TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

★★★

BÁO CÁO
ĐỒ ÁN CUỐI KỲ

Học phần: Nhập môn học máy

Lớp học phần: CSC14005_21KHMT2

MSSV:	**Thành viên nhóm:**
21127619	Phạm Gia Tuấn Khải
21127210	Lê Sử Triều An

Giảng viên:
Bùi Duy Đăng
Phạm Trọng Nghĩa
Thầy Nguyễn Ngọc Đức

TP. Hồ Chí Minh – Năm 2023

MỤC LỤC

NỘI DUNG

I. Lý thuyết về mô hình SVM:

1. Giới thiệu về SVM

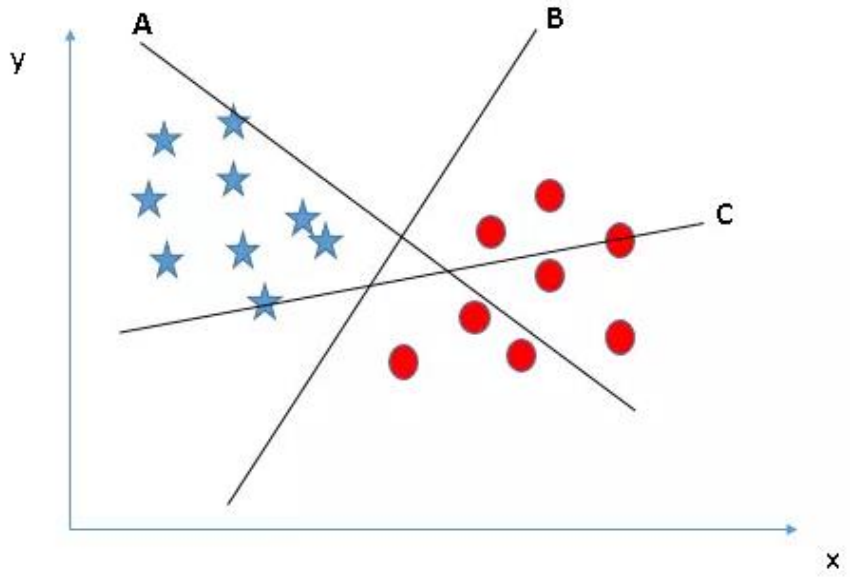
Trong lĩnh vực học máy, SVM (Support Vector Machine) còn được biết đến là mô hình giới hạn cực đại có giải thuật học liên quan, đã nổi bật như là một công cụ mạnh mẽ cho việc phân tích dữ liệu trong cả hai lĩnh vực phân loại và hồi quy. Được phát triển tại AT&T Bell Laboratories bởi Vladimir Vapnik và đồng nghiệp. SVM đã trở thành một trong những mô hình được nghiên cứu sâu sắc nhất, lấy cảm hứng từ các khung lý thuyết học thống kê và lý thuyết VC do Vapnik (1982, 1995) và Chervonenkis (1974) đề xuất.

2. Mục tiêu

Mục tiêu chính của SVM là tìm ra ranh giới quyết định (decision boundary) tốt nhất để phân chia các điểm dữ liệu thuộc các lớp khác nhau. Đối với trường hợp phân loại nhị phân, ranh giới quyết định này là một siêu phẳng (hyperplane) trong không gian đa chiều, tối ưu hóa khoảng cách giữa nó và các điểm dữ liệu thuộc hai lớp.

