Trường Đại Học Điện Lực

Khoa Công Nghệ Thông Tin

BÁO CÁO CHUYÊN ĐỀ HỌC PHẦN

MÁY VECTO HỖ TRỢ

ĐỀ TÀI :

PHÂN LOẠI OTO, XE MÁY SỬ DỤNG THUẬT TOÁN SVM

Sinh viên thực hiện:	**NGUYỄN MẠNH HƯNG**
	PHẠM ĐỨC MẠNH
Lớp:	**D15TTNT&TGMT**
Giảng Viên Hướng Dẫn:	**PGS. TS. NGÔ QUỐC TẠO**
Ngành:	**CÔNG NGHỆ THÔNG TIN**
Chuyên Ngành:	**TTNT & TGMT**
Khóa:	**2020-2025**

Hà Nội, Tháng 5 Năm 2023

PHIẾU CHẤM ĐIỂM

Sinh viên thực hiện

Họ và tên	Điểm	Chữ ký	Ghi Chú
Nguyễn Mạnh Hưng 20810310347			
Phạm Đức Mạnh 20810310398			

Giảng viên chấm

Họ và tên	Chữ ký	Ghi chú
Giảng viên chấm 1		
Giảng viên chấm 2		

MỤC LỤC

LỜI NÓI ĐẦU

Trong thời đại số hóa hiện nay, Công nghệ Thông tin và Truyền thông (CNTT) đang trở thành một ngành rất quan trọng và phát triển với tốc độ nhanh chóng. CNTT đóng vai trò vô cùng quan trọng trong nhiều lĩnh vực khác nhau, từ kinh doanh, y tế, giáo dục cho đến nghiên cứu khoa học.

Trong lĩnh vực CNTT, học máy đang là một chủ đề đang được quan tâm rất nhiều. Học máy được ứng dụng trong việc giải quyết các bài toán phức tạp, từ dự báo tài chính đến phân tích hình ảnh. Trong đó, Support Vector Machine (SVM) là một trong những thuật toán học máy được sử dụng phổ biến và hiệu quả nhất.

Trong bài báo cáo này, chúng em sẽ tìm hiểu về SVM - một trong những thuật toán học máy quan trọng nhất trong lĩnh vực CNTT. Chúng ta sẽ cùng nhau tìm hiểu về cách SVM hoạt động, cách nó phân loại và các ứng dụng thực tế của nó trong lĩnh vực CNTT. Bên cạnh đó, chúng ta cũng sẽ xem xét một số ví dụ và thực hành với các tập dữ liệu để hiểu rõ hơn về cách áp dụng SVM trong thực tế của CNTT. Chính sự hiệu quả trong công việc và các lợi ích vượt bậc mà thuật toán này đem lại đã góp phần giúp cho Machine Learning ngày càng được chú trọng và quan tâm nhiều hơn. Chính vì vậy, chúng em chọn đề tài "Phân loại oto, xe máy sử dụng thuật toán SVM".

CHƯƠNG 1: TÌM HIỂU VỀ KỸ THUẬT PHÂN LỚP DỮ LIỆU

1.1. Phát biểu bài toán

Trong quá trình hoạt động, con người tạo ra nhiều dữ liệu nghiệp vụ. Các tập dữ liệu được tích lũy có kích thước ngày càng lớn, và có thể chứa nhiều thông tin ẩn dạng những quy luật chưa được khám phá. Chính vì vậy, một nhu cầu đặt ra là cần tìm cách trích rút từ tập dữ liệu đó các luật về phân lớp dữ liệu hay dự đoán những xu hướng dữ liệu tương lai. Công nghệ phân lớp và dự đoán dữ liệu ra đời để đáp ứng mong muốn đó. Công nghệ phân lớp dữ liệu đã, đang và sẽ phát triển mạnh mẽ trước những khao khát tri thức của con người.

Trong những năm qua, phân lớp dữ liệu đã thu hút sự quan tâm các nhà nghiên cứu trong nhiều lĩnh vực khác nhau như học máy (machine learning), hệ chuyên gia (expert system), thống kê (statistics)... Công nghệ này cũng ứng dụng trong nhiều lĩnh vực thực tế như: thương mại, nhà băng, maketing, nghiên cứu thị trường, bảo hiểm, y tế, giáo dục... Nhiều kỹ thuật phân lớp đã được đề xuất như: Phân lớp cây quyết định (Decision tree classification), phân lớp Bayesian (Bayesian classifier), phân lớp K hàng xóm gần nhất (K-nearest neighbor classifier), mạng nơron, phân tích thống kê,…

Support Vector Machines (SVM) là kỹ thuật mới đối với việc phân lớp dữ liệu, là phương pháp học sử dụng không gian giả thuyết các hàm tuyến tính trên không gian đặc trưng nhiều chiều, dựa trên lý thuyết tối ưu và lý thuyết thống kê.

