### [HAY] Tóm tắt Logistic Regresion

#### **Mô hình chung cho bài toán trong Deep Learning.**

1. Visualize dữ liệu

2. Thiết lập model Logistic regresion là hàm sigmoid :

= (cho về khoảng [0,1])

3. Thiết lập loss function **binary crossentropy** :

4. Tìm tham số bằng việc tối ưu loss function (Gradient descent) :

5. Dự đoán dữ liệu mới bằng model vừa tìm được

*Giải thích :*

* *Ta có f(x) , để tìm cực trị của nó ta đi đạo hàm và cho để tìm ra x , tuy nhiên nhiều hàm f(x) phức tạo ta chỉ đạo hàm ra được còn việc tìm ra được x là rất khó . Chính vì thế nên ta dùng Gradient descent để đi tìm giá trị gần giống với nghiệm x nhất .*
* *Tóm lại :*

*Model + Gradient descent : Đi tìm* ***w mới*** *sau mỗi vòng lặp*

*Loss function : MSE hay binary crossentropy* ***để đánh giá*** *w mới đã tối ưu hay chưa thông qua việc xem* ***giá trị mất mát*** *mà* ***w mới*** *này mang lại sau mỗi vòng lặp từ đó điều chỉnh hệ số* ***learning rate*** *và* ***epochs*** *cho**hợp lí**để tìm ra được w chính xác mà không bị các tình trạng sớm , trễ hay không chính xác .*

* *Ở trong Linear regression điều đặc biệt là ta có thể tìm ra được nghiệm của hàm mất mát sau khi đạo hàm . Nhưng ở Logistic thì ta không thể tìm ra được nghiệm . Mặc khác việc giải toán và tìm ra nghiệm này cũng không quan trọng vì như đã nói ở trên .*

#### **Hàm Sigmoid**

**;**

*Giải thích : Mục đích là có đạo hàm với mọi x nhưng giá trị của f(x) chỉ ở phạm vi (0,1)*

#### **Thiết lập Model**

**=**

#### **Gradient Descent**

#### **Binary Crossentropy Loss function**

*Giải thích :*

* *Hàm này bắt nguồn từ công thức trong thuyết* [*Entropy (information theory)*](https://en.m.wikipedia.org/wiki/Entropy_(information_theory)?fbclid=IwAR3CdFdQ_VxEDq1r75LqI-WNUtezsklpGneX_0PnG9ZzmAcJ-W49JRDLN6w) *. Tham khảo* [*Cách xây dựng hàm Binary Cross Entropy*](https://www.youtube.com/watch?v=BoAH99NnWmY&t=1059s) *.*
* *Ta có thể dùng cơ số 2 , 10 , e . Tuy nhiên để cho phù hợp với hàm Sigmoid sau này thì ta dùng cơ số e .*

***Tóm lại :***

* *Sigmoid , Model và Gradient Descent để tìm ra và tối ưu dần w*
* *Binary Crossentropy Loss function để đánh giá mỗi khi tìm ra w mới thông qua giá trị mất mát mà w mới đó mang lại*

#### **Giải bài toán bằng đại số tuyến tính**

Quy tắc Chain Rule : Nếu và hay thì

Áp dụng Gradent descent

Ta có :

=

Như vậy ta đã có hàm loss fucntion theo biến w như thường lệ ta sẽ đi tìm đạo hàm của hàm loss fucntion nhằm mục đích sau này kết hợp với Gradient descent để đi tìm nghiệm w .

Áp dụng quy tắc Chain rule

Ta có

=

Đặt

=

=

**...**

=

Do đó

**...**

Đạo hàm trên **toàn bộ dữ liệu** ta sẽ có

**...**

Nếu để ý quan sát ta sẽ thấy

***Vậy Tóm lại ta đã có :***

* *Sigmoid , Model và Gradient Descent để tìm ra và tối ưu dần w*
* *Đạo hàm của Binary Crossentropy Loss function để đánh giá mỗi khi tìm ra w mới thông qua giá trị mất mát mà w mới đó mang lại*

#### **Code dự đoán pass hay không pass môn học**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra

def train\_test\_split(X, y, train\_ratio=0.7):

    m = len(X)

    train\_size = int(train\_ratio \* m)

    indices = np.random.permutation(m)

    train\_indices = indices[:train\_size]

    test\_indices = indices[train\_size:]

    X\_train, y\_train = X[train\_indices], y[train\_indices]

    X\_test, y\_test = X[test\_indices], y[test\_indices]

    return X\_train, y\_train, X\_test, y\_test

# Hàm sigmoid

def sigmoid(z):

    return 1 / (1 + np.exp(-z))

# Hàm loss binary crossentropy

def binary\_crossentropy(y\_true, y\_pred):

    return -np.mean(y\_true \* np.log(y\_pred) + (1 - y\_true) \* np.log(1 - y\_pred))

