[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 : $\sigma(x) = \frac{1}{1+e^{-x}}$

   $\widehat{y_i} = \sigma\left(w_0 + w_1 x_1^{(i)} + \cdots + w_m x_m^{i}\right) = \frac{1}{1+e^{-(w_0+w_1x_1^{(i)}+\cdots+w_m x_m^{i})}}$ (cho $\widehat{y_i}$ về khoảng [0,1])
3. Thiết lập loss function **binary crossentropy** : $L = -(y_i * \log(\hat{y}_i) + (1-y_i) * \log(1-\widehat{y_i}))$
4. Tìm tham số bằng việc tối ưu loss function (Gradient descent) : $\frac{\mathrm{dJ}}{\mathrm{dw}} = \frac{1}{\mathrm{N}} * \mathrm{X}^{\mathrm{T}} * (\hat{\mathrm{y}} - \mathrm{y})$
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 $f'(x) = 0$ để tìm ra x , tuy nhiên nhiều hàm f(x) phức tạo ta chỉ đạo hàm ra được $f'(x)$ 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

$$\boldsymbol{\sigma(x) = \frac{1}{1+e^{-x}}\ ;\forall\, x \in R \rightarrow \sigma(x) \in (0,1)}$$

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

$$\boldsymbol{\widehat{y_i} = \sigma\left(w_0 + w_1 x_1^{(i)} + \cdots + w_m x_m^{i}\right) = \frac{1}{1+e^{-(w_0+w_1x_1^{(i)}+\cdots+w_m x_m^{i})}}}$$

Gradient Descent

$$\mathbf{\frac{dJ}{dw} = \frac{1}{N} * X^T * (\hat{y} - y)}$$

Binary Crossentropy Loss function

$$\boldsymbol{L = -(y_i * \log_e(\widehat{y_i}) + (1-y_i) * \log_e(1-\widehat{y_i}))}$$

Giải thích :

- *Hàm này bắt nguồn từ công thức trong thuyết [Entropy (information theory)](#) . Tham khảo [Cách xây dựng hàm Binary Cross Entropy](#) .*
- *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 $z = f(y)$ và $y = g(x)$ hay $z = f(g(x))$ thì $\frac{dz}{dx} = \frac{dz}{dy} * \frac{dy}{dx}$

Áp dụng Gradent descent

Ta có : $L = -(y_i * \log_e(\widehat{y_\iota}) + (1 - y_i) * \log_e(1 - \widehat{y_\iota}))$

$\widehat{y_\iota} = \sigma\left(w_0 + w_1 x_1^{(i)} + \cdots + w_m x_m^i\right) = \frac{1}{1 + e^{-(w_0 + w_1 x_1^{(i)} + \cdots + w_m x_m^i)}}$

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

$$\frac{dL}{dw} = \frac{dL}{d\widehat{y_\iota}} * \frac{d\widehat{y_\iota}}{w}$$

Ta có

$$\frac{dL}{d\widehat{y_\iota}} = \frac{d(-(y_i * \log_e(\widehat{y_\iota}) + (1 - y_i) * \log_e(1 - \widehat{y_\iota})))}{d\widehat{y_\iota}} = -\left(\frac{y_i}{\widehat{y_\iota}} - \frac{1 - y_i}{1 - \widehat{y_\iota}}\right) = \frac{\widehat{y_\iota} - y_i}{\widehat{y_\iota} * (1 - \widehat{y_\iota})}$$

$$\frac{d(\sigma(x))}{dx} = \frac{d\left(\frac{1}{1 + e^{-x}}\right)}{dx} = \frac{d\left(\frac{1}{1 + e^{-x}}\right)}{d(1 + e^{-x})} * \frac{d(1 + e^{-x})}{d(-x)} * \frac{d(-x)}{x} = -\frac{1}{(1 + e^{-x})^2} * e^{-x} * -1$$

$$= \frac{e^{-x}}{(1 + e^{-x})^2} = \frac{1}{1 + e^{-x}} * \frac{e^{-x}}{1 + e^{-x}} = \frac{1}{1 + e^{-x}} * \left(1 - \frac{1}{1 + e^{-x}}\right) = \sigma(x) * (1 - \sigma(x))$$

Đặt $z^i = w_0 + w_1 x_1^{(i)} + \cdots + w_m x_m^{(i)}$

$$\frac{d\widehat{y_\iota}}{w_0} = \frac{d\left(\sigma\left(w_0 + w_1 x_1^{(i)} + \cdots + w_m x_m^i\right)\right)}{dw_0} = \frac{d(\sigma(z^i))}{dz^i} * \frac{dz^i}{dw_0} = \sigma(z^i) * (1 - \sigma(z^i)) * 1 = \widehat{y_\iota} * (1 - \widehat{y_\iota})$$

$$\frac{d\widehat{y_\iota}}{w_1} = \frac{d\left(\sigma\left(w_0 + w_1 x_1^{(i)} + \cdots + w_m x_m^i\right)\right)}{dw_0} = \frac{d(\sigma(z^i))}{dz^i} * \frac{dz^i}{dw_1} = \sigma(z^i) * (1 - \sigma(z^i)) * x_1^{(i)} = \widehat{y_\iota} * (1 - \widehat{y_\iota}) * x_1^{(i)}$$

