1. The process of sampling from a discrete distribution with 2 outcomes was described on Page 24 of Lecture 2. Describe the sampling algorithm that is able to sample from a discrete distribution with K outcomes. Write a Python program to to generate 100 samples from the following distribution:

P(X = 1) = 0.7

P(X = 2) = 0.2

P(X = 3) = 0.1

Are the actual outcomes close in number to what the distribution predicts?

Multinomial distribution is the algorithm used to sample a discrete distribution with k outcomes. It is also called as categorical distribution.

- x in {1, 2, 3, …, K}

It is a generalization of the Bernoulli distribution from a binary variable to a categorical variable, where the number of cases *K* for the Bernoulli distribution is set to 2, K=*2*.

A multinomial distribution is summarized by a discrete random variable with *K* outcomes, a probability for each outcome from *p1* to *pK*, each defining the probability of a given categorical outcome from 1 to *K*, and where all probabilities sum to 1.0.

- P(x=1) = p1
- P(x=2) = p1
- P(x=3) = p3
- …
- P(x=K) = pK

The repetition of multiple independent Multinoulli trials will follow a multinomial distribution. As we have 100 samples with K outcomes, we use this distribution.

Python Code:

```
from numpy.random import multinomial
# defining the parameters of the distribution
p = [0.7, 0.2, 0.1]
k = 100
# run a single simulation
cases = multinomial(k, p)
# summarize cases
for i in range(len(cases)):
    print('Case %d: %d' % (i+1, cases[i]))
```

```
Case 1: 72
Case 2: 16
Case 3: 12
```

For the distribution we have mentioned,

P(X = 1) = 0.7

P(X = 2) = 0.2

P(X = 3) = 0.1

We would expect first category to have about 70 events, second category to have about 20 events and third category to have about 10 events.

A different random sequence of 100 trials result each time the code is run, our specific results does differ. But the probability of first category remains higher than second followed by the third.

2. How are deep learning systems different from older machine learning systems?

Machine Learning is a method of statistical **learning** where each instance in a dataset is described by a set of features or attributes. In contrast, the term “**Deep Learning**” is a method of statistical **learning** that extracts features or attributes from raw data.

The key **difference** between **deep learning** vs **machine learning** stems from the way data is presented to the system. **Machine learning** algorithms almost always require structured data. It requires that we supply a good representation for the data. Whereas

deep learning networks rely on layers of the ANN (artificial **neural networks**), it creates higher level representations of data as part of the learning process.

Example: To categorize the images as dogs or cats,

In the case of **Machine learning,** we need to provide structured data. We label the pictures of dogs and cats in a way which will define specific features of both the animals. This data will be enough for the machine learning algorithm to learn, and then it will continue working based on the labels that it understood, and classify millions of other pictures of both animals as per the features it learned through the said labels.

In case of **Deep Learning**, they do not necessarily need structured/labeled data of the pictures to classify the two animals. The artificial neural networks using deep learning send the input (the data of images) through different layers of the network, with each network hierarchically defining specific features of images**.** After the data is processed through layers within deep neural networks, the system finds the appropriate identifiers for classifying both animals from their images.

3. There are 4 balls in an urn. Each ball is either red or black. You start by believing that the probabilities that the urn contains 0, 1, 2, 3, 4 red balls are all equal. You then reach into the urn and pull out a ball at random. It is red. Compute the new probabilities that the urn contains 0, 1, 2, 3 or 4 red balls. Hint: Use the Law of Conditional Probabilities

P(#r=i) = ⅕ {probability of any number of red ball is equal}

P(ball=r) = ½ {probability of each ball being red}

Using the formula,

P(A|B)*P(B)=P(B|A)*P(A)

