FINAL
TAKE-HOME EXAM
Fall 2018

Artificial Intelligence

CMSC 409

Nathan West, V00717736
Name & ID

Due Tuesday, Dec. 11 at 1pm

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. 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.
Further, I certify that I have neither given nor received any aid on this test.

Print Name: Nathan West *Date:* 12/10/18
Signed : Nathan West *(you can sign/scan or use e-signature)*

Note: Problems 2.3 and 2.4 are regular credit problems. Select one problem between Ex.2.1 and Ex.2.2 to solve for "regular" and the other as extra credit (if you wish).

Ex.2.1. Competitive learning *(25p)*
Examine the data set "Ex1_data.txt". This data set comes from IRIS data set https://archive.ics.uci.edu/ml/datasets/Iris, with labels and two out of four features removed. 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 applied. Report on the process of choosing weights, number of iterations, speed of convergence, and other "lessons learned". Attach the code and plots created during process. Provide intermediate and final plot describing clusters found.
2. Repeat 1. with 3 and 7 neurons. Discuss.

Ex.2.2. Bayes classifier *(25p)*
Examine the training and testing data sets: "Ex2_train.txt" and "Ex2_test.txt". This is also IRIS data set https://archive.ics.uci.edu/ml/datasets/Iris, but with labels this time (and three out of four features removed). Code the Bayes classifier.

1. Use training data set to estimate $P(x|C_i)$ and $P(C_i)$. Plot density probability distributions.
   a. Execute trained Bayes classifier on test data set. How accurate the algorithm is?

Attach the code and other plots you may have created. Discuss the solution.

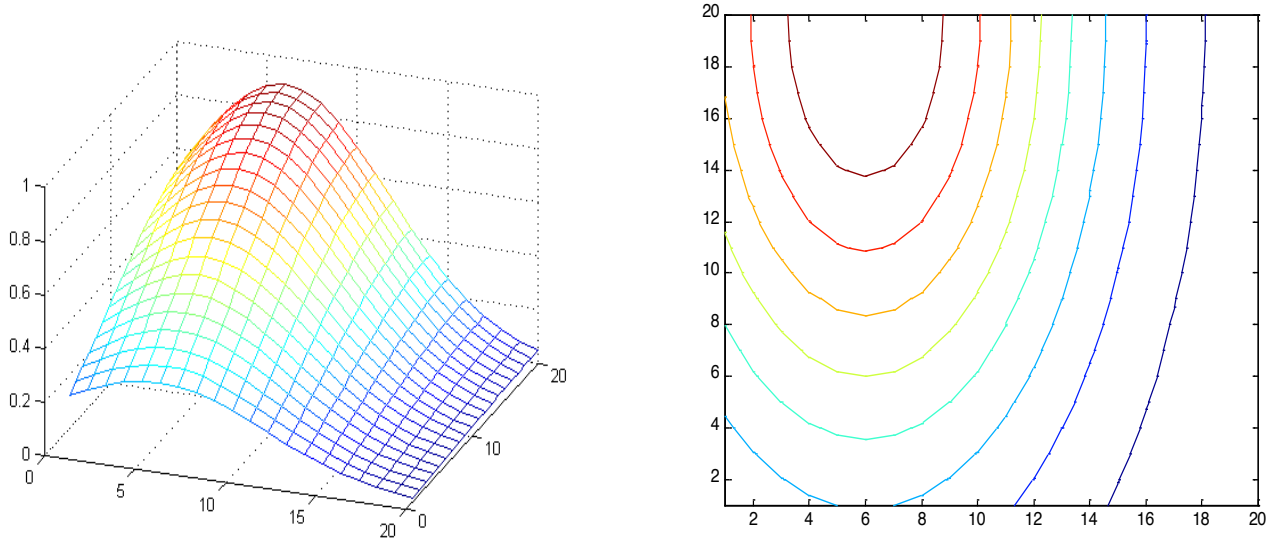
Note: Bayes classifier is a representative of supervised clustering techniques. Provided data set contain three classes.