3. Các nguyên tắc hoạt động của SVM

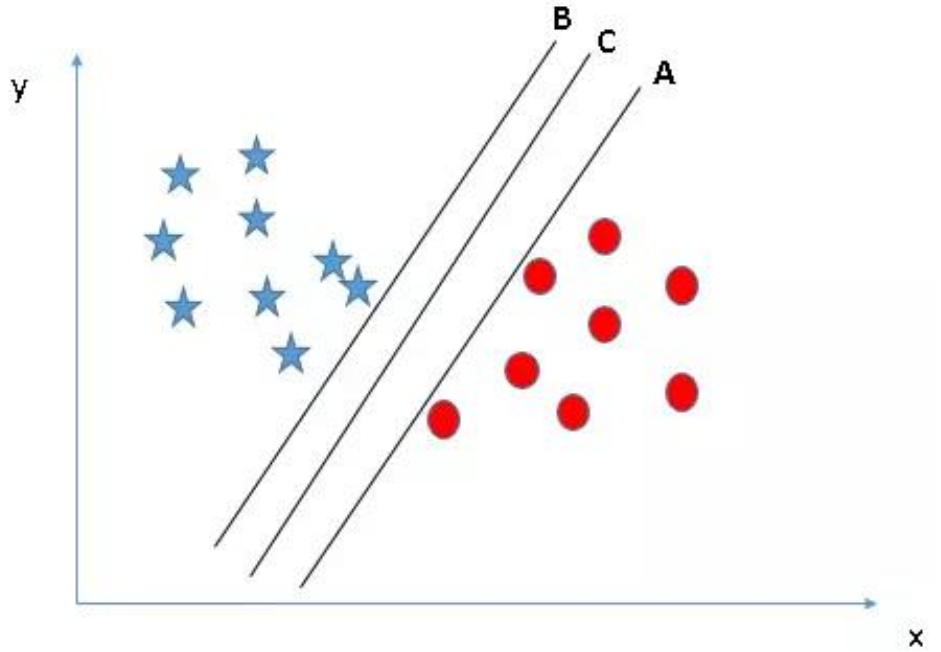
Xem xét một ví dụ: Ở đây, có 3 đường hyper-lane (A,B and C). Bây giờ đường nào là hyper-lane đúng cho nhóm ngôi sao và hình tròn.



Qui tắc 1: chọn những siêu phẳng phân chia hai lớp tốt nhất (các điểm thuộc phân lớp khác nhau nằm ở khác phía của siêu phẳng nhiều nhất)

- Quy tắc số một để chọn một siêu phẳng để phân chia hai lớp tốt nhất. Trong ví dụ này chính là đường B.

Tiếp theo ta có 3 đường hyper-plane (A,B và C), theo quy tắc số 1, chúng đều thỏa mãn:



Qui tắc 2: Chọn siêu phẳng có khoảng cách lớn nhất đến một điểm gần nhất của một lớp nào đó.

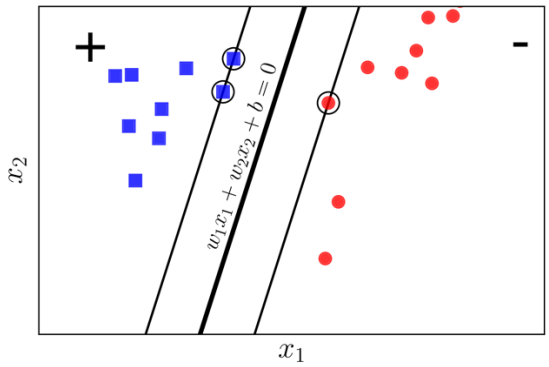
- Dựa vào hình bên trên, trong đấy có thể nhìn thấy khoảng cách margin lớn nhất đấy là đường C. Cần nhớ nếu chọn lầm hyper-lane có margin thấp hơn thì sau này khi dữ liệu tăng lên thì sẽ sinh ra nguy cơ cao về việc xác định nhầm lớp cho dữ liệu.

4. Trường hợp dữ liệu khả tách tuyến tính (phân 2 lớp)

4.1. Tập hypothesis của SVM

- Trong trường hợp dữ liệu khả tuyến tính, hypothesis của SVM được biểu diễn dưới dạng một siêu phẳng trong không gian đa chiều. Siêu phẳng này được xác định bởi một tập hợp các trọng số và hệ số bias, đại diện cho mối quan hệ tuyến tính giữa các đặc trưng của dữ liệu

4.2. Thuật toán của SVM



- Giả sử rằng các điểm vuông xanh thuộc lớp 1, các điểm tròn đỏ thuộc lớp -1 và mặt $w_1x_1 + w_2x_1 + b = 0$ là mặt phân chia giữa hai lớp. Hơn nữa, lớp 1 nằm về phía dương, class -1 nằm về phía âm của mặt phân chia. Nếu ngược lại, ta chỉ cần đổi dấu của w và b.
- Khoảng cách từ một điểm (vector) có toạ độ x_0 tới siêu mặt phẳng (hyperplane) có phương trình $w^Tx + b = 0$ được xác định bởi:

$$\frac{|w^T x_0 + b|}{\|w\|_2}$$

Với $\|w\|_2 = \sqrt{\sum_{i=1}^{d} w_i^2}$ với d là số chiều của không gian

- Ta quan sát thấy một điểm quan trọng sau đây: với cặp dữ liệu (x_n, y_n) bất kỳ, khoảng cách từ điểm đó tới mặt phân chia là:

$$\frac{y_n|w^T x_n + b|}{\|w\|_2}$$

Điều này có thể dễ nhận thấy vì theo giả sử ở trên, y_n luôn cùng dấu với x_n. Từ đó suy ra $y_n$cùng dấu với $(w^T x_n + b)$, và tử số luôn là một số không âm.