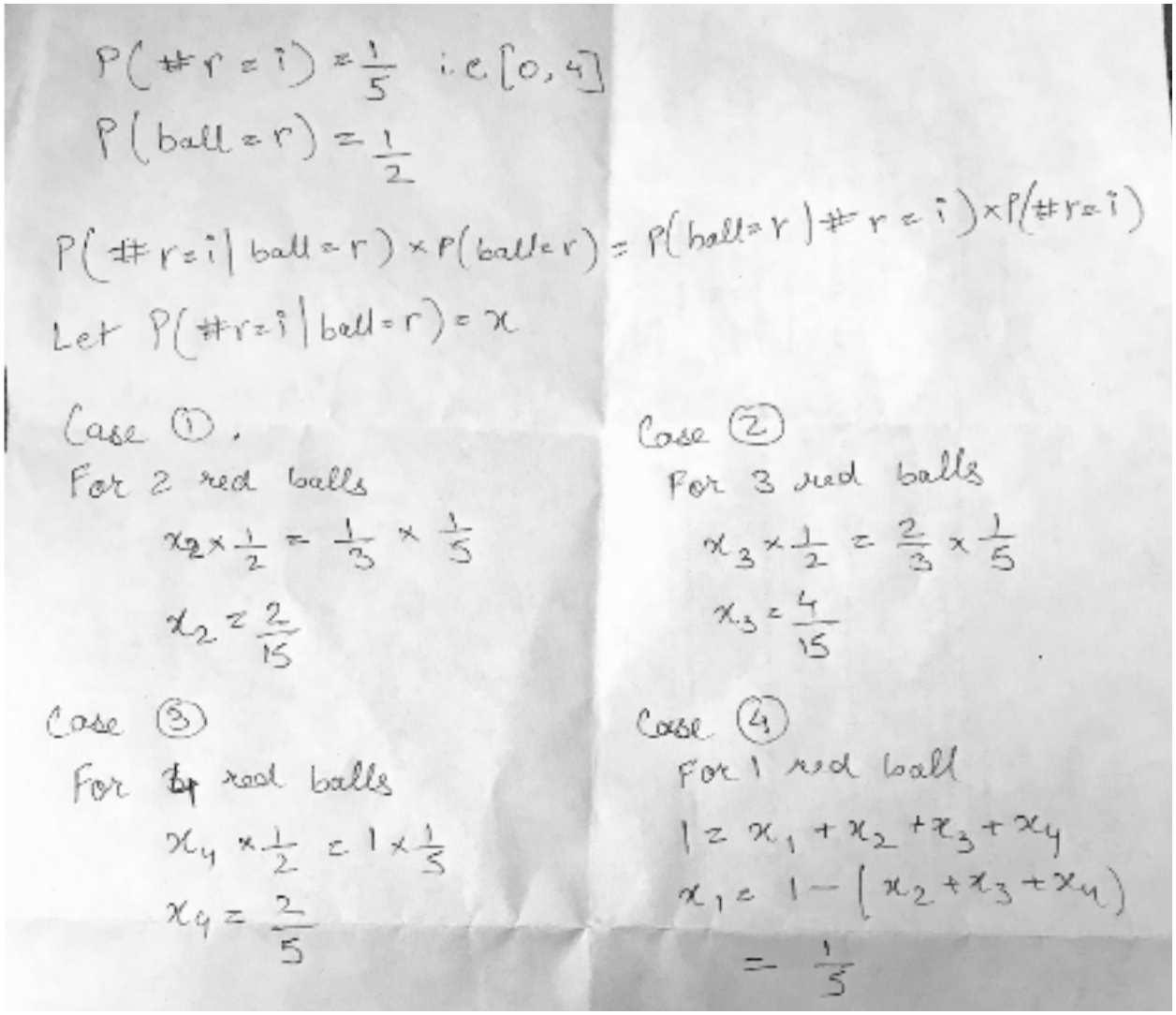
Where,

P(A|B)= probability of no.of red ball given first ball is red

P(B)= probability of ball being red

P(B|A)= probability of red ball given no.of balls

P(A)= probability of no.of red balls



4. a)

Q4. a) $y = \frac{1}{1+\exp^{(-x)}} \Rightarrow \frac{1}{1+e^{-x}}$

$$\frac{dy}{dx} = \left(\frac{1}{1+e^{-x}}\right)^2 \frac{d}{dx}(1+e^{-x})$$

$$= \left(\frac{1}{1+e^{-x}}\right)^2 e^{-x}(-1)$$

$$= \left(\frac{1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right)(-e^{-x})$$

$$= \left(\frac{1}{1+e^{-x}}\right)\left(\frac{-e^{-x}}{1+e^{-x}}\right)$$

$$\boxed{\frac{dy}{dx} = y(1-y)} \Rightarrow \text{Sigmoid function.}$$

4. b)

Ans. (b) $y_k = \dfrac{\exp(a_k)}{\sum_{i=1}^{k} \exp(a_i)} \Rightarrow \dfrac{e(a_k)}{\sum_{i=1}^{k} e(a_i)}$

