LOGISTIC REGRESSION

Implement Logistic Regression for classifying the IRIS dataset samples belong to two classes Iris-Virginica and Iris-Versicolor. The learning rate may be assumed as 0.02.

Logistic Regression is a supervised machine learning algorithm used for classification purposes. It is introduced due to failure of linear regression to outliers(i.e points that are far away from the classifier which are good examples but causes the error to make the classifier move in a wrong way).

sigmoid function is an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1,
Steps involved in logistic regression :

1. Make predictions (calculating y` or A)
   We use the values of X,w,b to calculate the expected output A
   A=sigmoid((w.T dot A) +b)
   --X is matrix of all training samples of X
   --w is all parameters of features of problem(dimensions of X)
   --b is bias

2. Calculating error we do A-Y , for calculating cost we do 1/m(summation((A-Y)^2)
   But since it has multiple local minima we use a log error function to calculate loss.
   J(w,b) is the cost function that is the summation of all errors in the training set.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$$

3. Calculating gradient descent:
   To minimize this cost function we use gradient descent technique to find the value of b and parameters (w) that minimize the value of the cost function.

$$
\begin{aligned}
&Repeat\ \{ \\
&\theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \\
&\}
\end{aligned}
$$

w= w - learningrate*(dw)
b = b - learningrate*(db)
dw = (1 /m) * np.dot(X, (A - Y).T)
db = (1 /m) * np.sum(A - Y)

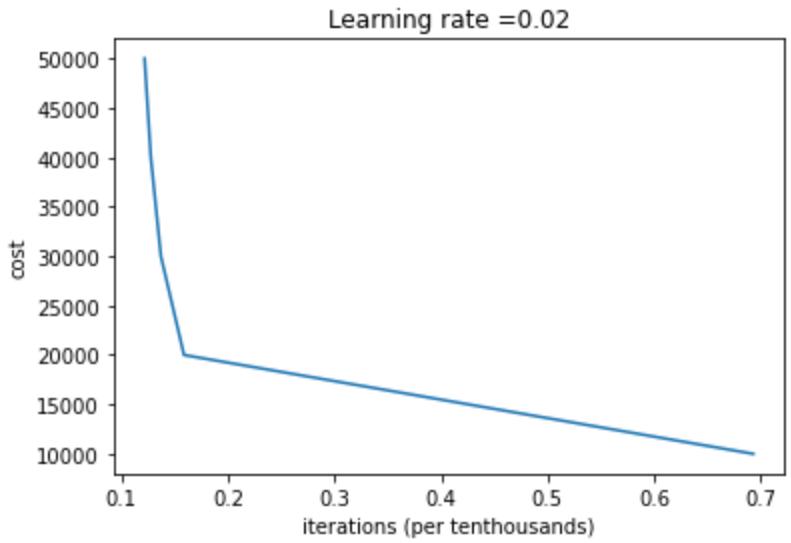
Question 1:

- The first stopping criteria is $||\nabla J|| < \varepsilon$. where $||\nabla J||$ is the Euclidean norm of vector ∇J and you may consider ε as 0.01.

Firstly we loaded the dataset using csv reader for test_2.csv and openpyxl for reading train_2.xlsx we store them into numpy arrays X_train and Y_train as float values .

- Then we normalize the features as X = X-min(X)/(max(X) - min(X)) to make all the data to be in the same bucket so that the contours will be circular instead of elliptical making our gradient descent to reach minimum quickly.
- With parameters(w) and bias b initialized to zero logistic regression is started the loop for updation of parameters with gradient descent will be stopped until the condition mentioned in the question is reached.
- Epochs until condition is reached is around 45,000
- The Parameters obtained here are used to test with a given test set.

Accuracy : On Train data : 94.3% On Test data : 100%



Question 2:
In the second case, the number of epochs are fixed to 80.

Firstly we loaded the dataset using csv reader for test_2.csv and openpyxl for reading train_2.xlsx we store them into numpy arrays X_train and Y_train as float values .

- Then we normalize the features as X = X-min(X)/(max(X) - min(X)) to make all the data to be in the same bucket so that the contours will be circular instead of elliptical making our gradient descent to reach minimum quickly.
- With parameters(w) and bias b initialized to zero ,logistic regression is started The loop for updation of parameters with gradient descent is ran for 80 epochs
- The Parameters obtained here are used to test with a given test set.

Accuracy : On Train data : 54.9% On Test data : 37.93%

Cost after iteration 1: 0.693147