ME 466 Introduction to AI Fall 2021
Programming Assignment 3 Submitted on: 30 December 2021

Name: Korkut Emre Arslantürk

Student ID: 250206039

Grade:

1	
2	
3	
Σ	

I hereby declare that the paper I am submitting under this cover is product of my own efforts only. Even if I worked on some of the problems together with my classmates, I prepared this paper on my own, without looking at any other classmate's paper. I am knowledgeable about everything that is written under this cover, and I am prepared to explain any scientific/technical content written here if a short oral examination about this paper is conducted by the instructor. I am aware of the serious consequences of cheating.

Signature:

[illegible]

TABLE OF CONTENTS

INTRODUCTION

We have 3 different problems in this assignment.

In the first part, we study on the implementation of logic gates with the McCulloch-Pitts model. McCulloch-Pitt's neuron weights can be ±1, while inputs can be one or zero. We worked on the logic function of [(x1 AND x2) OR x3] for every possible set of inputs. Output is calculated according to the General Neurode with Bias. We specified a threshold to make a decision. When the input is 1 and stimulating, it means that we sum up the total with 1. Also, 1 is subtracted from the total, when the input is 1 and inhibiting. The result is obtained, by doing that for all entries. Finally, we decide by comparing the result with the threshold. For instance, the output equals 1 if the result is higher than 1. Also, we get zero as an output if the result is fewer than the threshold.

In the second part, the activation function of the neuron and the neuron is given with three inputs. Also, the error is defined. The gradient descent method was employed with three inputs and an activation function to derive weight update equations. Subsequently, that neuron is trained to estimate function in the first problem. In order to do that, firstly, the weights are initialized randomly. Then, weight update equations are used, which are determined at the previous stage. In the last step, iteration is continued until getting fewer error than machine epsilon. The resulting values of weights are reported and the impact of the learning rate is observed on convergence. Finally, inputs and the discriminant are shown in 3D space.

In the third part, an artificial neural network is implemented in order to derive the daily activities of a person by smartphone's sensors. The dataset includes six activities and it was recorded by thirty volunteers. The feature matrix consists of 10299 columns and 561 rows. We implement a multilayer perception with the given input and output neurons. The hyperbolic tangent function is employed as an activation function. The neuron that corresponds to the performed activity output may equal 1 in the output layer, as the others output minus 1. The network is trained. Then, the performance of the network is calculated according to the different amounts of hidden layer neurons and the learning rate. 10-fold cross-validation and Leave-one-participant-out cross-validation schemes are used. Finally, the features number is decreased to 10 by using PCA and the previous part is repeated to make a performance comparison.

BODY OF THE REPORT

Question 1

According to McCulloch Pitts, neuron weights $w_i=\pm1$, while the inputs(x_i) are 1 or, and the unit step function is used as an activation function. In that question, the aim is to create a McCulloch-Pitts neuron network that calculates the logic function for each possible input set.

Logic Function = (x1 AND x2) OR x3 = $(X_1 \cap X_2) \cup X_3$

Thus, first I employed two neurons. First for AND,

That information is known:

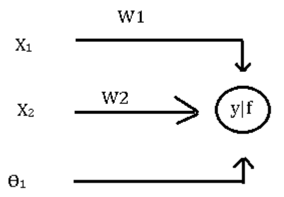
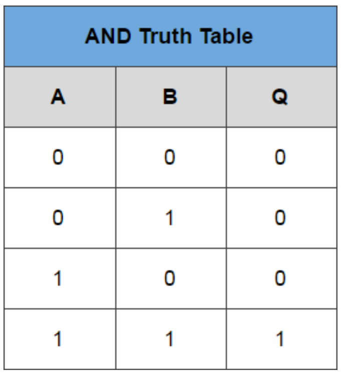


Figure 1: Diagram of Equation

$$f(g(v)) = \begin{cases} 1, & g(v) \geq 0 \\ 0, & g(v) < 0 \end{cases}$$

$$Y = b * w_b + \sum_{i=1}^{n} x_i * w_i$$



AND Truth Table		
A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

Figure 2: AND Gate Truth Table

If we accept that values, $W_1=1$, $W_2=1$. Also let's say $\Theta_1=1$ to try:

Row 1: $0 \text{ x } W_1 + 0 \text{ x } W_2 + \Theta_1 = (0\text{x}1) + (0\text{x}1) + 1 = 1 \Rightarrow g(v) \geq 0$.

