AI VIET NAM – COURSE 2022

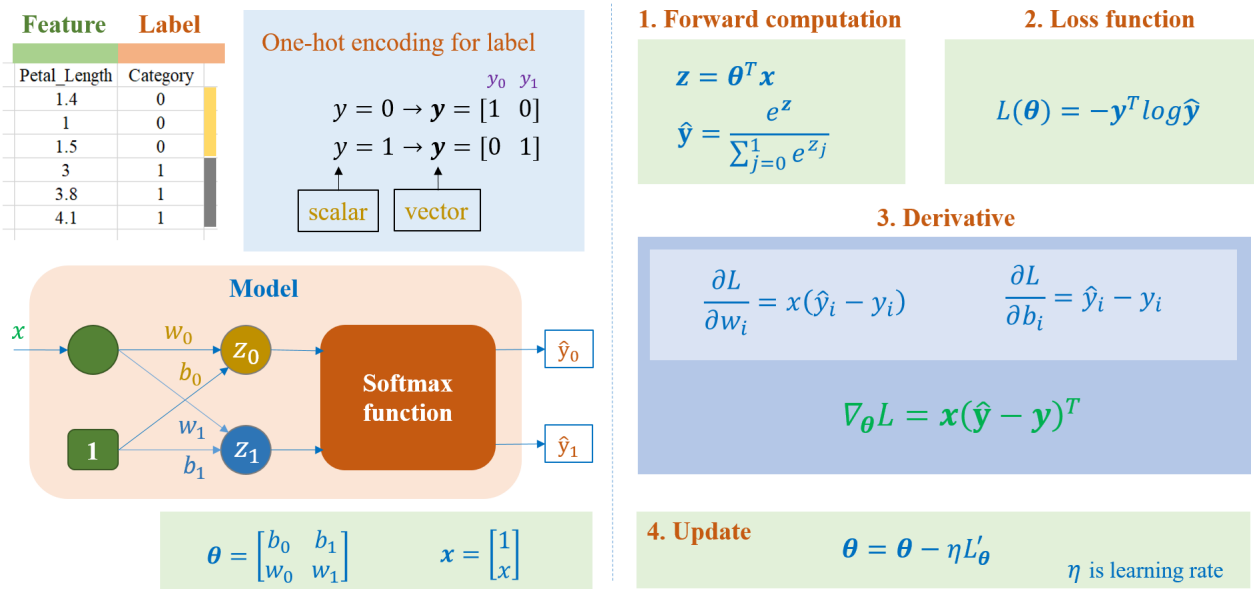
Softmax Regression - Exercise

Ngày 31 tháng 10 năm 2022

Softmax Regression là một trong những thuật toán Machine Learning cơ bản thuộc nhánh Supervised Learning. Tương tự như Logistic Regression, Softmax Regression cũng được dùng để giải quyết các bài toán phân lớp (classification). Tuy nhiên, trong khi về cơ bản Logistic Regression chỉ có thể áp dụng để giải quyết các bài toán phân lớp nhị phân (có 2 class), Softmax Regression lại có thể giải quyết các bài toán có số lượng class lớn hơn 2. Trong bài tập này, chúng ta sẽ triển khai thuật toán Softmax Regression sử dụng thư viện NumPy trong Python để giải quyết bài toán Phát hiện gian lận thẻ tín dụng (Credit Card Detection) và bài toán Tìm vị trí người sử dụng Wi-Fi (Wi-Fi User Localization).

Phần I: Cài đặt thuật toán

Để cài đặt thuật toán Softmax Regression, các bạn sẽ thực hiện theo các bước như sau (**Lưu ý:** phần mô tả dưới đây thể hiện cho kiểu cài đặt 1-sample):



Hình 1: Ảnh minh họa các bước tính toán của Softmax Regression phiên bản 1-sample

1. **Quy ước từ khóa**

   - **n_samples:** Số lượng mẫu dữ liệu trong bộ dữ liệu (số hàng của bảng dữ liệu).
   - **n_features:** Số lượng đặc trưng trong bộ dữ liệu (số cột của bảng dữ liệu trừ cột cuối cùng).
   - **n_classes:** Số lượng lớp đối tượng trong bộ dữ liệu (số lượng giá trị duy nhất của cột cuối cùng).

