FINAL
TAKE-HOME EXAM

Artificial Intelligence

Fall 2023 **CMSC 409** *Name & ID* **Group 1: Start Dec. 9 (noon), end Dec. 11 (noon**
Group 2: Start Dec. 12 (noon), end Dec. 14 (noon)

Student certification:
Please read and sign the following statement before you begin:
I fully understand that I am on my honor to do my own work on this examination. I will not share the exam or any portion of it with the rest of the group that may be taking the exam after me. Further, I certify that I have neither given nor received any aid on this test.
If I violate this confidence, I may receive a letter grade of "F" for this exam, or for the course, or be expelled from the program.

Print Name: ___Devon McDermott_________________________________
Date:_______12.13.23___________ Signed : ________________________________Devon McDermott________(you can sign/scan or use e-signature)

Note: This exam requires 3 out of 4 problems below to be solved. Problems Ex.2.3 and Ex.2.4 are mandatory. Select one problem between Ex.2.1 and Ex.2.2 to solve. You can solve the remaining one for extra credit.

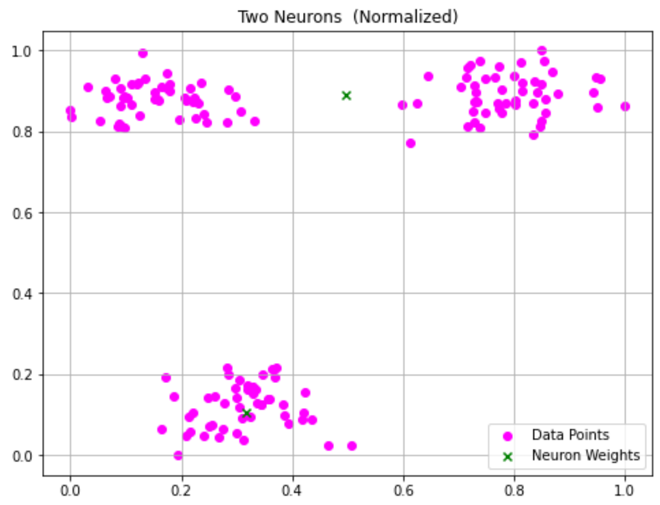
Ex.2.1. Competitive learning (35p) Examine the dataset "Ex1_data.csv". This dataset has 3 different clusters, is not labeled and it has two different features. Code and execute Kohonen's Winner Take All clustering algorithm. Test the following assumptions: 1. Use the two-neuron, single-layer network and assign weights randomly. First, run the algorithm once to explore the data. Use the outcome to understand the arrangement of data points (you can also plot the data and weights). Based on what you understood, choose more suitable initial weights and run the algorithm on the same network again. Observe the changes in weights (plot) after each pattern is applied. Report on the process of choosing weights, the number of iterations, speed of convergence, and other "lessons learned". Attach the code and plots created during the process. Provide intermediate and final plots describing clusters found. 2. Repeat 1. with 3 and 7 neurons. Discuss.

Selecting weights randomly versus manually in the Kohonen clustering algorithm, can have significant implications for the learning process and the final performance of the network. Random initialization introduces variability in the starting point of the learning process. This can be beneficial as it prevents the network from starting in a biased state. Randomly initialized weights may require more training epochs to converge. There's a risk of introducing bias if manual weights are chosen as selection allows incorporating prior knowledge. Manual initialization also provides control over the starting point, potentially speeding up convergence.

Console Output for Randomly Selected weights:

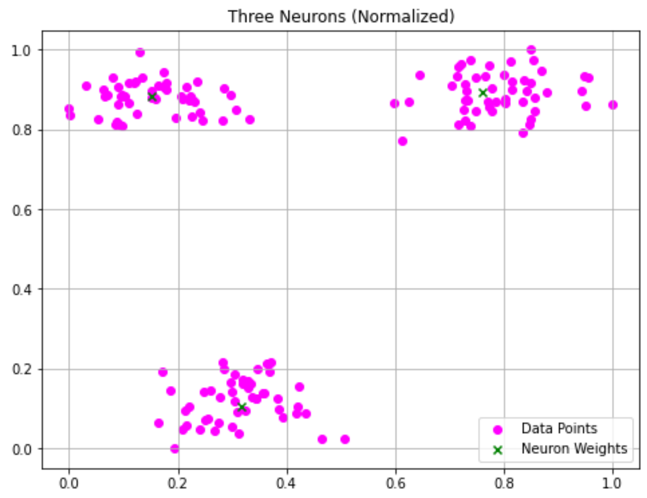
Part 1: Two Neurons

```
Final Weights for 2 Neurons:
[[-7.73940065 -8.46191444]
 [-5.46883564  6.06290676]]
```