Thus we can say it is not fit. Θ_1 should be equal -1.

In that case, $0 \text{ x } W_1 + 0 \text{ x } W_2 + \Theta_1 = -1 \Rightarrow g(v)<0$. So, Y=0. As can be seen in the table Q=0.

Row 2: $0 \text{ x } W_1 + 1 \text{ x } W_2 + \Theta_1 = (0\text{x}1)+(1\text{x}1)-1=0 \Rightarrow g(v) \geq 0$.

Thus we can say it is not fit. Θ_1 should be equal -2.

In that case, $0 \text{ x } W_1 + 0 \text{ x } W_2 + \Theta_1=-1 \Rightarrow g(v)<0$. So, Y=0. As can be seen in the table Q=0.

Row 3: $1 \text{ x } W_1 + 0 \text{ x } W_2 + \Theta_1 = (1\text{x}1)+(0\text{x}1)-2=-1 \Rightarrow g(v)< 0$.

Thus we can say it is fit as can be seen in the table.

Row 4: $1 \text{ x } W_1 + 1 \text{ x } W_2 + \Theta_1 = (1\text{x}1)+(1\text{x}1)-2= 0 \Rightarrow g(v) \geq 0$.

Thus we can say it is fit. However to get more accuracy, lets say $\Theta_1=-1.5$.

In that case, $0 \times W_1 + 0 \times W_2 + \Theta_1 = 0$ ➔ $g(v)<0$. So, Y=0. As can be seen in the table Q=1.

For second part, output of AND gate is used as an input. Also, let’s accept W1=1,W2=1 and $\Theta_2=1$.

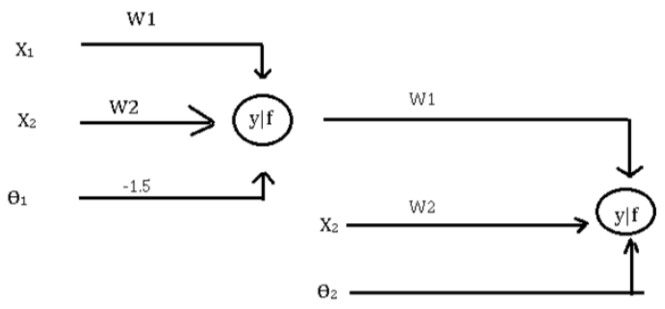


Figure 3: Diagram of the general Equation

OR Truth Table		
$Q=X_1$	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	1

Figure 4: Truth table of OR Gate

Row 1: $0 \times W_1 + 0 \times W_2 + \Theta_1 = (0x1) + (0x1) + 1 = 1$ ➔ $g(v) \geq 0$.

Thus we can say it is not fit. Θ_2 should be equal -1.

In that case, $0 \times W_1 + 0 \times W_2 + \Theta_1 = -1$ ➔ $g(v)<0$. So, Y=0. As can be seen in the table Q=0.

Row 2: $0 \times W_1 + 1 \times W_2 + \Theta_1 = (0x1)+(1x1)-1=0$ ➔ $g(v) \geq 0$.

Thus we can say it is fit as can be seen in the table.

Row 3: $1 \times W_1 + 0 \times W_2 + \Theta_1 = (1x1)+(0x1)-2=-1$ ➔ $g(v) < 0$.

Thus we can say it is fit as can be seen in the table.

Row 4: $1 \times W_1 + 1 \times W_2 + \Theta_1 = (1x1)+(1x1)-2= 0$ ➔ $g(v) \geq 0$.

Although we can say it is fit, theta is too high. To get more accuracy, let’s say $\Theta_2=0.5$.

Finally, design of given logic $(X_1 \cap X_2) \cup X_3$ is observed. Outputs are obtained according to the specified theta values ($\Theta_1=1.5$, $\Theta_2=0.5$).

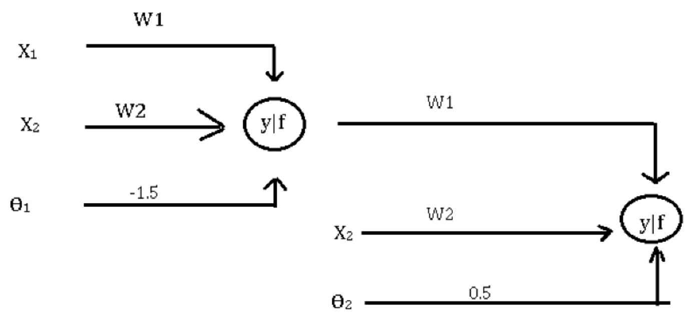


Figure 5: Final Design of given Logic.

