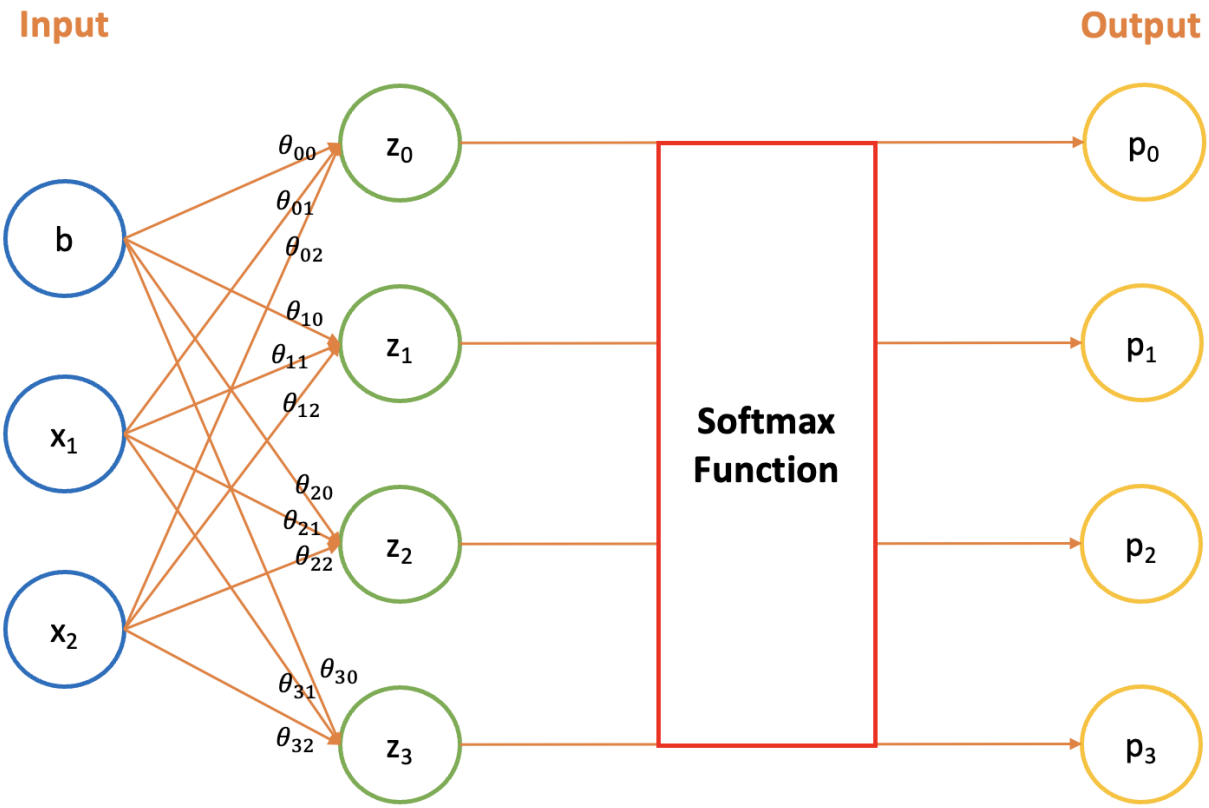
AI VIET NAM – COURSE 2023

Softmax Regression - Exercise

Ngày 27 tháng 10 năm 2023

Phần I: Giới thiệu

Softmax Regression là một trong những thuật toán supervised-learning Machine Learning nền tảng quan trong nhất, được sử dụng để giải quyết bài toán Phân loại đa lớp (Multiclass Classification). Với việc sử dụng hàm Softmax, thuật toán Softmax Regression trả về kết quả đầu ra là phân bố xác suất trên các phân lớp của đề bài. Từ đó, ta có thể sử dụng các ước lượng xác suất này để thực hiện dự đoán một cách linh hoạt.



Trong bài tập này ở phần lập trình, chúng ta sẽ thực hành cài đặt từ đầu quá trình xây dựng một mô hình Softmax Regression, áp dụng vào giải quyết hai bài toán phân loại là Credit Card Fraud Detection (2 class) và Twitter Sentiment Analysis (3 class). Đồng thời, ôn tập một số lý thuyết về Softmax Regression thông qua bài tập trắc nghiệm.