Part 2: Three Neurons

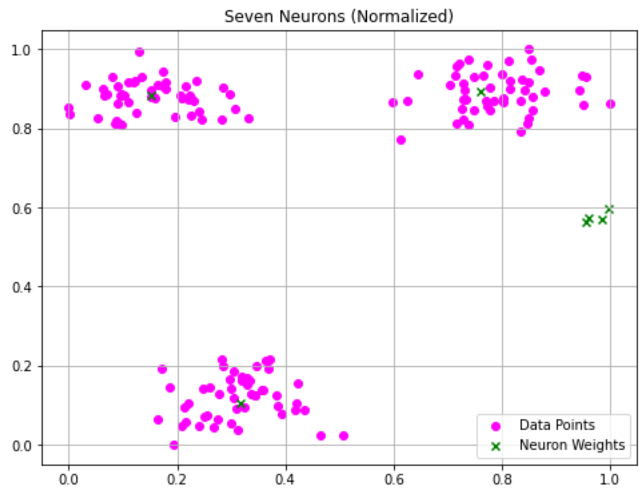
```
Final Weights for 3 Neurons:
[[-2.11819905  6.14664601]
 [-9.82366213  5.93561601]
 [-7.73940065 -8.46191444]]
```



Part 3: Seven Neurons

```
Final Weights for 7 Neurons:
[[ 0.94248722  0.68848859]
```

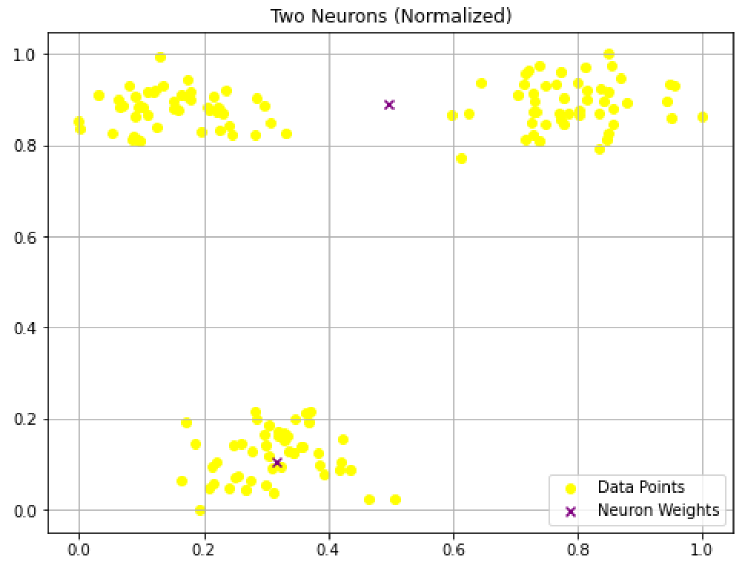
```
 [-9.82366213  5.93561601]
 [ 0.5557891   0.65371225]
 [ 0.26942981  0.54762069]
 [-2.11819905  6.14664601]
 [-7.73940065 -8.46191444]
 [ 0.7523362   0.48705955]]
```



Output in console for manually selected then updated weights:

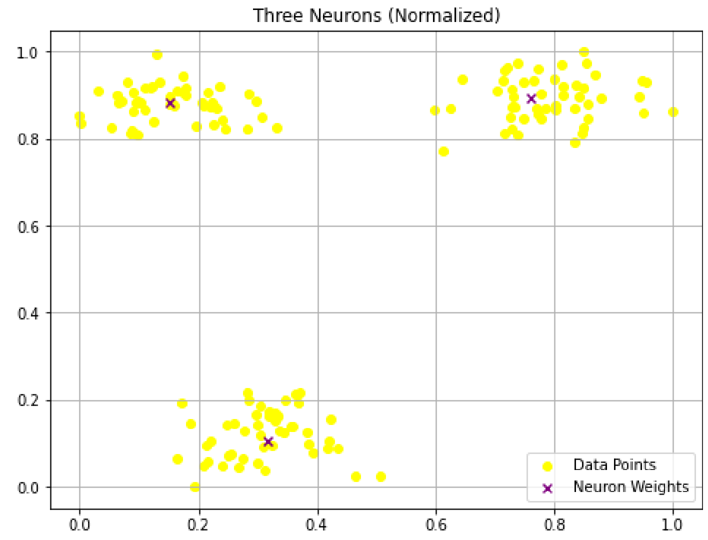
```
Part 1: Two Neurons with Manual Weights

Final Weights for 2 Neurons:
[[-5.46883564  6.06290676]
 [-7.73940065 -8.46191444]]
```



