# Lab 2 Explanation: **Neural Networks**

**PUSL3123 AI and Machine Learning**


**Neamah Al-Naffakh**

School of Engineering, Computing and Mathematics

**Neamah.al-naffakh@plymouth.ac.uk**


UNIVERSITY OF PLYMOUTH
School of Engineering, Computing and Mathematics

## Today's Topics

## Neural Networks

Lesson learning outcomes: By the end of today's lesson, you would be able to:

Implement ANN using MATLAB

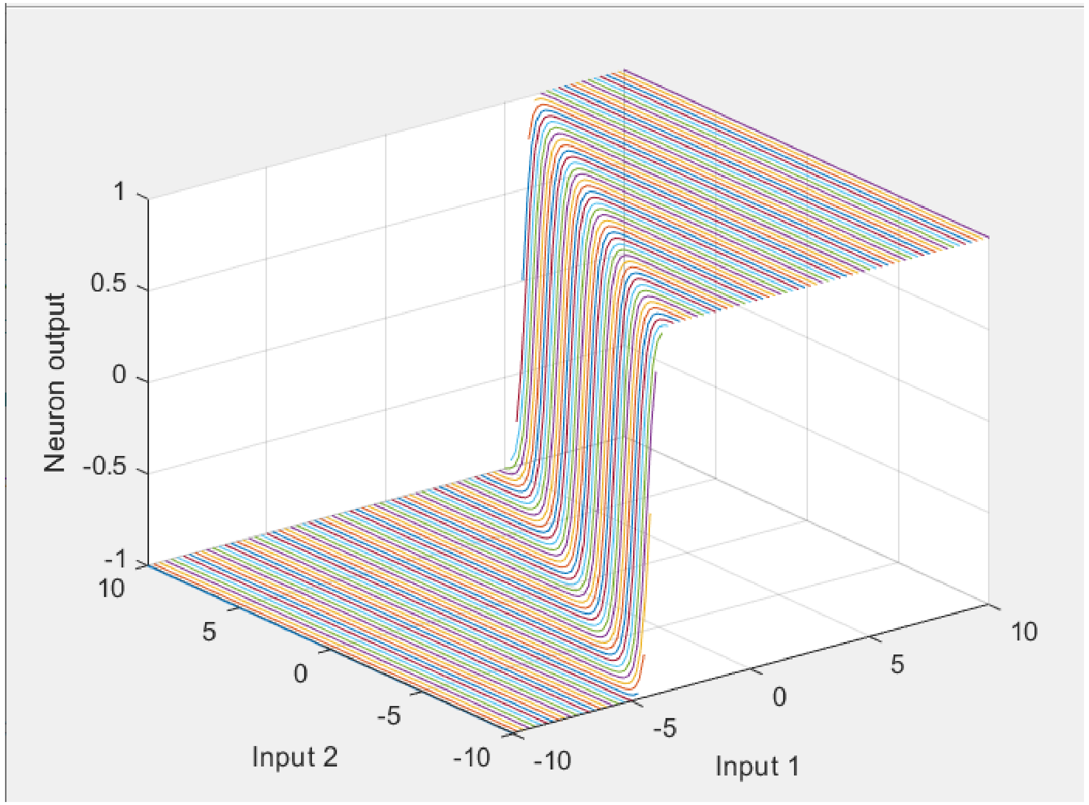
UNIVERSITY OF PLYMOUTH
School of Engineering, Computing and Mathematics

# Calculate the output of a simple neuron

% Define neuron parameters (i.e., Weights, bias, Activation function)
% Define input vector
% Calculate neuron output
% Plot neuron output over the range of inputs

```
close all, clear all, clc;
% Neuron weights
w = [4 -2];

% Neuron bias
b = -3;

% Activation function
func = 'tansig'; % Explore other Functions such as 'purelin' 'hardlim' or 'logsig'

% Define input vector
v = [2 3];

% Calculate neuron output
activation_potential = v*w'+b;
neuron_output = feval(func, activation_potential)
```

```
% Plot neuron output over the range of inputs
[p1,p2] = meshgrid(-10:.25:10);
z = feval(func, [p1(:) p2(:)]*w'+b );
z = reshape(z,length(p1),length(p2));
plot3(p1,p2,z)
grid on
xlabel('Input 1')
ylabel('Input 2')
zlabel('Neuron output')
```