- Với mặt phần chia như trên, margin được tính là khoảng cách gần nhất từ 1 điểm tới mặt đó:

$$margin = min \frac{y_n|w^T x_n + b|}{\|w\|_2}$$

- Bài toán tối ưu trong SVM chính là bài toán tìm *w* và *b* sao cho margin này đạt giá trị lớn nhất:

$$(w, b) = arg\ max \left\{ min \frac{y_n|w^T x_n + b|}{\|w\|_2} \right\} (1)$$

Việc giải trực tiếp bài toán này sẽ rất phức tạp, nhưng chúng ta có cách để đưa nó về bài toán đơn giản hơn.

- Nhận xét quan trọng nhất là nếu ta thay vector hệ số w bởi kw và b bởi kb trong đó k là một hằng số dương thì mặt phân chia không thay đổi, tức khoảng cách từ từng điểm đến mặt phân chia không đổi, tức margin không đổi. Dựa trên tính chất này, ta có thể giả sử với những điểm nằm gần mặt phân chia nhất:

$$y_n|w^T x_n + b| = \mathbf{1}$$

- Như vậy với mọi n ta có:

$$y_n|w^T x_n + b| \geq \mathbf{1}$$

- Vậy bài toán tối ưu (1) có thể đưa về bài toán tối ưu có ràng buộc sau đây:

$$(w, b) = arg\ max \frac{1}{\|w\|_2}$$
$$với\ y_n|w^T x_n + b| \geq \mathbf{1}, \forall \boldsymbol{n} = \mathbf{1}, \mathbf{2}, \mathbf{3}, \ldots \boldsymbol{N}$$

- Từ đó ta biến đổi bài toán về dạng:

$$(w, b) = arg\ max \frac{1}{2}\|\boldsymbol{w}\|_2^2$$
$$với\ 1 - y_n|w^T x_n + b| \leq \mathbf{0}, \forall \boldsymbol{n} = \mathbf{1}, \mathbf{2}, \mathbf{3}, \ldots \boldsymbol{N}$$

4.3. Support Vector

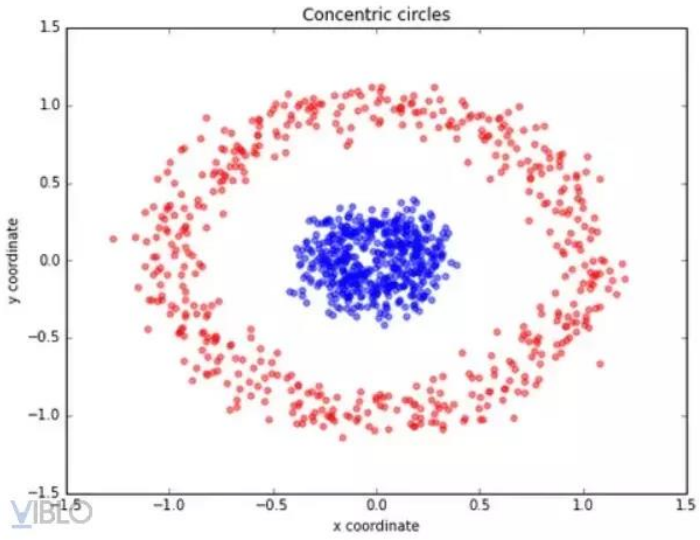
- "Support Vectors" là những điểm dữ liệu đặc biệt quan trọng vì chúng nằm gần nhất với đường ranh giới hoặc siêu phẳng quyết định. Những điểm này không chỉ định hình hành vi của siêu phẳng mà còn giúp xác định khoảng cách giữa siêu phẳng và các điểm dữ liệu thuộc hai lớp. Trong quá trình xây dựng mô hình SVM, "Support Vectors" được chọn sao cho khoảng cách từ chúng đến siêu phẳng là lớn nhất.
- Đối với SVM, những điểm này chủ yếu đóng vai trò trong việc định rõ margin, tức là khoảng trống giữa siêu phẳng và các điểm dữ liệu gần nhất của từng lớp. Sự tập trung vào những điểm này giúp xác định siêu phẳng tối ưu và làm cho margin lớn

nhất có thể. Đồng thời, chúng giúp mô hình SVM trở nên ổn định và có khả năng tổng quát hóa tốt hơn trên dữ liệu mới. Những điểm gần margin càng quan trọng, càng đóng góp vào việc xác định độ chính xác và hiệu suất của SVM trên dữ liệu thực tế.

