1.1 (b)
The features used are [1, area, perimeter^2/area, maximum-minimum]. Here, 1 is for the bias term and area describes how many black pixels are there in a given shape. If area is same, then for most shapes, perimeter^2/area serves as a metric because, it is proportional to the inverse of circularity of the shape(4*pi*area/perimeter^2). In case, the circularities are also close, the final metric is the difference between the maximum distance of a black pixel from the centre(25*25) of the figure and the minimum distance of a black pixel from the centre(25*25) of the figure. The final metric is specific to this dataset because, after observing the data set, I came to the conclusion that most shapes are centred at (25*25) after we reshape the (2500,) matrix to (50,50) matrix. The only shape that has the last metric zero is circle. Every other shape which are close to circle using the previous features can be distinguished from circle using this feature.

For the proofs below, x is used instead of phi(x) and X is the matrix having all x's as rows and Y is the one_hot encoded matrix of the original output column matrix Y.

2.1 a) $E = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} y_k^i \times \log(P(y=k|w_k, x_i))$

In class we had the case of two classes $\Rightarrow K=2$

A

B

$$E = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{2} y_k^i \times \log(P(y=k|w_k, x_i))$$

The two classes were 0 and 1

$$\Rightarrow E = -\frac{1}{N} \sum_{i=1}^{N} \left(y_0^i \log(P(y=0|w_0, x_i)) + y_1^i \log(P(y=1|w_1, x_i)) \right)$$

3

- $y_0^i = 1$ if $y^i = 0$ and $y_0^i = 0$ if $y^i = 1$
$\Rightarrow y_0^i = 1 - y^i$

$y_1^i = 1$ if $y^i = 1$ and $y_1^i = 0$ if $y^i = 0$
$\Rightarrow y_1^i = y^i$

Let w_0, w_1 be the weight vectors associated with 0 and 1 classes

$$P(y=0 \mid w_0, x_i) = \frac{e^{w_0^T x}}{e^{w_0^T x} + e^{w_1^T x}} = \frac{e^{-(w_1 - w_0)^T x}}{1 + e^{-(w_1 - w_0)^T x}}$$

$$P(y=1 \mid w_0, x_i) = \frac{e^{w_1^T x}}{e^{w_1^T x} + e^{w_0^T x}} = \frac{1}{1 + e^{-(w_1 - w_0)^T x}}$$

Assuming $w_1 - w_0 = w$, $P(y=1 \mid w, x_i) = \frac{1}{1+e^{-w^T x}} = \sigma_w(x)$

$$P(y=0 \mid w, x) = \frac{e^{-w^T x}}{1+e^{-w^T x}} = 1 - \sigma_w(x)$$

$\Rightarrow E = -\frac{1}{N} \sum_{i=1}^{N} (1-y^i) \log(1 - \sigma_w(x_i)) + y^i \log(\sigma_w(x_i))$, which is same as the expression in the slides for cross entropy used to train a binary logistic regression

b) Let $Z = XW \Rightarrow \frac{\partial Z}{\partial W} = \bullet X^T$

$$\frac{\partial E}{\partial W} = \frac{\partial Z}{\partial W} \cdot \frac{\partial E}{\partial Z}$$

$$E = -\frac{1}{N} \sum_{i=1}^{N} \sum_{K=1}^{K} y_k^i \log(P(Y=K \mid w_K, x^i))$$

$$= -\frac{1}{N} \sum_{i=1}^{N} \sum_{K=1}^{K} y_k^i \log\left(\frac{e^{w_K^T x^i}}{\sum_{K=1}^{K} e^{w_K^T x^i}}\right)$$

$$= -\frac{1}{N} \sum_{i=1}^{N} \sum_{K=1}^{K} y_k^i \left(w_K^T x_i - \log\left(\sum_{K=1}^{K} e^{w_K^T x_i}\right)\right)$$

$$= -\frac{1}{N} \sum_{i=1}^{N} \sum_{K=1}^{K} y_K^i \left(z_{iK} - \log \left(\sum_{K=1}^{K} e^{z_{iK}} \right) \right)$$

$$\frac{\partial E}{\partial Z} = \frac{\partial}{\partial Z} \frac{-1}{N} \sum_{i=1}^{N} \sum_{K=1}^{K} y_K^i \left(z_{iK} - \log \left(\sum_{K=1}^{K} e^{z_{iK}} \right) \right)$$

$$= -\frac{1}{N} \sum_{i=1}^{N} \sum_{K=1}^{K} \frac{\partial y_K^i \cdot z_{iK}}{\partial Z} + \left(\frac{1}{N} \sum_{i=1}^{N} \sum_{K=1}^{K} \frac{\partial}{\partial Z} y_K^i \log \left(\sum_{K=1}^{K} e^{z_{iK}} \right) \right)$$

$-\frac{1}{N} Y$ (By applying first principle)

$$\frac{1}{N} \sum_{i=1}^{N} \frac{\partial}{\partial Z} \log \left(\sum_{K=1}^{K} e^{z_{iK}} \right)_K \quad (\because y_K^i = 1 \text{ for only one } K)$$

Now, by applying first principle, we get the softmax matrix S, where, $S_{iK} = \frac{e^{z_{iK}}}{\sum_{K=1}^{K} e^{z_{iK}}}$

$$\therefore \frac{\partial E}{\partial Z} = -\frac{1}{N} Y + \frac{1}{N} S$$

$$\frac{\partial E}{\partial W} = \frac{\partial Z}{\partial W} \cdot \frac{\partial E}{\partial Z} \quad \text{(By chain rule)}$$

$$= X^T \left(-\frac{1}{N} Y + \frac{1}{N} S \right)$$

2.2 (b)

The test accuracy obtained for binary logistic regression is 0.8632707774798928.

The test accuracy for model M is 0.8418230563002681.

I don’t think accuracy is a good measure here, because, even a zero function has accuracy close to binary logistic regression. It maybe happening because, the test data set has more zeros as outputs, so, even if we predict zero for every test case, we acheive good accuracy.
So, for cases like these, where one class is dominant, accuracy is not a good measure.

2.2 (c)
F1 score for binary logistic regression is 0.3013698630136986.
F1 score for model M is 0.
F1 score is a good evaluation metric here, since we need to penalise false negatives more(which is done by using recall) as positive class is dominated by the negative class in the dataset. So, already there are less instances of positives and if we predict even them as negatives, our model is not reliable enough. Also, after training on such a dominant negative class dataset, we should be able to predict them correctly, so, even false positives are to be penalised highly(which is done by using precision).
So, for cases like these, where one class is dominant, F1 score is a good measure.

2.2 (e)
Logistic regression achieves more test accuracy. Beacuse, while training a perceptron, we find any separating hyperplane, not the best separating hyperplane. But while training a logistic regression model, we not only find a way to classify the train data, but also to best fit the data. This is because of the difference in loss functions. For perceptron, we used hinge loss, which cares only about finding a hyperplane for separating the data. While the cross entropy function we use for logistic regression, not only tries to maximise the probability for correct classification, it also tries to minimise the probability for incorrect classification. Hence, we get better test accuracy with logistic regression than perceptron.
Also consider a rare example where, there is noise in the input of the training data . Now if a perceptron just classifies the data(i.e. any one point lies very near to the separating hyper plane and the plane is not the best separating plane) and the test data is given from another source, which has different output for same input(the point very near to the hyperplane), which was in train data. Now, there is more chance for a logistic regression model to classify this correctly as it provides more “breathing space” than the perceptron we used. Note that if the perceptron has any guarantee, that it finds the best separating plane, like logistic regression, this case would not arise.