- **theta:** Bộ trọng số của mô hình. Trong Softmax Regression, shape của theta = (n_features + 1, n_classes). Giá trị của theta ban đầu sẽ được khởi tạo ngẫu nhiên.
- **epochs:** Số lần duyệt qua bộ dữ liệu trong quá trình huấn luyện.
- **learning_rate:** Tốc độ học của mô hình. Mặc định giá trị learning_rate = $1e-3$.
- **training_size:** Tỉ lệ kích thước của tập dữ liệu training.
- **val_size:** Tỉ lệ kích thước của tập dữ liệu validation.
- **test_size:** Tỉ lệ kích thước của tập dữ liệu test.

2. **Tiền xử lý dữ liệu**

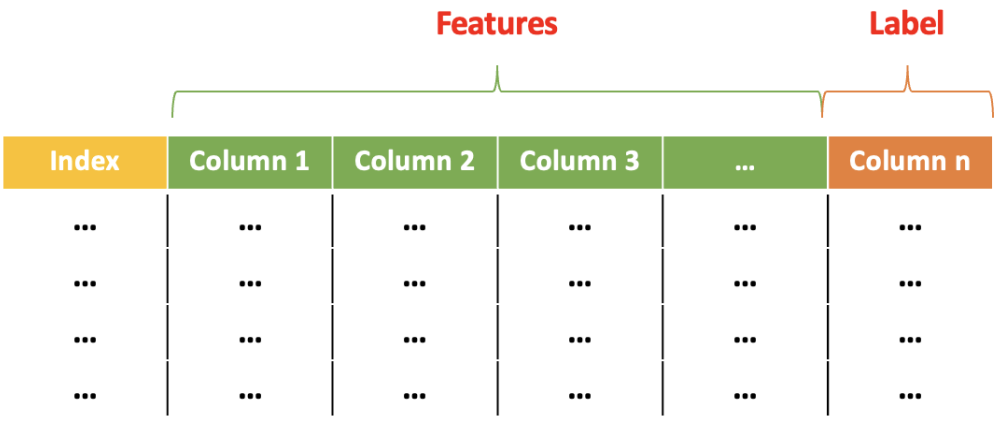
(a) **Đọc dữ liệu (Read Data):** Đối với hai bài toán trên, bộ dữ liệu cho trước thuộc dạng bảng (lưu trữ dưới dạng file .csv). Vì vậy, chúng ta sẽ sử dụng các hàm đọc các văn bản có định dạng theo bảng dữ liệu (**Gợi ý:** sử dụng hàm numpy.genfromtxt(), numpy.loadtxt()...). Sau bước này, ta sẽ có một numpy.ndarray lưu trữ dữ liệu với shape = (n_samples, n_features).

(b) **Trộn dữ liệu (Data Shuffle):** Từ array chứa dữ liệu đã đọc, thực hiện thay đổi vị trí ngẫu nhiên giữa các hàng trong bảng dữ liệu với nhau. (**Gợi ý:** sử dụng hàm np.random.permutation() hoặc sử dụng kỹ thuật trộn thông qua trộn chỉ mục).

Index	Column 1	Column 2	Column 3	Column 4
0	10	5	2	0
1	6	2	0	0
2	30	9	1	1
3	14	4	0	1

ndex	Column 1	Column 2	Column 3	Column 4
3	14	4	0	1
0	10	5	2	0
2	30	9	1	1
1	6	2	0	0

(c) **Tách đặc trưng và nhãn:** Từ array chứa dữ liệu đã được trộn, thực hiện tách thành hai biến X, y đại diện cho biến chứa đặc trưng (features) và nhãn (label) của dữ liệu. Đối với các bộ dữ liệu trong bài tập này, các cột từ cột đầu tiên đến cột thứ **n_features** sẽ là đặc trưng của dữ liệu, cột cuối cùng sẽ chứa nhãn.



