EEL 5840 Exam II

100 minutes CLOSED BOOK (one page of notes – front and back-). INCLUDE YOUR ANSWER IN THE EXAM PAGES. Full credit require full explanation of your answers.

NAME: UF ID:

By signing your name, you declare that you do not help or get help from others during the exam.

---

1. (4 points) Explain clearly the difference between regression and classification.

BOTH Methods are supervised, however the desired responses are different as well as the goals.

In regression we assume that the data is ~~belongs~~ homogeneous (same source) and the goal is to transform it (~~to~~ map it) to the desired.

In classification, we assume that the data we receive is heterogeneous, and the goal is to separate it in their homogeneous parts. Hence the difference in the labels that drive the solution

2. (4 points) In terms of the assumptions about the data, classifiers can be divided in parametric and non-parametric, where the former makes assumptions about the input data distributions. For the two classes of classifiers that you have studied (Bayesian and neural networks) put them in one of these two types and justify your answer.

Bayesian classifiers are parametric because we implicitely model each class by a Gaussian probability density function.

Neural networks are nonparametric because they do not impose any assumption a priori to define the separation surfaces

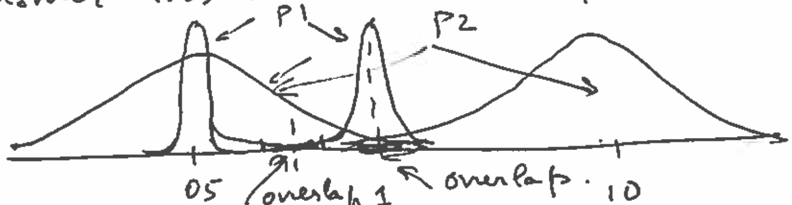
3. (8 points) You have two different classification problems, each involving two equiprobable classes.

First Problem: Class 1 mean= 0.5, variance = 0.01
Class 2 mean= 1.0, variance = 0.01

Second Problem: Class 1 mean= 0.5, variance=1
Class 2 mean= 10, variance= 1

Which one will give you smaller error using a Bayes classifier? Justify your answer.

the answer lies in the overlap between the PDFs in each case



Since in each problem the class variance is the same, we know immediately the separation point as difference in means

Prob I

$T_1 = \frac{\mu_1 + \mu_2}{2} = \frac{0.5+1}{2} = 0.75$

Problem II

$T_2 = \frac{0.5+10}{2} = 5.25$

the area under the Gaussian curve can also be easily estimated with the Q function, but you really don't need it, if you follow this reasoning. the scale is the S.d. so for P1 s.d=0.1 while P2 is s.d = 1.

In Prob I, the difference between the means and the separation PoinT T_1 is ± 0.25, which is ~ 2.5 the SD.

In Prob II, the difference between the mean and ~~the~~ is ± 4.75, which is about 5 the SD.

therefore the overlap is going to be much smaller in P2, and so is the error.

NOTE: If the class variances where not the same, you would have to use formulas + Q functions.

4. (4 points) What is the difference between a Bayes classifier and a LDA classifier in terms of the shape of the separation surface? For high dimensional data, which is the one that is more computationally complex? Justify your answer.

In Bayes classifiers the separation surface are always quadratic functions.
In LDA the separation surface is always linear, ~~and~~ fleaed ~~in~~ the line that links the means.

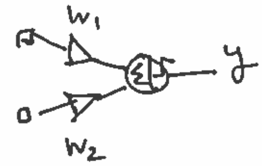
In high dimensional data LDA is much simpler because we do not need to use the correlation matrix to determine the discriminant fctn as required by Bayes

5. (4 points). What is the fundamental difference between classification and clustering? Can clustering help for classification?

Both methods attempt to separate the input data in homogeneous parts. However classification is a supervised method, i.e. it requires labels, while clustering only uses the structure of the input data

Clustering can NOT in general help classification, because when the data overlaps only external information (labels) can help place the bondary

6. (8 points). Assume you have the following simple network with two inputs and one output. The processing unit is M-P with a threshold nonlinearity (0, 1) centered at net=0. The initial condition for the weights is $w_{11}(0) = w_{21}(0) = 0.1$ and $y_1(0)=1$. The weights are trained with the following rule:



$$w_{ij}(n+1) = w_{ij}(n) + \eta y_i(n)(x_j(n) - w_{ij}(n))$$

6.1 In practice can the initial conditions for weights and outputs be zero?
6.2 Given the training data in the table compute the final weight values for a stepsize of 0.5.
6.3 Compute the outputs for the test set. Explain the possible applications of this network and justify your answer.

