INT 3405: Machine Learning	**First Semester 2023-2024**
Week name	
Student Name: Le Hoang Lan	*UID: 22028013*

- Feel free to talk to other students in the class when doing the homework. You should, however, write down your solution yourself. You also must indicate on each homework with whom you collaborated and cite any other sources you use including Internet sites.
- You will write your solution in LaTeX and submit the pdf file in zip files, including relevant materials, through courses.uet.vnu.edu.vn
- Dont be late.

1 Homework 1 - 10pts

XOR function.

1. The logistic regression model eventually converges after a high enough number of loops (the loss converges at around 0.69). However, the accuracy of the model is not high - at only around 50% at best. This is because Logistic Regression only gives a linear decision boundary in case if we driectly apply it to the XOR dataset. However, it is unable to find a linear boundary to satisfy all 4 data points of the function - the best boundary can only go through 2 out of the 4 points.

2. The accuracy of training Logistic Regression on sample with 10000, 1000000 points, using learning rate 0.1 and 0.15.

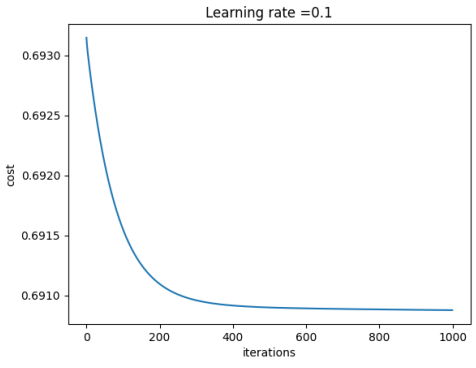

Figure 1: 10000 datapoints, lr = 0.1

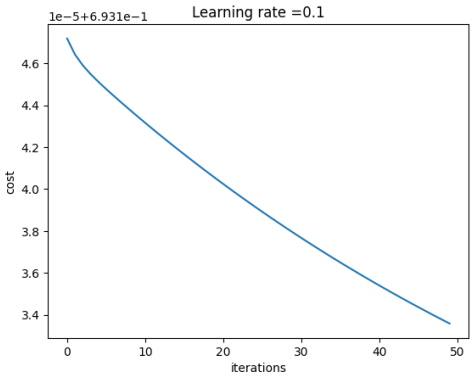

Figure 2: 1000000 datapoints, lr = 0.1

2 Homework 2 - 10 pts

1. Applying Newton's method

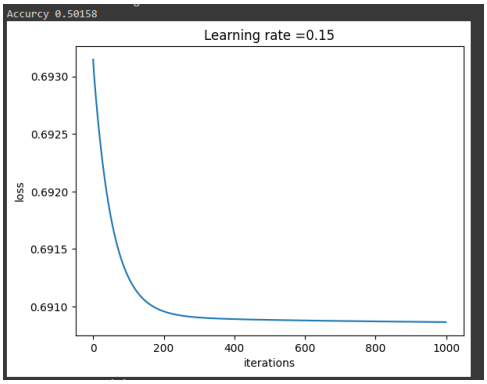

Figure 3: 10000 datapoints, lr = 0.15

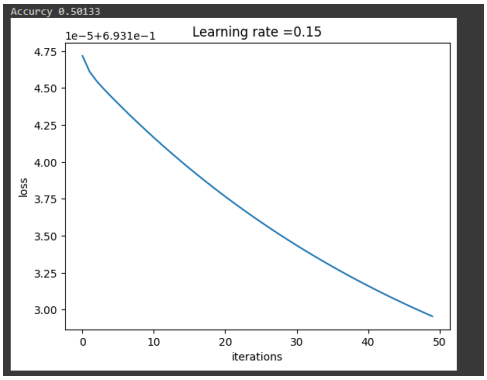

Figure 4: 1000000 datapoints, lr = 0.15

The sigmoid function:

$$g(x) = \frac{1}{1 + e^{(} - z)}, z = w^T x$$

We have the NLL:

$$l(w) = -\frac{1}{N}\sum_{i=1}^{N} y \ln g(x) + (1-y)\ln(1 - g(x))$$

The first-derivative of $l(w)$ with respecto to $w_j (0 \leq j \leq d)$ is:

$$\frac{\partial l(w)}{\partial w_j} = -\frac{1}{N}\sum_{i=1}^{N} y_i \ln g(x_i) + (1-y_i)\ln(1 - g(x_i)) \tag{1}$$

$$= -\frac{1}{N}\sum_{i=1}^{N} y_i . \frac{1}{g(x_i)} . \frac{e^{-z}}{1+e^{-z}} . (z_i)'_{w_j} - (1-y_i) . \frac{1}{1-g(x_i)} . \frac{e^{-z}}{1+e^{-z}} . (z_i)'_{w_j} \tag{2}$$

$$= -\frac{1}{N}\sum_{i=1}^{N} y_i . \frac{1}{g(x_i)} . g(x_i)(1-g(x_i)) . x_{ij} - (1-y_i) . \frac{1}{1-g(x_i)} . g(x_i) . (1-g(x_i)) . x_{ij} \tag{3}$$

$$= -\frac{1}{N}\sum_{i=1}^{N} (y - g(x_i)) x_{ij} \tag{4}$$

$$= \frac{1}{N}\sum_{i=1}^{N} (g(x_i) - y) x_{ij} \tag{5}$$

$$\tag{6}$$

Hence:

$$H_{jk} = \frac{\partial^2 l(w)}{\partial w_j \partial w_k} = \frac{1}{N}\sum_{i=1}^{N} (g(x_i))'_{w_k} x_{ij} \tag{7}$$

$$= \frac{1}{N}\sum_{i=1}^{N} g(x_i)(1-g(x_i)) x_{ij} x_{ik} \tag{8}$$

, with H_{jk} is then entry of the j-th row and k-th column of the Hessian matrix.

The Hessian matrix can also be written as:

$$H = \frac{1}{N} X S X^T$$

, with X is a $d \times N$ matrix, each column of X is x_i^T, $S = diag(g(x)(1 - g(x)))$.

2. Plotting It can be seen that in both dataset, Newton method finds the optimal parameters after only a few iterations, while the Gradient Descent method slowly goes to the minimum loss.