(d) **One-hot encoding:** Đối với Softmax Regression, sau khi có biến chứa nhãn dữ liệu y, cần phải thay đổi cách biểu diễn giá trị của nhãn trước khi thực hiện tính toán. Cụ thể (giả định với bộ dữ liệu có 2 class), với dạng biểu diễn số nguyên ban đầu y = [0, 0, 1,...1], ta biến đổi thành các vector toàn giá trị 0 với số phần tử bằng **n_classes** và gán bằng 1 tại vị trí chỉ mục theo giá trị tại nhãn ban đầu của mẫu dữ liệu tương ứng. Lúc này, ta sẽ được y_one_hot = [[1, 0], [1, 0], [0, 1],...[0, 1]].

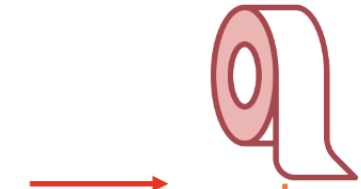
Label
0
0
1
1

Label
[1, 0]
[1, 0]
[0, 1]
[0, 1]

(e) **Chuẩn hóa dữ liệu (Data Normalization):** Để cải thiện kết quả huấn luyện, ta sẽ thực hiện chuẩn hóa dữ liệu đầu vào X (không chuẩn hóa nhãn y) sử dụng kỹ thuật Min-Max Normalization. Theo đó, với mỗi cột đặc trưng trong bảng dữ liệu, tìm giá trị X_{max} và X_{min}. Sau đó, cập nhật lại các giá trị trong cột tương ứng theo công thức sau:

$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

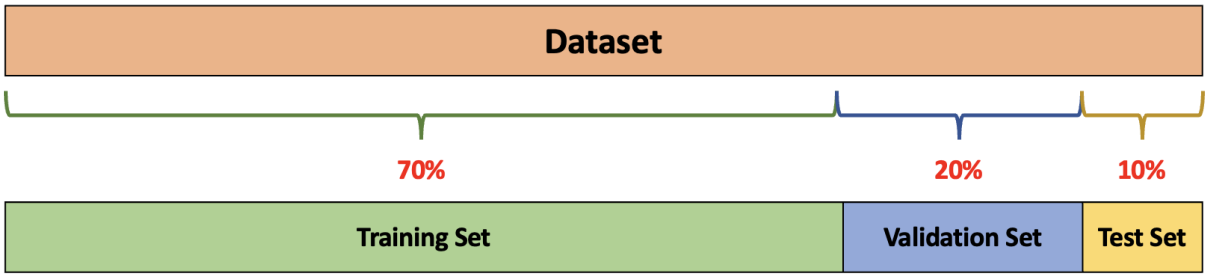
(f) **Thêm bias:** Sau khi chuẩn hóa, ta thêm vào vector đặc trưng của mỗi mẫu dữ liệu một giá trị 1 tượng trưng cho bias. Điều này có thể thực hiện đơn giản bằng cách tạo một vector toàn giá trị 1 có số phần tử bằng **n_samples**, sau đó sử dụng hàm numpy.concatenate() để nối vector này vào vector đặc trưng X.



Bias
1
1
1
1

Feature 1	Feature 2	Feature 3	Feature 4
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...

(g) **Chia bộ dữ liệu train/val/test:** Từ hai biến X, y ban đầu, thực hiện tách bộ dữ liệu thành 3 bộ train, val, test theo tỉ lệ **TRAIN_SIZE**, **VAL_SIZE**, **TEST_SIZE** (trong bài này ta sẽ chia theo tỉ lệ 7/2/1). Sau bước này, ta sẽ có các cặp **X**, **y** cho từng bộ train/val/test (X_train, X_val, X_test, y_train, y_val, y_test).