Ex.2.3. Designing controller *(35p)*
The required control surface is given by the following formula:

$$z = 0.9 \exp\left[-0.003 \left(x - 20 \right)^2 - 0.015 \left(y - 6 \right)^2 \right]$$

and the following MATLAB code:

```
clear all; x=[1:20]; y=[1:20];
for i=1:20;   for j=1:20;
    z(i,j)=0.9*exp(-0.003*(x(i)-20).^2-0.015*(y(j)-6).^2);
end; end;
figure(1); clf; contour(z);
figure(2); clf; mesh(z); view(20,30);
```



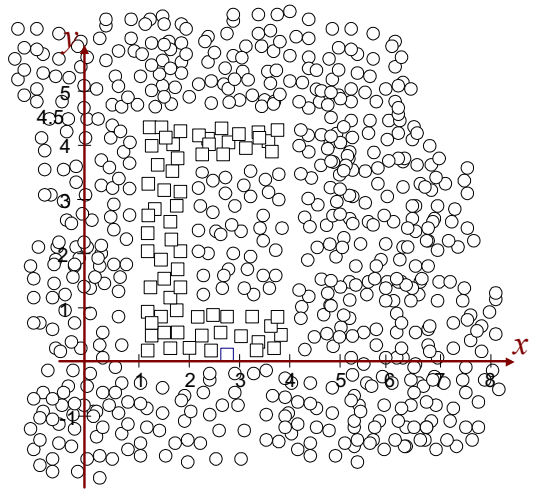
Manually design a fuzzy controller for the given control surface. Select adequate number membership functions for both input variables (you may select equal or not equal spacing). Use triangular membership functions. Identify adequate number of output singletons with values of your choice. Fully document the design process of your controller.

a) Test your controller on two points:
   Point 1: values (X,Y)=(6, 6).
   Point 2: values (X,Y)=(18, 18).
b) Plot required 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.

Ex.2.4 Creating neural network via design (selecting 2D area) *(15p)*

A character presented via square patterns is shown in the figure below.

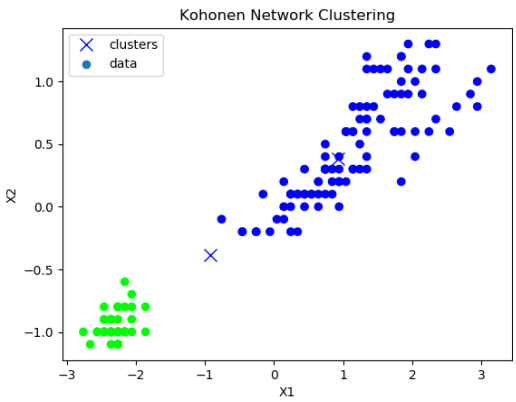
a) Design a network that extracts a character formed by rectangular patterns.
b) Clearly indicate your choice of transfer function and neuron definition. Draw the network architecture and clearly indicate weights. Present desired outputs.
c) Comment.



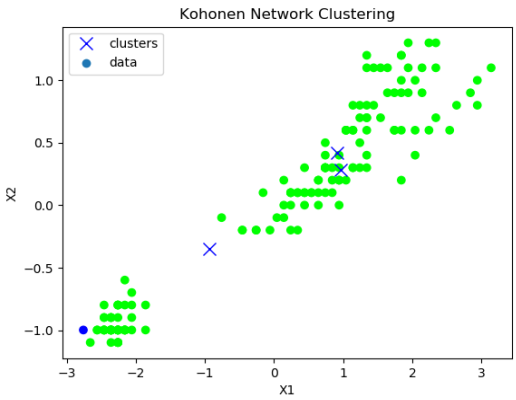
Nathan West
CMSC 409 - Final Exam
12/10/18