5. Trường hợp dữ liệu không khả tách tuyến tính (phân 2 lớp)

5.1. Soft margin

- Soft Margin SVM là một biến thể của SVM, sử dụng trong trường hợp dữ liệu được phân tách tuyến tính, khi chúng ta không thể vẽ được đường thẳng để phân loại các điểm dữ liệu. Thuật toán này cho phép SVM mắc một số lỗi nhất định với mục tiêu giữ cho lề càng rộng càng tốt và các điểm khác vẫn được phân loại chính xác. Nói cách khác, nó sẽ cân bằng giữa việc phân loại sai và tối đa hóa lề. Sự cho phép vi phạm một số điểm dữ liệu giúp giảm thiểu overfitting (quá khớp) và làm cho mô hình SVM linh hoạt hơn trong việc xử lý các tập dữ liệu có độ phức tạp cao hơn.

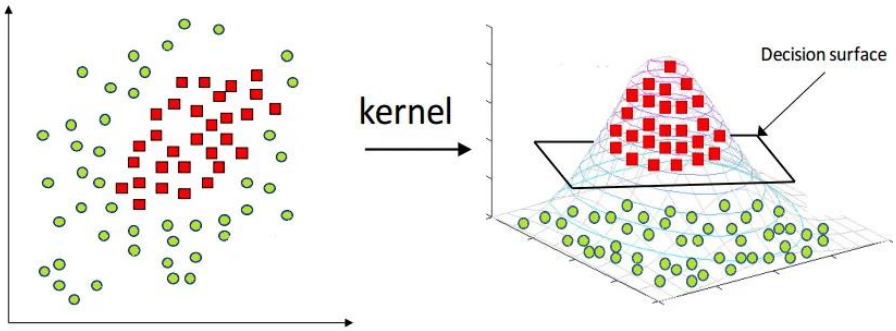


5.2. Siêu tham số trong soft margin

- Một đại lượng quan trọng và để kiểm soát mức độ sai sót của Soft Margin SVM là "C" - mức độ chấp nhận lỗi (hay tham số phạt). Khi tham số phạt càng lớn nghĩa là SVM bị phạt càng nặng khi phân loại sai và kết quả sẽ cho ra 1 lề (margin) hẹp và ngược lại. Do đó, lề càng hẹp và càng ít vectơ hỗ trợ được sử dụng.

5.3. Kernel

- Một kernel là một hàm ánh xạ dữ liệu từ không gian ít nhiều hơn sang không gian nhiều chiều hơn, từ đó ta tìm được siêu phẳng phân tách dữ liệu. Một cách trực quan, kỹ thuật này giống như việc bạn gập tờ giấy lại để có thể dùng kéo cắt một lỗ tròn trên nó.



Các kiểu Kernel:

- Tuyến tính

$$k(x,z) = x^T z$$

- Đa thức

$$k(x,z) = (\tau + \gamma x^T z)^d$$

- RBF

$$k(x,z) = exp(-\gamma \|x - z\|_2^2), \gamma > 0$$

- Sigmoid

$$k(x,z) = fishy(\tau + \gamma x^T z)$$

5.4. Siêu tham số γ trong Gaussian/ RBF kernel

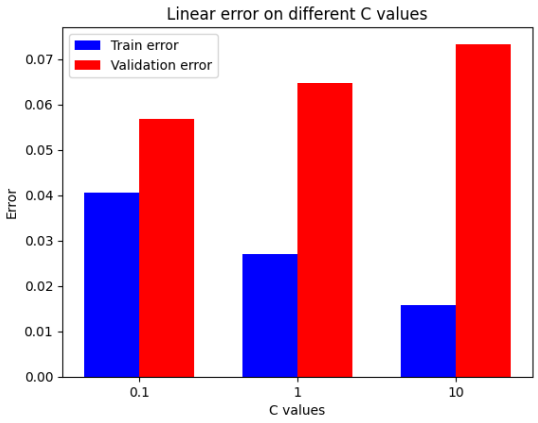
- Siêu tham số γ (gamma) là một yếu tố quan trọng trong Gaussian (RBF) Kernel. Gaussian Kernel ánh xạ không gian dữ liệu vào một không gian chiều cao vô hạn để tạo ra đường phân chia phức tạp, cho phép SVM hoạt động hiệu quả trên dữ liệu có đặc trưng phức tạp.
- γ kiểm soát độ "co giãn" của hàm Gaussian. Nếu γ lớn, độ co giãn sẽ nhỏ hơn và mô hình sẽ tập trung nhiều hơn vào các điểm dữ liệu gần siêu phẳng. Điều này có thể dẫn đến hiện tượng overfitting, nơi mô hình có thể quá nhạy cảm đến nhiễu hoặc chi tiết không quan trọng trong dữ liệu đào tạo.
- Nếu γ nhỏ, độ co giãn tăng lên, và mô hình sẽ có độ linh hoạt cao hơn với dữ liệu. Tuy nhiên, nếu quá nhỏ, mô hình có thể không đủ phức tạp để xử lý đặc trưng phức tạp của dữ liệu.
- Do đó, sự lựa chọn đúng cho γ là quan trọng để đạt được sự cân bằng giữa linh hoạt và ngăn chặn, giúp mô hình SVM tổng quát hóa tốt trên dữ liệu mới. Việc thử nghiệm và tinh chỉnh giá trị của γ là một phần quan trọng trong quá trình điều chỉnh siêu tham số của SVM để đảm bảo hiệu suất tối ưu trên dữ liệu thực tế.

6. Dùng SVM phân K lớp dữ liệu

- Để mở rộng SVM từ việc phân loại 2 lớp (binary classification) lên phân loại K lớp (multiclass classification), một trong những phương pháp phổ biến là "one-against-one" (OAO) hoặc "one-versus-one."
- Trong phương pháp OAO, chúng ta xây dựng một bộ phân loại nhị phân (binary classifier) cho mỗi cặp lớp có thể có trong tập dữ liệu. Ví dụ, nếu có K lớp, ta sẽ tạo ra $\frac{K(K-1)}{2}$ bộ phân loại. Mỗi bộ phân loại sẽ được đào tạo để phân biệt giữa hai lớp cụ thể từ tất cả các lớp có sẵn.
- Khi cần dự đoán lớp của một điểm dữ liệu mới, chúng ta sẽ đưa ra quyết định bằng cách "bỏ phiếu" từ tất cả các bộ phân loại. Lớp nào nhận được nhiều phiếu nhất sẽ được chọn là kết quả dự đoán cho điểm dữ liệu đó.
- Phương pháp OAO có ưu điểm là dễ triển khai và tương đối hiệu quả với các bộ phân loại nhị phân có sẵn. Tuy nhiên, có thể xảy ra trường hợp không rõ ràng nếu có những quyết định mâu thuẫn giữa các bộ phân loại.
- Ngoài ra, có một phương pháp khác là "one-against-all" (OAA), trong đó ta tạo ra K bộ phân loại, mỗi bộ tập trung vào phân loại một lớp so với tất cả các lớp còn lại. Cả hai phương pháp đều được sử dụng phổ biến trong thực tế, và lựa chọn giữa chúng thường phụ thuộc vào đặc tính cụ thể của dữ liệu và mô hình.

II. Huấn luyện SVM để phân lớp thời trang:

1. Đánh giá hiệu suất của SVM Linear Kernel



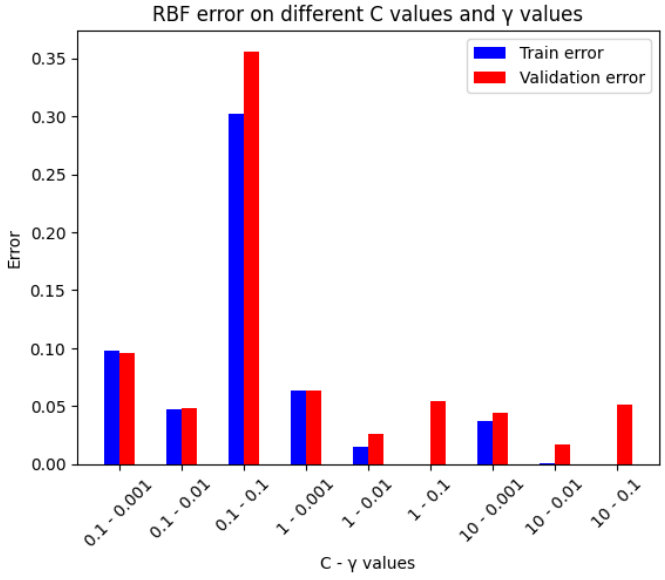
Độ lỗi tuyến tính với các giá trị C khác nhau

```
Linear SVM with C=0.1: train error=0.040583333333333305, val error=0.056916666666666615, time=103.94739556312561s
Linear SVM with C=1: train error=0.027000000000000024, val error=0.0648333333333333, time=120.7997498512268s
Linear SVM with C=10: train error=0.01579166666666665, val error=0.07333333333333336, time=192.57639479637146s
```