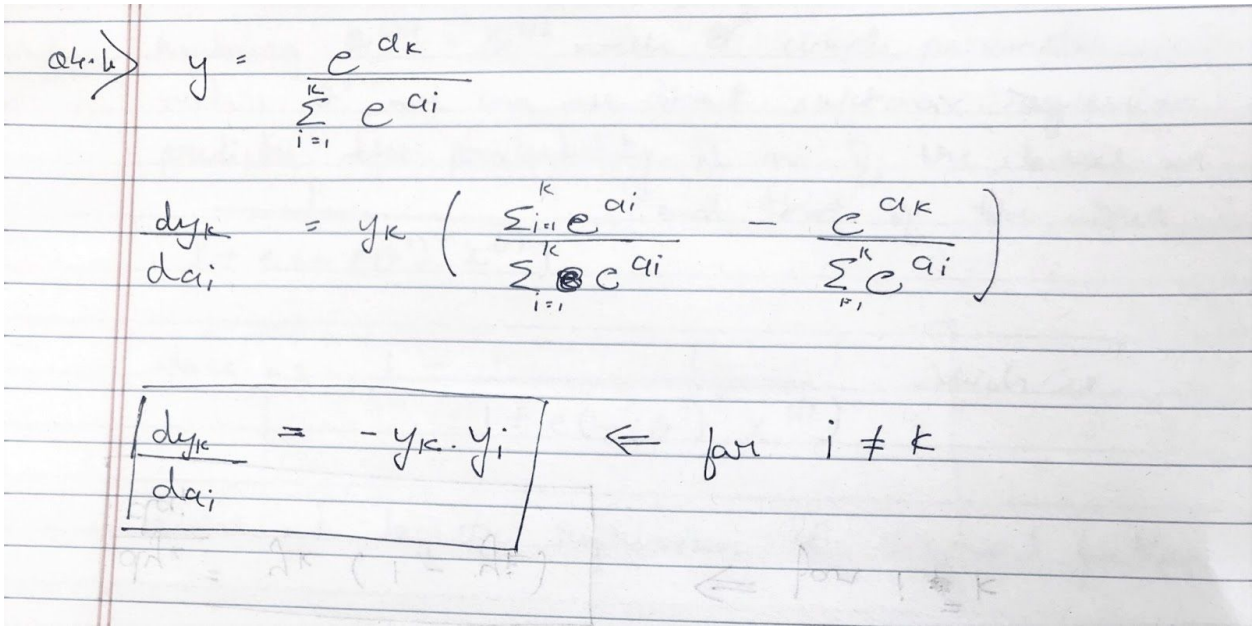
$$\frac{dy_k}{da_i} = \frac{e^{a_k} \sum_{i=1}^{k} e^{a_i} - e^k e^{a_k}}{\left(\sum_{i=1}^{k} e^{a_i}\right)^2}$$

$$= \frac{e^{a_k}\left(\sum_{i=1}^{k} e^{a_i} - e^k\right)}{\left(\sum_{i=1}^{k} e^{a_i}\right)^2}$$

$$= \frac{e^{a_k}}{\sum_{i=1}^{k} e^{a_i}} \times \frac{\left(\sum_{i=1}^{k} e^{a_i} - e^{a_k}\right)}{\sum_{i=1}^{k} e^{a_i}}$$

$$= y_k \left(\frac{\sum_{i=1}^{k} e^{a_i}}{\sum_{i=1}^{k} e^{a_i}} - \frac{e^{a_k}}{\sum_{i=1}^{k} e^{a_i}} \right)$$

$$\boxed{\frac{dy_k}{da_i} = y_k (1 - y_k)} \Leftarrow \text{for } i = k$$


Q4.b)
$y = \frac{e^{a_k}}{\sum_{i=1}^{k} e^{a_i}}$
$\frac{dy_k}{da_i} = y_k \left(\frac{\sum_{i=1}^{k} e^{a_i}}{\sum_{i=1}^{k} e^{a_i}} - \frac{e^{a_k}}{\sum_{i=1}^{k} e^{a_i}} \right)$
$\frac{dy_k}{da_i} = -y_k \cdot y_i \Leftarrow$ for $i \neq k$