2.1 - Competitive Learning (Extra credit)

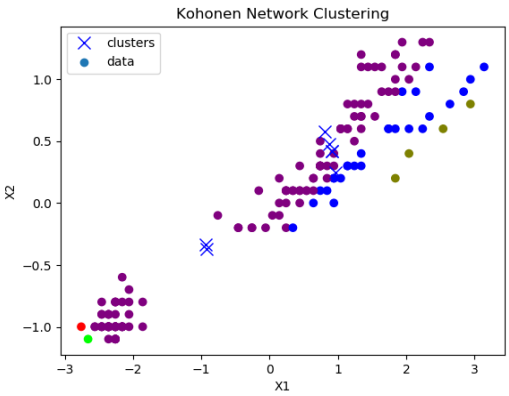
- I first test the Kohenon network using random weights for 2 neurons. My learning rate started as 0.3, with 500 iterations of training. I noticed how the network wasnt clustering properly, so I decreased the learning rate to 0.05, which affected the clustering greatly and correctly. I chose the static weights for each of the neuron architecture with a range of 1-10. I decreased the learning rate to 0.01 and decreased the number of iterations to 300. Overall, I found that the network performed somewhat the same as when randomly selecting weights. Below are my results graphically.



Graph for 2 neurons



Graph for 3 neurons

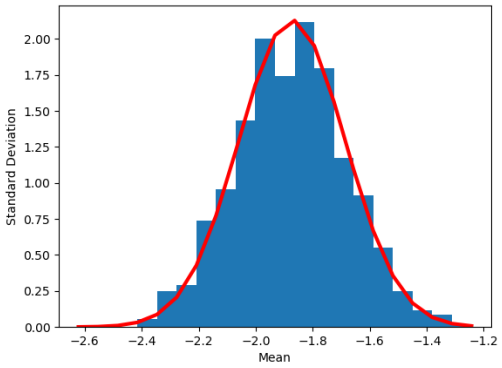


Graph for 7 neurons

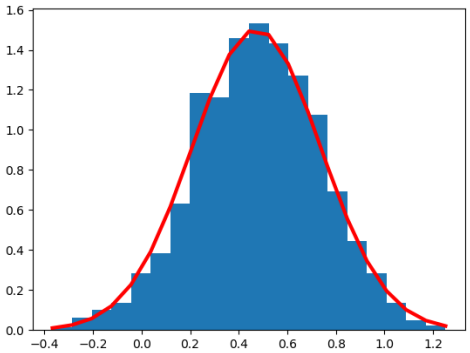
2.2 – Bayes Classifier

a) After training the classifer, it made predictions using the mean and standard deviations from the training data set. The classifier received an accuracy of 100%. The classifier works by training on each. The dataset is read into a dictonary using the class as the key and its values are a list of each classes float values from the file. Using the dictionary the means and standard deviations are calculated for each class. Using the test dataset, we perform the Guassian normal distribution on each value in the test set for all 3 classes. The max probability between the 3 classes is the predicted class for that test set value.

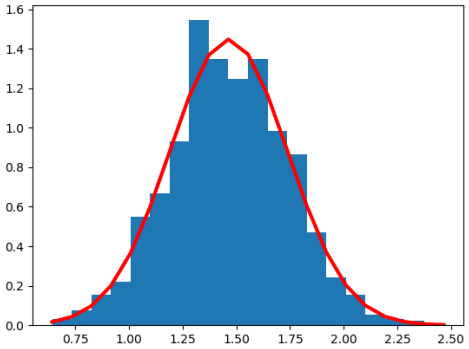
* note class probabilities are printed to console



