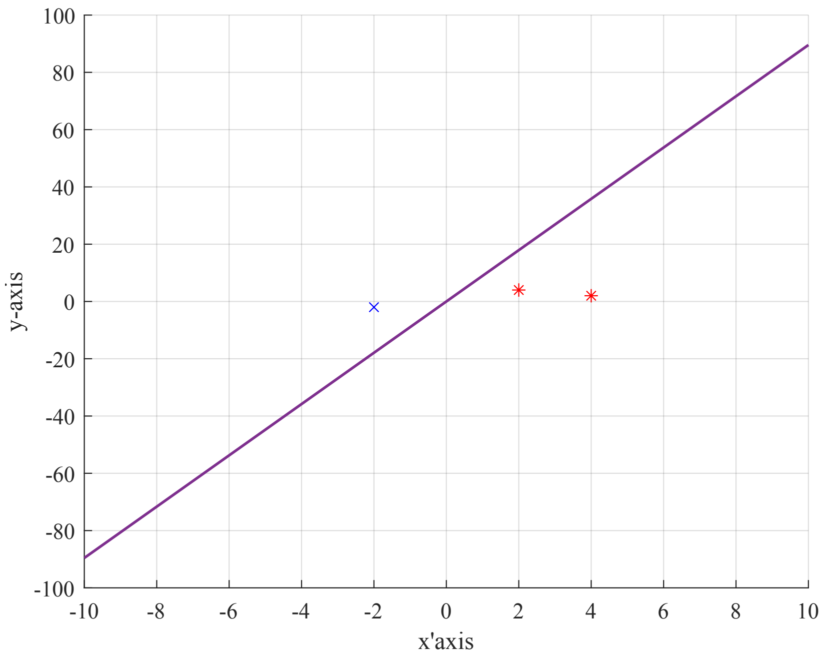


Figure 13: Optimal Decision Boundary line and the patterns

From the figure 13 it is obvious that the optimal decision boundary separates and correctly classifies the patterns. For a binary classification problem,as in our case, all points(*) on one side of the decision boundary are predicted to belong to one class, while points (x) on the other side are predicted to belong to the other class.

(d) Question d

The maximum stable learning rate for the LMS algorithm can be calculated by $lr_{max} = \dfrac{1}{\lambda_{max}}$, where λ_{max} is the largest eigenvalue of the correlation matrix R. With the help of matlab i calculated and found that the eigenvalue matrix of R is

$$eigenvalue = \begin{bmatrix} 1.2293 \\ 17.5707 \end{bmatrix}$$

Thus the largest eigenvalue is $\lambda_{max} = 17.5707$ and in conclusion the maximum learning rate is $lr_{max} = \dfrac{1}{17.5707} \rightarrow \boxed{lr_{max} = 0.0569}$

As we have already mentioned, the learning rate is related to the correlation matrix R. The matrix R depends only on the properties of the input data -$R = E[p \cdot p^T]$ - and not on the output. The learning rate does not depend on the target values but on the properties of the input data.
This means that, changing the target values could potentially affect the convergence of the LMS algorithm and the final solution.However, the maximum stable learning rate itself, as a parameter of the algorithm, would not be directly affected by changes in the target values.

(e) Question e

To perform one iteration of the LMS algorithm we will need a learning rate, the initial weight values and an input. It is given that

- Input data: pattern $p_1 = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$
- $w(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$
- Learning rate: lr = 0.05

For a single-layer ADALINE network we recall the rule

$$W_{k+1} = W_k + 2 \cdot lr \cdot e_k \cdot p_k^T$$

where e_k is the error on that step.

To begin with, we must follow these steps for the iterations. For k=0 we get:

- Step1: Find output a_k
  $a_0 = purelin(W(0) \cdot p_1) = purelin(w(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 4 \end{bmatrix} = 0$
- Step2: Calculate error e_k
  $e_0 = t_0 - a_0 = t_1 - a_0 = 26 - 0 = 26$
- Step3: Calculate weight W_{k+1}
  $W_1 = W_0 + 2 \cdot lr \cdot e_0 \cdot p_0^T = w(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 2 \cdot 0.05 \cdot e_0 \cdot \begin{bmatrix} 2 \\ 4 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix} = \begin{bmatrix} 5.2000 & 10.4000 \end{bmatrix}$

Problem 9

Again, suppose that we have the following two reference patterns and their targets:

$$\left\{p_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, t_1 = [-1]\right\} \quad \left\{p_2 = \begin{bmatrix} -2 \\ 1 \end{bmatrix}, t_2 = [1]\right\}$$

The probabilities of vectors p_1 and p_2 are equiprobable, so that means $P_1 = P_2 = 0.5$.

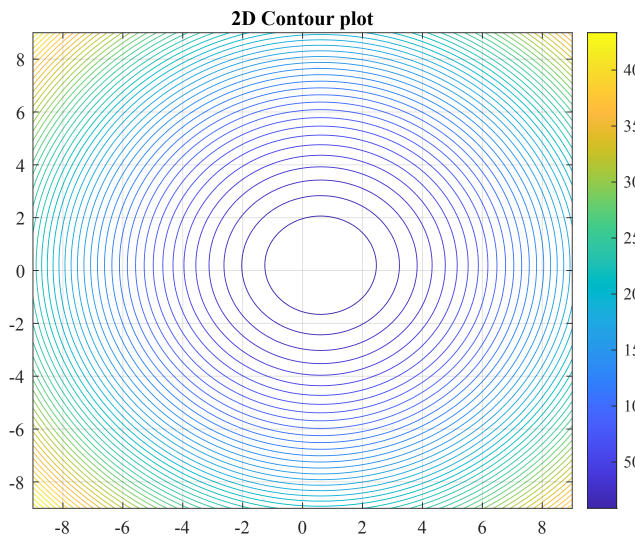
(a) Question a

In order to sketch the contour plot of the Mean Square Error performance, we ought to calculate the arguments of the quadratic function F(x). As we did on Problem8-Question (b), with the help of Matlab we calculate the c,R,h and we get the following results

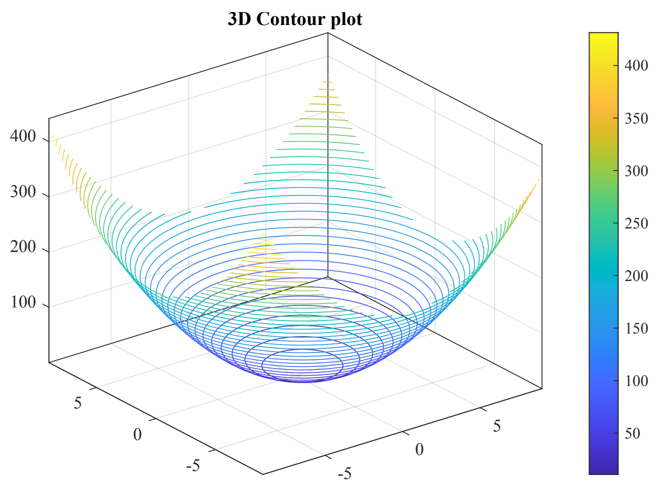
- $R = \begin{bmatrix} 2.5 & 0 \\ 0 & 2.5 \end{bmatrix}$
- $h = \begin{bmatrix} -1.5 \\ -0.5 \end{bmatrix}$
- c = 1

and the MSE function is: $F(x) = 1 - 3 \cdot w_{11} - w_{12} + 2.5 \cdot w_{11}^2 + 2.5 \cdot w_{12}^2$

From the form of the equation we can understand that it is a circle with a center of $x^* = R^-1 \cdot h$.
By the Matlab code that i provided to calculate the F(x), we get the 2D and 3D contour plots of the MSE index as shown in figures 14a and 14b



(a) 2D plot of MSE index



(b) 3D plot of MSE index

(b) Question b

Working with the same method as i did in Problem8-Question (c) but this time with the help of Matlab code, we got the following sketch about the optimal decision boundary.

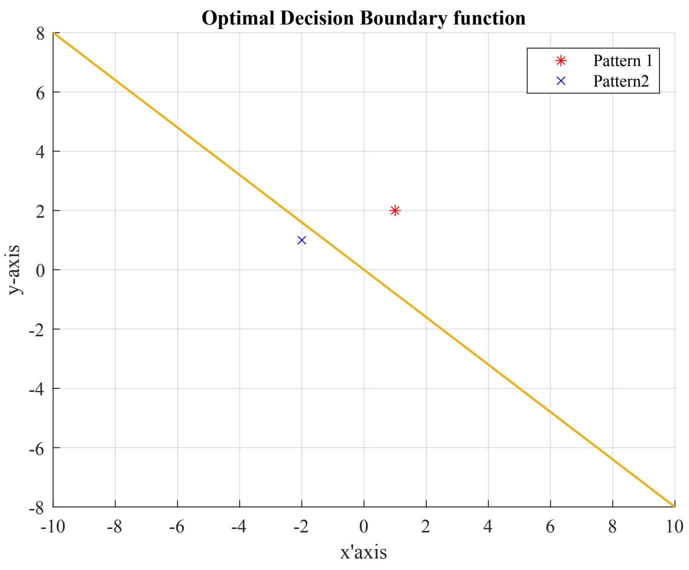


Figure 15: Optimal decision Boundary plot

We can verify that indeed the optimal decision boundary separates and correctly classifies the patterns.

(c) Question c

If we want to sketch the trajectory of the LMS algorithm on our contour plot, we follow the methodology of Problem8- Question(e), but for the number of iterations N = 100.

Here our learning rate is lr = 0.025 and the initial weight is $w(0) = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

Using Matlab, we exported this trajectory of the LMS algorithm to our 2D contour plot of the mean square error performance index.

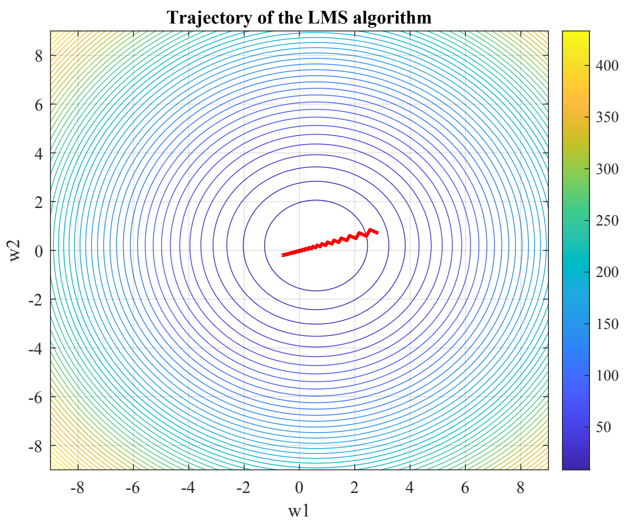


Figure 16: Trajectory of the LMS algorithm

Problem 10

(a) Question A

The patterns that we want to separate are plotted in figure 17.

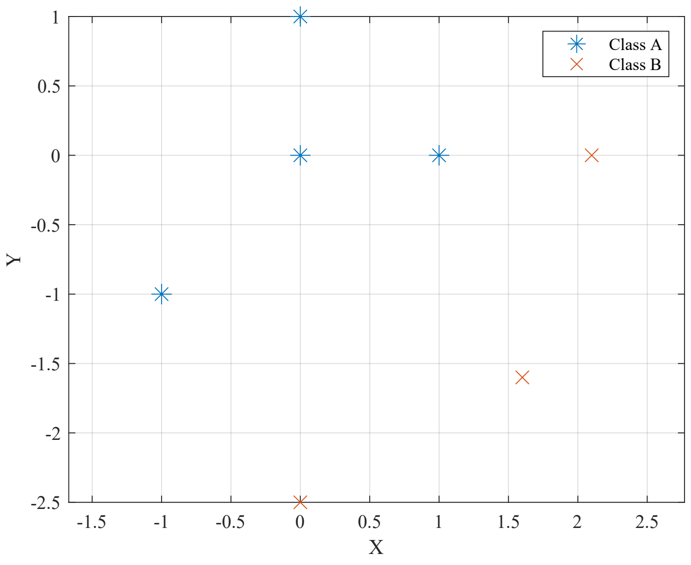


Figure 17: Plot of patterns

We can clearly see that there can be a straight line that can separate the two classes, thus an ADALINE neural network can work in classification for this system.

(b) Question B

The designed ADALINE neural network will be of the following architecture

- Input Layer: Since the patterns are two-dimensional (each pattern has two values), the input layer will have two nodes.
- Output Layer: The output layer will have one node. This is because the task is a binary classification. The output node will use a linear activation function, as is standard in ADALINE networks.
- Weights and Bias: There will be two weights (one for each input node) and one bias. The weights and bias are parameters that the network will learn during the training process.
- Learning Rule: The network will use the LMS learning rule (*Least Mean Square*) to update the weights and bias. This rule minimizes the mean square error between the network's output and the target output.

This architecture described beforehand is shown in figure 18.

(c) Question c

The ADALINE neural network mentioned above was coded in MATLAB. During training, we plotted its weights and bias in order to check their progression. Maximum iteration value was defined in code to be 10^4

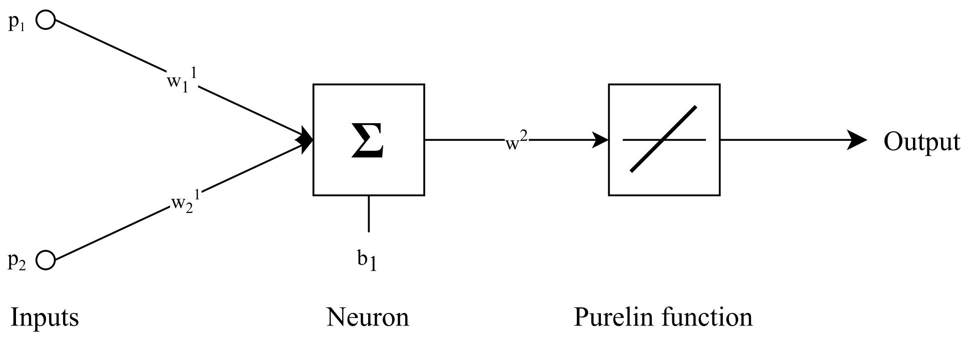


Figure 18: ADALINE neural network architecture

and minimum error to end train and consider the solutions converged is $epsilon$ of the machine, specifically $eps = 2.2204\ 10^{-16}$.
After converging, the final weights and bias are presented in table 1 below.

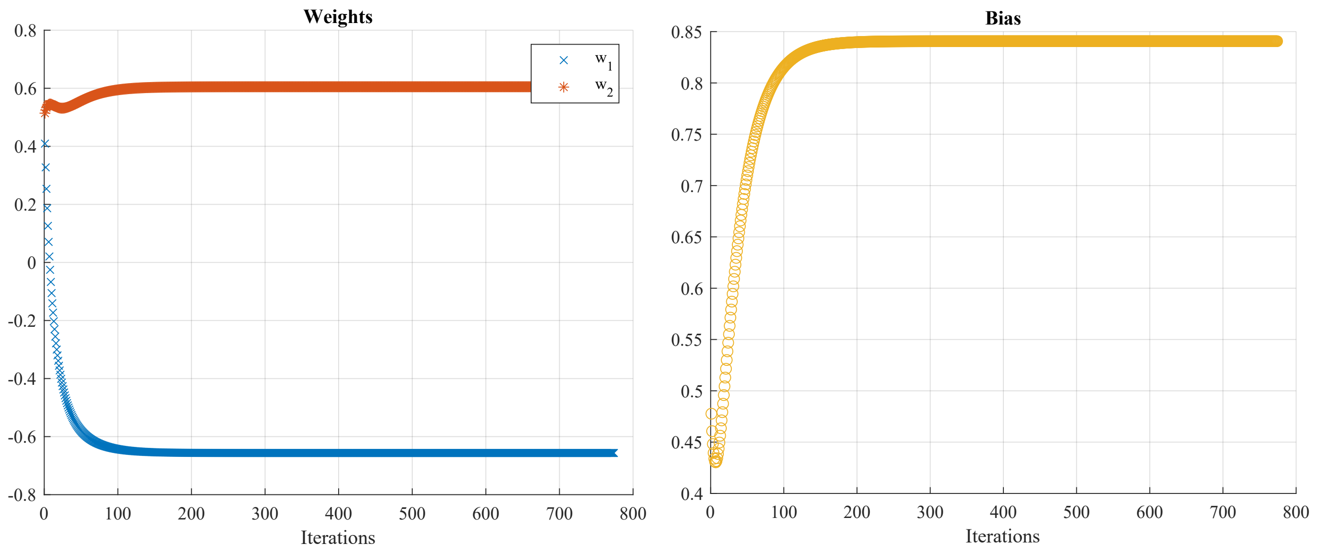


Figure 19: Plots of weights and biases during training

Weight 1	**Weight 2**	**Bias**
-0.6564	0.6052	0.8407

Table 1: Final table of weights and bias

Problem 11

Fuzzy logic is a type of logic that deals with vague, imprecise, or uncertain information. It is based on the concept of fuzzy sets, which are sets that can have any degree of membership between 0 and 1. The value zero is used to represent complete non-membership, the value one is used to represent complete membership, and values in between are used to represent intermediate degrees of membership.This means that an element can be a member of a fuzzy set to some degree, rather than all or nothing.
The uniqueness of fuzzy logic is that fuzzy logic can handle imprecise and uncertain information,which makes it a valuable tool for dealing with real-life problems that are inherently vague or fuzzy.

On this exercise, we are dealing with the linguistic variable *Truth* with a possible membership set:

T = Absolutely false, Very false, False, Fairly true, True, Very true, Absolutely true

Based on that set we may define the membership function of truth as:

True(u) = u   *False(u) = 1-u*

for each $u \in [0, 1]$.