X_1	X_2	X_3	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Figure 6: Truth table of given Logic

Question 2

The aim of that question is to train the given neuron in order to calculate the function in question 1. Also, the gradient descent method should be used to derive the weight update equations for that neuron.

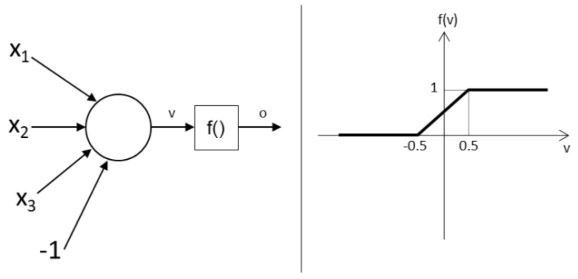


Figure 7: Given figure in the Question

Firstly, neuron function is defined according to the given figure in the question. In that function, there are 3 inputs and Bayes variable. Output is observed according to the right side of the figure.

```
function f=neuron(x1,x2,x3,w)

a = x1*w(1)+x2*w(2)+x3*w(3)-1*w(4);

if a<-0.5
f=0;
elseif -0.5<a && a<0.5
f=a+0.5;
elseif 0.5<a
```

```
f=1;
end
end
```

Gradient descent is an effective strategy if the parameters must be found using an optimization methodology. The gradient descent method is employed according to the request of the problem in order to observe the weight update equations. Therefore, the derivative function of neurons is required. The derivative function is defined according to the slope in the graph given in the question.

```
function f=der_neuron(x1,x2,x3,w)
a = x1*w(1)+x2*w(2)+x3*w(3)-1*w(4);
if -0.5<a && a<0.5
    f=1;
else
    f=0;
end

end
```

In the main, first of all, weights were initialized randomly. To compare how it changed after that, ‘ww’ variable was defined. The data were written according to the truth table. The trE variable was initially defined as 1 to start the while loop. The variable 'e' equalized the machine epsilon in the question. The count variable is variable, which indicates how many attempts the desired error has been reached. While loop stops when the error rate(trE) is lower than the desired rate (machine epsilon). Also, the eta variable is defined below. Learning rate affects convergence. When it is very high, it cannot properly reach the weight values. It also needs to rotate a lot when it is too small. The optimum rate effect value is 0.6 in that program, it takes around 19 try to reach the actual value. For instance, the count variable increases to around 2791 when the rate effect variable is equal to 0.05.

```
w = rand(1,4);
ww=w;
datas = [0 0 0 0; 0 0 1 1; 0 1 0 0; 0 1 1 1; 1 0 0 0; 1 0 1 1; 1 1 0 1; 1 1 1 1];
eta=0.06;
trE=1;
e=eps;
count=0;
```

One is subtracted from the trE variable at the end of the for loop because the trE variable equalized to 1 in the while loop. For loop is returned 8 times because the data has 8 rows.’x1,x2,x3 inputs, and the target value are taken from the data. Then the neuron function is called with the variables x1,x2,x3,w and the value is calculated. Then, the error is calculated using the expected and calculated value according to the formula given in the question. In order to avoid the possibility of exiting the loop in the event of an error failure, a total of 8 errors are found and then averaged. Delta is calculated by multiplying the difference between the target value and the received value by the derivative function of the neuron. Then, according to the formula, a new weight value is obtained and also the count value is calculated.

```
while trE>e
    trE=1;
    for i = 1:8
        x1 = datas(i,1);
        x2 = datas(i,2);
        x3 = datas(i,3);
        t = datas(i,4);
        o = neuron(x1,x2,x3,w);
        Err=((t-o)^2)/2;
        trE = trE+Err;
        delta = (t-o)*der_neuron(x1,x2,x3,w);
        w = w+(eta*delta*[x1 x2 x3 -1]);
    end
    trE=(trE-1)/8;
    count=count+1;
end
```

Subsequently, empty arrays are determined. After that, classes are separated. Then, the values of each axis have recorded the arrays according to each statement. There are 3 inputs, so graph is plotted in 3D space.

```
x11=[]; x22=[]; y11=[]; y22=[]; z11=[]; z22=[];
for i=1:8
    x(i)=[datas(i,1)];
    y(i)=[datas(i,2)];
    z(i)=[datas(i,3)];
    o = neuron(x(i),y(i),z(i),w);
    if o<0.5
        x11=[x11 datas(i,1)];
        y11=[y11 datas(i,2)];
        z11=[z11 datas(i,3)];
    elseif o>0.5
        x22=[x22 datas(i,1)];
        y22=[y22 datas(i,2)];
        z22=[z22 datas(i,3)];
    end
end
plot3(x11,y11,z11,'o')
title("The Class Distinction ")
hold on
plot3(x22,y22,z22,'x')
xlabel('X')
ylabel('Y')
Zlabel('Z')
```

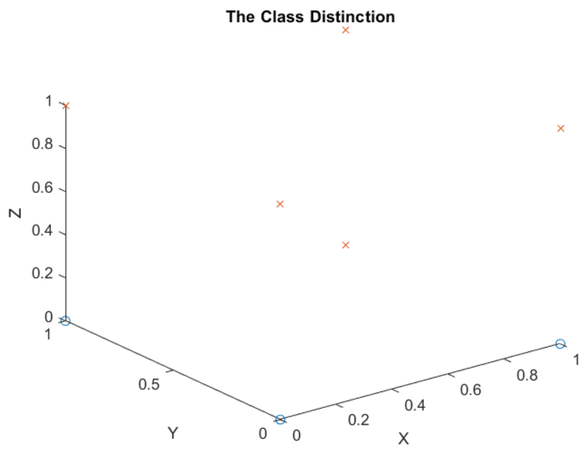


Figure 8: Output of the Algorithm

As can be seen in the figure, the class distinction has been made according to the input values. If this problem is to be interpreted as a linear discriminant problem separating two sets of points, it can be evaluated separately for both axes for all classes.

Question 3

In that problem, the implementation of an artificial neural network is done. The problem focuses on the recognition of daily activities using the sensors on a smartphone. The data consists of 30 people and 6 different activities. The features matrix consists of 10299 columns and 561 rows. The variable classes have the activity labels (classes) for each column, whereas the variable participants contain the participant who did the activity.

B)

i)

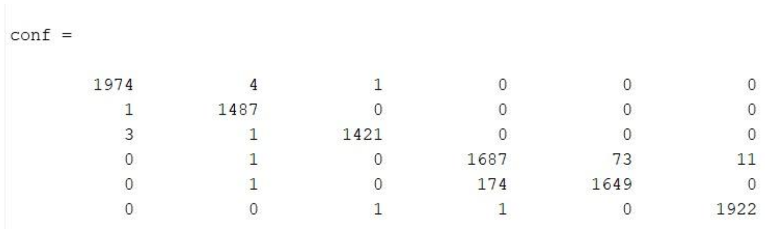
10-fold cross-validation is employed. Thus, data is splited randomly into 10 sets, 9 sets for training and 1 set for testing. Then, performance is calculated. Network is trained 10 times.

First data set is loaded and required variables are defined.

```
load uci_har.mat

Targets_ref=-1*ones(6,10299);

for k=1:10299
    Targets_ref(classes(k),k)=1;
end
c = cvpartition(size(features,2),'KFold',10);
num_input=561;
num_hidden=900;
num_output=6;
eta=0.001;
relE=inf;
prevE=inf;
conf=zeros(6);
```

After that, for loop is employed and the size of the train and test datasets are determined. Subsequently, training, test arrays are built. While the loop continues to process if the error is bigger than 0.15. Inside of while loop, there is a for aloop. For loop continues during the size of num_training.

```
for j = 1:c.NumTestSets

    Wih=0.02*(rand(num_input+1,num_hidden)-0.5);
    Who=0.02*(rand(num_hidden+1,num_output)-0.5);

    ind=randperm(10299);
    Inputs=f(:,ind);
    Targets=Targets_ref(:,ind);
    num_training=c.TrainSize(1,j);
    num_test=c.TestSize(1,j);
    Training=Inputs(:,1:num_training);
    TrainingTargets=Targets(:,1:num_training);

    Test=Inputs(:,num_training+1:end);
    TestTargets=Targets(:,num_training+1:end);
    epoch=0;
    error = 1;
    trErrorhistory=[];
    while (error>0.15)
        trE=0;
        for i=1:num_training
            v=Wih'*[Training(:,i);-1];
            o=tanh(v);

            vv=Who'*[o;-1];
            oo=tanh(vv);

            trE=trE+sum((oo-TrainingTargets(:,i)).^2);

            deltao=(oo-TrainingTargets(:,i)).*(1-oo.^2);
            Who=Who+(-eta*[o;-1]*deltao');
            deltah=(Who*deltao).*(1-[o;-1].^2);
            Wih=Wih+(-eta*[Training(:,i);-1]*deltah(1:end-1)');
        end
        epoch=epoch+1;

        trE=trE/num_training;
        error = trE;
        trErrorhistory=[trErrorhistory,trE];

        plot(trErrorhistory)
        hold on;

    end
    est=tanh(Who'*[tanh(Wih'*[Test;-1*ones(1,num_test)]);-1*ones(1,num_test)]);

    for k=1:num_test
        [~,I]=max(est(:,k));
        [~,J]=max(TestTargets(:,k));
        conf(J,I)=conf(J,I)+1;
    end
end
```



```
conf =

        1974           4           1           0           0           0
           1        1487           0           0           0           0
           3           1        1421           0           0           0
           0           1           0        1687          73          11
           0           1           0         174        1649           0
           0           0           1           1           0        1922
```