4. c)

w.k.t $\Rightarrow \quad L = -\sum_{k=1}^{K} t_k \log y_k$

$$y_k = \frac{e^{(a_k)}}{\sum_{i=1}^{k} e^{(a_i)}}$$

$$L = -\frac{1}{M} \sum_{m=1}^{M} \sum_{k=1}^{K} t_k(m) \log y_k(m)$$

$$= \frac{1}{M} \sum_{m=1}^{M} L(m)$$

where $L(m) = -\sum_{k=1}^{K} t_k(m) \log y_k(m)$

$$\frac{dL}{dw_{ki}} = \frac{dL}{da_k} \cdot \frac{da_k}{dw_{ki}} \quad \text{— (1)}$$

$$\Rightarrow \frac{dL}{da_k} = \sum_{j=1}^{K} \frac{dL}{dy_j} \frac{dy_j}{da_k} \quad \text{— (2)}$$

$$\Rightarrow \frac{da_k}{dw_{ki}} = x_i \quad \text{— (3)}$$

Once the form of softmax equation

$$\frac{dL}{dy_j} = -\frac{t_j}{y_j} \quad \text{— (4)}$$

From Q4. b) we know

$$\frac{dy_j}{da_j} = y_j(1 - y_j) \quad \text{— (5)}$$

and that $\frac{dy_j}{da_k} = -y_j y_k$ for $j \neq k$ — (6)

Substituting (4), (5), (6) in (2)

$$\frac{dL}{da_k} = \frac{-t_k}{y_k} y_k(1 - y_k) - \sum_{j \neq k} \frac{t_k}{y_j}(-y_j y_k)$$