Density graph for Class 1

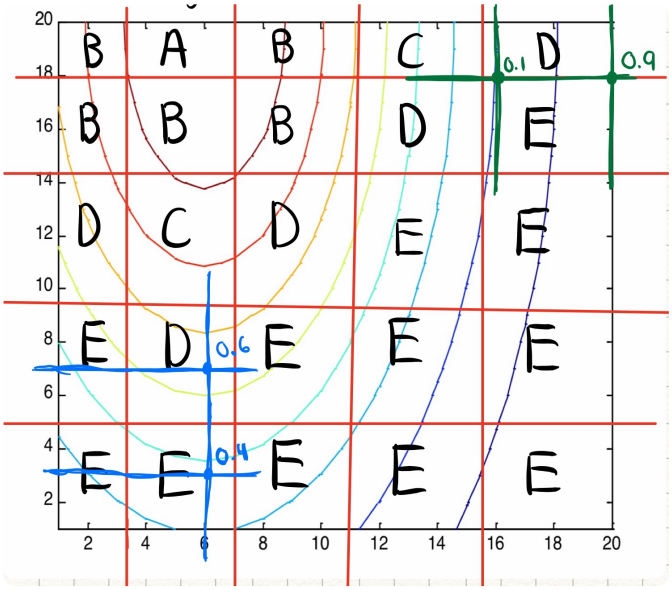


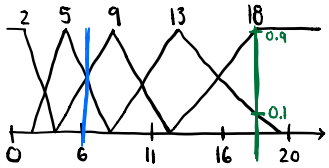
Density graph for Class 2

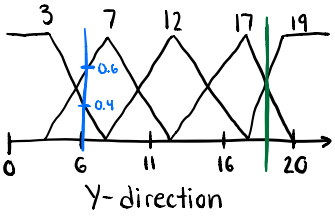


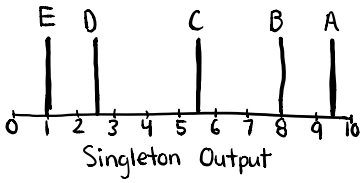
Density graph for Class 3

2.3- Design Fuzzy Controller





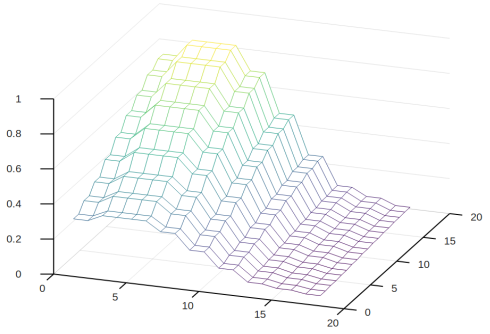




- The control surface on the graph shows the area which determines the condensed areas for the fuzzy controller. The X-direction fuzzy controller condenses when $x = 6$ and the fuzzy controller for the Y-direction fuzzy controller condenses when $y \approx 19$. The triangular member functions were chosen because I felt they best represent the "A" are of the control surface.

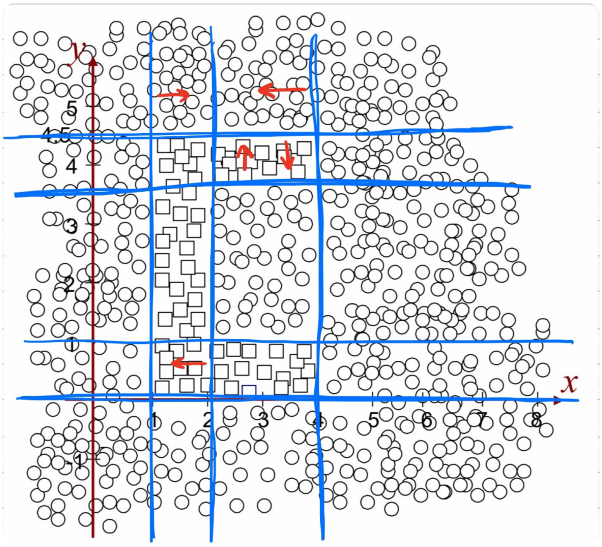
a) Point: (6, 6) ● out = $0.6 \cdot 2.5_{(D)} + 0.4 \cdot 1.3_{(E)}$
max (D) = 0.6 = 2.02
max (E) = 0.4