...

$$\frac{d\widehat{y_\iota}}{w_m} = \frac{d\left(\sigma\left(w_0 + w_1 x_1^{(i)} + \cdots + w_m x_m^i\right)\right)}{dw_0} = \frac{d(\sigma(z^i))}{dz^i} * \frac{dz^i}{dw_m} = \sigma(z^i) * (1 - \sigma(z^i)) * x_m^{(i)} = \widehat{y_\iota} * (1 - \widehat{y_\iota}) * x_m^{(i)}$$

Do đó

$$\frac{dL}{dw_0} = \frac{dL}{d\widehat{y_\iota}} * \frac{d\widehat{y_\iota}}{w_0} = \frac{\widehat{y_\iota} - y_i}{\widehat{y_\iota} * (1 - \widehat{y_\iota})} * \widehat{y_\iota} * (1 - \widehat{y_\iota}) = \widehat{y_\iota} - y_i$$

$$\frac{dL}{dw_1} = \frac{dL}{d\widehat{y_\iota}} * \frac{d\widehat{y_\iota}}{w_1} = \frac{\widehat{y_\iota} - y_i}{\widehat{y_\iota} * (1 - \widehat{y_\iota})} * \widehat{y_\iota} * (1 - \widehat{y_\iota}) * x_1^{(i)} = x_1^{(i)} * (\widehat{y_\iota} - y_i)$$

...

$$\frac{dL}{dw_m} = \frac{dL}{d\widehat{y_\iota}} * \frac{d\widehat{y_\iota}}{w_m} = \frac{\widehat{y_\iota} - y_i}{\widehat{y_\iota} * (1 - \widehat{y_\iota})} * \widehat{y_\iota} * (1 - \widehat{y_\iota}) * x_m^{(i)} = x_m^{(i)} * (\widehat{y_\iota} - y_i)$$

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

$$J = \frac{1}{N} * \sum_{i=1}^{N} L = -\frac{1}{N} * \sum_{i=1}^{N} x_m^{(i)} * (y_i * \log_e(\widehat{y_i}) + (1 - y_i) * \log_e(1 - \widehat{y_i}))$$

$$\frac{dJ}{dw_0} = \frac{1}{N} * \sum_{i=1}^{N} \widehat{y_i} - y_i$$

$$\frac{dJ}{dw_1} = \frac{1}{N} * \sum_{i=1}^{N} x_1^{(i)} * (\widehat{y_i} - y_i)$$

…

$$\frac{dJ}{dw_m} = \frac{1}{N} * \sum_{i=1}^{N} x_m^{(i)} * (\widehat{y_i} - y_i)$$

Nếu để ý quan sát ta sẽ thấy

$$\frac{dJ}{dw} = \frac{dJ}{dw_0} + \frac{dJ}{dw_1} + \cdots + \frac{dJ}{dw_m} = \frac{\mathbf{1}}{\boldsymbol{N}} * \boldsymbol{X^T} * (\widehat{\boldsymbol{y}} - \boldsymbol{y})$$

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))
```

```python
# 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
```

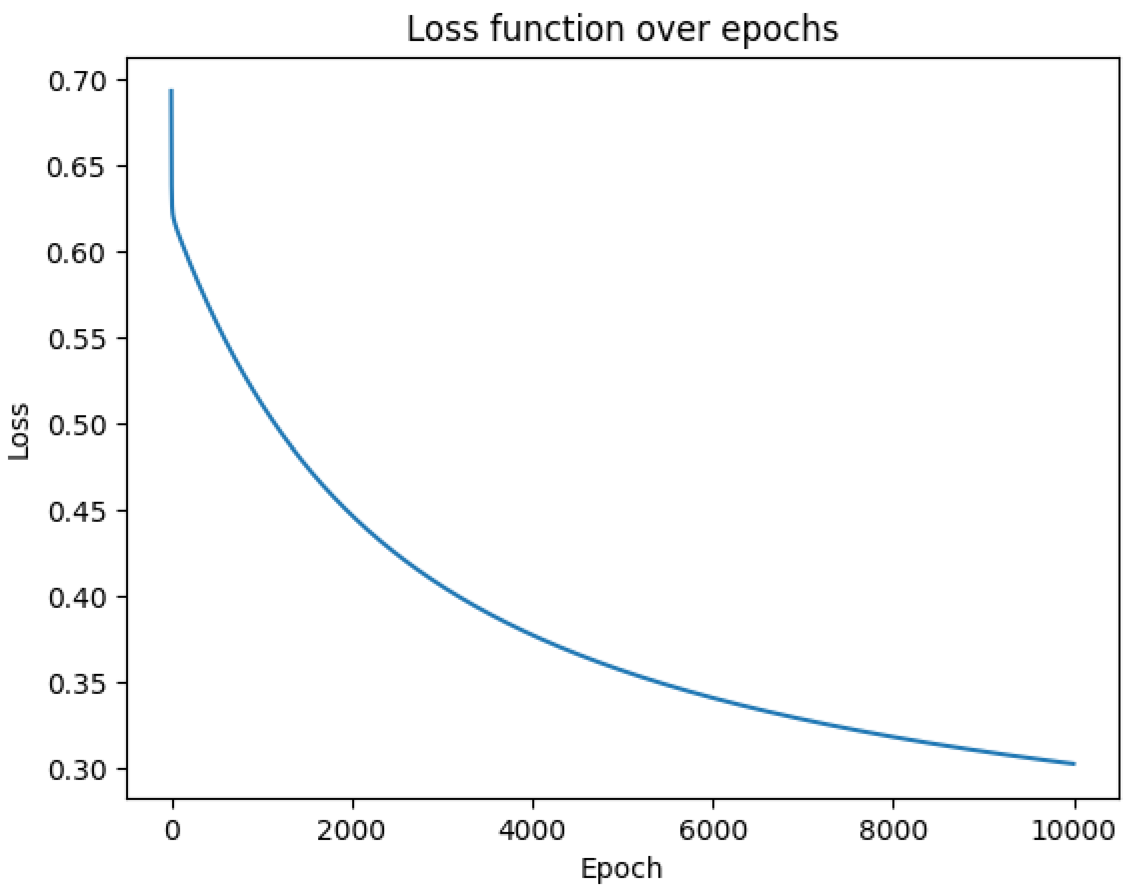
```
w, losses = logistic_regression(X, y)

# 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
```

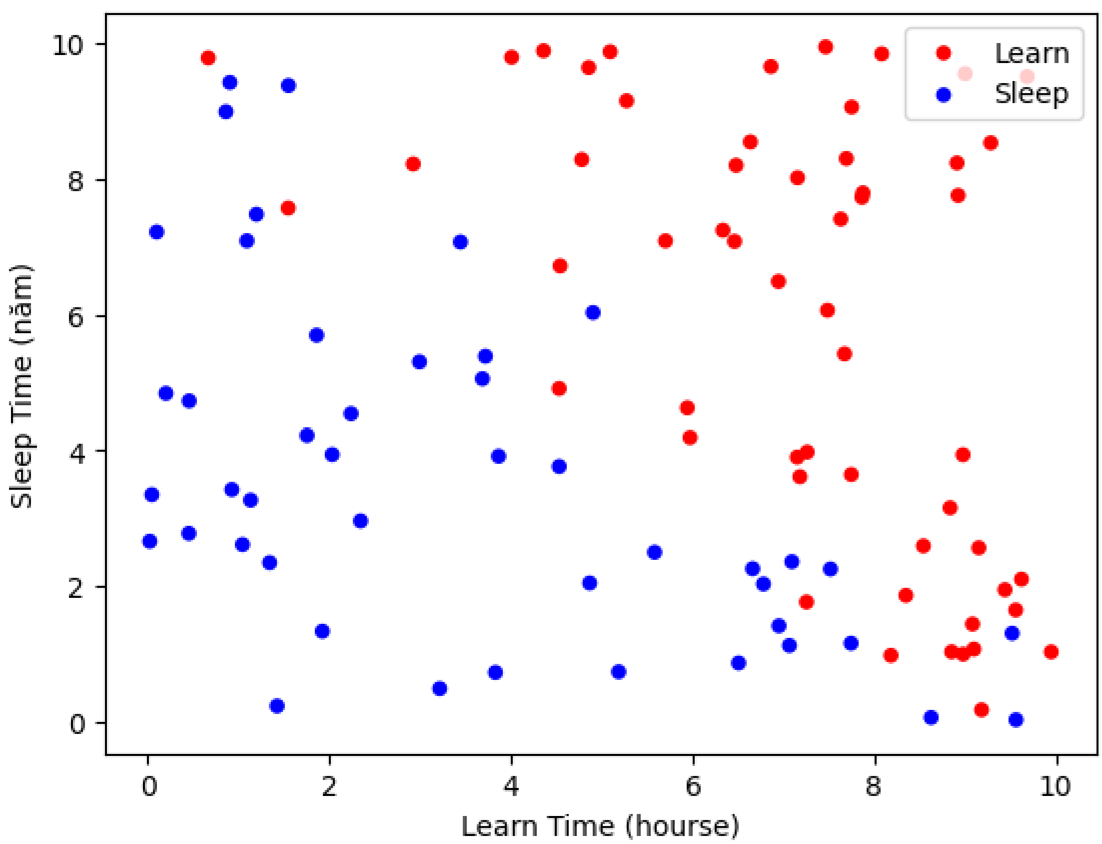



```
# Load data từ file csv
```

```python
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)')
```




```python
# 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]])
```

```python
# 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.
```