Hình 2: Chia bộ train/val/test

3. **Khai báo các hàm tính toán cần thiết**

(a) **Hàm softmax():** Nhận đầu vào là vector chứa các giá trị tuyến tính z. Đầu ra là vector chứa các giá trị xác suất được tính bởi công thức:

$$softmax(z) = \frac{e^z}{\sum_{i=1}^{C} e^{z_i}}$$

Trong đó: C là số lượng class.

(b) **Hàm predict():** Nhận đầu vào là vector đặc trưng X của mẫu dữ liệu và vector theta. Thực hiện tích vô hướng giữa hai vector này theo công thức $z = \theta^T \cdot X$ và truyền vào hàm softmax đã khai báo trước đó. Cuối cùng, trả về kết quả đầu ra chính là kết quả từ hàm softmax.

(c) **Hàm loss():** Nhận đầu vào là vector nhãn thực tế y và vector dự đoán y_hat. Trả về giá trị loss được tính theo công thức:

$$loss(y, y\_hat) = -\log\left(y^T \cdot y\_hat\right)$$

(d) **Hàm gradient():** Nhận đầu vào là vector đặc trưng X, vector nhãn thực tế y và vector dự đoán y_hat. Trả về giá trị gradient được tính theo công thức:

$$gradient(X, y, y\_hat) = X \cdot (y\_hat - y)^T$$

(e) **Hàm fit():** Nhận đầu vào là bộ dữ liệu huấn luyện X_train y_train, theta, epochs và learning_rate. Sau đó, thực hiện các bước tính toán như sau:

i. Khởi tạo vòng lặp với số lần lặp bằng số epochs.

ii. Với mỗi lần lặp tại bước 1, khởi tạo vòng lặp mới với số lần lặp chính bằng số lượng mẫu dữ liệu trong tập train (n_samples của tập train). Với mỗi lần lặp tại bước 2, thực hiện:

A. Đọc mẫu dữ liệu train tại vị trí thứ i (X_i và y_i).

B. Reshape X_i thành shape = (n_features, 1) và y_i thành shape = (n_classes, 1).

C. Thực hiện dự đoán với đặc trưng X_i và theta sử dụng hàm **predict()**.

D. Thực hiện tính loss với y_hat vừa tính được so với y_i thực tế sử dụng hàm **loss()**. Sau đó, lưu trữ giá trị loss này vào một list **losses** (dùng trong việc trực quan hóa kết quả huấn luyện về sau).

E. Thực hiện tính gradient với X_i, y và y_hat sử dụng hàm **gradient()**.

F. Cập nhật lại bộ trọng số theta theo công thức:

$$theta = theta - gradient \times learning\_rate$$

(f) **Hàm evaluate():** Nhận đầu vào là bộ dữ liệu đánh giá X y (test) và bộ trọng số theta. Thực hiện đánh giá bằng duyệt qua từng mẫu dữ liệu trong bộ dữ liệu đánh giá, tính y_hat. Sau đó, tính loss giữa y_hat và y thực tế, lưu lại kết quả này vào trong một list **losses**. Khi duyệt qua hết toàn bộ mẫu dữ liệu, trả về kết quả loss cuối cùng bằng công thức:

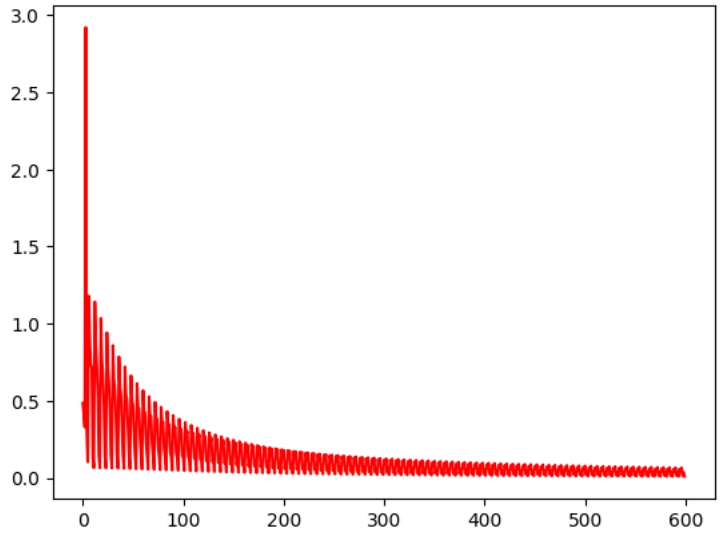
$$l = \frac{\sum_{i=1}^{n\_samples} losses_i}{n\_samples}$$

Ngoài ra, ta có thể bổ sung thêm độ đo Accuracy vào trong hàm này với công thức:

$$acc = \frac{\sum_{i=1}^{n\_samples} (\arg\max_{y\_hat_i} == \arg\max_{y_i})}{n\_samples}$$

Lưu ý: n_samples ở đây tính trong bộ dữ liệu đánh giá.

4. **Huấn luyện, đánh giá và trực quan hóa:** Thực hiện huấn luyện sử dụng hàm **fit()** trên bộ dữ liệu train. Sau khi hoàn thành quá trình huấn luyện, sử dụng danh sách giá trị **losses** đã lưu để trực quan hóa kết quả lên đồ thị. Việc này có thể dễ dàng thực hiện được với hàm plot() trong thư viện matplotlib. Khi trực quan hóa thành công, ta sẽ nhận được một đồ thị giống như sau:

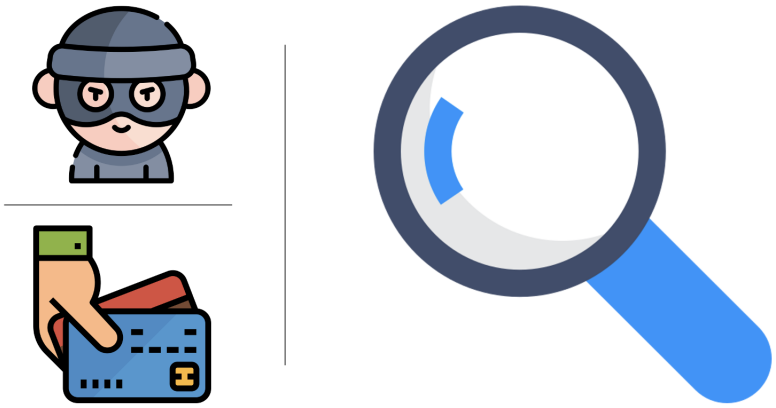


Hình 3: Minh họa đồ thị trực quan hóa kết quả loss

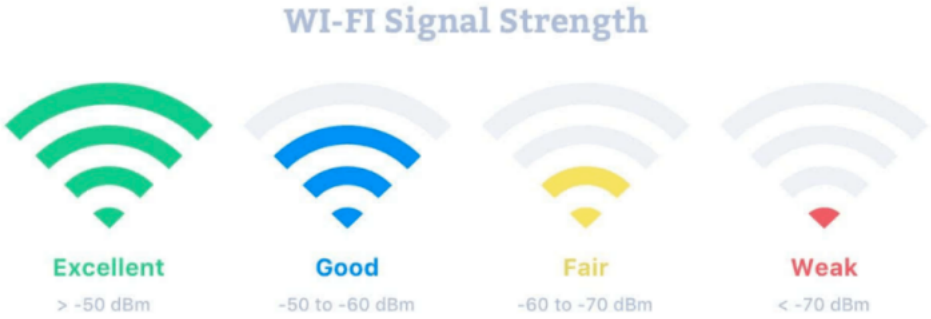
Đồng thời, với bộ trọng số theta mới tìm được, ta thực hiện đánh giá mô hình trên hai bộ dữ liệu val và test sử dụng hàm **evaluate()**.