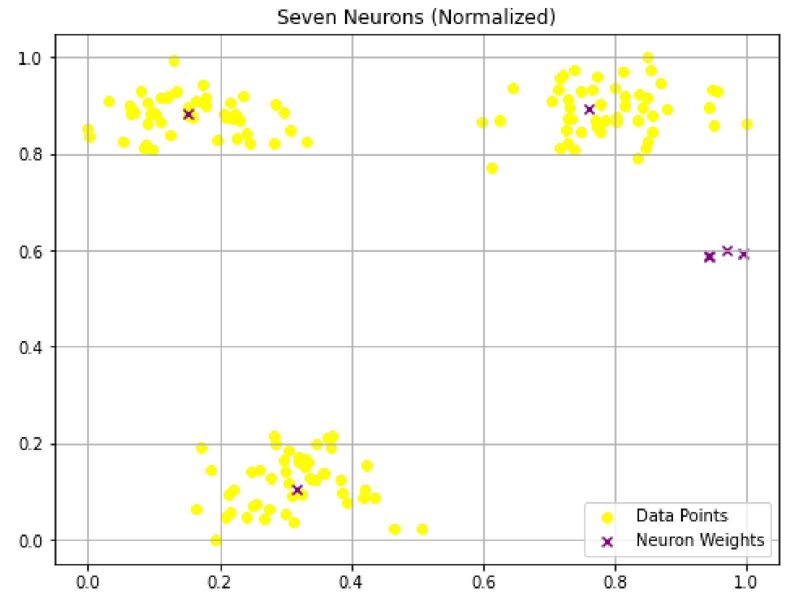
```
Part 2: Three Neurons with Manual Weights

Final Weights for 3 Neurons:
[[-7.73940065 -8.46191444]
[-2.11819905  6.14664601]
[-9.82366213  5.93561601]]
```



Part 3: Seven Neurons with Manual Weights

```
Final Weights for 7 Neurons:
[[-7.73940065 -8.46191444]
 [ 0.61193085  0.74446426]
 [ 0.99339816  0.65638705]
 [-9.82366213  5.93561601]
 [-2.11819905  6.14664601]
 [ 0.96633422  0.67852476]
 [ 0.64783209  0.77000935]]
```



Ex.2.3. Designing controller *(45p)*
Manually design a fuzzy controller for the given control surface. Select adequate number membership functions for both input variables (you may select equal or not equal spacing). Use triangular membership functions. Identify an adequate number of output singletons with values of your choice. Fully document the design process of your controller. a) Test your controller on two points:
Point 1: values (X,Y)=(5, 5).
Point 2: values (X,Y)=(16, 16).
b) Plot the given control surface and control surface from your controller. Compare and comment. c) What is the error of your controller? Can you plot this error (the difference between the two)? Discuss and comment.

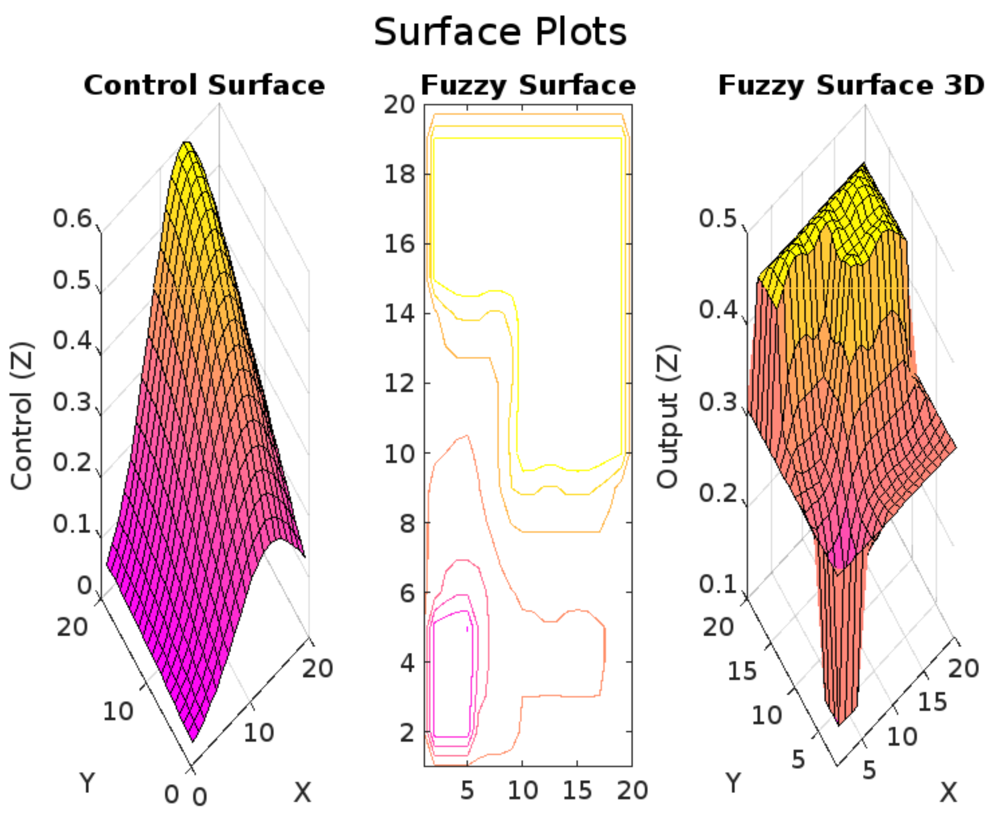
Console Output:
```
Output at Point 1 (X=5, Y=5): 0.099488
Output at Point 2 (X=16, Y=16): 0.48871
```
These values represent the degree of membership

Visually we can easily identify there are differences between the given control surface and the fuzzy controller surface. The 3D surface plots allow you to visually inspect how well the fuzzy controller output approximates the control surface. Differences between the two surfaces indicate areas where the fuzzy system may not be accurately capturing the behavior of the control surface.

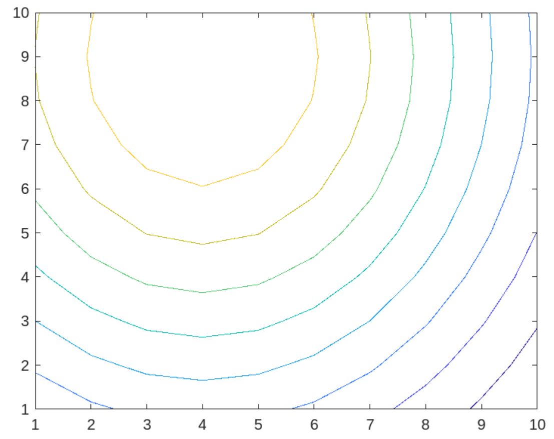
The absolute error is calculated as the absolute difference between the control surface and the output of the fuzzy controller. Mathematically, it's expressed as error = abs(zControl - outputGrid). The contour plot of the absolute error provides a clear visualization of regions where the error is larger. Sharp contours differences indicate larger errors. Areas with higher absolute error tell us that the fuzzy system is not capturing the control surface accurately. This could be due to the membership functions, or rules of the fuzzy system.

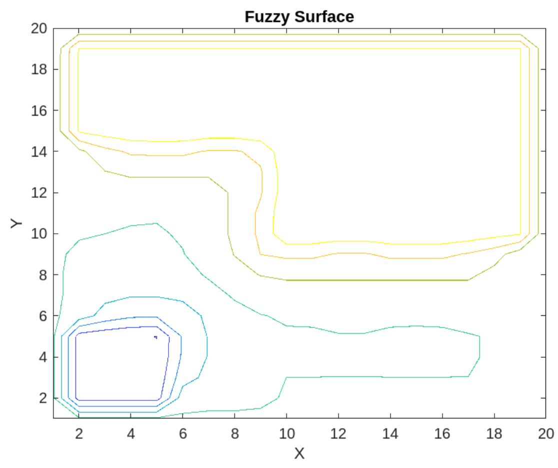
Analysis can help to identify specific regions where the fuzzy system performs well and areas where improvements are needed.

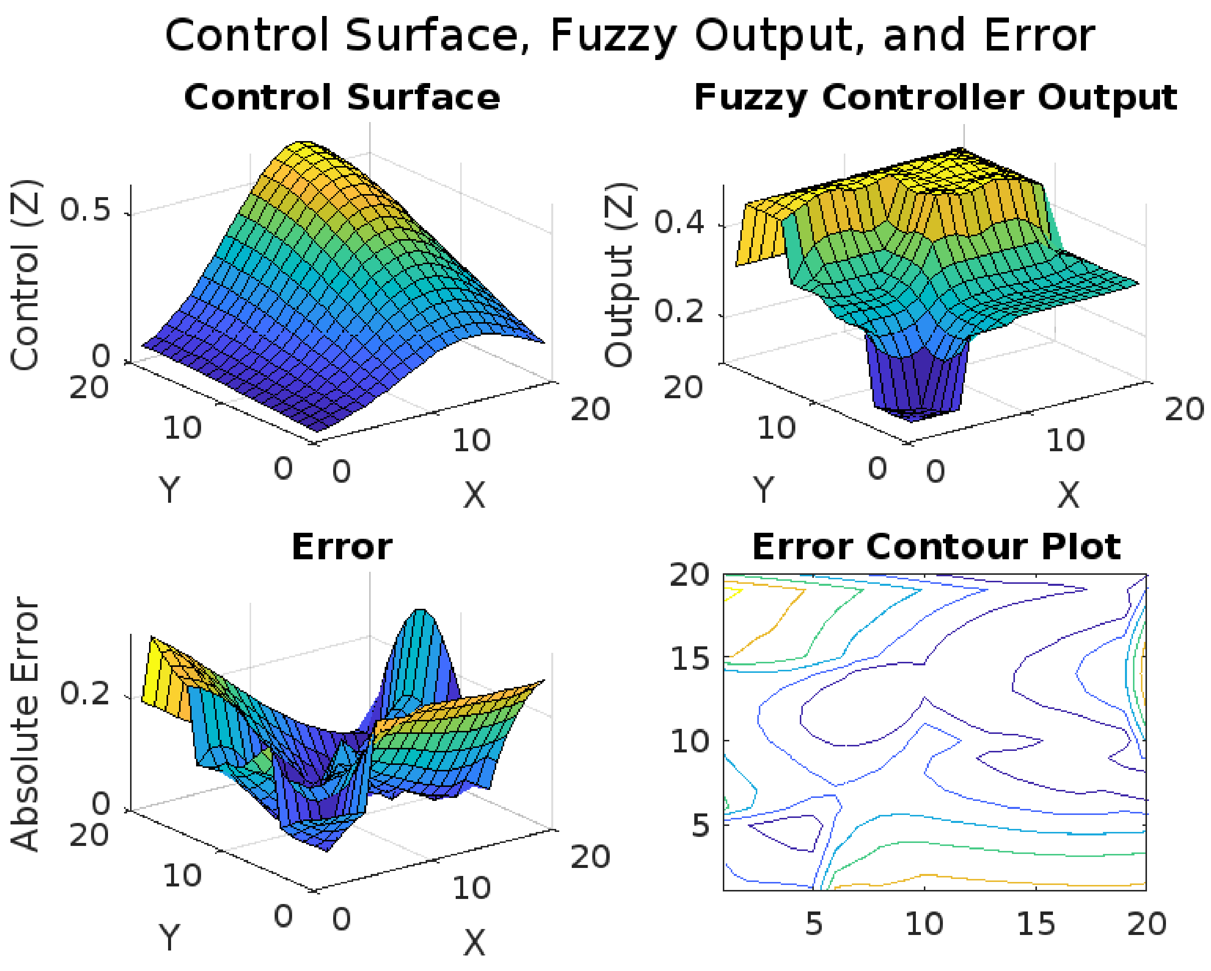
Console Output: `Maximum Absolute Error: 0.31366`