1.1.1. Giới thiệu về kĩ thuật phân lớp dữ liệu

Phân lớp dữ liệu là một kỹ thuật trong khai phá dữ liệu được sử dụng rộng rãi nhất và được nghiên cứu mở rộng hiện nay.

Mục đích: Để dự đoán những nhãn phân lớp cho các bộ dữ liệu hoặc mẫu mới.

- Đầu vào: Một tập các mẫu dữ liệu huấn luyện, với một nhãn phân lớp cho mỗi mẫu dữ liệu.
- Đầu ra: Bộ phân lớp dựa trên tập huấn luyện, hoặc những nhãn phân lớp.

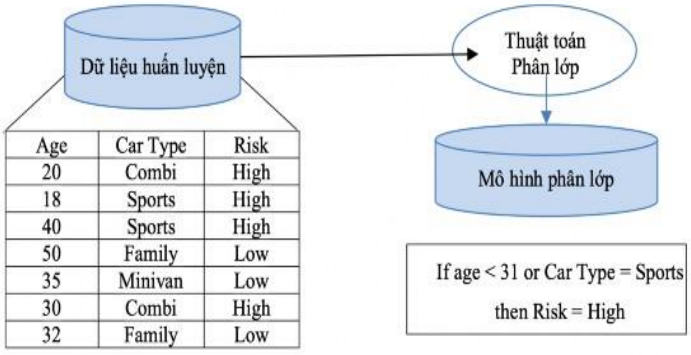
Phân lớp dữ liệu dựa trên tập huấn luyện và các giá trị trong một thuộc tính phân lớp và dùng nó để xác định lớp cho dữ liệu mới.

***Kỹ thuật phân lớp dữ liệu được tiến hành bao gồm 2 bước:**

- Bước 1: Xây dựng mô hình từ tập huấn luyện.
- Bước 2: Sử dụng mô hình - kiểm tra tính đúng đắn của mô hình và dùng nó để phân lớp dữ liệu mới.

Bước 1. Xây dựng mô hình

- Mỗi bộ / mẫu dữ liệu được phân vào một lớp được xác định trước.
- Lớp của một bộ / mẫu dữ liệu được xác định bởi thuộc tính gán nhãn lớp.
- Tập các bộ / mẫu dữ liệu huấn luyện - tập huấn luyện - được dùng để xây dựng mô hình.
- Mô hình được biểu diễn bởi các luật phân lớp, các cây quyết định hoặc các công thức toán học.

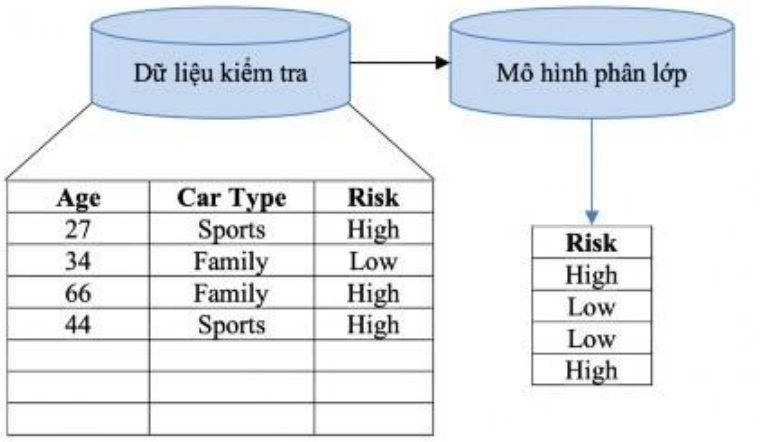


Age	Car Type	Risk
20	Combi	High
18	Sports	High
40	Sports	High
50	Family	Low
35	Minivan	Low
30	Combi	High
32	Family	Low

Hình 1.1: Ví dụ xây dựng mô hình

Bước 2: Phân lớp dữ liệu mới

- Phân lớp cho những đối tượng mới hoặc đối tượng chưa được phân lớp.
- Đánh giá độ chính xác của mô hình:
    - Lớp biết trước của một mẫu/bộ dữ liệu đem kiểm tra được so sánh với kết quả thu được từ mô hình.
    - Tỉ lệ chính xác bằng phần trăm các mẫu/bộ dữ liệu được phân lớp đúng bởi mô hình trong số các lần kiểm tra.