Yêu cầu bài tập: Dựa vào các bước hướng dẫn cài đặt thuật toán Softmax Regression, các bạn hãy code theo hai phiên bản (1-sample và N-sample). Với 1-sample, các bạn làm theo đúng mô tả trên. Đối với N-sample, các bạn tham khảo lại slide bài giảng phần này và cài đặt lại. Sau đó, huấn luyện và đánh giá với hai phiên bản cài đặt trên hai bộ dữ liệu sau (**Các thư viện được sử dụng:** NumPy, matplotlib):

1. **Bài toán Phát hiện gian lận thẻ tín dụng (Credit Card Fraud Detection)** (tải bộ dữ liệu này tại đây): Để giúp khách hàng tránh việc bất thình lình bị trừ tiền thẻ tín dụng do bị mạo danh, các công ty tín dụng cần phát triển một phần mềm có thể xác định các giao dịch thẻ tín dụng bất thường. Dựa trên bảng dữ liệu cho trước, hãy xây dựng một mô hình phân loại giao dịch thẻ tín dụng sử dụng Softmax Regression?



2. **Bài toán Tìm vị trí người sử dụng Wi-Fi (Wi-Fi User Localization)** (tải bộ dữ liệu này tại đây): từ thống kê Received Signal Strength Indicator (RSSI) (dùng để đo chất lượng sóng nhận được từ các access point. Ở đây cụ thể là router trên một thiết bị) từ 7 router của một chiếc điện thoại và vị trí của điện thoại là 4 căn phòng khác nhau trong nhà. Hãy xây dựng mô hình phân loại dùng Softmax Regression để xác định xem người dùng điện thoại đang ở trong phòng nào?



	router1	router2	router3	router4	router5	router6	router7	location
0	-64	-56	-61	-66	-71	-82	-81	0
1	-68	-57	-61	-65	-71	-85	-85	0
2	-63	-60	-60	-67	-76	-85	-84	0
3	-61	-60	-68	-62	-77	-90	-80	0
4	-63	-65	-60	-63	-77	-81	-87	0

Phần II: Trắc nghiệm

1. Softmax Regression là một thuật toán Machine Learning thuộc nhánh:

   (*a*) Self-supervised Learning
   (*b*) Supervised Learning
   (*c*) Semi-supervised Learning
   (*d*) Unsupervised Learning

2. Softmax Regression thường được dùng để giải quyết dạng bài toán nào dưới đây?

   (*a*) Binary Classification
   (*b*) Multiclass Classification
   (*c*) Cả a, b đều đúng
   (*d*) Cả a, b đều sai

3. Trong các phát biểu sau, phát biểu nào miêu tả đúng về tính chất của hàm Softmax?

   (*a*) Đảm bảo các giá trị đầu ra luôn trong khoảng (0, 1)
   (*b*) Tổng các giá trị đầu ra bằng 1
   (*c*) Giá trị tuyến tính đầu vào càng lớn thì xác suất đầu ra tương ứng càng cao
   (*d*) Tất cả các phương án trên

4. Cho kết quả dự đoán từ mô hình Softmax Regresion trên một mẫu dữ liệu y_hat = (0.4; 0.15; 0.05; 0.4) và nhãn thực tế y = (1; 0; 0; 0). Giá trị loss cross entropy của mô hình là (làm tròn đến hàng thập phân thứ 3):

   (*a*) 0.872
   (*b*) 0.916
   (*c*) 0.943
   (*d*) 0.890

5. Cho bộ giá trị tuyến tính Z = (-1; -2; 3; 2). Sử dụng công thức hàm Softmax, bộ giá trị xác suất đầu ra tương ứng là (làm tròn đến hàng thập phân thứ 3):

