Submission for Machine Learning Assignment 2

Team: 47
Students: Juan Alvarez, Prabhav Gupta, Prafful Ravuri

October 28, 2025

1 Linear Regression

See the folder "Task_1". You can run *main.py* after installing the required packages by opening a terminal and running: `pip install -r requirements.txt`.

The coefficients for age and area are the following:

Coefficients [age, area]: **[-1481.44628961 9742.08372386].**

The predicted cost of a house that was built 10 years ago and that has an area of 50.0 m2 is: **472,289.72€**

If the real value of the house is 427,451.10€:
Least-squares Error: 2010502139.1610332 **L1 Error:** 44838.623296896985

2 Logistic Regression

2.1 2.1 and 2.2

See the attached resolution below.

Team-ID-47

2. Logistic Regression

(2.1) $$\boxed{J = -\sum_{n=1}^{N} y_n \log P_n + (1-y_n)\log(1-P_n)}$$

(Ignoring the bias term as instructed)

$$P_n = h_w(x_n) = P(y=1 \mid X = x_n, w) = \sigma(x_n)$$

$$\boxed{\sigma(x_n) = \frac{e^{x_n w}}{1+e^{x_n w}}}$$

(Sigmoid / Logistic function)

Soln: The primary rule for update of w for gradient descent goes like this -

$$\boxed{W \leftarrow W - \underbrace{\alpha}_{\text{learning rate}} \underbrace{\nabla_W J}_{\text{Jacobian}}}$$

(~~ESPS~~)

→ ~~Let's~~ assume that X is of ~~the~~ the shape $N \times M$, where N is the # of datapoints and M is the degree of each datapoint.

$N \times M$, $M \times 1$, $1 \times M$, $1 \times M$

→ W is the weight vector of shape $M \times 1$. $W = \begin{pmatrix} w_1 \\ \vdots \\ w_M \end{pmatrix}$.

[Bias term can be included in W using augmented notation, but we ignore that too since we were asked to do so in the problem statement]

→ $\nabla_W J$:- $\left[\nabla_W J = \begin{pmatrix} \frac{\partial J}{\partial w_1} \\ \vdots \\ \frac{\partial J}{\partial w_M} \end{pmatrix} \right]$

Let's compute $\frac{\partial J}{\partial w_i}$ ($\forall i = 1, 2, \ldots M$).

$$\frac{\partial}{\partial w_i} J = -\sum_{n=1}^{N} \left[y_n \left(\frac{\partial \log P_n}{\partial w_i} \right) + (1-y_n)\left(\frac{\partial \log(1-P_n)}{\partial w_i} \right) \right]$$ (Since y_n is const wrt w)

$$\boxed{\frac{\partial J}{\partial w_i} = -\sum_{n=1}^{N} \left[(y_n)\left(\frac{1}{P_n}\right)\left(\frac{\partial P_n}{\partial w_i}\right) + (1-y_n)(-1)\left(\frac{1}{1-P_n}\right)\left(\frac{\partial P_n}{\partial w_i}\right) \right]} \rightarrow (1)$$

$$\frac{\partial P_n}{\partial w_i} = \frac{\partial}{\partial w_i}\left[\frac{e^{x_n w}}{1+e^{x_n w}} \right] = \frac{\partial}{\partial w_i}\left[1 - \frac{1}{1+e^{x_n w}} \right] = \left[(-1)\left(e^{x_n w}\right)\left(\frac{-1}{(1+e^{x_n w})^2}\right)\left(\frac{\partial x_n w}{\partial w_i}\right) \right]$$

$$\boxed{\frac{\partial P_n}{\partial w_i} = \frac{e^{x_n w}}{(1+e^{x_n w})^2} x_{ni}} \rightarrow (2)$$

Substituting eqⁿ (1) in (2), we get:

$$\frac{\partial J}{\partial w_i} = -\sum_{n=1}^{N}\left[(y_n)\left(\frac{1+e^{x_n w}}{e^{x_n w}}\right)\left(\frac{e^{x_n w}}{(1+e^{x_n w})^{2}}\right)x_{ni} + (1-y_n)\left(\frac{(-1)(1+e^{x_n w})}{1}\right)\times\left(\frac{e^{x_n w}}{(1+e^{x_n w})^{2}}\right)x_{ni}\right]$$

$$= -\sum_{n=1}^{N}\left[\frac{y_n x_{ni} - (1-y_n)\,e^{x_n w}\cdot x_{ni}}{1+e^{x_n w}}\right]$$

$$= -\sum_{n=1}^{N}\left[\frac{y_n x_{ni}(1+e^{x_n w}) - x_{ni}\cdot e^{x_n w}}{1+e^{x_n w}}\right] = -\sum_{n=1}^{N}\left[x_{ni} y_n - x_{ni} P_n\right].$$

$$\left(\text{Since, } P_n = \frac{e^{x_n w}}{1+e^{x_n w}}\right)$$

$$\boxed{\frac{\partial J}{\partial w_i} = \sum_{n=1}^{N} x_{ni}(P_n - y_n)} \Longrightarrow \boxed{\nabla_w J = \sum_{n=1}^{N} x_n (P_n - y_n)}$$