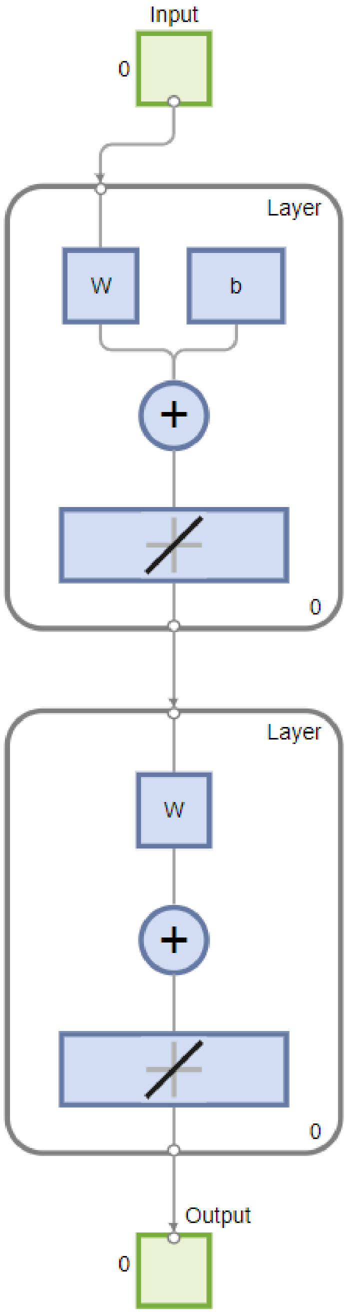
# Create and view custom neural networks

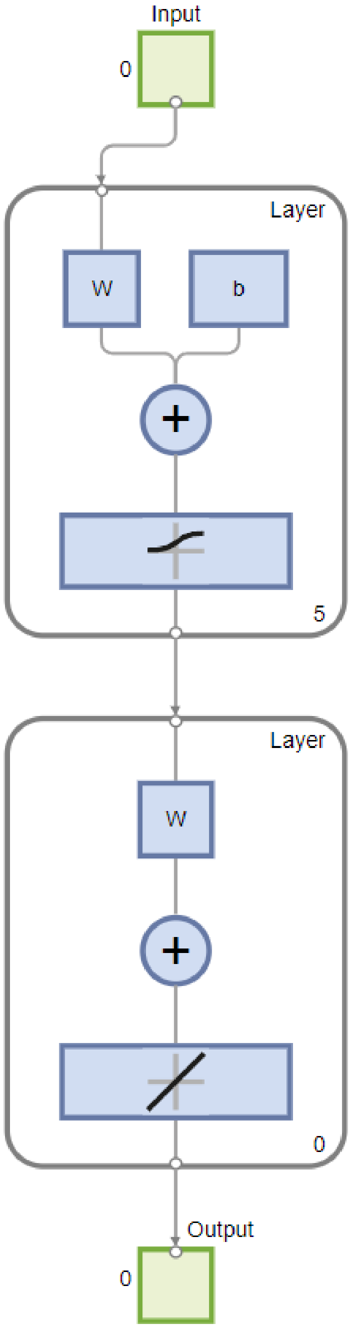
% Define one sample: inputs and outputs
% Define and custom network
% Define topology and transfer function
% Configure network
% Train net and calculate neuron output