ORIGINAL SURFACE



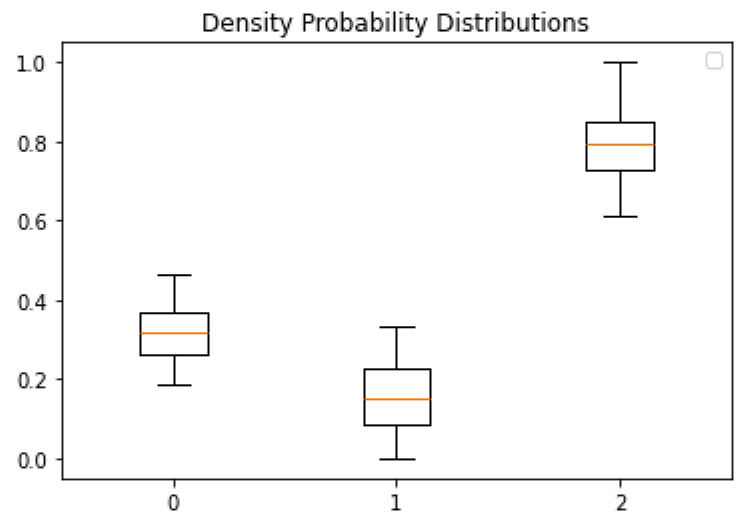




Ex.2.2. Bayes classifier (35p)
Examine the training and testing datasets: “Ex2_train.csv” and “Ex2_test.csv”. This is the same dataset of the first
exercise except now it is labeled, and has one feature removed (the first column represents the feature and the second
column represents the label). Code the Bayes classifier.
a. Use training dataset to estimate P(x|Ci) and P(Ci). Plot density probability distributions.
b. Execute trained Bayes classifier on test data set. How accurate is the algorithm?
Attach the code and other plots you may have created. Discuss the solution.
Note: Bayes classifier is a representative of supervised clustering techniques. Provided dataset contains three classes.

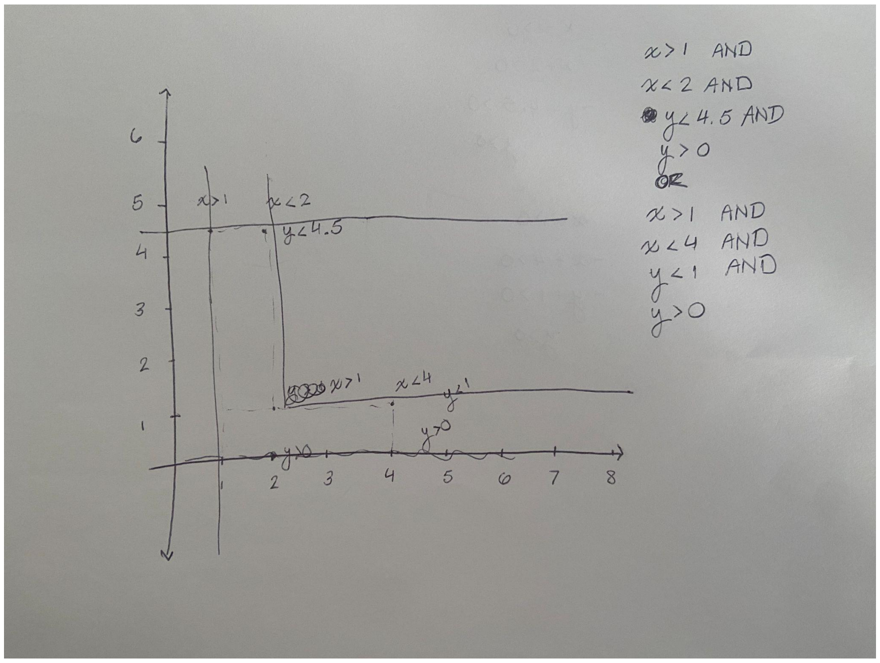
Output:
Accuracy: 84.31%

An accuracy of 84.31% is indicative that Naive Bayes classifier is correctly predicting the class labels for approximately 84.31% of the instances in the test set.
Naive Bayes assumes that features are conditionally independent given the class label. If this assumption doesn't hold well in the data, it might impact performance. The quality of features used for training the model are essential as well. If important features are missing or irrelevant features are included, it can affect the accuracy.