The resulting update rule ~~for~~ w is

$$\boxed{w^{(k)} \leftarrow w^{(k-1)} - \alpha.\left(\sum_{n=1}^{N} x_n (P_n - y_n)\right)}$$

2.2 What is Gradient Descent?

Gradient Descent is an iterative optimization technique to identify optimal parameters that optimize a given function. We begin with a randomly chosen parameters and then iteratively make corrections to the parameters at the rate of gradient of the optimization function at ~~each~~ the initial set. The updates to parameters (lets say w) look as below -

$$\boxed{w^{(k)} \leftarrow w^{(k-1)} - \eta \left.\frac{dJ(D,w)}{dw}\right|_{w=w^{(k-1)}}}$$

learning rate - controls the rate at which optimal point is reached. [It might not be converged when too high!]

J is the function we try to optimize. It is ~~[illegible]~~ a function both on the given data D and params w.

Why do we use it for Logistic Regression?

Logistic Regression doesn't have an analytical solution, but the ~~function~~ loss function is convex. ~~Thus logistic regression turns out~~ Thus gradient descent proves to be a very efficient method to identify the optimal parameters. The loss function being convex, guarantees convergence — given that a proper learning rate is used. (not too high).

Feasibility to use Stochastic Gradient Descent can speed up the convergence too.

2.2 2.3 and 2.4

See the attached resolution below.

Team ID - 47

Q2 3) initial $w_0 = [0, 0, 0]^T$

$$\Rightarrow w_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

and learning rate, $\alpha = 0.25$

b	x_1	x_2	y
1	-5	0	0
1	-3	-2	0
1	2	5	1
1	4	1	1

We ~~add~~ can add a column of ones in the data (as above) to add the bias term.

Therefore, the augmented data matrix X $\Rightarrow$

$$X = \begin{bmatrix} 1 & -5 & 0 \\ 1 & -3 & -2 \\ 1 & 2 & 5 \\ 1 & 4 & 1 \end{bmatrix} ; \text{ and}$$

target values y $\Rightarrow$

$$y = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Now, calculating $h_w(x)$

$$\Rightarrow \quad h_w(x) = \frac{1}{1+e^{-w^T x}}$$

Since $w_0 = [0 \ \ 0 \ 0]^T$,

$$\Rightarrow \quad w_0^T x_i = (0 \times 1) + (0 \times x_{1i}) + (0 \times x_{2i}) = 0$$

for all $i \in [1, \ldots 4]$

$\Rightarrow$ The predicted probability for all data points is:

$$p_i = h_{w_0}(x_i) = \frac{1}{1+e^{-0}} = \frac{1}{1+1} = 0.5$$

$$\Rightarrow \quad p = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$$

Now, error vector $\underline{\underline{e}}$ is the difference b/w the prediction and the target value :

$$e = p - y = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \\ -0.5 \\ -0.5 \end{bmatrix}$$

$\rightarrow$

The batch gradient descent update rule for logistic regression (using L_2 loss) is :

$$\nabla J(\omega) = \frac{1}{N} X^T(p-y)$$

$$= \frac{1}{N} X^T e$$

wher $N = 4$
(no. of data samples)

$$\Rightarrow \nabla J(\omega) = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ -5 & -3 & 2 & 4 \\ 0 & -2 & 5 & 1 \end{bmatrix}\begin{bmatrix} 0.5 \\ 0.5 \\ -0.5 \\ -0.5 \end{bmatrix}$$

$$= \frac{1}{4}\begin{bmatrix} 0 \\ -7 \\ -4 \end{bmatrix}$$

$$\Rightarrow \nabla J(\omega_0) = \begin{bmatrix} 0 \\ -1.75 \\ -1 \end{bmatrix}$$

Now, the update rule is :

$$\omega_{k+1} = \omega_k - \alpha \cdot \nabla J(\omega_k)$$

$$\Rightarrow \quad w_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} - 0.25 \times \begin{bmatrix} 0 \\ -1.75 \\ -1 \end{bmatrix}$$

$$\Rightarrow \quad w_1 = \begin{bmatrix} 0 \\ 0.4375 \\ 0.25 \end{bmatrix} \quad \underline{\underline{\text{Ans}}}$$

Q~~3~~

4) using w_1 above, find $P(y=1 / x=[-1,1]^T)$

~~we show~~

augmenting input x with bias term

$$x = [1, -1, 1]^T$$

Now, $$w_1^T x = [0, 0.4375, 0.25] \times \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

$$= 0 - 0.4375 + 0.25$$

$$\Rightarrow w_1^T x = -0.1875$$

Now, $P(Y=1 / x=[1,-1,1]^T)$

$$= \frac{1}{1+e^{-w_1^T x}} \rightarrow$$

$$= \frac{1}{1 + e^{-(-0.1875)}}$$

$$= \frac{1}{1 + e^{0.1875}}$$

$$\cong \frac{1}{1 + 1.2062}$$

$$\cong 0.4532$$

Therefore, using w_1,

$P(y=1 / x=[1,-1,1]^T) \cong \underline{\underline{0.4532}}$ Ans