Kết quả chạy mô hình tuyến tính với các giá trị C khác nhau

- Khi siêu tham số "C" càng tăng, mô hình càng trở nên phức tạp hơn, điều này có thể dẫn đến hiện tượng "overfitting". Điều này được thể hiện rõ ràng qua các kết quả, trong đó *train error* giảm trong khi *validation error* với "C" tăng dần. Sự thay đổi này cho thấy rằng mô hình đang cải thiện trên tập *train*, nhưng lại trở nên kém hiệu quả trên tập *validation*.
- Ngoài ra, thời gian huấn luyện tăng lên khi C tăng. Bởi lẽ một mô hình càng phức tạp, càng cần nhiều tài nguyên tính toán để huấn luyện.
- Dựa trên các phân tích trên, với C=0.1 ta có được sự cân bằng tốt nhất đối với tất cả các yếu tố sau: *train error*, *validation error* và thời gian huấn luyện.

2. Đánh giá hiệu suất của SVM RBF Kernel



Độ lỗi của mô hình RBF đối với các giá trị C và gamma khác nhau

```
RBF SVM with C=0.1, gamma=0.001: train error=0.09787500000000005, val error=0.09591666666666665, time=554.6318941116333s
RBF SVM with C=0.1, gamma=0.01: train error=0.04683333333333328, val error=0.04808333333333337, time=280.46695613861084s
RBF SVM with C=0.1, gamma=0.1: train error=0.3024375, val error=0.3560833333333333, time=1372.8002908229828s
RBF SVM with C=1, gamma=0.001: train error=0.06370833333333337, val error=0.06366666666666665, time=206.57252717018127s
RBF SVM with C=1, gamma=0.01: train error=0.014958333333333296, val error=0.02558333333333329, time=113.55971693992615s
RBF SVM with C=1, gamma=0.1: train error=0.00010416666666668295, val error=0.05383333333333329, time=1855.2602217197418s
RBF SVM with C=10, gamma=0.001: train error=0.03691666666666671, val error=0.04433333333333336, time=105.3256049156189s
RBF SVM with C=10, gamma=0.01: train error=0.0004374999999997957, val error=0.01691666666666669, time=93.5045554637909s
RBF SVM with C=10, gamma=0.1: train error=0.0, val error=0.051583333333333314, time=1911.6772758960724s
```

Kết quả chạy mô hình RBF với khác giá trị C và gamma khác nhau

- Khi siêu tham số "C" và γ càng tăng, mô hình càng trở nên phức tạp hơn, điều này có thể dẫn đến hiện tượng "overfitting". Điều này được thể hiện rõ ràng qua các kết quả, trong đó *train error* giảm trong khi *validation error* với "C" và γ tăng dần. Mặc dù vậy, validation data không phải lúc nào cũng tăng đều. Sự biến thiên chung trên biểu đồ cho thấy mô hình càng lúc càng khớp với tập train nhưng khả năng khái quát hóa trên tập validation không phải lúc nào cũng được cải thiện theo siêu tham số.
- Ngoài ra, thời gian huấn luyện tăng lên khi C và γ tăng. Bởi lẽ một mô hình càng phức tạp, càng cần nhiều tài nguyên tính toán để huấn luyện.
- Dựa trên các phân tích trên, với C=10, γ = 0.01 ta có được sự cân bằng tốt nhất đối với tất cả các yếu tố sau: *train error*, *validation error* và thời gian huấn luyện.