Point: (18, 18) ● out = $0.9 \cdot 1.3_{(E)} + 0.1 \cdot 2.5_{(D)}$
max (E) = 0.9 = 1.23
max (D) = 0.1

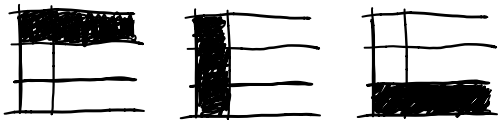


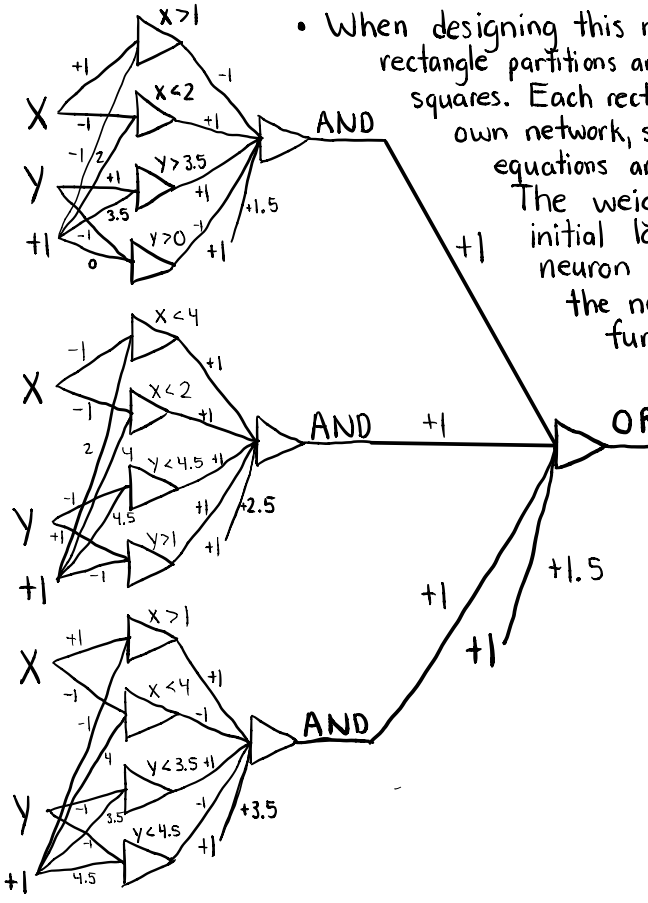
- With my member functions inputted in the equation, it looks like more of a trapezoidal representation of the graph. This effect could be the error and possibly selecting a member functions closer to 0 could have reduced this effect.

2.4 – Designing neural network



neuron	equations	weights
$x > 1 \longrightarrow$	$x - 1 > 0$	1 0 -1
$x < 2 \longrightarrow$	$-x + 2 > 0$	-1 0 2
$x < 4 \longrightarrow$	$-x + 4 > 0$	-1 0 4
$y > 0 \longrightarrow$	$y > 0$	0 1 0
$y < 1 \longrightarrow$	$-y - 1 > 0$	0 -1 -1
$y > 3.5 \longrightarrow$	$-y + 3.5 > 0$	0 -1 3.5
$y < 4.5 \longrightarrow$	$-y + 4.5 > 0$	0 -1 4.5
		x y bias





- When designing this neural network, I first draw several rectangle partitions around the areas that contain lots of squares. Each rectangle I draw gets represented as its own network, shown by the figures above. The neuron equations are chosen from the edges of each rectangle. The weights for the inputs and bias of the initial layer of the network come from the neuron equations. The weights for the rest of the network are randomly selected. The transfer function is a hard activation function because we're trying to discover an edge of overall figure the squares on the graph.