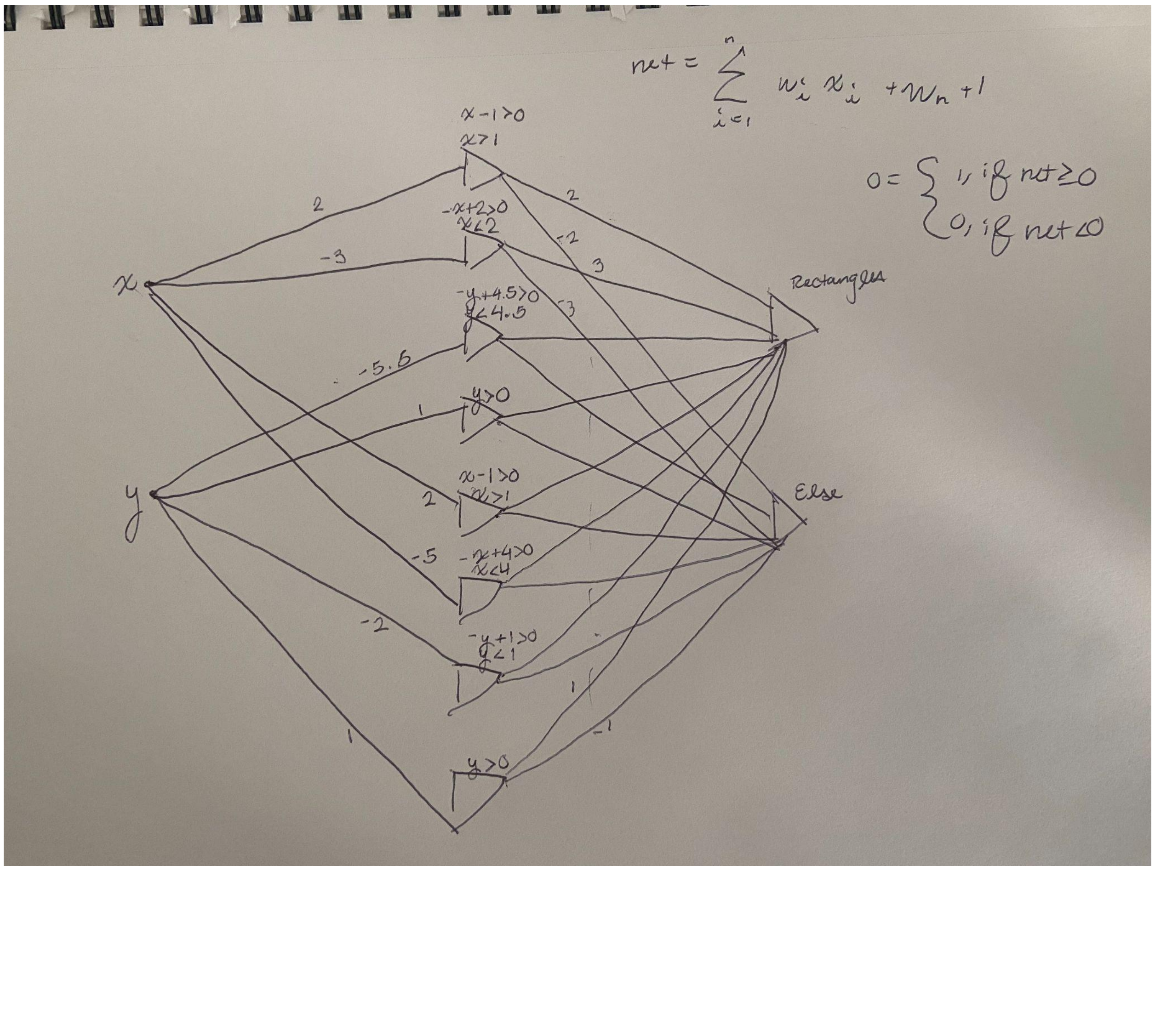


Ex.2.4 Creating neural network via design (selecting 2D area) (20p) A character presented via square patterns is shown in the figure below. a) Design a network that extracts a character formed by rectangular patterns. b) Clearly indicate your choice of activation function and neuron definition. Draw the network architecture and clearly indicate weights. c) Comment. Is there any other architecture that could solve it?

(Neuron image below)



$x > 1$ AND
$x < 2$ AND
$y < 4.5$ AND
$y > 0$
OR
$x > 1$ AND
$x < 4$ AND
$y < 1$ AND
$y > 0$



$$net = \sum_{i=1}^{n} w_i x_i + w_n + 1$$

$$o = \begin{cases} 1, \text{ if } net \geq 0 \\ 0, \text{ if } net < 0 \end{cases}$$