Bài 3: Phân tích mô hình hồi quy logistic

(Bài tập này giúp bạn nắm bắt cách chuyển các công thức toán học sang mã lệnh Python đúng đắn)

Hướng dẫn: điền đoạn code Python đúng vào phần ô trống ở cột Mã lệnh Python.

Dữ liệu của bài tập chứa trong các tập tin ex2data1.txt và ex2data2.txt, theo cấu trúc từng bộ dữ liệu (x, y) xếp trên từng hàng. Các giá trị trên từng hàng phân tách nhau bởi dấu ',' .

STT	Biểu diễn Toán học	Mã lệnh Python

1	Tập dữ liệu $D = \{(x_i, y_i) \mid x_i \in R^n, y_i \in \{0,1\}, \forall i = \overline{1,m}\}$.
	Lấy ra $X = \{x_i \mid x_i \in R^n, \forall i = \overline{1,m}\}$ và $y\{y_i \mid y_i \in \{0,1\}, \forall i = \overline{1,m}\}$ từ tập D.
	Ghi chú:
	- $X \in R^{m \times n}$
	- $y \in \{0,1\}^{m \times 1}$
		`def readData(filePath: str, filename: str):`
		`  data = np.loadtxt(os.path.join(filePath, filename), delimiter = ',')`
		`  X = data[:,:-1]`
		`  y = data[:, -1]`
		`  m = X.shape[0]`
		`  n = X.shape[1]`
		`  X = np.reshape(X, (m,n))`
		`  y = np.reshape(y, (m,1))`
		`  return X, y`

2	Chuyển đổi không gian của X từ R^n sang $R^{(n+1)}$, bằng cách thêm 1 cột chứa các giá trị 1 vào bên trái ma trận X.
	- Tạo vector (ma trận $m \times 1$) chứa các số 1, $x_0 = \begin{bmatrix} x_{01} = 1 \\ \vdots \\ x_{0m} = 1 \end{bmatrix}$
	- Tạo ma trận X ~~mới: $X = [x_0, X]$~~
	Ghi chú: lúc này $X \in R^{m \times n}$ (ngầm hiểu $n = n + 1$)
		`  # Tạo vector (ma trận m × 1) chứa các số 1`
		`  x0 = np.ones((m,1))`
		`  # Thêm cột x0 vào bên trái ma trận X`
		`  X = np.column_stack([x0, X])`

3	Về lý thuyết, ta có:
	$h_w(x) = \frac{1}{1+e^{-(w_0.x_0+ w_1x_1+ \cdots+ w_nx_n)}}$ (1)
	hay
	$h_w(x) = \frac{1}{1+e^{-w^Tx}}$ (2)
	với:
	- $w \in R^{n\times1}, x \in R^{n\times1}$
	Viết lại (1) và (2) dưới dạng phép toán ma trận
	$h_w(X) = \frac{1}{1+e^{-X.w}}$ (3)
	Ghi chú:
	- $X \in R^{m\times n}, w \in R^{n\times1} \rightarrow X.w \in R^{m\times1}$
	- $h_w(X) \in R^{m\times1}$
		*- Hãy lập trình Python tính giá trị **h** phương trình (3) (gợi ý: hàm np.exp() – tính số mũ)*
		`def sigmoid(X, w):`
		`  result = 1/(1 + np.exp(-np.dot(X, w)))`
		`  return result`

4	Hàm mất mát
	$J(w) = -\frac{1}{m}\sum_{i=1}^{m}[y_i.\log(h_w(x_i)) + (1- y_i).\log(1-h_w(x_i))]$ (4)
	Viết lại (4) theo phép toán ma trận
	$J(w) = -\frac{1}{m}\sum[y^T \log(h_w(X)) + (1-y)^T log(1-h_w(X))]$ (5)
	Ghi chú:
	- $y \in R^{m\times1} \rightarrow y^T \in R^{1\times m}$
	- $h_w(X) \in R^{m\times1} \rightarrow \log(h_w(X)) \in R^{m\times1}$
	- $(1-y) \in R^{m\times1} \rightarrow (1-y)^T \in R^{1\times m}$
	- $(1-h_w(X)) \in R^{m\times1} \rightarrow log(1-h_w(X)) \in R^{m\times1}$
	- $J(w) \in R$
		`def loss(w, X, y):`
		`  m = X.shape[0]`
		`  #Sử dụng biến tạm h để giảm số lần gọi hàm sigmoid`
		`  h = sigmoid(X, w)`
		`  result = (-1 / m) * np.sum(np.dot(y.T, np.log(h)) + np.dot((1 - y).T, np.log(1 - h)))`
		`  return result`

5	Thay vì áp dụng thuật toán Gradient Descent, scipy cung cấp nhiều thuật toán tối ưu (vd: TNC, BFGS, L-BFGS-B, .v.v.). Cú pháp sử dụng chung như sau:
	`from scipy import optimize`
	`result = optimize.minimize(fun=loss, x0=w, args=(X,y), method='L-BFGS-B', options={"maxiter":n_iters})`
	`return result.x, result.fun`
	- fun=loss: ghi tên của hàm mất mát là hàm cần tối thiểu hóa giá trị
	- *x0=w: vector trọng số w*
		`def toi_uu_bang_scipy(X,y,w,n_iters):`
		`  #Thứ tự xuất hiện các tham số của hàm loss phải đổi lại theo thứ tự trong optimize.minimize là`
		`  #loss(w, X, y)`
		`  result = optimize.minimize(fun=loss, x0=w, args=(X,y), method='L-BFGS-  B', options={"maxiter":n_iters} )`
		`  w_optimal = result.x`
		`  J_optimal = result.fun`
		`  return w_optimal, J_optimal`

5	- args=(X,y): tập dữ liệu X và vecgtor nhãn lớp y - method='BFGS': ghi tên thuật toán tối ưu được sử dụng - options={"maxiter":n_iters}: tạo dictionary khai báo số bước lặp tối đa là n_iters. - result.x: chứa vector trọng số tối ưu w∗ - result.fun: chứa giá trị của hàm mất mát tương ứng với trọng số tối ưu - J(w∗) Ghi chú: thứ tự các tham số của hàm loss do các bạn lập trình cũng phải được thay đổi tuân theo thứ tự xuất hiện các tham số trong hàm optimize này, như sau: `def loss(w, X, y):`	
7	*Hãy hoàn thành chương trình xây dựng mô hình Logistic có xét đến trường hợp chuẩn hóa dữ liệu và không chuẩn hóa dữ liệu.Ghi chú: Áp dụng 2 phương pháp chuẩn hóa dữ liệu:* - *Mean normalization* - *Max-Min normalization* - *Thay thế thuật toán tối ưu khác mà scipy cung cấp, tham khảo link*	