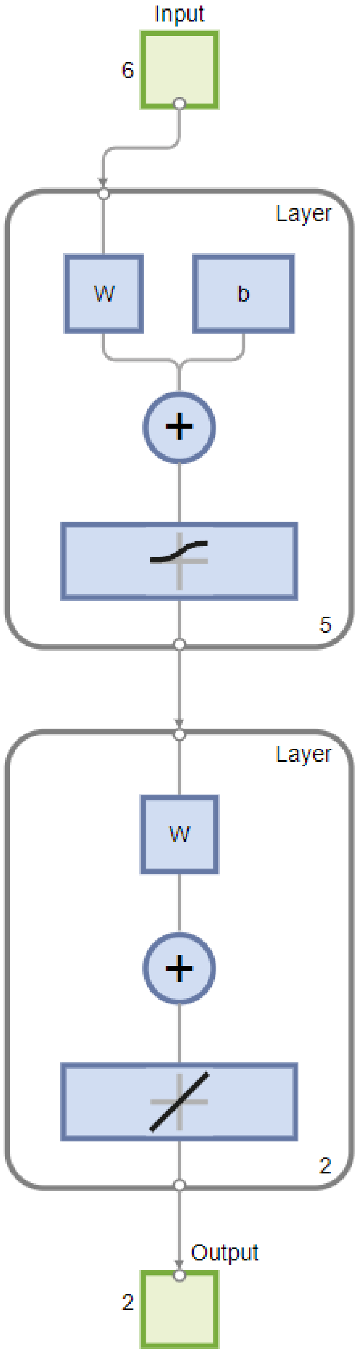
```
close all, clear all, clc;
% Define one sample: inputs and outputs
inputs = [1:6]'; % input vector (6-dimensional pattern)
outputs = [1 2]'; % corresponding target output vector

% Define and custom network
net = network( ...
1, ... % numInputs, number of inputs,
2, ... % numLayers, number of layers
[1; 0], ... % biasConnect, numLayers-by-1 Boolean vector,
[1; 0], ... % inputConnect, numLayers-by-numInputs Boolean
matrix,
[0 0; 1 0], ... % layerConnect, numLayers-by-numLayers
Boolean matrix
[0 1] ... % outputConnect, 1-by-numLayers Boolean vector
);
```

```
% View network structure
view(net);

% Define topology and transfer function
net.layers{1}.size = 5; % number of hidden layer neurons
net.layers{1}.transferFcn = 'logsig'; % hidden layer transfer
function
view(net);

% Configure network
net = configure(net,inputs,outputs);
view(net);

% Train net and calculate neuron output
initial_output = net(inputs) % initial network response without
training

% network training
net.trainFcn = 'trainlm';
net.performFcn = 'mse'; % Mean Square Error
net = train(net,inputs,outputs);
% network response after training
final_output = net(inputs)
```

# NN - Output


Input
0
Layer
W
b
0
Layer
W
0
Output
0

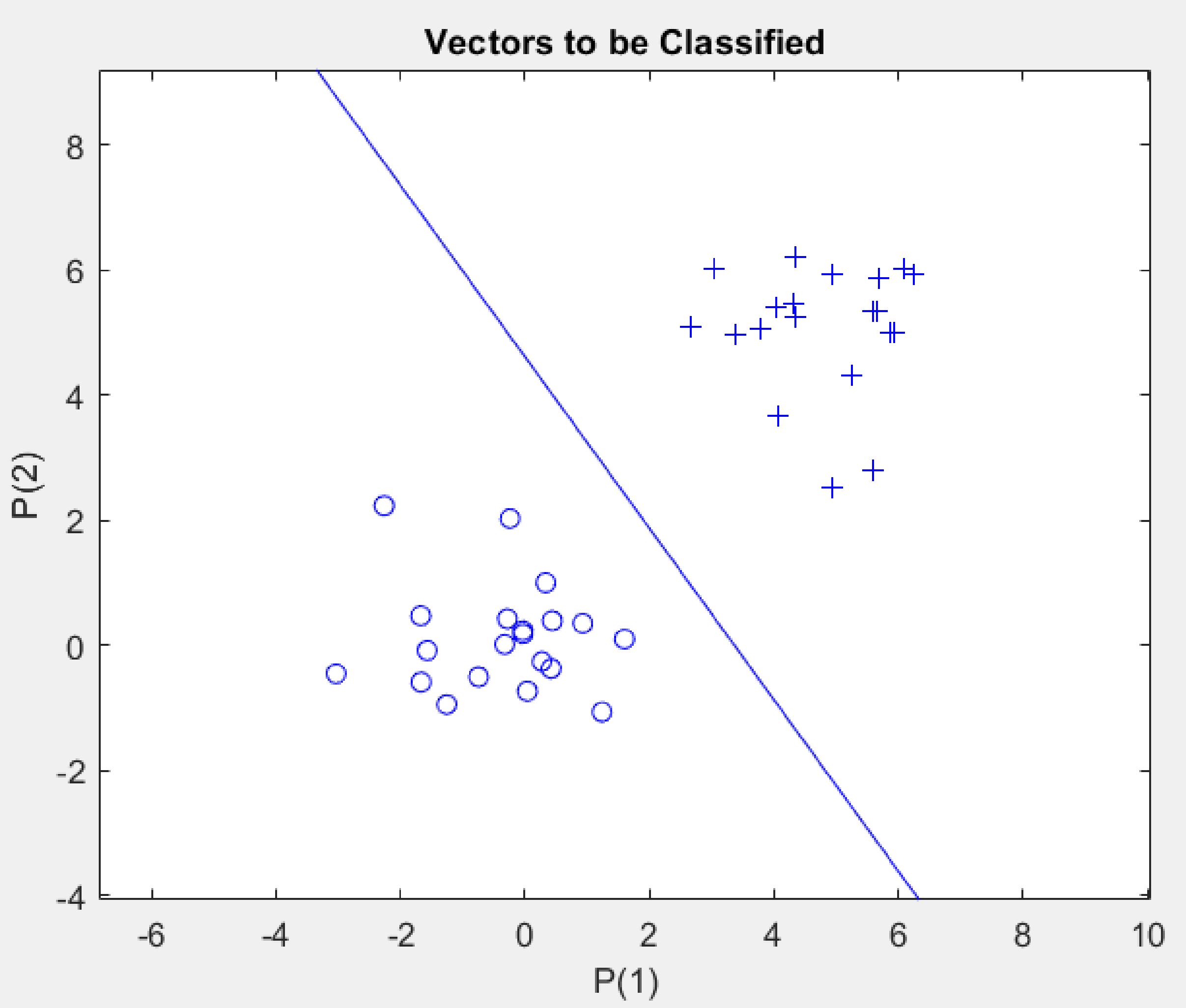

Input
0
Layer
W
b
5
Layer
W
0
Output
0


Input
6
Layer
W
b
5
Layer
W
2
Output
2

# Classification of linearly separable data with a perceptron

% Define input and output data
% Create and train perceptron
% Plot decision boundary

```
close all, clear all, clc
N = 20; % number of samples of each class

% define inputs and outputs
offset = 5;
x = [randn(2,N) randn(2,N)+offset];     % inputs
y = [zeros(1,N) ones(1,N)];             % outputs

% (Plot perceptron input/target vectors)
figure(1)
plotpv(x,y);

% Create and train perceptron
net = perceptron;
net = train(net,x,y);
view(net);

% Plot decision boundary
figure(1)
plotpc(net.IW{1},net.b{1});
```

