THỰC HÀNH 2: MÁY HỌC CƠ BẢN PHẦN 2

Mục tiêu: Thao tác cơ bản với python. Hiểu được cách sử dụng python để hiện thực một số thuật toán máy học cơ bản.

1. Logistic regression

Hàm mục tiêu của **Logistic Regression**: $\hat{y} = \sigma\left(X\theta^T\right)$

Hàm mất mát: $L(\theta) = \frac{-1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(\hat{y}^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-\hat{y}^{(i)}\right)\right]$

Bộ dữ liệu áp dụng: **Iris**. Bộ này được cung cấp và hỗ trợ sẵn bởi thư viện *sklearn*.

Dữ liệu huấn luyện cho bài toán phân lớp sẽ gồm 2 phần:

- X: các thuộc tính của dữ liệu. Có tổng cộng 4 thuộc tính gồm Sepal Length, Sepal Width, Petal Length, Petal Width.
- y: thuộc tính nhãn. Có 3 nhãn gồm Setosa, Versicolour, and Virginica

Tổng cộng có 150 điểm dữ liệu.

Code load dữ liệu:

```
from sklearn.datasets import load_iris
iris = load_iris()
```

Trong bài này, chúng ta sẽ dùng thuộc tính Petal Width để phân loại xem loài hoa đang xét có phải là Virginica hay không => phân lớp nhị phân với 2 nhãn: Virginica (1) và không phải Virginica (2).

Tạo dữ liệu huấn luyện:

```
X = iris["data"][:, 3:]
y = (iris["target"] == 2).astype(np.int)
```

Tạo dữ liệu dự đoán: sinh ra ngẫu nhiên 100 phần tử tương ứng với *Petal Width* trong khoảng giá trị từ 0 - 3 cm.

```
X_new = np.linspace(0, 3, 1000).reshape(-1, 1)
```

2. Gradient descent

Thuật toán Gradient descent:

```
W := θ0          // Khởi tạo trọng số
Repeat {
    θ := θ - α*dL(θ,b)/dθ
}
```

Vector gradient của θ: $\frac{dL(\theta)}{d\theta} = \frac{1}{m} X^T(\hat{y} - y)$

Ghi chú: Cách lập trình sử dụng hàm trong Python:

```
def <tên hàm> (<tham số 1>, ..., <tham số n>):
    // code xử lý (nhớ thụt vào 1 tab - 4 space)
    return <giá trị> (nếu có)
```

Các bước thực hiện:

Bước 1: Thêm giá trị bias_term vào vector X ban đầu.

```
import numpy as np
intercept = np.ones((X.shape[0], 1))
X = np.concatenate((intercept, X), axis=1)
```

Bước 2: Viết hàm tính sigmoid cho 1 vector. $\hat{y} = \sigma(X\theta^T)$. Đặt tên hàm là **sigmoid**.

Gợi ý: Dùng thư viện *np.exp()* để tính giá trị e.

```
sig = 1/(1+np.exp(-z))
```

Bước 3: Viết hàm tính giá trị hàm loss. Đặt tên hàm là **compute_loss**.

$$L(\theta) = \frac{-1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right]$$

Gợi ý: dùng hàm *np.log()* để tính log cho y mũ. Sau đó dùng *np.mean()* để tính cho giá trị $\frac{-1}{m} * giá trị hàm loss$.

```
-np.mean(y_true * np.log(y_hat) + (1 - y_true) * np.log(1 - y_hat))
```

Bước 4: Viết hàm tính giá trị Gradient descent. Đặt tên hàm là **compute_gradient**.

Vector gradient descent được tính như sau: $\frac{dL(\theta)}{d\theta} = \frac{1}{m} X^T(\hat{y} - y)$ (*)

Gợi ý: dùng *np.dot()* và *np.mean()* để tính cho công thức (*)

Bước 5: Viết hàm khởi tạo tham số cho w. Đặt tên là **initializers**.

Gợi ý: Dùng *np.zeros(A)* để khởi tạo 1 vetor chứa giá trị 0. A là số chiều của vector. Số chiều vector w khởi tạo ban đầu bằng số thuộc tính của bộ dữ liệu. Để lấy số thuộc tính của bộ dữ liệu X, ta dùng hàm *X.shape[1]*.

Bước 6: Dùng gradient descent để tìm ra tham số tối ưu. Đặt tên hàm là **fit**.

Gợi ý: dựa theo thuật toán gradient descent đã trình bày ở trên. Tham số θ_0 chính là trọng số đã khởi tạo ban đầu bằng hàm **initilizers** (ở bước 4). Thay **Repeat** bằng vòng lặp for với số bước chạy được xác định trong biến ***iter***. Đặt giá trị tham số ***iter*** mặc định là 100.

Lưu ý: biến iter và alpha sẽ là 2 siêu tham số cho mô hình.

Kết quả trả về của hàm này là tham số w cuối cùng => tham số tối ưu cho mô hình.

Bước 7: Viết hàm dự đoán, đặt tên là **predict**.

Gợi ý: Với dữ liệu đầu vào X_new, ta xác định giá trị dự đoán cho X_new bằng cách $\widehat{y_{new}} = \sigma(X\theta^T)$. θ chính là tham số tối ưu tìm được ở bước 5.

Lưu ý: Nhãn dự đoán được xác định như sau:

$$class = \begin{cases} 0 \, nếu \, \widehat{y_{new}} < 0.5 \\ 1 \, nếu \, \widehat{y_{new}} \geq 0.5 \end{cases}$$

3. Bài tập

Bài 1: Các bạn hiện thực lại mô hình Logistic Regression theo các hướng dẫn ở mục 2. Huấn luyện mô hình với siêu tham số alpha = 0.1 và iter = 100.

Bài 2: Dùng tham số θ vừa huấn luyện được từ mô hình dự đoán cho 100 dòng dữ liệu đầu tiên từ tập dữ liệu gốc. So sánh kết quả dự đoán với nhãn thực sự của dữ liệu.
Gợi ý: Để lấy ra 100 dòng dữ liệu đầu tiên, ta dùng lệnh: `X[1:100]` và `y[1:100]`.
Có thể dùng *accuracy_score* để tính độ chính xác của dự đoán.

Bài 3: Trong Bước 5, với mỗi lần lặp để cập nhật trọng số, hãy tính giá trị hàm mất mát của mỗi lần lặp và đưa vào list loss. Vẽ biểu đồ giá trị loss sau mỗi lần lặp.
Gợi ý: dùng hàm *lineplot* trong seaborn để vẽ biểu đồ giá trị của hàm loss sau mỗi lần cập nhật trọng số. Tính giá trị hàm mất mát bằng hàm **compute_loss** ở Bước 3.

Bài 4: Dùng tham số θ vừa huấn luyện được từ mô hình dự đoán cho dữ liệu `X_new`.

Bài 5: Các bạn hãy dùng thư viện Logistic Regression trong sklearn để dự đoán, và so sánh kết quả giữa dùng thư viện và làm bằng tay đối với Bài 2 và Bài 4.

Bài 6*: Hãy sử dụng các thuộc tính khác như Petal Length, Sepal Length và Sepal Width để huấn luyện cho mô hình, và cho biết kết quả.

Các bạn làm trực tiếp trên file jupyter notebook, đặt tên là: **MSSV_BaiThucHanh2.ipynb** (hoặc .jpynb) Các bạn nộp trên course theo thời gian quy định nhé.