Training

X1	X2
0.3	0.5
0.5	0.1

Test

X1	X2
1	0.5
-0.5	0.2
0.3	-1

6.1. If the initial weight and outputs are zero the weights never adapt! because the update is zero.

6.2. $W_{11}(1) = 0.1 + 0.5 \times 1 \, (0.3 - 0.1) = 0.1 + 0.1 = 0.2$

$W_{21}(1) = 0.1 + 0.5 \times 1 \, (0.5 - 0.1) = 0.1 + 0.2 = 0.3$

$net(1) = 0.3 \times 0.2 + 0.5 \times 0.3 = 0.06 + 0.15 = 0.21 \Rightarrow y(1) = 1$

$W_{11}(2) = 0.2 + 0.5 \times 1 \, (0.5 - 0.2) = 0.2 + 0.5 \times 0.3 \sim 0.2 + 0.15 = 0.35$

$W_{21}(2) = 0.3 + 0.1 \times 1 \, (0.1 - 0.3) = 0.3 - 0.2 \times 0.1 \approx 0.28$

$net(2) = 0.5 \times 0.35 + 0.1 \times 0.28 > 0 \Rightarrow y(2) = 1$

Test set output

no update, so weights are

$w_{11} = 0.35$

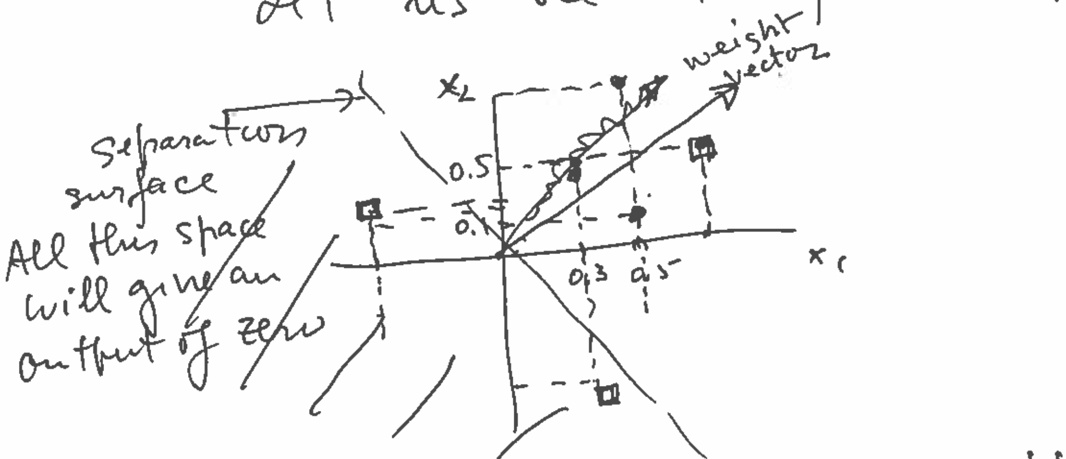
$w_{21} = 0.28$

So $net(1) = 1 \times 0.35 + 0.5 \times 0.28 > 0 \Rightarrow y(1) = 1$

$net(2)\ -0.5 \times 0.35 + 0.2 \times 0.28 = -0.19 + 0.06 < 0 \quad y(2) = 0$

$net(3)\ 0.3 \times 0.35 - 1 \times 0.28 < 0 \quad y(3) = 0$

Let us see the points from traing. + test



x_1	x_2	y
1	0.5	1
-0.5	0.2	0
0.3	-1	0

So the system gives an output of 1 when the input is close to the weight vector, and gives an output of zero when the output is far away from the input.