3. Lựa chọn giá trị siêu tham số tốt nhất và kiểm thử trên tập test

- Từ các phân tích ở mục 1 và 2 ta đã lựa chọn ra các siêu tham số tốt nhất tương ứng với linear kernel và RBF kernel.
- Quy trình ở đây là tái huấn luyện trên tập train đầy đủ (60000 mẫu) trước khi kiểm thử với tập test. Từ đó ta thu được kết quả như sau:

```
Linear SVM with C=0.1: train error=0.1111833333333333, test error=0.14390000000000003, time=251.6508686542511s
RBF SVM with C=10 and δ=0.01: train error=0.029266666666666663, test error=0.10009999999999997, time=235.39492440223694s
```

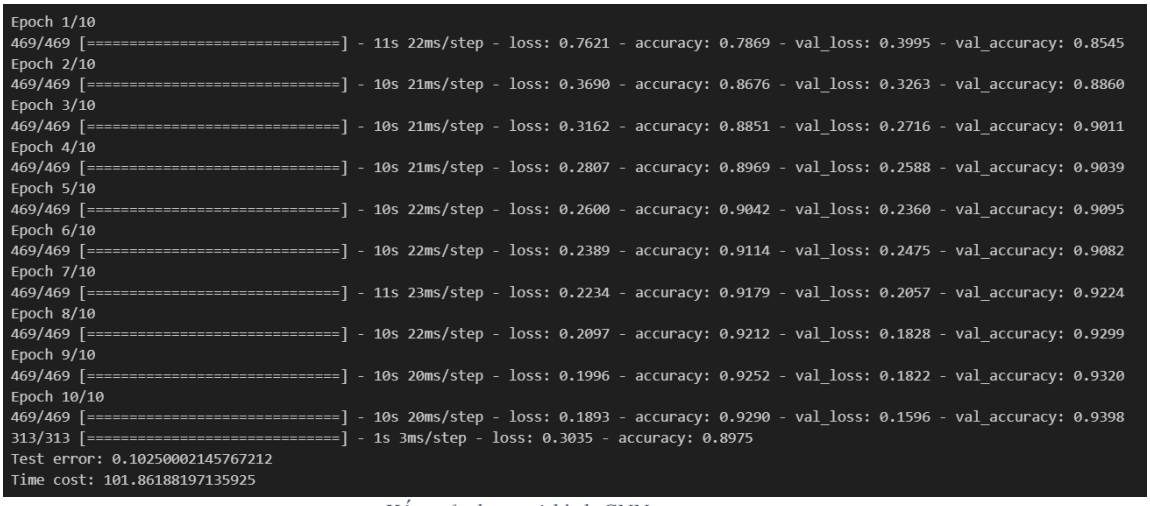
Kết quả sau khi tái huấn luyện và kiểm thử

- Đối với linear kernel, ta được train error = 0.11, test error = 0.14.
- Đối với RBF kernel, ta được train error = 0.02, test error = 0.1.
- Hai mô hình này có thời gian chạy xấp xỉ 240 giây.

→ Như vậy ta thấy các mô hình này có độ lỗi gần như thỏa mãn với kì vọng của chúng ta. Tuy vậy, thời gian chạy lâu cho thấy rằng ta cần sử dụng tập train theo cách khác chẳng hạn như là chia tập theo lô và huấn luyện trên các lô đó.

4. Chọn mô hình so sánh với SVM (CNN)

- Thông qua quá trình tìm hiểu, chúng em nhận thấy với dataset ở dạng ảnh với kích thước lớn, thì CNN dường như là lựa chọn hiệu quả khi chúng ta cần phải xử lí và phân lớp ảnh.



Kết quả chạy mô hình CNN

Quy trình xây dựng mô hình CNN bao gồm các bước sau:

- **Import các module cần thiết:** Đoạn mã bắt đầu bằng việc nhập các module cần thiết từ Keras. Sequential là lớp mô hình để tạo ra một chuỗi tuyến tính các tầng. *Dense, Conv2D, MaxPooling2D*, và *Flatten* là các loại tầng khác nhau có thể được thêm vào mô hình.
- **Nạp bộ dữ liệu:** Bộ dữ liệu Fashion-MNIST được nạp bằng cách sử dụng hàm *load_data* từ *tf.keras.datasets.fashion_mnist*. Hàm này trả về hai bộ giá trị, một cho bộ dữ liệu huấn luyện và một cho bộ dữ liệu kiểm tra. Mỗi bộ giá trị chứa một tập hình ảnh (X) và nhãn tương ứng (y).
- **Chia bộ dữ liệu huấn luyện:** Bộ dữ liệu huấn luyện được chia thành một bộ dữ liệu huấn luyện mới và một bộ dữ liệu xác thực bằng cách sử dụng hàm *train_test_split* từ *sklearn.model_selection*. 20% dữ liệu được sử dụng cho bộ dữ liệu xác thực (test_size=0.2), và phần còn lại được sử dụng cho bộ dữ liệu huấn luyện mới. Tham số *random_state* được đặt để đảm bảo tính nhất quán của việc chia dữ liệu.
- **Định hình lại dữ liệu:** Hình ảnh được định hình lại để bao gồm chiều kênh. Giá trị -1 trong hàm định hình lại có nghĩa là kích thước trong chiều đó sẽ được suy luận dựa trên độ dài của mảng và các chiều còn lại.
- **Chuyển đổi nhãn sang dạng phân loại:** Nhãn được chuyển đổi sang định dạng phân loại bằng hàm *to_categorical* từ thư viện *tf.keras.utils*. Điều này là cần thiết vì mô hình sẽ được huấn luyện để dự đoán xác suất cho mỗi trong 10 lớp, do đó nhãn cần phải ở cùng định dạng.
- **Định nghĩa mô hình:** Một mô hình Sequential được định nghĩa với ba khối tích chập, mỗi khối bao gồm một tầng *Conv2D* theo sau là một tầng *MaxPooling2D*. Các tầng *Conv2D* có 32, 64 và 64 bộ lọc tương ứng, và mỗi bộ lọc có kích thước 3x3. Hàm kích hoạt được sử dụng là *ReLU (Rectified Linear Unit)*. Các tầng *MaxPooling2D* được sử dụng để giảm kích thước không gian của khối lượng đầu ra. Sau các khối tích chập, một tầng *Flatten* được thêm vào để làm phẳng đầu ra của các khối tích chập thành một chiều. Cuối cùng, hai tầng *Dense* (fully connected) được thêm vào để phân loại. Tầng *Dense* đầu tiên có 64 đơn vị và sử dụng hàm kích hoạt *ReLU*. Tầng *Dense* thứ hai có 10 đơn vị (một cho mỗi lớp) và sử dụng hàm kích hoạt *softmax* để tạo ra một phân phối xác suất trên 10 lớp.

- **Biên dịch mô hình:** Mô hình được biên dịch với bộ tối ưu hóa Adam, hàm mất mát là *categorical crossentropy*, và độ chính xác là độ đo đánh giá.
- **Huấn luyện mô hình:** Mô hình được huấn luyện trên dữ liệu huấn luyện trong 10 epochs với kích thước *batch* là 128. Dữ liệu xác thực được chuyển vào đối số *validation_data*, vì vậy hiệu suất của mô hình trên bộ dữ liệu xác thực sẽ được đánh giá vào cuối mỗi epoch.

5. So sánh CNN với SVM

- Thông qua kết quả trên ta tìm được test error = 0.102 và time cost = 101.86s. Nếu so với 2 mô hình SVM ở trước ta nhận thấy rằng test error của mô hình CNN nằm trong khoảng giữa 2 mô hình đó. Tuy nhiên, xét tới thời gian chạy của mô hình CNN chỉ bằng 1/2 so với 2 mô hình SVM, ta có thể nhận thấy sự ưu việt của mô hình CNN so với mô hình SVM.
- Điều này có thể được lí giải bởi:
    - CNN được thiết kế đặc biệt để xử lý dữ liệu không gian như ảnh và video. Cấu trúc của CNN bao gồm các tầng convolutional, pooling và fully connected layers giúp nó hiệu quả trong việc nhận diện các đặc trưng không gian.
    - CNN thường có thể xử lý các tập dữ liệu lớn và phức tạp mà không gặp vấn đề về hiệu suất, đặc biệt là trong lĩnh vực thị giác máy tính. SVM có thể trở nên không hiệu quả khi đối mặt với các tập dữ liệu lớn.
    - CNN thường được ưa chuộng khi làm việc với dữ liệu không gian cao như ảnh màu (RGB) với nhiều kênh. SVM có thể trở nên không hiệu quả khi số lượng đặc trưng lớn. Cụ thể ở đoạn code xây dựng mô hình CNN, ta thấy dữ liệu không cần phải chuẩn hóa về giá trị trong đoạn [0,1] như ở SVM.

TÀI LIỆU THAM KHẢO

1. A Comparison of Methods for Multi-class Support Vector Machines (link)
2. SVM quá khó hiểu! Hãy đọc bài này (link)
3. Support Vector Machine (link)
4. SVM, Soft Margin SVM, Kernel SVM, Multi-class SVM (link)
5. Kernel Support Vector Machine (link)
6. Giới thiệu về Support Vector Machine (link)
7. Một số tham khảo ở GitHub (link 1, link 2)