Phần II: Bài tập

A. Phần lập trình

- **Credit Card Fraud Detection**

1. **Tải bộ dữ liệu:** Các bạn tải bộ dữ liệu tại đây.
2. **Import libraries:**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

3. **Đọc dữ liệu:** Sử dụng thư viện pandas để đọc file .csv thành DataFrame như sau:

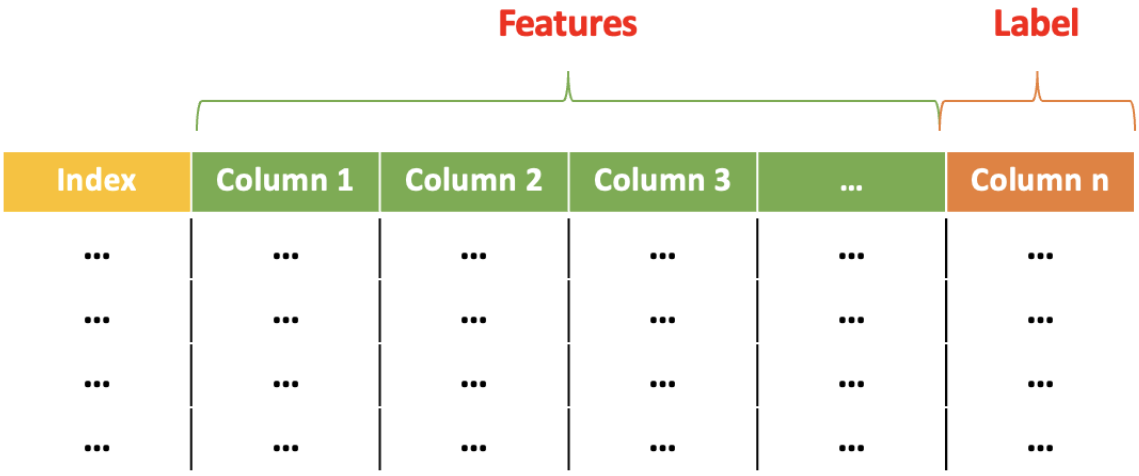
```
dataset_path = 'creditcard.csv'
df = pd.read_csv(
    dataset_path
)
```

...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0
...	...	...	...	...	...	...	...	...	...	...
...	0.213454	0.111864	1.014480	-0.509348	1.436807	0.250034	0.943651	0.823731	0.77	0
...	0.214205	0.924384	0.012463	-1.016226	-0.606624	-0.395255	0.068472	-0.053527	24.79	0
...	0.232045	0.578229	-0.037501	0.640134	0.265745	-0.087371	0.004455	-0.026561	67.88	0
...	0.265245	0.800049	-0.163298	0.123205	-0.569159	0.546668	0.108821	0.104533	10.00	0
...	0.261057	0.643078	0.376777	0.008797	-0.473649	-0.818267	-0.002415	0.013649	217.00	0

Hình 1: Một phần DataFrame của bộ dữ liệu Card Fraud Detection.

4. **Tách biến X, y:** Chuyển đổi DataFrame hiện tại thành array và tách hai biến X, y:

```
dataset_arr = df.to_numpy()
X, y = dataset_arr[:, :-1].astype(np.float64), dataset_arr[:, -1].astype(np.
    uint8)
```

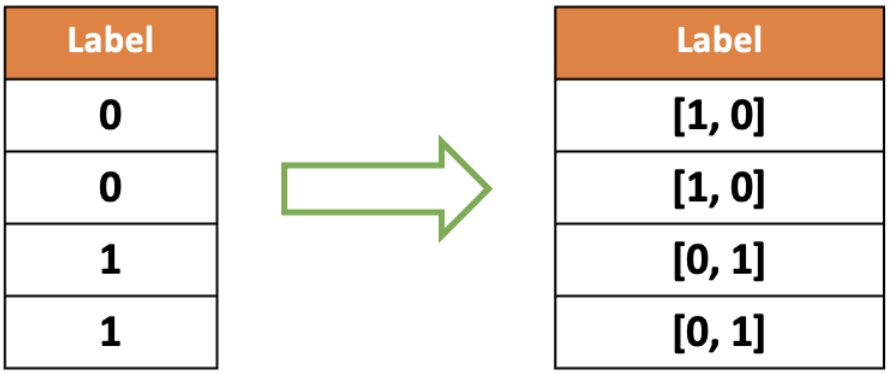


Hình 2: Mô phỏng việc tách biến X và y từ bộ dữ liệu gốc.

5. **Thêm bias vào X:** Khi sử dụng thư viện, bias sẽ được thêm tự động vào X. Tuy nhiên, khi triển khai lại từ đầu, chúng ta cần phải tự thêm bias vào mỗi mẫu dữ liệu, nhằm thỏa mãn công thức hàm dự đoán:

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

6. **One-hot encoding label:** Đối với Softmax Regression, sau khi có biến chứa nhãn dữ liệu y, ta 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]]. Như vậy, ta cài đặt one-hot encoding cho bài này như sau:

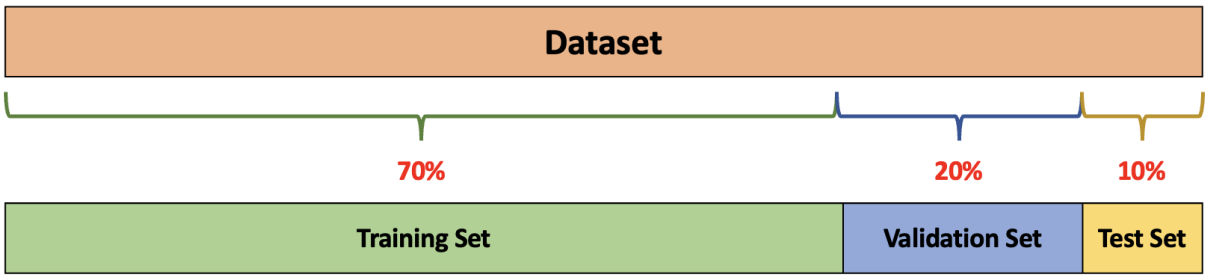


Hình 3: Mô phỏng việc one-hot encoding cho label.

```
n_classes = np.unique(y, axis=0).shape[0]
n_samples = y.shape[0]

y_encoded = np.array(
    [np.zeros(n_classes) for _ in range(n_samples)]
)
y_encoded[np.arange(n_samples), y] = 1
```

7. **Chia tập train, val, test:** Sau khi đã hoàn chỉnh biến X, chúng ta tiến hành chia ba bộ train, val, test với tỉ lệ 7:2:1. Thực hiện như sau:

```
val_size = 0.2
test_size = 0.125
random_state = 2
is_shuffle = True

X_train, X_val, y_train, y_val = train_test_split(
    X_b, y_encoded,
    test_size=val_size,
    random_state=random_state,
    shuffle=is_shuffle
)

X_train, X_test, y_train, y_test = train_test_split(
    X_train, y_train,
    test_size=test_size,
    random_state=random_state,
    shuffle=is_shuffle
)
```



Hình 4: Mô phỏng chia bộ dữ liệu gốc thành ba bộ train, val, test với tỉ lệ 7:2:1.

8. **Chuẩn hóa dữ liệu:** Ta sử dụng X_train vừa tạo ở bước trên fit vào hàm chuẩn hóa StandardScaler. Sau đó, đem scaler này chuẩn hóa cho tập X_val và X_test (lưu ý rằng ta không chuẩn hóa bias nên sẽ bỏ qua cột đầu tiên trong X):

```
normalizer = StandardScaler()
X_train[:, 1:] = normalizer.fit_transform(X_train[:, 1:])
X_val[:, 1:] = normalizer.transform(X_val[:, 1:])
X_test[:, 1:] = normalizer.transform(X_test[:, 1:])
```

9. **Cài đặt các hàm quan trọng:** Để thuận tiện trong việc cài đặt chương trình, ta định nghĩa sẵn một số hàm sẽ được dùng trong quá trình huấn luyện mô hình:
   - **Hàm softmax:** Xây dựng hàm softmax với công thức như sau:

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

Trong đó: C là số class.

```
def softmax(z):
    exp_z = np.exp(z)

    return exp_z / exp_z.sum(axis=1)[:, None]
```

– **Hàm dự đoán:**

```
def predict(X, theta):
    z = np.dot(X, theta)
    y_hat = softmax(z)

    return y_hat
```

– **Hàm tính loss:** Xây dựng hàm tính loss với công thức Cross-entropy như sau:

$$L(\hat{y}, y) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log \hat{y}_i$$

```
def compute_loss(y_hat, y):
    n = y.size

    return (-1 / n) * np.sum(y * np.log(y_hat))
```

– **Hàm tính gradient:** Xây dựng hàm tính gradient với công thức như sau:

$$\nabla_\theta L(X, y, y\_hat) = \frac{X^T \cdot (y\_hat - y)}{N}$$

```
def compute_gradient(X, y, y_hat):
    n = y.size

    return np.dot(X.T, (y_hat - y)) / n
```

– **Hàm cập nhật trọng số:** Khi áp dụng giải thuật Gradient Descent, trọng số θ sẽ được cập nhật bằng công thức như sau:

$$\theta = \theta - \eta \nabla_\theta L$$

Trong đó: η là learning rate.

```
def update_theta(theta, gradient, lr):
    return theta - lr * gradient
```

– **Hàm tính độ chính xác:** Xây dựng hàm tính độ chính xác với công thức như sau:

$$accuracy = \frac{\text{Số lần dự đoán đúng}}{\text{Tổng số lần dự đoán}}$$

```
def compute_accuracy(X, y, theta):
    y_hat = predict(X, theta)
    acc = (np.argmax(y_hat, axis=1) == np.argmax(y, axis=1)).mean()

    return acc
```