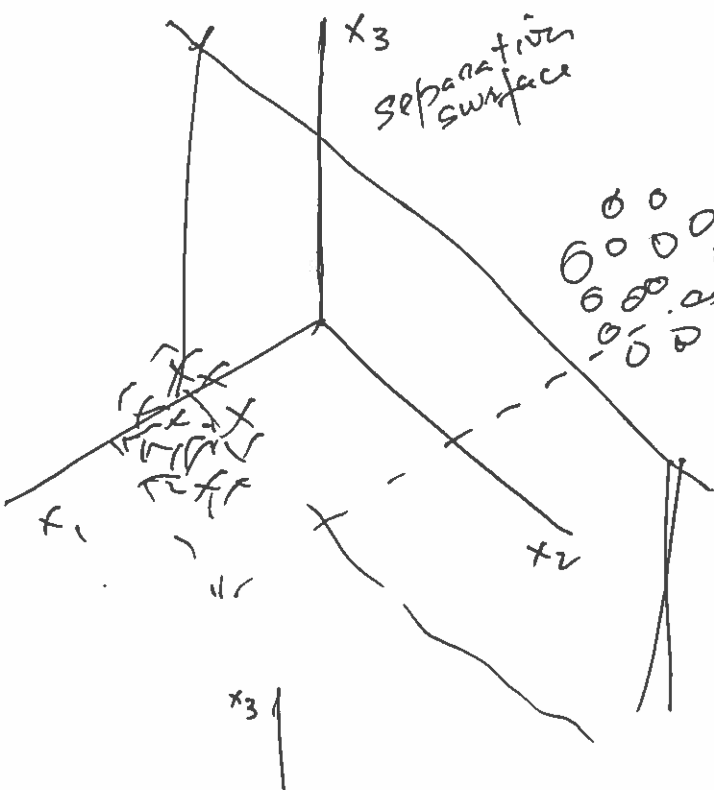
So it is a form of clustering.

it remembers past inputs, and colors the space 1/0

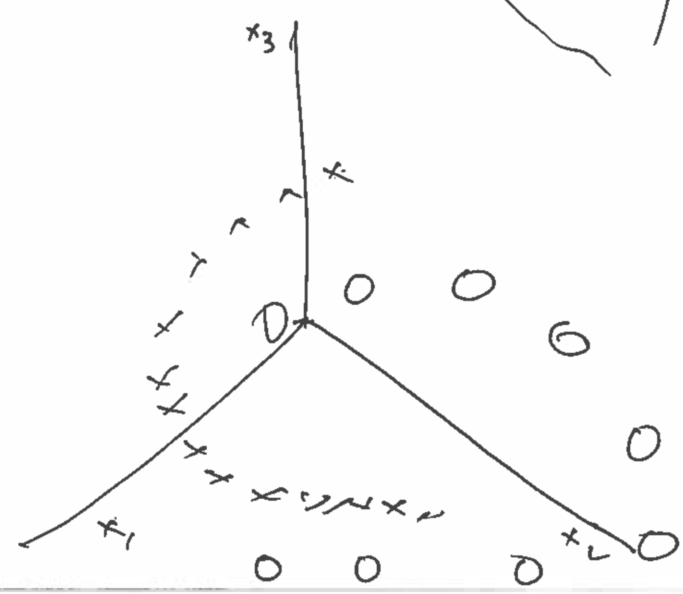
7. (4 points) Is Rosenblatt's perceptron a universal function approximator? Justify your answer by explaining the geometric type of regions that perceptrons can construct. Give two examples in 3D spaces (in tables) that corroborate your answer.

No the one ~~bottom~~ layer perceptron is only able to discriminate linear separable classes.