   (*a*) P = (0.579; 0.213; 0.078; 0.129)
   (*b*) P = (0.013; 0.005; 0.718; 0.264)
   (*c*) P = (0.024; 0.486; 0.003; 0.486)
   (*d*) P = (0.087; 0.237; 0.644; 0.032)

6. Cho mô hình Softmax Regression nhận đầu vào là 5 đặc trưng, số class cần dự đoán là 3. Không kể bias, tổng số lượng trọng số trong mô hình này là:

   (*a*) 15
   (*b*) 5
   (*c*) 18
   (*d*) 8

7. Cho kết quả dự đoán từ mô hình Softmax Regression y_hat = (0; 1; 3; 2; 0; 2; 1; 2) và kết quả thực tế y = (0; 0; 3; 2; 1; 2; 2; 1) (mỗi giá trị trong ngoặc tương ứng với tên class). Như vậy, độ chính xác (Accuracy) của mô hình trên là:

   (*a*) 75%
   (*b*) 25.5%
   (*c*) 50%
   (*d*) 80%

8. Với số class cần dự đoán là 2, công thức Softmax p nào sau đây là sai (với $z_i = \theta_i^T \cdot X$)?

(a) $p_i = \frac{e^{z_i}}{e^{z_1}+e^{z_2}}$

(b) $p_1 = \frac{1}{1+e^{(\theta_2^T-\theta_1^T)\cdot X}}$

(c) $p_i = \frac{e^{z_i+\ln a}}{\sum_{j=1}^{C} e^{z_j+\ln a}}$ (a là hằng số)

(d) $p_2 = 1 - \frac{e^{z_2}}{e^{z_1}+e^{z_2}}$

9. Với $z = \theta^T \cdot X$ và $p = softmax(z)$, phát biểu nào sau đây là đúng?

(a) $z_i < 0$, $p_i > 0$

(b) $z_1 = z_2$, $p_1 \neq p_2$

(c) $z_1 < z_2$, $p_1 > p_2$

(d) $0 < p_i < 1$ ($\forall p_i \in p$), $\sum p_i \geq 1$

10. Dựa trên chương trình cài đặt Softmax Regression ở phần I (theo chuẩn file gợi ý), trả lời các câu hỏi sau:

(a) Với bộ dữ liệu credit_card_fraud_detection, khi huấn luyện mô hình với cài đặt tham số learning_rate = 1e-3, epoch = 1 phiên bản 1-sample. Giá trị loss và accuracy trả về trên tập val là:

(a) 0.023, 0.975

(b) 0.015, 0.964

(c) 0.011, 0.998

(d) 0.020, 0.947

(b) Với bộ dữ liệu wifi_localization, khi huấn luyện mô hình với cài đặt tham số learning_rate = 1e-3, epoch = 100 phiên bản 1-sample. Giá trị loss và accuracy trả về trên tập test là:

(a) 0.483, 0.966

(b) 0.483, 0.975

(c) 0.459, 0.975

(d) 0.464, 0.966

(c) **(Optional)** Với bộ dữ liệu credit_card_fraud_detection, khi huấn luyện mô hình với cài đặt tham số learning_rate = 1e-3, epoch = 50, batch_size = 128 phiên bản N-sample. Giá trị loss và accuracy trả về trên tập test là:

(a) 0.011, 0.998

(b) 0.025, 0.915

(c) 0.012, 0.962

(d) 0.026, 0.989

(d) **(Optional)** Với bộ dữ liệu wifi_localization, khi huấn luyện mô hình với cài đặt tham số learning_rate = 1e-1, epoch = 500, batch_size = 128 phiên bản N-sample. Giá trị loss và accuracy trả về trên tập val là:

(a) 0.483, 0.966

(b) 0.483, 0.975

(c) 0.464, 0.975

(d) 0.464, 0.912

- Hết -