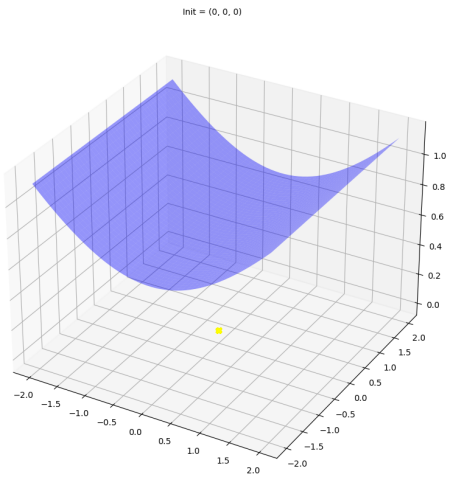

Figure 5: XOR Gradient Descent, $w0, w1$ axis, (0, 0, 0) weights

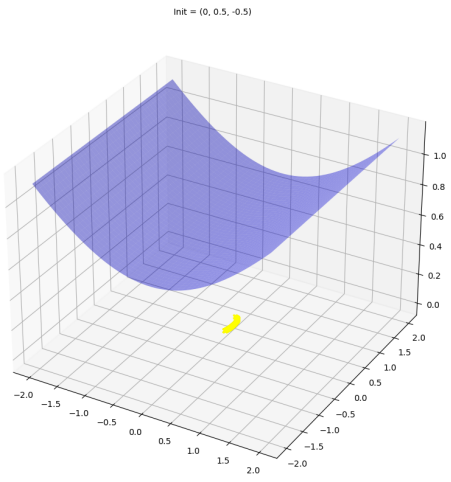

Figure 6: XOR Gradient Descent, $w1, w2$ axis, (0, 0, 0) weights

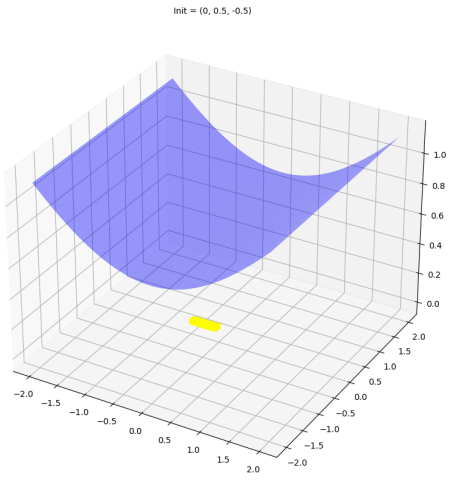

Figure 7: XOR GD, $w0, w1$ axis, (0, 0.5, -0.5) weights

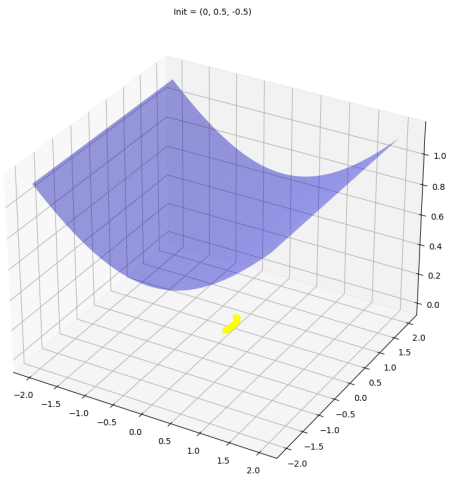

Figure 8: XOR GD $w1, w2$ axis, (0, 0.5, -0.5) weights

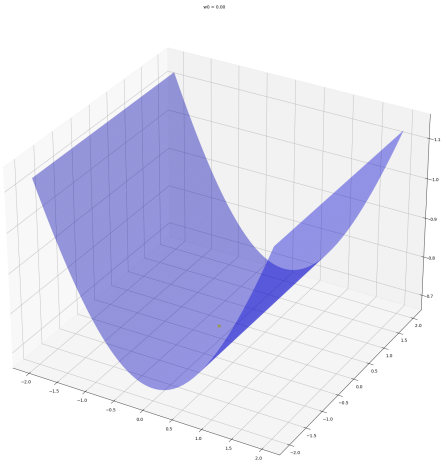

Figure 9: XOR Newton, $w0, w1$ axis, (0, 0, 0) weights

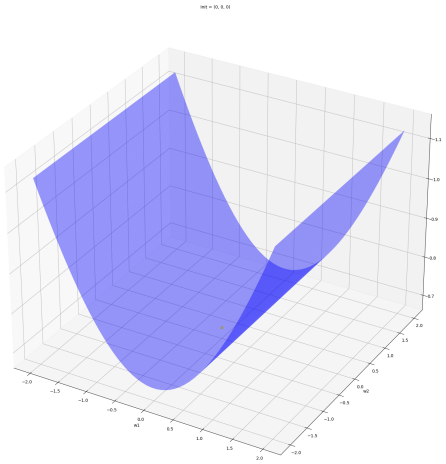

Figure 10: XOR GD $w1, w2$ axis, (0, 0.5, -0.5) weights

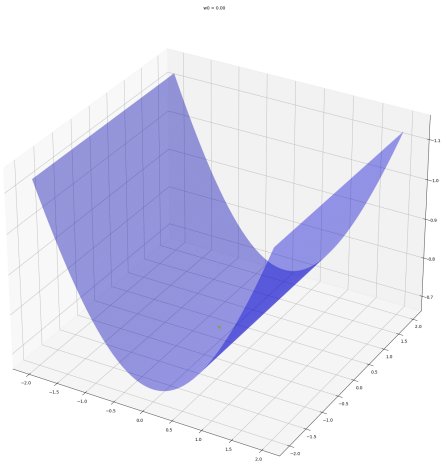

Figure 11: XOR Newton, $w0, w1$ axis, (0, 0, 0) weights

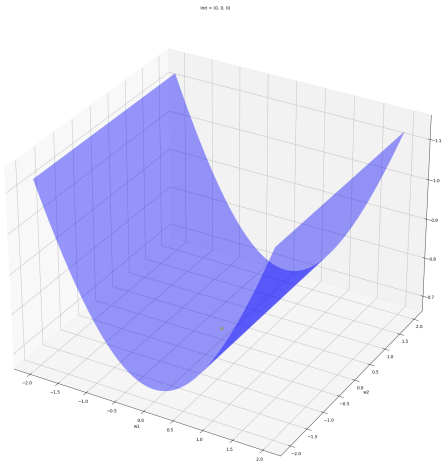