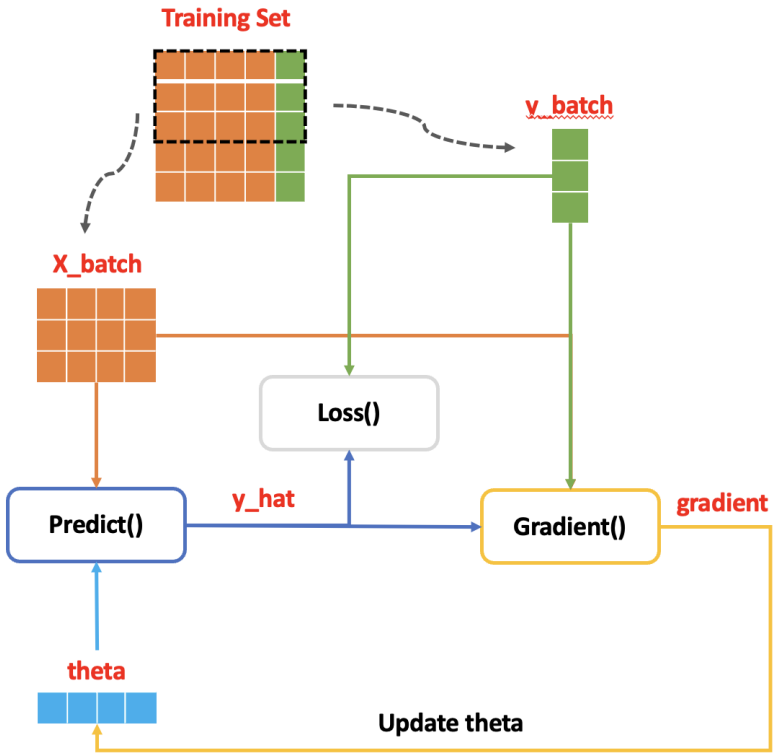
Ở bản cài đặt cho Softmax Regression, vì kết quả dự đoán là một vector phân bố xác suất, ta sẽ sử dụng hàm argmax để đưa dự đoán về class ID có ước lượng xác suất cao nhất.

10. **Khai báo các siêu tham số và khởi tạo weights:**

```
lr = 0.01
epochs = 30
batch_size = 1024
n_features = X_train.shape[1]

np.random.seed(random_state)
theta = np.random.uniform(
    size=(n_features, n_classes)
)
```

Lưu ý rằng, với Softmax Regression, mỗi class sẽ có một bộ trọng số riêng ứng với các đặc trưng đầu vào (bao gồm cả bias). Vì vậy, ta cần khởi tạo trọng số θ với shape = (n_features, n_classes).

11. **Huấn luyện mô hình:** Chúng ta sẽ triển khai quá trình huấn luyện mô hình sử dụng thuật toán Gradient Descent, với ý tưởng chính như sau: Khởi tạo vòng lặp với số lần lặp bằng số **epochs**. Với mỗi lần lặp, duyệt qua toàn bộ mẫu dữ liệu (trong training set) theo từng bộ mẫu dữ liệu có kích thước **batch_size** (tạm gọi là cặp **X_i** và **y_i**) và thực hiện các bước tính toán sau:
    (a) Tính **y_hat** sử dụng hàm **predict(X_i, theta)**. Đây là kết quả dự đoán của mô hình với các mẫu dữ liệu tại batch đang xét.
    (b) Tính **loss** sử dụng hàm **compute_loss(y_hat, y_i)**. Lưu trữ giá trị này vào một list **batch_losses**, dùng cho việc trực quan hóa kết quả huấn luyện sau này.
    (c) Tính **gradient** sử dụng hàm **compute_gradient(X_i, y_i, y_hat)**.
    (d) Sử dụng kết quả gradient vừa tìm được để cập nhật bộ trọng số **theta** sử dụng hàm **update_theta(theta, gradient, lr)**.



Hình 5: Mô tả quá trình huấn luyện mô hình Softmax Regression sử dụng Gradient Descent.

Tổng kết lại, chúng ta sẽ có toàn bộ code cài đặt như sau:

```
train_accs = []
train_losses = []
val_accs = []
val_losses = []

for epoch in range(epochs):
    train_batch_losses = []
    train_batch_accs = []
    val_batch_losses = []
    val_batch_accs = []

    for i in range(0, X_train.shape[0], batch_size):
        X_i = X_train[i:i+batch_size]
        y_i = y_train[i:i+batch_size]

        y_hat = predict(X_i, theta)

        train_loss = compute_loss(y_hat, y_i)

        gradient = compute_gradient(X_i, y_i, y_hat)

        theta = update_theta(theta, gradient, lr)


        train_batch_losses.append(train_loss)

        train_acc = compute_accuracy(X_train, y_train, theta)
        train_batch_accs.append(train_acc)

        y_val_hat = predict(X_val, theta)
        val_loss = compute_loss(y_val_hat, y_val)
        val_batch_losses.append(val_loss)

        val_acc = compute_accuracy(X_val, y_val, theta)
        val_batch_accs.append(val_acc)

    train_batch_loss = sum(train_batch_losses) / len(train_batch_losses)
    val_batch_loss = sum(val_batch_losses) / len(val_batch_losses)
    train_batch_acc = sum(train_batch_accs) / len(train_batch_accs)
    val_batch_acc = sum(val_batch_accs) / len(val_batch_accs)

    train_losses.append(train_batch_loss)
    val_losses.append(val_batch_loss)
    train_accs.append(train_batch_acc)
    val_accs.append(val_batch_acc)

    print(f'\nEPOCH {epoch + 1}:\tTraining loss: {train_batch_loss:.3f}\
    tValidation loss: {val_batch_loss:.3f}')
```