Figure 9: Output of 10-fold cross-validation

The covariance matrix computation process's aim is to check how the variables of the input data set are varying from the average. It means, to understand is there any relation between them or not since variables might be highly correlated to the point that they include redundant information.

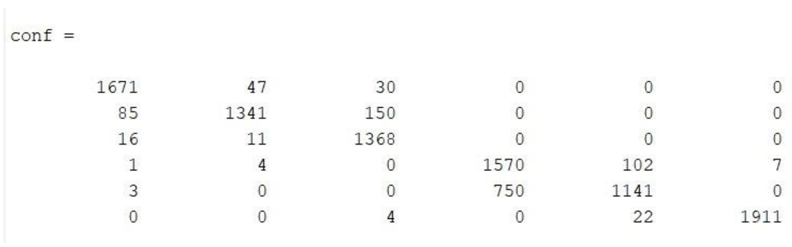
ii) The model is created and then tested. Using Leave-one-participant-out cross-validation, key features are extracted. Training and testing models can be applied after the feature selection. The network is trained 30 times with the data of 29 people and tested with the data of the remaining 1 participant, with each test using a different participant's data.

First of all, data is loaded and required variables are determined. Consequently, number of input layer, hidden layer, output layer and eta are specified.

```
load uci_har.mat

Targets_ref=-1*ones(6,10299);

for k=1:10299
    Targets_ref(classes(k),k)=1;
end

num_input=561;
num_hidden=800;
num_output=6;
eta=0.001;
for i= 1:30
    ind(i,:)=~(participants == i);
end

f=features;
relE=inf;
prevE=inf;
trErrorhistory=[];

conf=zeros(6);
```

After that, for loop is employed and size of train and test datasets are determined. Subsequently, training, test arrays are builded. While loop continues to process if the error is bigger than 0.15. Inside of while loop, there is an for loop. For loop continues during the size of num_training.

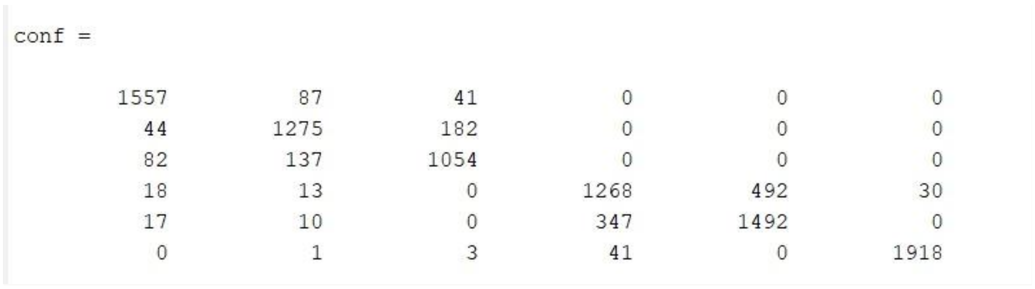
```
for j = 1:30

    Wih=0.02*(rand(num_input+1,num_hidden)-0.5);
    Who=0.02*(rand(num_hidden+1,num_output)-0.5);


    Inputs=f;
    Targets=Targets_ref;
    num_training=sum(ind(j,:) == 1);
    num_test=10299-num_training;
    Training=Inputs(:,ind(j,:)==1);
    TrainingTargets=Targets(:,ind(j,:)==1);

    Test=Inputs(:,ind(j,:)==0);
    TestTargets=Targets(:,ind(j,:)==0);
    epoch=0;
    error = 1;
    while (error>0.15)
        trE=0;
        for i=1:num_training
            v=Wih'*[Training(:,i);-1];
            o=tanh(v);

            vv=Who'*[o;-1];
            oo=tanh(vv);

            trE=trE+sum((oo-TrainingTargets(:,i)).^2);

            deltao=(oo-TrainingTargets(:,i)).*(1-oo.^2);
            Who=Who+(-eta*[o;-1]*deltao');
            deltah=(Who*deltao).*(1-[o;-1].^2);
            Wih=Wih+(-eta*[Training(:,i);-1]*deltah(1:end-1)');
        end
        epoch=epoch+1;

        trE=trE/num_training;
        error = trE;
        trErrorhistory=[trErrorhistory,trE];

        plot(trErrorhistory)
        hold on;

    end

    est=tanh(Who'*[tanh(Wih'*[Test;-1*ones(1,num_test)]);-1*ones(1,num_test)]);



    for k=1:num_test
        [~,I]=max(est(:,k));
        [~,J]=max(TestTargets(:,k));
        conf(J,I)=conf(J,I)+1;
    end
end
```



```
conf =

        1671          47          30           0           0           0
          85        1341         150           0           0           0
          16          11        1368           0           0           0
           1           4           0        1570         102           7
           3           0           0         750        1141           0
           0           0           4           0          22        1911
```