Problem 12

In order to evaluate the expression "`not(A(x) OR B(x))`", we must first take a look at how fuzzy logic differs from binary logic at operation level. In binary logic we have three basic operations: `AND(x,y)`, `OR(x,y)` and `NOT(x)`. But, in fuzzy logic, where a function can have a value in the range of $[0...1]$, things are slightly different. The binary operation `AND(x,y)` is equivalent to `MIN(x,y)` from fuzzy logic, `OR(x,y)` to `MAX(x,y)` and `NOT(x)` to `1-x`.

We need to find the proper x for which the previous expression has the maximum value. First, we calculate the expression and then find the correct x. To achieve this, we need to divide our calculations into ranges.

Starting for $x \leq 2$, "$A(x)$ AND $B(x)$" is equal to "$max\,(A(x), B(x))$" $= 1$. Applying De Morgan's law, we therefore have $max\,(A(x), B(x)) \Rightarrow not\,(A(x)\ or\ B(x)) = 0$. Exactly the same result is obtained with $x \geq 7$.

Things are a bit different in $2 \leq x \leq 7$. The function $A(x)$ starts to fall while $B(x)$ starts to rise. The point at which the two functions cross is important for the definition of the required expression and can be obtained by solving the equation:

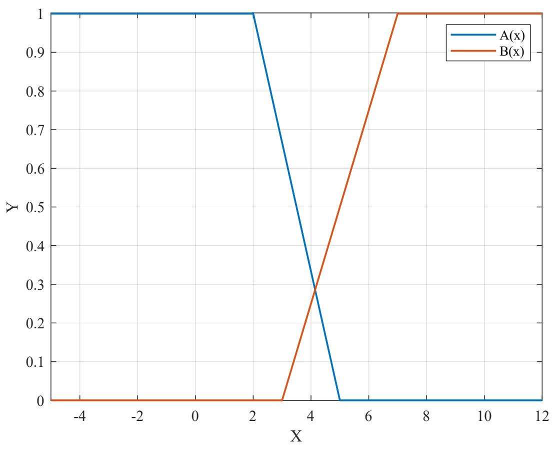


Figure 20: Plot of A(x), B(x)

$$A(x_{crit}) = B(x_{crit}) \Leftrightarrow 1 - \frac{x_{crit} - 2}{3} = \frac{x_{crit} - 3}{4} \Rightarrow x_{crit} = \frac{29}{7}$$

For $2 \leq x \leq \frac{29}{7}$, $max\,(A(x), B(x)) = 1 - \dfrac{x-2}{3} = A(x)$, because in this region $A(x)$ lies above $B(x)$. Thus, $not\,(A(x)\ or\ B(x)) = \dfrac{x-2}{3}$.

Using the same logic, we find out that for $\frac{29}{7} \leq x \leq 7$, $not\,(A(x)\ or\ B(x)) = 1 - \dfrac{x-3}{4}$.

Therefore, the expression $f(x) = not\,(A(x)\ or\ B(x))$ is summarized below:

$$f(x) = \begin{cases} 0 & x \leq 2, \\ \dfrac{x-2}{3} & 2 \leq x \leq \frac{29}{7}, \\ 1 - \dfrac{x-3}{4} & \frac{29}{7} \leq x \leq 7, \\ 0 & x \geq 7 \end{cases} \tag{7}$$

By plotting this function in figure 21, we can clearly see that the maximum occurs at $x = x_{crit} = \dfrac{29}{7}$ and its value is $\dfrac{10}{14}$ or 0.715465.

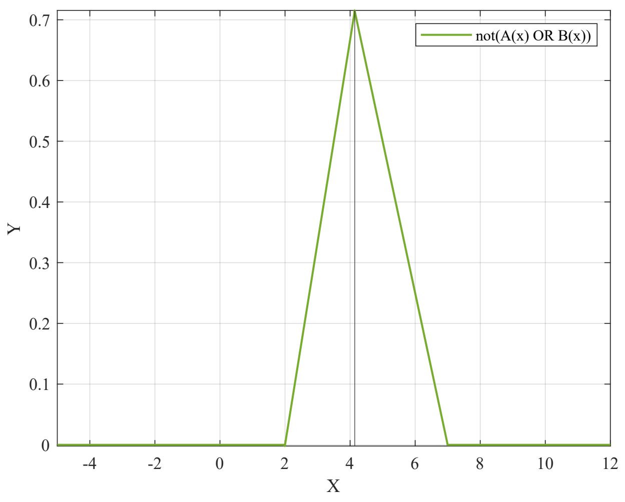


Figure 21: Expression's plot