$$= -(t_k - t_k y_k - \sum_{j \neq k} t_j y_k)$$

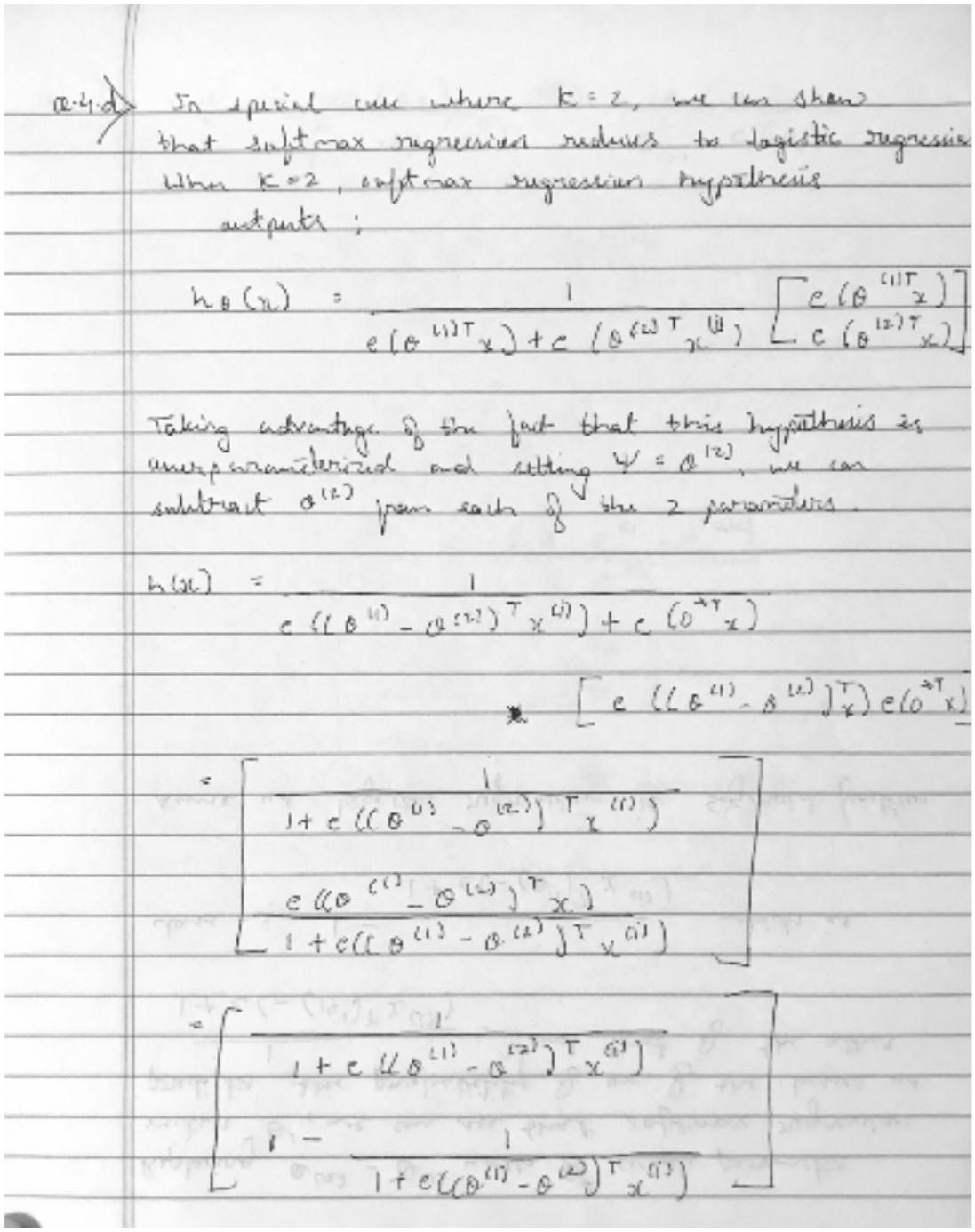
$$= -(t_k - \sum_{j=1}^{k} t_j y_k)$$

$$\boxed{\frac{dL}{da_k} = y_k - t_k} \quad \text{— (7)}$$

Substituting (7), (3) in (1)

$$\boxed{\frac{dL}{dw_{kj}} = x_j(y_k - t_k)}$$

4. d)



Q-4-d) In special case where $K = 2$, we can show that softmax regression reduces to logistic regression. When $K = 2$, softmax regression hypothesis outputs;

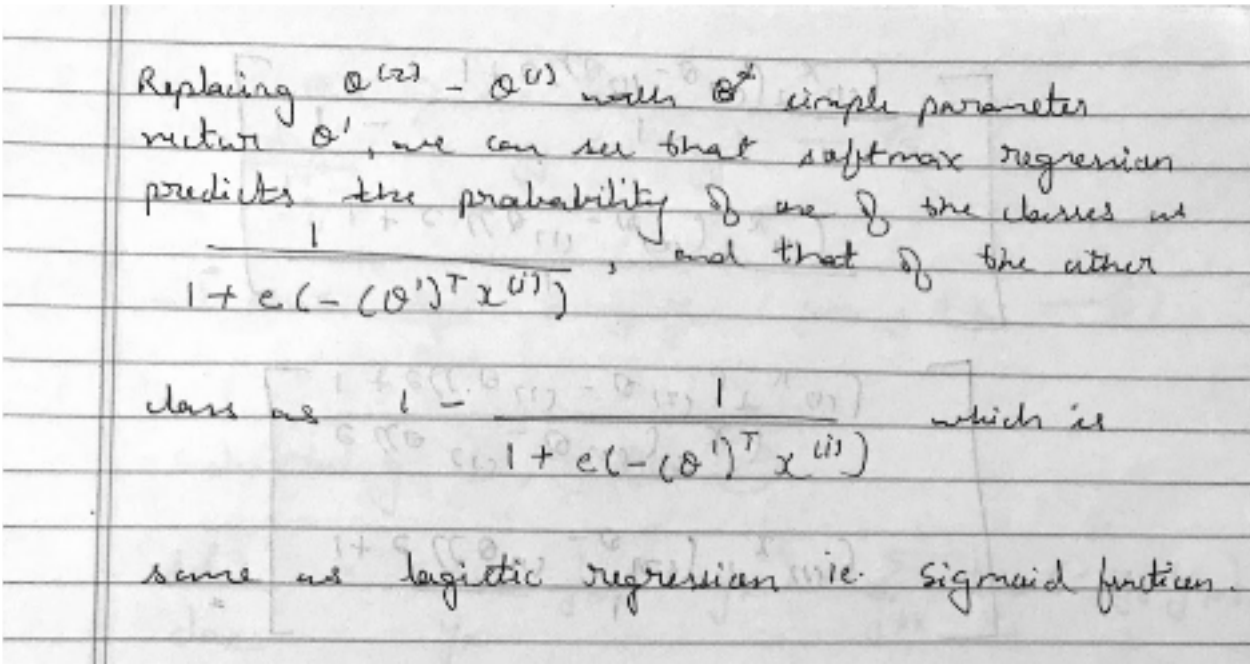
$$h_\theta(x) = \frac{1}{e^{(\theta^{(1)T}x)} + e^{(\theta^{(2)T}x^{(i)})}} \begin{bmatrix} e^{(\theta^{(1)T}x)} \\ e^{(\theta^{(2)T}x)} \end{bmatrix}$$