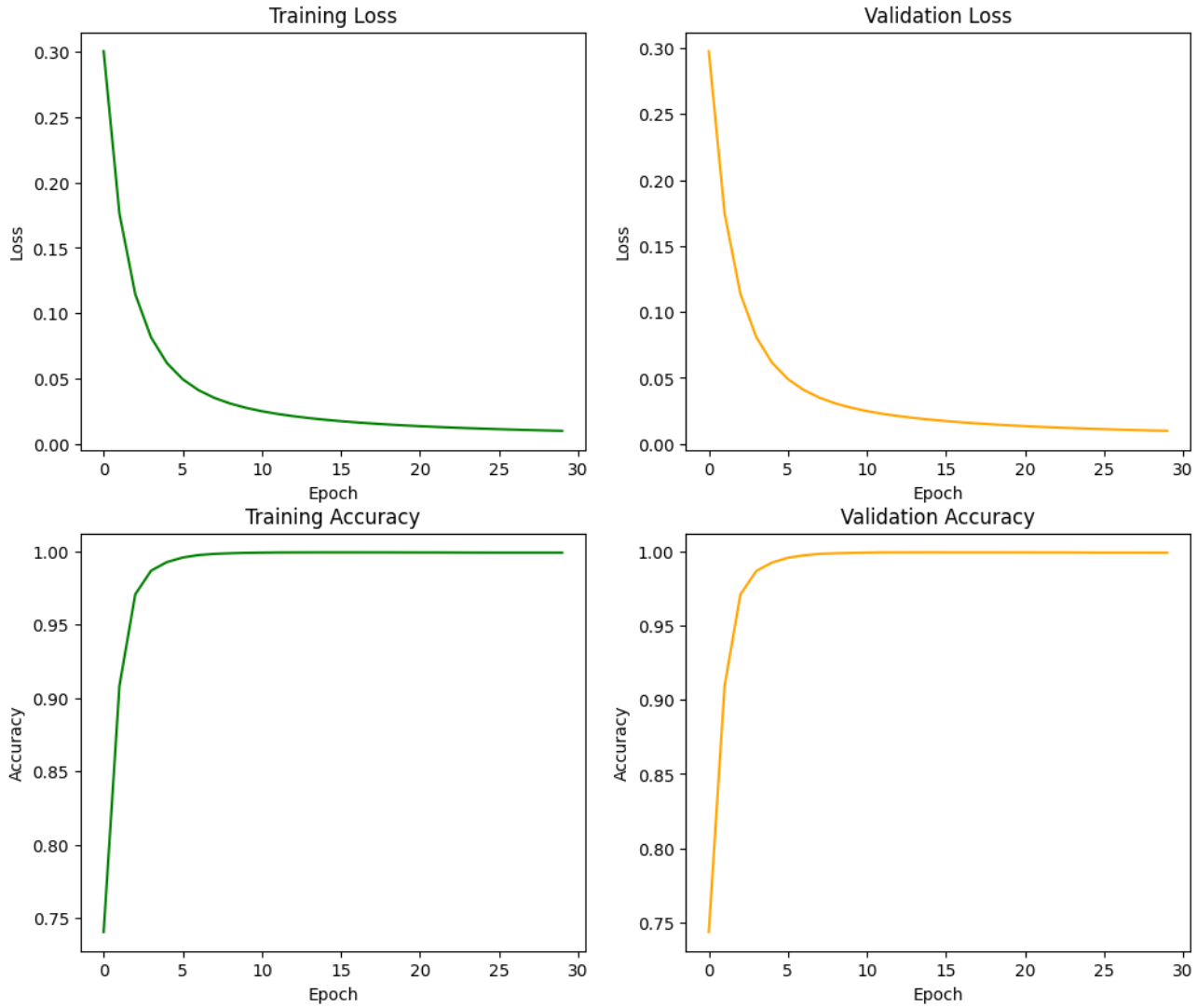
Khi chạy thuật toán, nếu các bạn quan sát thấy giá trị loss giảm và độ chính xác tăng dần khi số epoch tăng, điều đó là dấu hiệu cho thấy code huấn luyện mô hình của chúng ta hoạt động ổn.

```
EPOCH 20:       Training loss: 0.014    Validation loss: 0.014

EPOCH 21:       Training loss: 0.014    Validation loss: 0.013

EPOCH 22:       Training loss: 0.013    Validation loss: 0.013

EPOCH 23:       Training loss: 0.013    Validation loss: 0.012

EPOCH 24:       Training loss: 0.012    Validation loss: 0.012

EPOCH 25:       Training loss: 0.012    Validation loss: 0.011

EPOCH 26:       Training loss: 0.011    Validation loss: 0.011

EPOCH 27:       Training loss: 0.011    Validation loss: 0.011

EPOCH 28:       Training loss: 0.011    Validation loss: 0.010

EPOCH 29:       Training loss: 0.010    Validation loss: 0.010

EPOCH 30:       Training loss: 0.010    Validation loss: 0.010
```

Hình 6: Kết quả huấn luyện in trên màn hình ở những epoch cuối cùng

Bên cạnh đó, với các danh sách batch loss và batch accuracy trên hai bộ dữ liệu train và val, chúng ta còn có thể trực quan hóa kết quả huấn luyện lên đồ thị như sau:

```
fig, ax = plt.subplots(2, 2, figsize=(12, 10))
ax[0, 0].plot(train_losses)
ax[0, 0].set(xlabel='Epoch', ylabel='Loss')
ax[0, 0].set_title('Training Loss')

ax[0, 1].plot(val_losses, 'orange')
ax[0, 1].set(xlabel='Epoch', ylabel='Loss')
ax[0, 1].set_title('Validation Loss')

ax[1, 0].plot(train_accs)
ax[1, 0].set(xlabel='Epoch', ylabel='Accuracy')
ax[1, 0].set_title('Training Accuracy')

ax[1, 1].plot(val_accs, 'orange')
ax[1, 1].set(xlabel='Epoch', ylabel='Accuracy')
ax[1, 1].set_title('Validation Accuracy')

plt.show()
```



Hình 7: Hình ảnh trực quan kết quả huấn luyện trên tập train và val cho bài Card Fraud Detection.

12. **Đánh giá mô hình:** Sử dụng bộ trọng số mô hình tìm được sau quá trình huấn luyện, ta đánh giá độ chính xác của mô hình trên hai tập val và test:

```
val_set_acc = compute_accuracy(X_val, y_val, theta)
test_set_acc = compute_accuracy(X_test, y_test, theta)
print('Evaluation on validation and test set:')
print(f'Accuracy: {val_set_acc}')
print(f'Accuracy: {test_set_acc}')
```

- **Twitter Sentiment Analysis**

1. **Tải bộ dữ liệu:** Các bạn tải bộ dữ liệu tại đây.
2. **Import libraries:**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import re
import nltk
nltk.download('stopwords')

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer
```

3. **Đọc bộ dữ liệu:** Sử dụng thư viện pandas để đọc file .csv thành DataFrame:

```
dataset_path = 'Twitter_Data.csv'
df = pd.read_csv(
    dataset_path
)
df
```

	clean_text	category
0	when modi promised “minimum government maximum...	-1.0
1	talk all the nonsense and continue all the dra...	0.0
2	what did just say vote for modi welcome bjp t...	1.0
3	asking his supporters prefix chowkidar their n...	1.0
4	answer who among these the most powerful world...	1.0
...	...	...
162975	why these 456 crores paid neerav modi not reco...	-1.0
162976	dear rss terrorist payal gawar what about modi...	-1.0
162977	did you cover her interaction forum where she ...	0.0
162978	there big project came into india modi dream p...	0.0
162979	have you ever listen about like gurukul where ...	1.0

162980 rows × 2 columns

Hình 8: DataFrame của bộ dữ liệu Twitter Sentiment Analysis.

4. **Xóa missing data:** Bộ dữ liệu này có tồn tại một vài hàng chứa giá trị null:

	clean_text	category
148	NaN	0.0
130448	the foundation stone northeast gas grid inaugu...	NaN
155642	dear terrorists you can run but you cant hide ...	NaN
155698	offense the best defence with mission shakti m...	NaN
155770	have always heard politicians backing out thei...	NaN
158693	modi government plans felicitate the faceless ...	NaN
158694	NaN	-1.0
159442	chidambaram gives praises modinomics	NaN
159443	NaN	0.0
160559	the reason why modi contested from seats 2014 ...	NaN
160560	NaN	1.0

Hình 9: Các mẫu dữ liệu null trong bộ dữ liệu.

Vì số lượng mẫu dữ liệu là rất lớn, nên phương án tiện nhất là ta sẽ loại bỏ các hàng này đi như sau:

```
df = df.dropna()
```

5. **Tiền xử lý bộ dữ liệu:** Dữ liệu đầu vào của chúng ta lúc này hiện đang ở dạng văn bản (string), chưa có đặc trưng rõ ràng cũng như không thể đưa vào huấn luyện mô hình được. Vì vậy, chúng ta sẽ tiền xử lý dữ liệu văn bản đầu vào để đưa về một dạng vector đặc trưng nào đó:

   (a) **Xây dựng hàm chuẩn hóa văn bản:** Văn bản gốc có rất nhiều kí tự thừa thải, vô nghĩa... Vì vậy, ta cần loại bỏ chúng cũng như áp dụng thêm vài các bước chuẩn hóa văn bản khác để văn bản đầu vào trở nên ít phức tạp hơn, nhằm tăng cường hiệu quả biểu diễn của vector đặc trưng sau này:

```
def text_normalize(text):
    # Lowercasing
    text = text.lower()

    # Retweet old acronym "RT" removal
    text = re.sub(r'^rt[\s]+', '', text)

    # Hyperlinks removal
    text = re.sub(r'https?:\/\/.*[\r\n]*', '', text)

    # Punctuation removal
    text = re.sub(r'[^\w\s]', '', text)

    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    words = text.split()
    words = [word for word in words if word not in stop_words]
    text = ' '.join(words)
```

```
19
20      # Stemming
21      stemmer = SnowballStemmer('english')
22      words = text.split()
23      words = [stemmer.stem(word) for word in words]
24      text = ' '.join(words)
25
26      return text
```

Trong đó:

- **Dòng 1:** Khai báo hàm text_normalize() nhận đầu vào là một string (text).
- Dòng 5, 6: Loại bỏ các từ "RT" trong text (đây là một cụm từ viết tắt cũ cho "Retweet").
- **Dòng 8, 9:** Loại bỏ các đường dẫn trong text.
- **Dòng 11, 12:** Loại bỏ các dấu câu.
- **Dòng 14, 15, 16, 17, 18:** Loại bỏ stopwords.
- **Dòng 20, 21, 22, 23, 24:** Thực hiện kỹ thuật stemming.
- **Dòng 22:** Trả về văn bản đã được chuẩn hóa.

(b) **Khởi tạo tf-idf vectorizer:** Trong bài này, chúng ta sẽ sử dụng một dạng vector biểu diễn đặc trưng mới cho văn bản, đó là tf-idf. Trong phạm vi của bài, chúng ta sẽ không đi sâu vào kỹ thuật tf-idf, các bạn có thể đọc thêm về kỹ thuật này tại đây. Code biến đổi văn bản đầu vào thành vector tf-idf như sau:

```
vectorizer = TfidfVectorizer(max_features=2000)
X = vectorizer.fit_transform(df['clean_text']).toarray()
```

Trong đó:

- **Dòng 1:** Khai báo TfidfVectorizer với số lượng đặc trưng tối đa biểu diễn là 2000.
- **Dòng 2:** Thực hiện fit vetorizer với cột **'clean_text'**, sau đó transform cột này và gán vào biến X.

(c) **Thêm bias vào X:** Thực hiện tương tự như bài Card Fraud Detection:

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

6. **One-hot encoding label:** Thực hiện tương tự như ở bài trước:

```
n_classes = df['category'].nunique()
n_samples = df['category'].size

y = df['category'].to_numpy() + 1
y = y.astype(np.uint8)
y_encoded = np.array(
    [np.zeros(n_classes) for _ in range(n_samples)]
)
y_encoded[np.arange(n_samples), y] = 1
```

7. **Chia bộ train, val, test:** Thực hiện tương tự như bài Card Fraud Detection.

```
val_size = 0.2
test_size = 0.125
random_state = 2
```

```
is_shuffle = True

X_train, X_val, y_train, y_val = train_test_split(
    X_b, y_encoded,
    test_size=val_size,
    random_state=random_state,
    shuffle=is_shuffle
)