Figure 12: XOR Newton $w1, w2$ axis, (0, 0, 0) weights

Code:

XOR Gradient Descent, XOR Newton's method, AND Gradient Descent, AND Newton's method

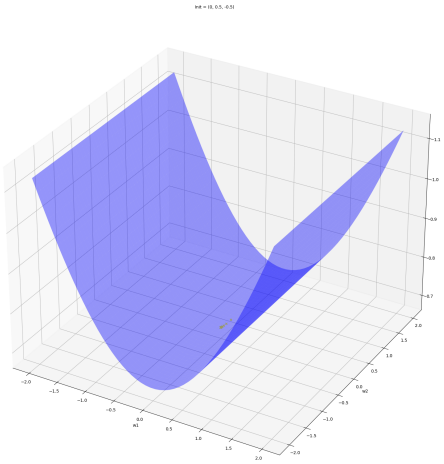

Figure 13: XOR Newton, $w0, w1$ axis, (0, 0.5, -0.5) weights

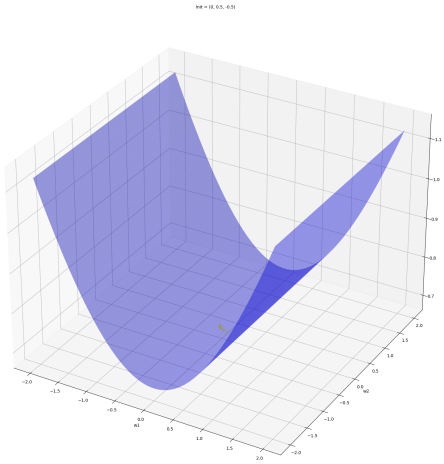

Figure 14: XOR Newton $w1, w2$ axis, (0, 0.5, -0.5) weights

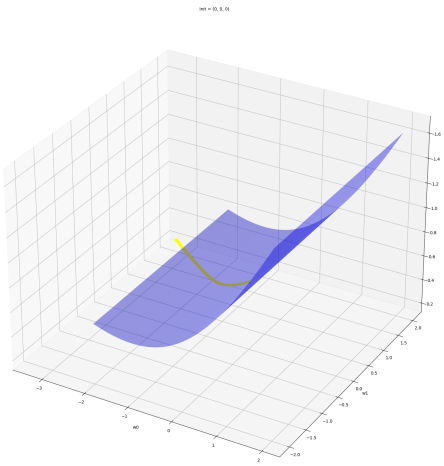

Figure 15: AND GD, $w0, w1$ axis, (0, 0, 0) weights

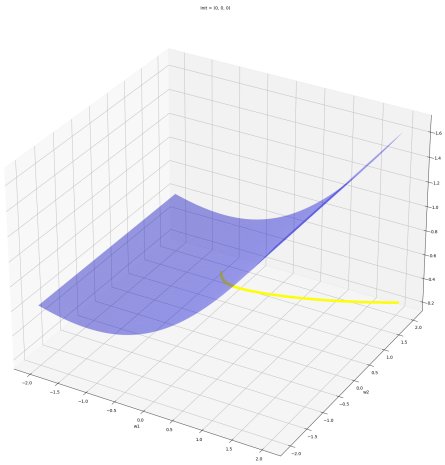

Figure 16: XOR Newton $w1, w2$ axis, (0, 0, 0) weights

3 Homework 3 - 10pts

Multiclass Learning

1. Calculate MLE

   We have the probability of the data D is $P(D)$, the number of x_i equals to k is N_k, and $\sum_{k=1}^{K} \theta_k = 1$. It can be seen that $\sum_{k=1}^{C} N_k = n$.