# Huấn luyện mô hình bằng gradient descent

def logistic\_regression(X, y, lr=0.01, epochs=10000, epsilon=1e-6):

    m, n = X.shape

    w = np.zeros((n, 1))

    losses = []  # List để lưu giá trị loss function qua các epoch

    for epoch in range(epochs):

        z = np.dot(X, w)

        y\_pred = sigmoid(z)

        loss = binary\_crossentropy(y, y\_pred)

        losses.append(loss)

        gradient = np.dot(X.T, (y\_pred - y)) / m

        w -= lr \* gradient

        if np.linalg.norm(gradient) < epsilon:

            break

    return w, losses

# Dự đoán và tính các độ đo chất lượng mô hình

def evaluate\_model(X, y, w):

    y\_pred = sigmoid(np.dot(X, w))

    y\_pred\_labels = (y\_pred > 0.5).astype(int)

    accuracy = np.mean(y\_pred\_labels == y)

    TP = np.sum((y\_pred\_labels == 1) & (y == 1))

    FN = np.sum((y\_pred\_labels == 0) & (y == 1))

    FP = np.sum((y\_pred\_labels == 1) & (y == 0))

    precision = TP / (TP + FP)

    recall = TP / (TP + FN)

    f1\_score = 2 \* (precision \* recall) / (precision + recall)

    return accuracy, recall, f1\_score

# Đọc dữ liệu từ file

data = pd.read\_csv('data\_logistic.csv').values

N, d = data.shape

X = data[:, 0:d-1].reshape(-1, d-1)

y = data[:, 2].reshape(-1, 1)

X = np.hstack((np.ones((N, 1)), X)) # thêm cột 1 vào cho X

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra

X\_train, y\_train, X\_test, y\_test = train\_test\_split(X, y)

# Huấn luyện mô hình

w, losses = logistic\_regression(X, y)

# Đánh giá mô hình trên tập kiểm tra

accuracy, recall, f1\_score = evaluate\_model(X\_test, y\_test, w)

print("Accuracy:", accuracy)

print("Recall:", recall)

print("F1-score:", f1\_score)

# Vẽ biểu đồ hàm loss function qua các epoch

plt.plot(losses)

plt.xlabel('Epoch')

plt.ylabel('Loss')

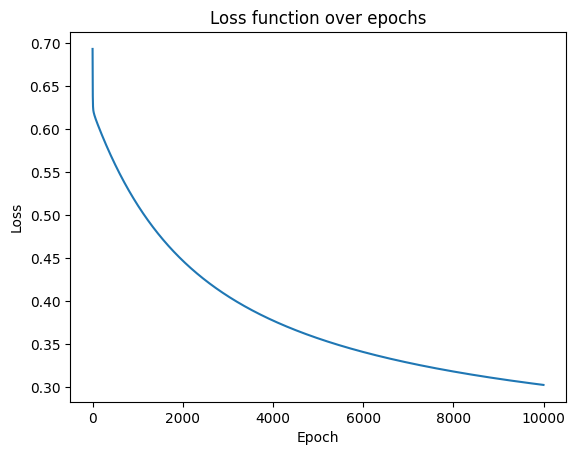
plt.title('Loss function over epochs')

plt.show()

Accuracy: 0.9333333333333333

Recall: 1.0

F1-score: 0.9411764705882353



# Load data từ file csv

data = pd.read\_csv('data\_logistic.csv').values

N, d = data.shape

x = data[:, 0:d-1].reshape(-1, d-1)

y = data[:, 2].reshape(-1, 1)

# Vẽ data bằng scatter

x\_learn = x[y[:,0]==1]

x\_sleep = x[y[:,0]==0]

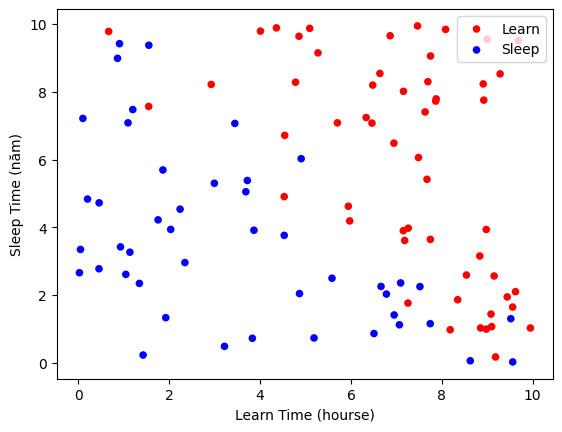
plt.scatter(x\_learn[:, 0], x\_learn[:, 1], c='red', edgecolors='none', s=30, label='Learn')

plt.scatter(x\_sleep[:, 0], x\_sleep[:, 1], c='blue', edgecolors='none', s=30, label='Sleep')

plt.legend(loc=1)

plt.xlabel('Learn Time (hourse)')

plt.ylabel('Sleep Time (năm)')



# Tính giá trị đầu vào cho mô hình logistic regression

learn\_time , sleep\_time = 6 , 4

X\_new = np.array([[1, learn\_time, sleep\_time]])

# Dự đoán kết quả sử dụng mô hình đã huấn luyện

y\_pred = sigmoid(np.dot(X\_new, w))

print(y\_pred)

# Kiểm tra kết quả dự đoán

if y\_pred > 0.5:

    print("Bạn có thể qua.")

else:

    print("Bạn không thể qua.")

[[0.59147665]]

Bạn có thể qua.