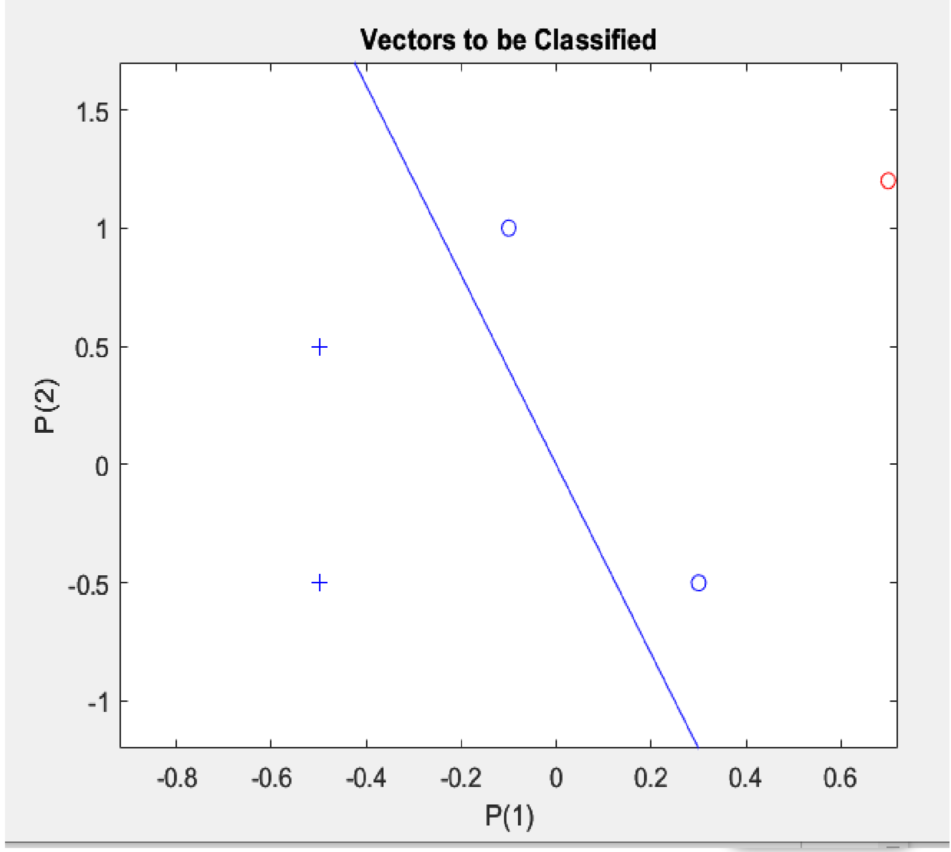


# Classification of linearly separable data with a perceptron

% Classification with a Two-Input Perceptron
% Define input and output data
% Create and train perceptron
% Plot decision boundary
% Plot new input

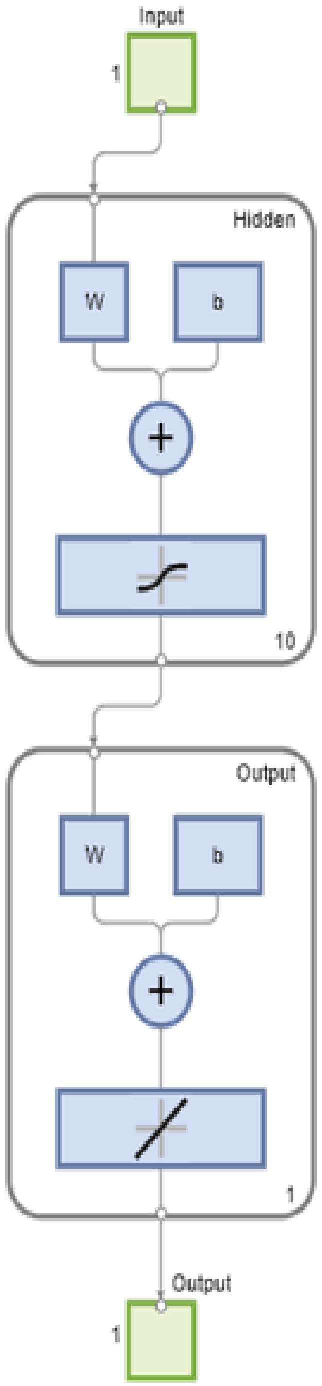
```
close all, clear all, clc
% Define input and output data
X = [ -0.5 -0.5 +0.3 -0.1; ...
-0.5 +0.5 -0.5 +1.0] % Input Vector
T = [1 1 0 0];          % Target Vector
plotpv(X,T);

% Create perceptron
net = perceptron;
net = configure(net,X,T);

% Replot with the neuron's
  initial attempt at classification.
plotpv(X,T);
plotpc(net.IW{1},net.b{1});
```

```
% convert the input and target data into
   sequential data
XX = repmat(con2seq(X),1,3);
TT = repmat(con2seq(T),1,3);
net = adapt(net,XX,TT);
plotpc(net.IW{1},net.b{1});

% classify new input vector, for example [0.7; 1.2].
x = [0.7; 1.2];
y = net(x);
plotpv(x,y);
point = findobj(gca,'type','line');
point.Color = 'red';

hold on;
plotpv(X,T);
plotpc(net.IW{1},net.b{1});
hold off;
```