Hình 1.2: Ví dụ phân lớp dữ liệu mới

1.1.2. Áp dụng thuật toán SVM trong phân lớp dữ liệu

- SVM rất hiệu quả để giải quyết bài toán dữ liệu có số chiều lớn (ảnh của dữ liệu biểu diễn gene, protein, tế bào).
- SVM giải quyết vấn đề *overfitting* rất tốt (dữ liệu có nhiễu và tách rời nhóm hoặc dữ liệu huấn luyện quá ít).
- Là phương pháp phân lớp nhanh.
- Có hiệu suất tổng hợp tốt và hiệu suất tính toán cao.

1.2. Thuật toán SUPPORT VECTOR MACHINE (SVM)

1.2.1. Giới thiệu

Bài toán phân lớp (*Classification*) và dự đoán (*Prediction*) là hai bài toán cơ bản và có rất nhiều ứng dụng trong tất cả các lĩnh vực như: học máy, nhận dạng, trí tuệ nhân tạo, .v.v .

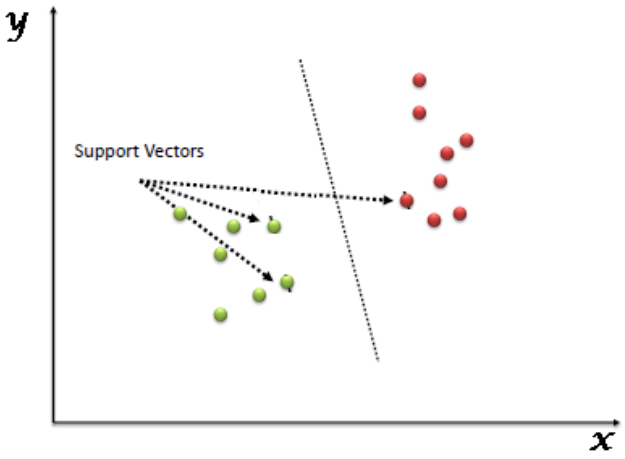
Phương pháp SVM được coi là công cụ mạnh cho những bài toán phân lớp phi tuyến tính được các tác giả Vapnik và Chervonenkis phát triển mạnh mẽ năm 1995. Phương pháp này thực hiện phân lớp dựa trên nguyên lý Cực tiểu hóa Rủi ro có Cấu trúc SRM (*Structural Risk Minimization*), được xem là một trong các phương pháp phân lớp giám sát không tham số tinh vi nhất cho đến nay. Các hàm công cụ đa dạng của SVM cho phép tạo không gian chuyên đổi để xây dựng mặt phẳng phân lớp.

1.2.2. Định nghĩa

- Là phương pháp dựa trên nền tảng của lý thuyết thống kê nên có một nền tảng toán học chặt chẽ để đảm bảo rằng kết quả tìm được là chính xác.
- Là thuật toán học giám sát (*supervied learning*) được sử dụng cho phân lớp dữ liệu.
- Là 1 phương pháp thử nghiệm, đưa ra 1 trong những phương pháp mạnh và chính xác nhất trong số các thuật toán nổi tiếng về phân lớp dữ liệu.
- SVM là một phương pháp có tính tổng quát cao nên có thể được áp dụng cho nhiều loại bài toán nhận dạng và phân loại.

1.2.3. Nội dung

Cho trước một tập huấn luyện, được biểu diễn trong không gian vector, trong đó mỗi tài liệu là một điểm, phương pháp này tìm ra một siêu phẳng quyết định tốt nhất có thể chia các điểm trên không gian này thành hai lớp riêng biệt tương ứng là lớp vuông và lớp tròn. Sau đó chúng ta thực hiện tìm "đường bay" (hyper-plane) phân chia các lớp. Hyper-plane nó chỉ hiểu đơn giản là 1 đường thẳng có thể phân chia các lớp ra thành các miền và mỗi miền chứ một loại dữ liệu.

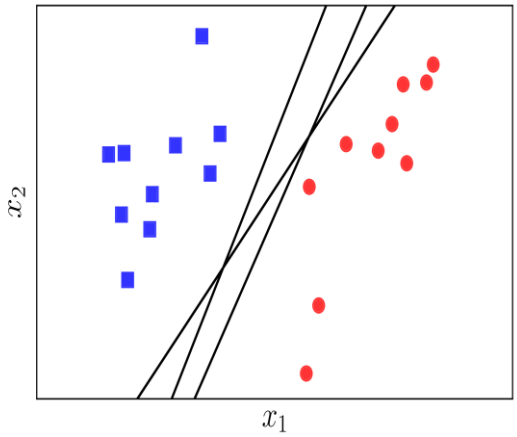


Hình 1.3: Ví dụ minh họa cho Support Vectors

Support Vectors là các đối tượng trên đồ thị tọa độ quan sát.