X_train, X_test, y_train, y_test = train_test_split(
    X_train, y_train,
    test_size=test_size,
    random_state=random_state,
    shuffle=is_shuffle
)
```

8. **Cài đặt các hàm quan trọng:** Sử dụng lại các hàm đã định nghĩa trong bài Card Fraud Detection.

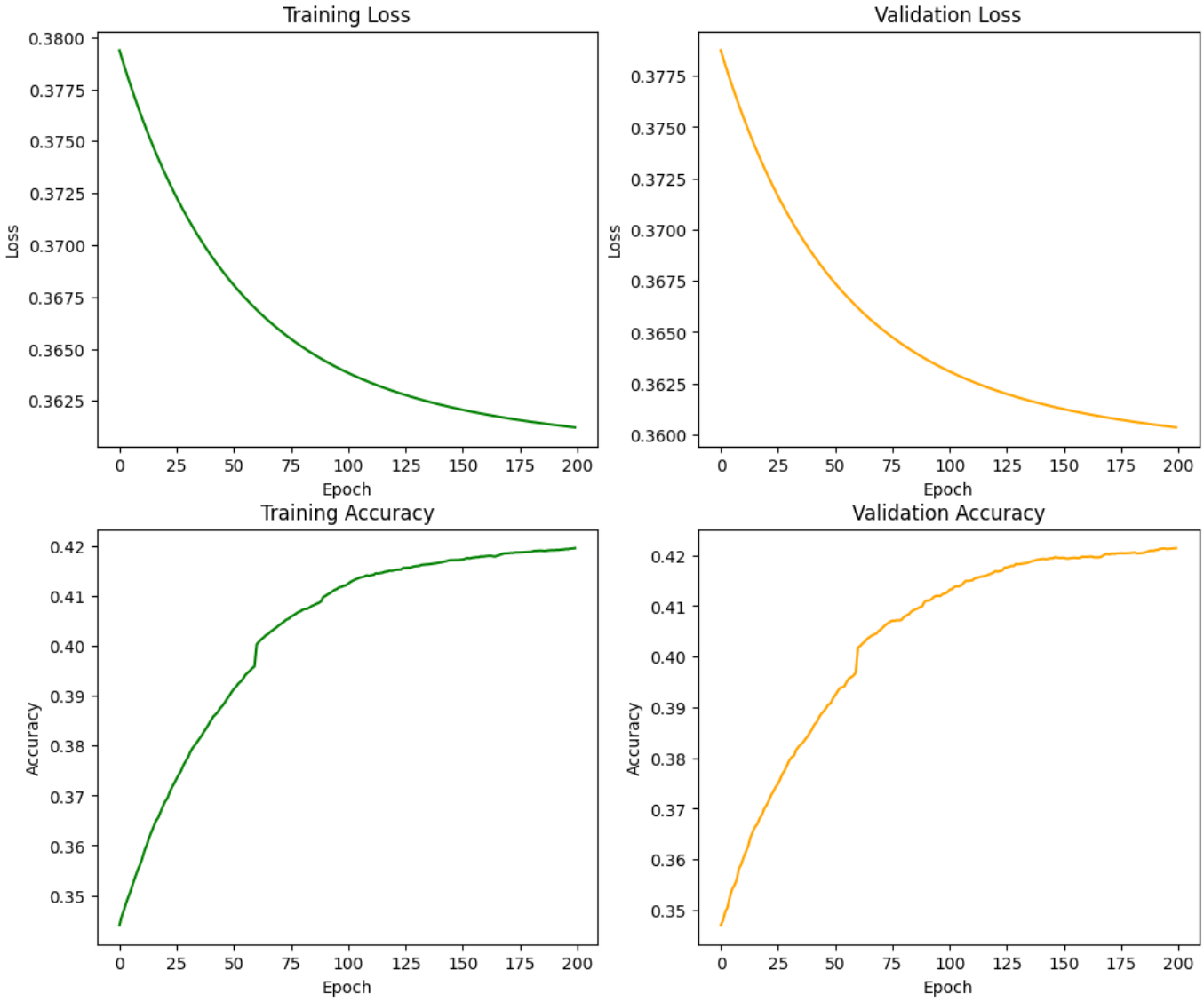
```
def softmax(z):
    exp_z = np.exp(z)

    return exp_z / exp_z.sum(axis=1)[:, None]

def compute_loss(y_hat, y):
    n = y.size

    return (-1 / n) * np.sum(y * np.log(y_hat))

def predict(X, theta):
    z = np.dot(X, theta)
    y_hat = softmax(z)

    return y_hat

def compute_gradient(X, y, y_hat):
    n = y.size

    return np.dot(X.T, (y_hat - y)) / n

def update_theta(theta, gradient, lr):
    return theta - lr * gradient

def compute_accuracy(X, y, theta):
    y_hat = predict(X, theta)
    acc = (np.argmax(y_hat, axis=1) == np.argmax(y, axis=1)).mean()

    return acc
```

9. **Khai báo các siêu tham số và khởi tạo weights:** Trong bài này, vì số lượng mẫu dữ liệu là rất lớn, ta có thể cân nhắc tăng số batch size lên để tăng tốc độ huấn luyện (ví dụ ở đây ta cài batch_size=n_samples).

```
lr = 0.1
epochs = 200
batch_size = X_train.shape[0]
n_features = X_train.shape[1]

np.random.seed(random_state)
theta = np.random.uniform(
    size=(n_features, n_classes)
```

```
)
```

10. **Huấn luyện mô hình:** Sử dụng code huấn luyện tương tự như bài Card Fraud Detection. Kết quả của quá trình huấn luyện được trực quan trên đồ thị như sau:



Hình 10: Hình ảnh trực quan kết quả huấn luyện trên tập train và val cho bài Twitter Sentiment Analysis.

11. **Đánh giá mô hình:** Sử dụng code đánh giá tương tự như bài Card Fraud Detection:

```
val_set_acc = compute_accuracy(X_val, y_val, theta)
test_set_acc = compute_accuracy(X_test, y_test, theta)
print('Evaluation on validation and test set:')
print(f'Accuracy: {val_set_acc}')
print(f'Accuracy: {test_set_acc}')
```

B. Phần 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. Softmax Regression còn có tên gọi khác là?

   (*a*) Logistic Regression
   (*b*) Linear Regression
   (*c*) Multinomial Logistic Regression
   (*d*) Multinomial Linear Regression

4. 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.

5. 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

6. 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)

7. 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

8. 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%

9. 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) $p_2 = 1 - \frac{e^{z_2}}{e^{z_1}+e^{z_2}}$

10. 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) $0 < p_i < 1$ ($\forall p_i \in p$), $\sum p_i \geq 1$

- Hết -