Taking advantage of the fact that this hypothesis is overparameterized and setting $\psi = \theta^{(2)}$, we can subtract $\theta^{(2)}$ from each of the 2 parameters.

$$h(x) = \frac{1}{e^{((\theta^{(1)} - \theta^{(2)})^T x^{(i)})} + e^{(0^T x)}} * \left[e^{((\theta^{(1)} - \theta^{(2)})^T x)} \; e^{(0^T x)} \right]$$

$$= \begin{bmatrix} \frac{1}{1 + e^{((\theta^{(1)} - \theta^{(2)})^T x^{(i)})}} \\ \frac{e^{((\theta^{(1)} - \theta^{(2)})^T x)}}{1 + e^{((\theta^{(1)} - \theta^{(2)})^T x^{(i)})}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{1 + e^{((\theta^{(1)} - \theta^{(2)})^T x^{(i)})}} \\ 1 - \frac{1}{1 + e^{((\theta^{(1)} - \theta^{(2)})^T x^{(i)})}} \end{bmatrix}$$



5. Consider the Linear Model with K = 2, as described on Page 20 of Lecture 3. Suppose that the Cross Entropy Loss Function was replaced by the Mean Square Error (MSE) Loss Function (see Lecture 3, Page 10 for a definition of MSE Loss Function).

Replacing the loss function in slide 20 with MSE loss function

(a) Compute the gradient for the MSE Loss Function. Hint: Follow the steps shown on Lecture 3, Page 25

Mean Square Error (MSE)

$$\mathcal{L}(s) = \frac{1}{K} \sum_{k=1}^{K} [y_k(s) - t_k(s)]^2$$

y_k = output Neural Network should give

t_k = Input

Using Chain Rule of Derivatives:

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial a} \frac{\partial a}{\partial w_i}$$

$$\mathcal{L} = \frac{1}{K} \sum_{k=1}^{K} [y_k - t_k]^2$$

$$y = \frac{1}{1+e^{-a}} \Rightarrow \frac{dy}{da} = y(1-y)$$

$$a = \sum_{i=1}^{n} w_i x_i + b \Rightarrow \frac{da}{dw_i} = x_i$$

$$\frac{\partial L}{\partial y} = \frac{1}{K} \sum_{k=1}^{K} 2(y_k - t_k) \frac{\partial}{\partial y}(y_k - t_k)$$

$$= \frac{1}{K} \sum_{k=1}^{K} 2(y_k - t_k) \times \left(\frac{\partial}{\partial y} y_k - \frac{\partial}{\partial y} t_k \right)$$

$$\frac{\partial L}{\partial y} = \frac{1}{K} \sum_{k=1}^{K} 2(y_k - t_k)$$

$$\frac{\partial L}{\partial w_i} = \frac{1}{K} \sum_{k=1}^{K} 2(y_k - t_k) \times y(1-y) \times x_i$$

(b) Compare the expressions derived in Part (a) for the MSE gradient, with the expression derived for the gradient of the Cross Entropy Loss Function (on Page 25 of Lecture 3). Based on this, explain why the Cross Entropy Loss Function is better than the MSE Loss Function for doing Classification.

Expression derived using Cross Entropy Loss function:

$$\frac{\partial L}{\partial w_i} = (y - t) x_i$$

Expression derived using MSE Loss function:

$$\frac{\partial L}{\partial w_i} = \frac{1}{K} \sum_{k=1}^{K} 2(y_k - t_k) \times y(1-y) \times x_i$$

Gradient is used to Cross-entropy is prefered for **classification**, while mean squared error is one of the best choices for **regression**. This comes directly from the statement of the problem itself - in classification we work with very particular set of possible output values where Cross Entropy is used. MSE is used when we assume the error follows Normal Distribution and cross entropy when we assume binomial distribution.

When we perform logistic regression for example, we will use the sigmoid function to estimate the probability, the cross entropy as the loss function and gradient descent to minimize it. Doing the same using MSE as loss function would lead to a non-convex problem where we find local minima. Only when we use cross entropy, it will lead to a convex problem where we can find the optimal solution.