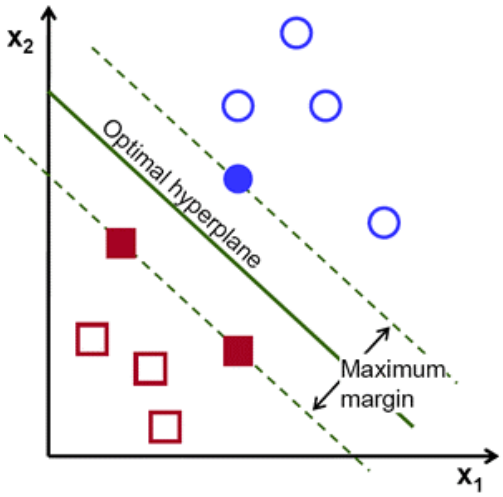
Support Vector Machine là một biên giới để chia hai lớp tốt nhất.

Chúng ta có thể phân tách 2 miền dữ liệu xanh và đỏ thành 2 miền khác biệt với các đường hyper-plane.



Hình 1.4: Ví dụ minh họa cho đường hyper-plane

Vấn đề đặt ra ở đây sẽ là: có rất nhiều siêu phẳng (hyper-plane), vậy ta phải chọn cái nào, và đâu sẽ là tối ưu nhất?



Hình 1.5: Ví dụ minh họa cho Support Vectors

Để xác định được đường bay hyper-plane tối ưu nhất ta cần xác định được khoảng cách từ điểm gần nhất của một lớp nào đó đến hyper-plane. Khoảng cách này được gọi là "Margin". Điều quan trọng ở đây đó là phương pháp SVM luôn cố gắng cực đại hóa margin này. Nhờ vậy, SVM có thể giảm thiểu việc phân lớp sai (missclassification) đối với điểm dữ liệu mới đưa vào.

Để tìm được margin chúng ta cần xem lại các kiến thức khoảng cách từ một điểm tới một siêu phẳng.

Trong không gian 2 chiều, ta biết rằng khoảng cách từ một điểm có toạ độ (x_0, y_0) tới đường thẳng có phương trình $w_1 x + w_2 y + b = 0$ được xác định bởi:

$$\frac{|w_1 x_0 + w_2 y_0 + b|}{\sqrt{w_1^2 + w_2^2}}$$

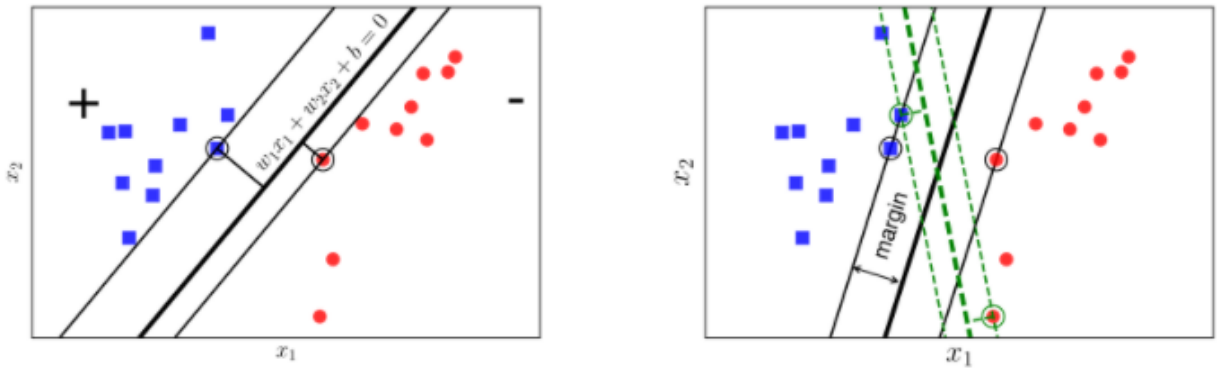
Trong không gian ba chiều, khoảng cách từ một điểm có toạ độ (x_0, y_0, z_0) tới một mặt phẳng có phương trình $w_1 x + w_2 y + w_3 z + b = 0$ được xác định bởi:

$$\frac{|w_1 x_0 + w_2 y_0 + w_3 z_0 + b|}{\sqrt{w_1^2 + w_2^2 + w_3^2}}$$

Vậy ta có hể được tổng quát lên không gian nhiều chiều: Khoảng cách từ một điểm (vector) có toạ độ $\mathbf{x}_0$ tới siêu mặt phẳng (hyperplane) có phương trình $\mathbf{w}^T \mathbf{x} + b = 0$ được xác định bởi:

$$\frac{|\mathbf{w}^T \mathbf{x}_0 + b|}{||\mathbf{w}||_2}$$

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



Hình 1.6: Ví dụ minh họa cho Margin

Nếu ta định nghĩa mức độ hạnh phúc của một class tