Tuần 1 – Phân loại SMS Spam bằng MLP: Từ Activation Function đến F1-Score

🧩 Giới thiệu

Bạn đã từng nhận được những tin nhắn SMS như:

> “Chúc mừng bạn đã trúng xe máy SH!”
> “Click ngay để nhận ưu đãi trị giá 50 triệu đồng!”

Rõ ràng đó là **spam**. Nhưng làm thế nào để máy tính cũng phân biệt được điều đó một cách **tự động và chính xác** như con người? Trong tuần học này, mình đã áp dụng kiến thức về **activation function** và **F1-Score** để xây dựng một mô hình AI có thể **phân loại SMS là spam hay ham (bình thường)**.

Đây không chỉ là bài thực hành, mà còn mở ra tiềm năng ứng dụng thực tế như:

- Tự động lọc tin rác,
- Bảo vệ người dùng khỏi lừa đảo,
- Phân loại email, bình luận, tin nhắn trong các hệ thống lớn.

🧠 Kiến thức chính đã học

1. Activation Function – Trái tim của mạng nơ-ron

Các **activation function** quyết định xem một "nơ-ron" có được kích hoạt hay không. Chúng tạo ra tính phi tuyến cần thiết để mô hình học được mối quan hệ phức tạp trong dữ liệu.

Activation	Công thức/Đặc điểm	Ưu điểm	Nhược điểm
Sigmoid	$\sigma(x) = \frac{1}{1 + e^{-x}}$	Dễ hiểu, cho đầu ra (0,1)	Dễ bị **vanishing gradient**
ReLU	$\max(0, x)$	Nhanh, đơn giản, phổ biến	“Dying ReLU” nếu đầu vào luôn âm
ELU	$\text{ELU}(x) = x \text{ if } x>0, \alpha(e^x - 1) \text{ if } x<=0$	Giảm dead neuron	Tốn phép tính hơn
Softmax	Chuyển logits → xác suất tổng = 1	Dùng cho phân loại đa lớp	Phụ thuộc vào toàn bộ vector

2. F1-Score – Thước đo công bằng

Khi phân loại spam/ham, ta thường gặp **dữ liệu lệch class**: rất nhiều tin nhắn “ham” và ít “spam”. Do đó:

- Nếu chỉ dùng **accuracy**, mô hình dễ "ăn gian" bằng cách luôn dự đoán “ham”.
- **F1-Score** giải quyết bằng cách cân bằng giữa:

- **Precision**: phần dự đoán spam có đúng là spam không?
- **Recall**: phần spam thật, mô hình bắt được bao nhiêu?

[F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}]

F1 cao khi mô hình **vừa phát hiện đủ** (recall), vừa **không nhầm nhiều** (precision).

🩹 Ứng dụng thực tế: Phân loại SMS spam với MLPClassifier

1. Dataset từ đâu?

Ban đầu, mình sử dụng **dữ liệu công khai** từ UCI & Kaggle:

> Tập SMS Spam Collection gồm ~5.500 tin nhắn đã được con người gán nhãn "spam" hoặc "ham".

2. Mô hình nào?

Mình dùng `MLPClassifier` – một **mạng nơ-ron nhiều lớp (Multi-Layer Perceptron)** có thể học các quan hệ phi tuyến giữa nội dung SMS và nhãn.

Không bắt buộc dùng MLP, bạn có thể thay bằng Logistic Regression, Random Forest, SVM, v.v. Nhưng MLP có ưu thế là:

- Hỗ trợ nhiều activation,
- Linh hoạt với dữ liệu phi tuyến như văn bản.

3. Quy trình huấn luyện:

```
from sklearn.neural_network import MLPClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import f1_score

# TF-IDF vectorization
vectorizer = TfidfVectorizer(max_features=2000)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# Train MLP
clf = MLPClassifier(hidden_layer_sizes=(100,), activation='relu', max_iter=300)
clf.fit(X_train_vec, y_train)
y_pred = clf.predict(X_test_vec)

# Evaluate
f1 = f1_score(y_test, y_pred)
print(f"F1-Score: {f1:.4f}")
```

4. Phân loại real-time?

Mình triển khai API bằng **FastAPI**, nhận nội dung SMS từ hệ thống SMS Gateway (ví dụ: Twilio), sau đó:

- Vector hóa nội dung,
- Dùng mô hình phân loại,
- Nếu là spam → không forward tin nhắn,
- Nếu là ham → forward đến người nhận.