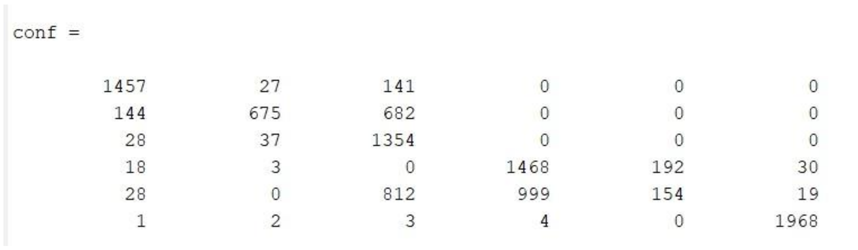
Figure 10: Output of Leave-one-participant-out cross validation

c) In that part, Principle components analysis is employed to declined features number to 10. Then, same process is repeated. Thus, same algorithm is used like previous part that’s why code is not explained again.



```
conf =

        1557          87          41           0           0           0
          44        1275         182           0           0           0
          82         137        1054           0           0           0
          18          13           0        1268         492          30
          17          10           0         347        1492           0
           0           1           3          41           0        1918
```

Figure 11: 10-fold cross validation with 10 features



```
conf =

        1457          27         141           0           0           0
         144         675         682           0           0           0
          28          37        1354           0           0           0
          18           3           0        1468         192          30
          28           0         812         999         154          19
           1           2           3           4           0        1968
```

Figure 12: Output of Leave-one-participant-out cross validation with 10 features

Covariance matrices are acquired by reducing number of feature and also input values. It can be said that, closer results may be obtained when the number of eta rises and neuron’s number declined. The simplicity of use improves.

CONCLUSION

This assignment includes three problems.

The McCulloch-Pitts model is used to investigate the implementation of logic gates in the first section. For every potential set of inputs, we worked on the logic function [(x1 AND x2) OR x3]. The General Neurode with Bias is used to calculate the output. Finally, we make our decision by comparing the outcome to the threshold. The optimal theta values were found.

To develop weight update equations, the gradient descent approach was used with three inputs and an activation function. Moreover, the neuron is trained to calculate function in the first problem. To do this, the weights are first randomly initialized. The weight update equations, which were computed at the previous stage, are employed. Iteration is repeated until there are less errors than machine epsilon. As a result, the discriminant and inputs are displayed in 3D space. The resulting weight values are presented, as well as the impact of the learning rate on convergence.

In the third section, an artificial neural network is implemented to extract a person's everyday activities using smartphone sensors. With the input and output neurons provided, I created a multilayer perception. As an activation function, the hyperbolic tangent function is used. In the output layer, the neuron that corresponds to the completed activity output may equal 1, while the others minus 1. After train of network, the network's performance is calculated based on the number of hidden layer neurons and the learning rate. The cross-validation strategies employed are 10-fold cross-validation and Leave-one-participant-out cross-validation. Finally, using PCA, the number of features is reduced to ten, and the preceding portion is performed to compare performance.