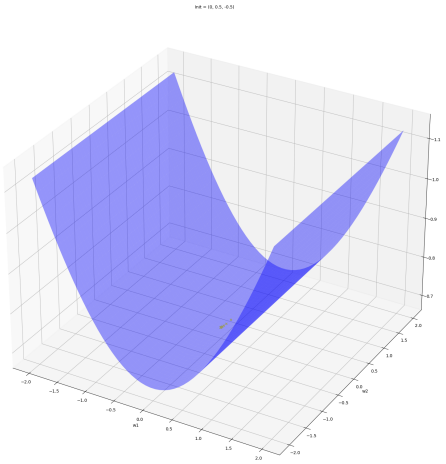

Figure 17: AND GD, $w0, w1$ axis, (0, 0.5, -0.5) weights

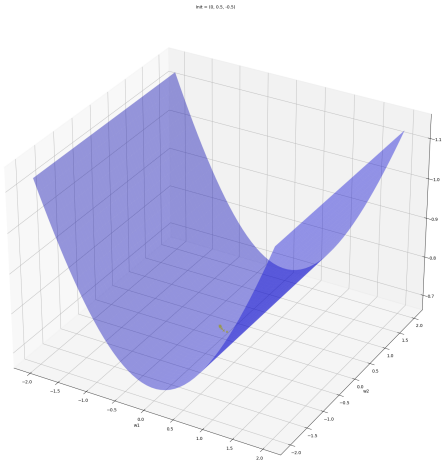

Figure 18: AND GD $w1, w2$ axis, (0, 0.5, -0.5) weights

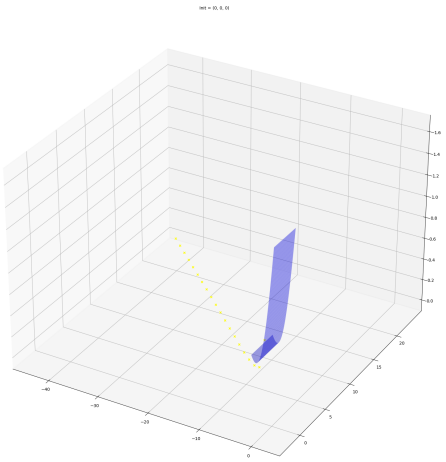

Figure 19: AND Newton, $w0, w1$ axis, (0, 0, 0) weights

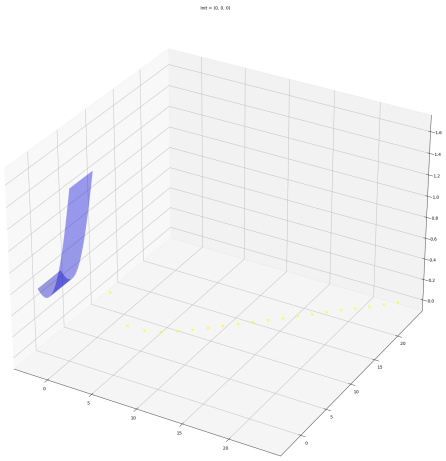

Figure 20: AND Newton $w1, w2$ axis, (0, 0, 0) weights

$$L(\theta) = P(D) = \prod_{i=1}^{n} P(x_i) = \prod_{i=1}^{n} \prod_{k=1}^{C} \theta_k^{[x_i=k]}$$

$$l(\theta) = logL(\theta) = \sum_{i=1}^{n} \sum_{k=1}^{C} log(\theta_k^{[x_i=k]})$$