Tại thời điểm webhook, SMS **chưa được gửi đến người dùng cuối**, nên bạn hoàn toàn có thể "chặn" hoặc "cho qua" tùy vào nhãn phân loại.

5. Mô hình phân biệt spam/ham dựa vào đâu?

- Giai đoạn huấn luyện: từ dữ liệu đã được gán nhãn.
- Khi vận hành: từ mô hình đã học → phân loại tin nhắn mới.
- Dữ liệu mới (SMS + phản hồi của người dùng) được lưu lại → dùng để retrain định kỳ.

📊 Phân tích kết quả & bài học rút ra

Ưu điểm:

- **MLP + ReLU** hội tụ nhanh, đơn giản.
- **F1-Score** là chỉ số tin cậy trong bài toán phân loại spam thật sự.
- API hoạt động **real-time**, đủ nhanh để xử lý tin nhắn ngay khi đến.

Nhược điểm:

- Mô hình cần retrain định kỳ vì spam thay đổi liên tục.
- MLP khá "đen hộp", khó giải thích vì là mạng nơ-ron.

Điều thú vị:

- SMS nhìn đơn giản nhưng thực chất rất phức tạp về ngôn ngữ, kiểu viết (viết tắt, dấu chấm, link ẩn).
- Việc thu thập phản hồi từ người dùng giúp mô hình **ngày càng thông minh hơn**.

🚀 Kết luận & mở rộng

Tổng kết:

Tuần 1 không chỉ là luyện tập activation function và F1-Score lý thuyết. Mình đã áp dụng vào **project thật**:

- Hiểu activation ảnh hưởng đến học như thế nào,
- Biết dùng F1 để đánh giá mô hình cân bằng,
- Triển khai API nhận SMS, phân loại và phản hồi real-time.

Mở rộng:

- Dùng thêm **LeakyReLU**, **ELU** hoặc các mô hình khác như `XGBoost`.
- Tích hợp phản hồi người dùng và thiết lập **pipeline tự động retrain mô hình**.
- Ứng dụng tương tự vào email, bình luận mạng xã hội, phát hiện gian lận giao dịch,...

🎯 Tuần 1 có thể là nền tảng – nhưng với một chút ứng dụng thực tế, bạn sẽ thấy AI thật sự chạm vào cuộc sống mỗi ngày.

🛠️ Mở rộng thực hành nâng cao

1. Test API với Postman

Sau khi khởi chạy FastAPI (`uvicorn main_api:app --reload`), bạn có thể test phân loại tin nhắn SMS bằng Postman như sau:

- **Method**: POST
- **Endpoint**: `http://localhost:8000/classify`
- **Body (JSON)**:

```
{
  "message": "Bạn đã trúng thưởng! Click ngay để nhận quà"
}
```

- **Response**:

```
{
  "label": "spam",
  "score": 0.9213
}
```

2. Lưu kết quả phân loại vào MySQL

Thay vì dùng PostgreSQL, bạn có thể thay cấu hình kết nối sang MySQL:

- Cài thêm thư viện:

```
pip install mysql-connector-python
```

- Tạo bảng MySQL:

```
CREATE TABLE sms_log (
    id INT AUTO_INCREMENT PRIMARY KEY,
    content TEXT,
    predicted_label VARCHAR(10),
    confidence FLOAT,
    user_feedback VARCHAR(10),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

- Cập nhật `create_engine` trong code:

```
from sqlalchemy import create_engine
engine = create_engine("mysql+mysqlconnector://user:password@localhost/smsdb")
```

3. Huấn luyện lại mô hình mỗi 24 giờ

Bạn có thể dùng cronjob hoặc scheduler để retrain mô hình mỗi ngày:

Cronjob mẫu trên Linux:

```
0 2 * * * /usr/bin/python3 /path/to/retrain_model.py >> /var/log/retrain.log 2>&1
```

- `0 2 * * *`: chạy lúc 2h sáng mỗi ngày.
- `retrain_model.py`: script đã viết để load dữ liệu mới từ `sms_log`, train lại model, đánh giá F1 và ghi đè file `.joblib`.

Sau khi retrain, FastAPI sẽ tự nhận diện file mô hình mới nhờ `mtime` và tự động reload.

☑ Với các bước này, bạn đã có hệ thống phân loại spam **tự động, có lưu log, có học lại và test nhanh** bằng Postman hoặc giao diện bạn tự xây dựng.