# Use a Feedforward NN

```
% Load the training data.
% Construct a feedforward One hidden layer of size 10.

close all, clear all, clc
% Load the training data.
[x,t] = simplefit_dataset;
% Construct a feedforward network- One layer - 10 Nodes
net = feedforwardnet(10);

%Train the network net using the training data.
net = train(net,x,t);
view(net)

% Assess the performance using the input and target
% TT= perform(net,x,t)

%Estimate the targets using the trained network.
y = net(x);
%Assess the performance after training using MSE
perf = perform(net,y,t)
```




Network Diagram

**Training Results**

Training finished: Met validation criterion

**Training Progress**

| Unit | Initial Value | Stopped Value | Target Value |
|---|---|---|---|
| Epoch | 0 | 19 | 1000 |
| Elapsed Time | - | 00:00:16 | - |
| Performance | 324 | 9.25e-05 | 0 |
| Gradient | 414 | 0.000292 | 1e-07 |
| Mu | 0.001 | 1e-05 | 1e+10 |
| Validation Checks | 0 | 6 | 6 |

**Training Algorithms**

Data Division: Random dividerand
Training: Levenberg-Marquardt trainlm
Performance: Mean Squared Error mse
Calculations: MEX

**Training Plots**

Performance | Training State
Error Histogram | Regression

# Train a Feedforward NN to predict temperature

```
% Read Data from the Weather Station ThingSpeak Channel
% Assign Input Variables and Target Values
% Create and Train the Two-Layer Feedforward Network
% Use the Trained Model to Predict Data


close all, clear all, clc

% Read Data from the Weather Station ThingSpeak Channel

data = thingSpeakRead(12397,'Fields',[2 3 4 6],'DateRange',[datetime('January 7, 2018'),datetime('January 9, 2018')],...

'outputFormat','table’);


% Assign Input Variables and Target Values

inputs = [data.Humidity'; data.TemperatureF’; data.PressureHg'; data.WindSpeedmph'];

tempC = (5/9)*(data.TemperatureF-32); % Convert temperature from Fahrenheit to Celsius


% specify the constants for water vapor (b) and barometric pressure (c).

b = 17.62;

c = 243.5;
```

# Use a Feedforward NN

```
% Calculate the intermediate value 'gamma', and assign target values for the network.
gamma = log(data.Humidity/100) + b*tempC ./ (c+tempC);
dewPointC = c*gamma ./ (b-gamma);
dewPointF = (dewPointC*1.8) + 32;
targets = dewPointF’;

% Create and Train the Two-Layer Feedforward Network
net = feedforwardnet(10);
[net,tr] = train(net,inputs,targets);

% Use the Trained Model to Predict Data
output = net(inputs(:,7))
```

UNIVERSITY OF PLYMOUTH
School of Engineering, Computing and Mathematics