Example of clusters separable in

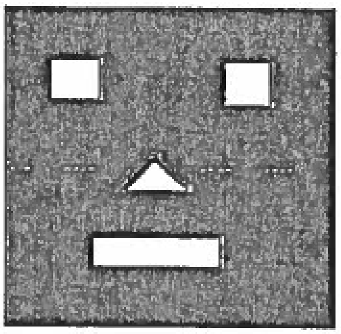


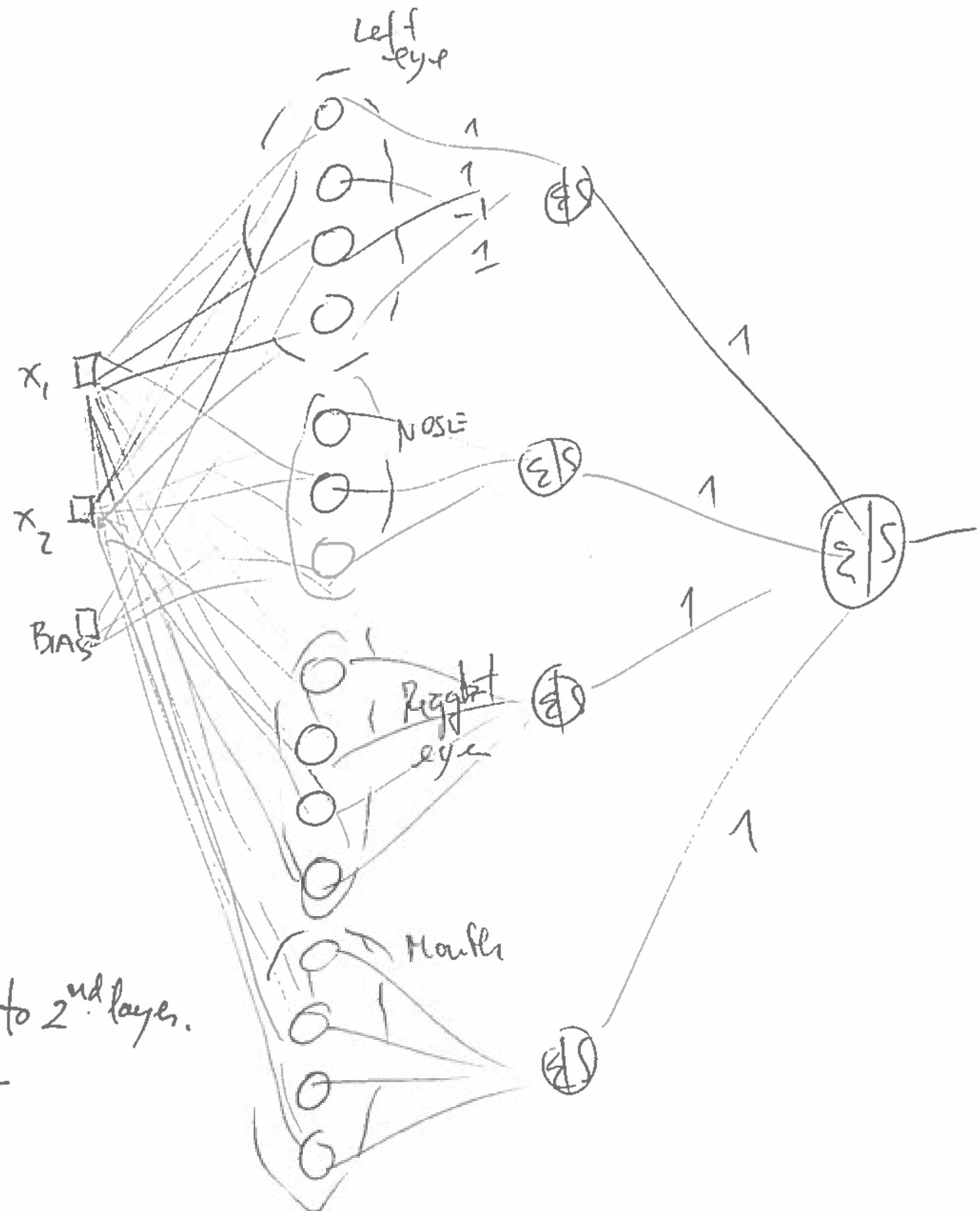
CLASS I			CLASS II		
x_1	x_2	x_3	x_1	x_2	x_3
1	1	1	-1	1	1
0.8	1	0.7	-0.8	1	0.7
⋮			⋮		



CLASS I			CLASS II		
x_1	x_2	x_3	x_1	x_2	x_3
1	1	0	-1	0.5	0
0.5	0.5	0	-0.5	1	0
		⋮			⋮

8. (4 points) Suppose you want to create a mask (see Figure) with a two hidden layer MLP with M-P units. State the smallest number of hidden units you will need in each layer and explain their role in creating the mask. Assume that black is -1 and white is 1. Can you achieve the same goal with a single hidden layer network? Justify your answer.





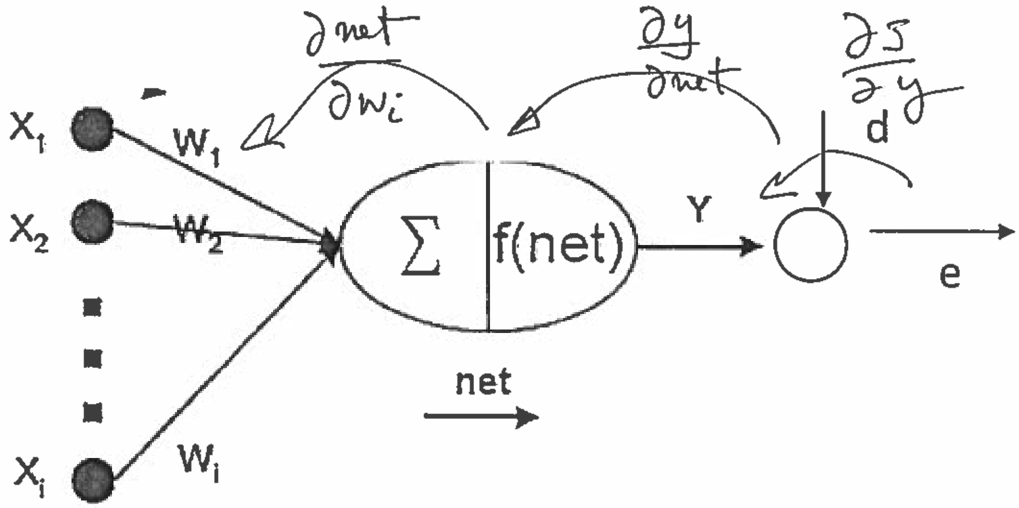
Straight from the picture one needs 15 Hidden PEs In first layer.