Let $g(\theta) = \sum_{k=1}^{C} \theta_k - 1$. Using Lagrange multiplier for $l(\theta)$:

$$l(\theta, \lambda) = \sum_{i=1}^{n} \sum_{k=1}^{C} log(\theta_k^{[x_i=k]}) + \lambda g(\theta)$$
$$= \sum_{i=1}^{n} \sum_{k=1}^{K} log(\theta_k^{[x_i=k]}) + \lambda(\sum_{k=1}^{K} \theta_k - 1)$$

Calculating the derivative of each θ_k and λ:

$$\frac{\partial l}{\partial \lambda} = \sum_{k=1}^{K} \theta_k - 1 = 0$$
$$\frac{\partial l}{\partial \theta_k} = \frac{\sum_{i=1}^{n}[x_i = k]}{\theta_k} + \lambda = 0$$
$$\Rightarrow -\frac{\sum_{i=1}^{n}[x_i = k]}{\theta_k} = \lambda$$
$$\Rightarrow -\frac{N_k}{\theta_k} = \lambda$$

We can gather that

$$\frac{N_1}{\theta_1} = \frac{N_2}{\theta_2} = ... = \frac{N_C}{\theta_K} = -\lambda$$
$$\Rightarrow \frac{N_k}{\theta_k} = \frac{\sum_{k=1}^{C} N_k}{\sum_{k=1}^{C} \theta_k} \Rightarrow \frac{N_k}{\theta_k} = \frac{n}{1} \Rightarrow \theta_k = \frac{N_k}{n}$$

Hence, $\theta_k^{MLE} = \frac{N_k}{n}, 1 \leq k \leq C$.

2. Multi-label Logistic Regression Model

Let $w \in R^{(d+1)\times C}, x_i \in R^{(d+1)\times 1}, x_{i0} = 1$,and w_0 is the bias. The probability of a classifier will be:

$$P[Y = c|x] = g_w(x)_i = \frac{e^{w_i^T x}}{\sum_j^C e^{w_j^T x}} = \frac{e^{f_c}}{\sum_{k=1}^{C} e^{f_j}}$$

We have:

$$\arg\max_w P(\{x_i, y_i\}_{i=1}^N | w) = \arg\max_w \prod_{i=1}^{N} \Big(\prod_{k=1}^{C} g_\theta(x)_k^{y_{ik}} \Big) \tag{9}$$
$$= \arg\max_w \sum_{i=1}^{N} \sum_{k=1}^{C} [y_i = k] \ln g_w(x)_k \tag{10}$$
$$= \arg\min_w L(w, X, y) \tag{11}$$

, with

$$L(w, X, y) = -\sum_{i=1}^{N}\sum_{k}^{C}[y_i = k]\ln g_w(x_i)_j$$

Hence, we can obtain the MLE of the model by minimizing the cross entropy loss function on the dataset.

Let $z_m = w_m^T x, 1 \leq m \leq C$ and $Z = \sum_{k=1}^{C} e^{z_k}$, then:

$$\frac{\partial \ln f_w(x)_k}{\partial z_m} = \frac{Z}{e^{z_k}}\Big[\frac{[m = k]Ze^{z_k} - e^{z_k + z_m}}{Z^2}\Big] = [m = k] - g_w(x)_m \tag{12}$$

substitute the above into the gradient of cross entropy

$$\nabla_{w_m} L(w, X, y) = -\sum_{i=1}^{N}\sum_{k=1}^{C}[y_i = k][[m = k] - g_w(x_i)_m]x_i$$

Hence, we will update w_k after each iteration:

$$w_m = w_m - \nabla_{w_m} L(w, X, y) \tag{13}$$

$$= w_m - -\sum_{i=1}^{N}\sum_{k=1}^{C}[y_i = k][[m = k] - g_w(x_i)_m]x_i \tag{14}$$

$$\tag{15}$$

The process of Gradient Descent will be as followed:

1. Initialize **w**.
2. for $i = 1, 2, .., epochs$ {

   Calculate loss.

   For each $w_m, 1 \leq m \leq C$, update:

   $$w_m := w_m - \alpha \nabla_{w_m} L(w, X, y)$$

   } .