Question 1 - What is the optimal Bayes test to minimize the probability of error

We know that the optimal test according to Bayes is the the likelihood ratio:

$$\frac{f_1(X)}{f_0(X)} \underset{H_0}{\overset{H_1}{\gtrless}} \frac{P(H_0)}{P(H_1)} \Leftrightarrow \frac{f_1(X)}{f_0(X)} \underset{H_0}{\overset{H_1}{\gtrless}} 1 \quad \textbf{(1)}$$, since it is given that $P(H_0) = P(H_1) = 0.5$

It is also given that $f_0(X) = f_0(x_1) \cdot f_0(x_2)$ and $f_1(X) = f_1(x_1) \cdot f_1(x_2)$, because x_1 and x_2 are independent. Thus, for H_0 it is $f_0(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$ (2) and similarly for H_1 it is $f_1(x) = 0.5\left(\frac{1}{\sqrt{2\pi}} e^{-\frac{(x-1)^2}{2}} + \frac{1}{\sqrt{2\pi}} e^{-\frac{(x+1)^2}{2}}\right)$ (3), using the Probability Density Function (PDF) of the normal distribution $\mathcal{N}(\mu, \sigma^2)$:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

We can log both sides of the inequation (1), since the function $\omega(x)=\ln(x)$ is monotonically increasing, so that we derive the following equivalent optimal test:

$$\ln\left(\frac{f_1(x_1) \cdot f_1(x_1)}{f_0(x_1) \cdot f_0(x_1)}\right) \underset{H_0}{\overset{H_1}{\gtrless}} 0$$

We then proceed doing simplifications using the logarithm properties.

$$\ln\left(\frac{f_1(x_1) \cdot f_1(x_2)}{f_0(x_1) \cdot f_0(x_2)}\right) \Leftrightarrow ln\left(\frac{f_1(x_1)}{f_0(x_1)}\right) + \ln\left(\frac{f_1(x_2)}{f_0(x_2)}\right) \Leftrightarrow \ln(f_1(x_1)) - \ln(f_0(x_1)) + \ln(f_1(x_2)) - \ln(f_0(x_2)) \overset{(2),(3)}{\Longleftrightarrow}$$
$$ln\left(0.5\left(e^{-\frac{(x_1-1)^2}{2}} + e^{-\frac{(x_1+1)^2}{2}}\right)\right) - ln\left(e^{-\frac{x_1^2}{2}}\right) + ln\left(0.5\left(e^{-\frac{(x_2-1)^2}{2}} + e^{-\frac{(x_2+1)^2}{2}}\right)\right) - ln\left(e^{-\frac{x_2^2}{2}}\right) \quad (4)$$

where the constant terms $\frac{1}{\sqrt{2\pi}}$ are simplified from both numerator and denumerator, before "breaking" the log.
So,

$$(4) \Leftrightarrow \boxed{\ln(0.5) + \ln\left(e^{-\frac{(x_1-1)^2}{2}} + e^{-\frac{(x_1+1)^2}{2}}\right) + \frac{x_1^2}{2} + \ln(0.5) + \ln\left(e^{-\frac{(x_2-1)^2}{2}} + e^{-\frac{(x_2+1)^2}{2}}\right) + \frac{x_2^2}{2}} \quad \textbf{(5)}$$

We can also define a function $h(x) = ln(0.5) + ln(e^{-\frac{(x-1)^2}{2}} + e^{-\frac{(x+1)^2}{2}}) + \frac{x^2}{2}$, so that finally (5) can be written as $h(x_1) + h(x_2)$ and the optimal Bayes test as:

$$h(x_1) + h(x_2) \underset{H_0}{\overset{H_1}{\gtrless}} 0,$$

Questions 2 & 3 - Calculate error using simulations

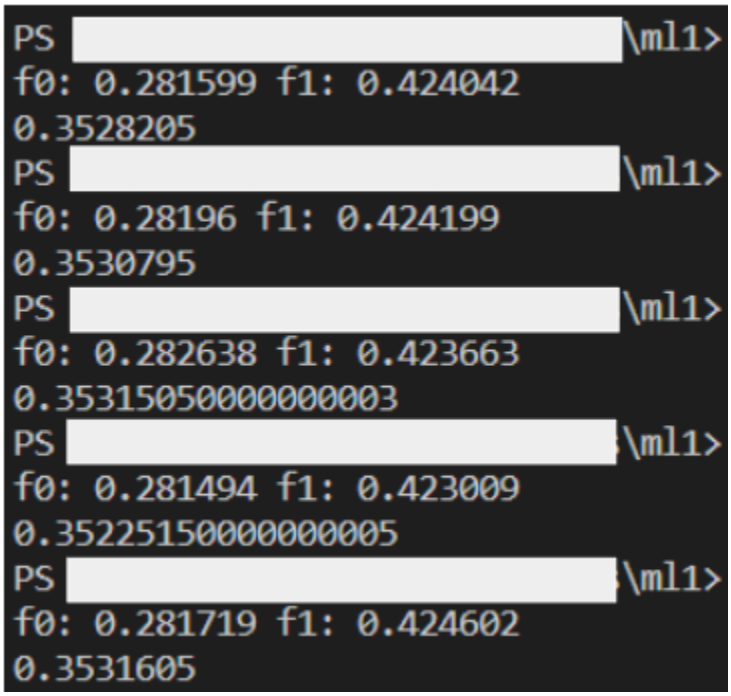
Given a vector X=[x_1,x_2], it suffices to calculate h(x_1) + h(x_2). If this is quantity positive (>0), iit is decided that the data results from hypothesis H1, if the result is negative (<0), it is concluded that the data results from the hypothesis H0, while finally if it is equal to 0 we rank X either under hypothesis H0 or under H1 with probability 0.5.

In order to implement the above, in this question we use simulations: we manufacture 10^6 data (pairs [x_1,x_2]) using the distribution under H_0 and as much under hypothesis H_1. We then calculate h(x_1) + h(x_2) as mentioned in question 1 and count the number of errors, i.e. those cases in which a data pair under H_0 had a positive result (f_{0_errors}) and those in which a data pair under H_1 had a negative result (f_{1_errors}). Finally, it suffices to calculate the total error as

$$\varepsilon = 0.5\left(\frac{f_{0-errors}}{K} + \frac{f_{1-errors}}{K}\right),\ where\ K = 10^6\ is\ the\ total\ number\ of\ data.$$

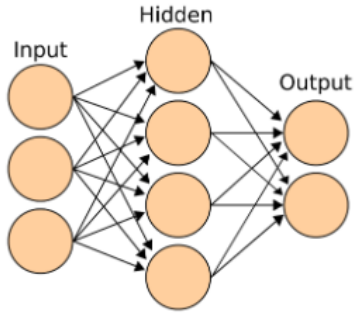
Specifically, the programming implementation was performed using the programming language python (version 3.8) for all questions. The numpy library was also used which allows the fast execution of table operations. This library also contains probability elements, allowing the use of normal distributions etc. The aforementioned function h for example, performs operations with arrays directly and not element by element.

Running the program (ex1.py) prints f_{0_errors}, f_{1_errors} and the overall optimal error to the screen, which amounts to about 35%. The program was executed several times and each of them the Bayes error differs to the 3rd decimal place.



Question 4 - Classification using Neural Network (NN)

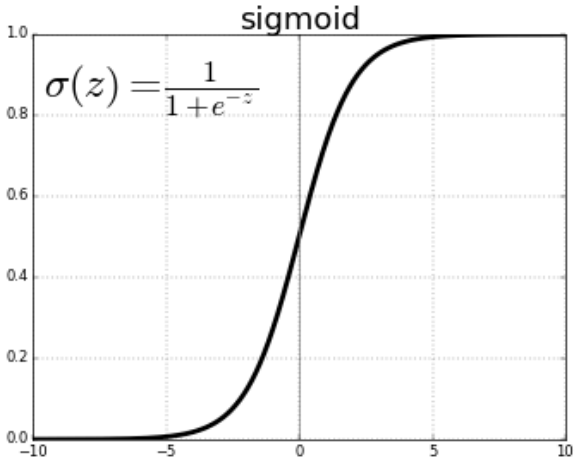
In this question we will repeat the previous exercise, but using a neural network of size 2x20x1, which means the network accepts two inputs: x_1, x_2, has a hidden layer with 20 neurons and finally has a single output (corresponding to the categorization of the input as derived under hypothesis H_0 or H_1). **This happens, as according to theory, given a suitable neural network we can approximate any function.** The following neural network is indicative and does not correspond to the actual dimensions of the exercise network.

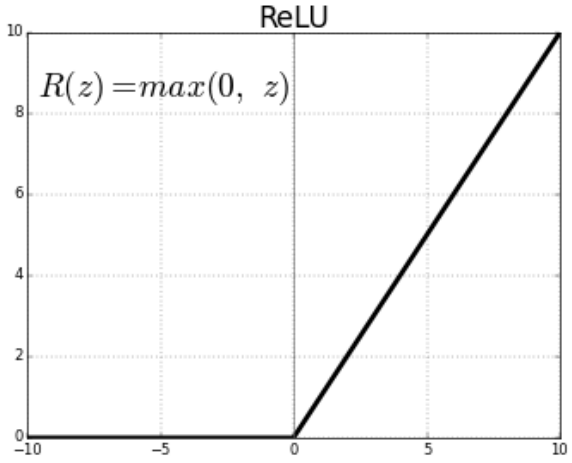


The feed forward process is quite simple and consists of operations upon tables. The implementation has adopted the following notation for the arrays (the exponent of each array signifies its dimensions).

$z_1^{20x1} = W_1^{20x2} \cdot x^{2x1} + b_1^{20x1}$	z_1 is the output of the first layer, W_1 its weights and b_1 its bias.
$a_1^{20x1} = max(0, z_1^{20x1})$	a_1 is the output after applying the ReLU activation function.
$z_2^{1x1} = W_2^{1x20} \cdot a_1^{20x1} + b_2^{1x1}$	z_2 is the output of the second layer, W_2 its weights and b_2 its bias.

There are made two different test cases, using either the cross entropy loss function (in which case a sigmoid function is used as activation, so that final output $\hat{y} = \frac{1}{1+e^{-z_2}}$) or the exponential loss function (in which case $\hat{y} = z_2$). The $\hat{y}$ notation signifies the prediction (predicted outcome) of the neural network.





Of greater interest is neural network training (a process known as backpropagation). According to this process, the parameters of the network (weights and biases) change according to the corresponding partial derivatives of the cost function. Let us consider the case of the cross-entropy method, as a similar process is performed in the exponential method.
If the class of the input is indeed H_0, then the cost function is $\phi(\hat{y}) = -\ln(1-\hat{y})$, while on the contrary if it is under H_1 the cost is $\psi(\hat{y}) = -\ln(\hat{y})$. Here are the partial derivatives with respect to all of the the variables, which are used to improve the θ parameters of the network

$$Y = 0$$

$$L = -\log(1-\hat{y}_0)$$

$$\frac{\partial L}{\partial \hat{y}_0} = \frac{1}{1-\hat{y}_0}$$

$$\frac{\partial L}{\partial z_2} = \frac{\partial L}{\partial \hat{y}_0} * \frac{\partial \hat{y}_0}{\partial z_2} = \hat{y}_0$$

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial z_2} * \frac{\partial z_2}{\partial W_2} = \hat{y}_0 * \alpha_1$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial z_2} * \frac{\partial z_2}{\partial b_2} = \hat{y}_0$$

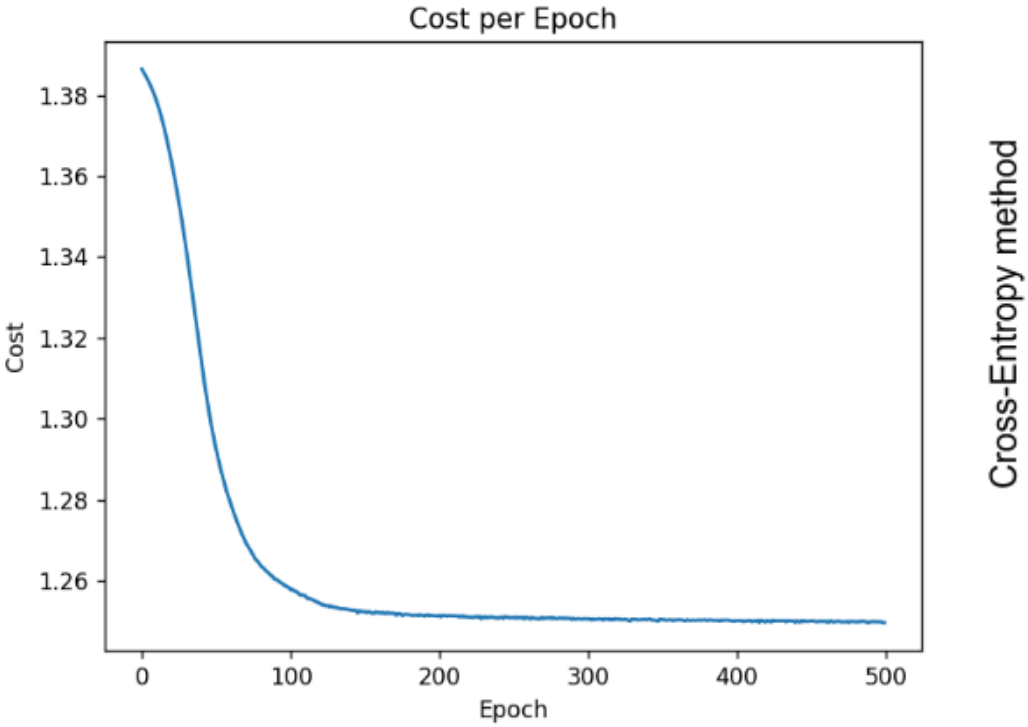
$$\frac{\partial L}{\partial \alpha_1} = \frac{\partial L}{\partial z_2} * \frac{\partial z_2}{\partial \alpha_1} = \hat{y}_0 * W_2$$

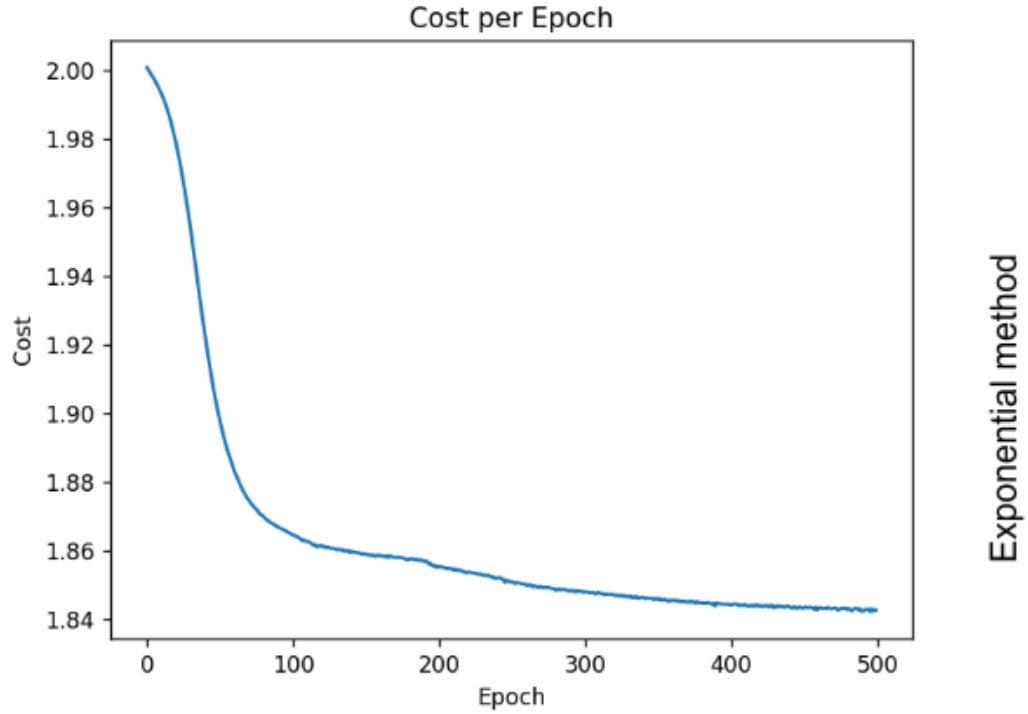
$$\frac{\partial L}{\partial z_1} = \hat{y}_0 * W_2 * (z_1 > 0)$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial z_1} * \frac{\partial z_1}{\partial b_1} = \hat{y}_0 * W_2 * (z_1 > 0)$$

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial z_1} * \frac{\partial z_1}{\partial W_1} = \hat{y}_0 * W_2 * (z_1 > 0) * X$$

The training parameters (learning rate=0.003 and epochs=500) were arbitrarily chosen after some tests. As one epoch we define the passage of all 2*200 data through the network.
Thus, the training process is as follows: At each epoch we initially shuffle the existing data stored in separate structures. Then we first feedforward one data from H_0 hypothesis and one from H_1 and calculate the cost in each case. After each feedforward the algorithm performs automatic backpropagation in order to improve the parameters. The cost of the iteration is the sum of the two errors, while at the end of the season we calculate the average cost of the season.





Finally, we present a comparison of the two methods with each other as well as with the optimum Bayes error.

```
Bays test:
f0: 0.282141 f1: 0.423691
0.352916

cross entropy:
data under H0 misclassified as H1 (e0):  0.358316
data under H1 misclassified as H0 (e1):  0.359674
Total error (e): 0.358995

exponential:
data under H0 misclassified as H1 (e0):  0.13529
data under H1 misclassified as H0 (e1):  0.616679
Total error (e): 0.3759845
```

Question 5 - Classification of handwritten digits.

The process followed for the implementation of a neural network to classify handwritten digits 0 and 8 from the MNIST dataset follows the exact same process as in the previous question, modifying the matrices dimensions as needed.