But can reuse 4 units for the eyes(2) and mouth (2) so

11 units is minimum in first layer + 4 to 2nd layer.

The one hidden Layer should solve the same problem but it will be more difficult to train, need more units and the solution is approximate (larger error)

9. (4 points) Use the figure below and the chain rule of derivatives to explain how the sensitivity computation procedure can be used to train the weights on a sigmoid unit (the delta rule).



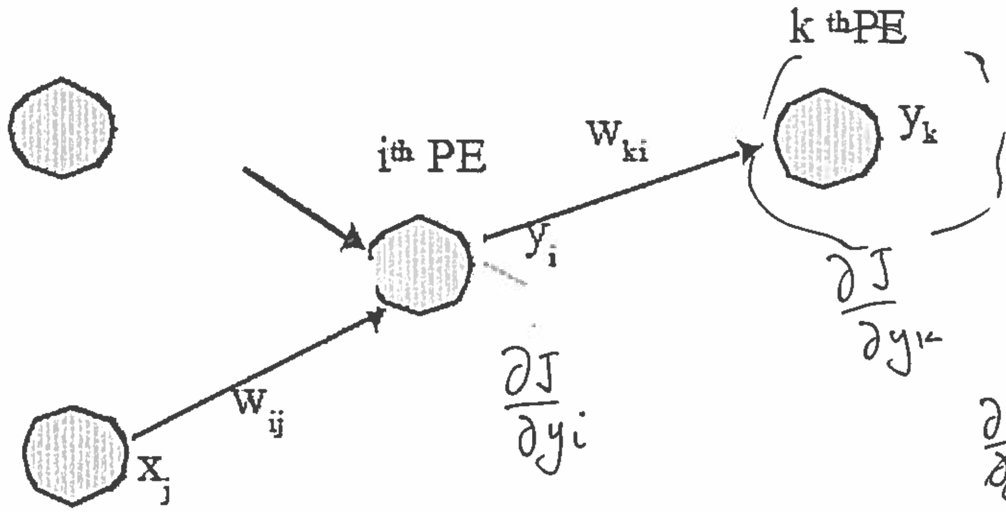
$$\frac{\partial J}{\partial w_i} = \frac{\partial J}{\partial y} \cdot \frac{\partial y}{\partial net} \cdot \frac{\partial net}{\partial w_i}$$

$$= -e \cdot f'(net) \cdot x_i$$

What is interesting is that we are applying chain rule to the topology, going from the top layer toward the input.

10. (6 points) What is the difficulty in training the hidden layer weights in MLP networks? Justify your answer in general terms first. Then show, using the chain rule of derivatives applied to the weight w_{ij} in the figure below how you can effectively solve the problem. Assume that the k^{th} unit is an output unit, for which you have access to the error e_k.

Note: I am not interested in the formula for backpropagation but how you get there…..



Has access to error. so can compute

$\frac{\partial J}{\partial y_k} = -e_k$

the problem is that a hidden unit does not have access directly to the error!

However, we can use the chain rule over the network to get this

$\boxed{\text{note that it only uses states } (y)}$

$$\frac{\partial J}{\partial y_i} = \frac{\partial J}{\partial y_k} \cdot \frac{\partial y_k}{\partial y_i} = -e_k f'(net_k)\, w_{ki}$$

So to compute the weight update for w_{ij} we have

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial y_i} \frac{\partial y_i}{\partial net_i} \cdot \frac{\partial net_i}{\partial w_{ij}} = -e_k f'(net_k)\, w_{ki} f'(net_i)\, x_i$$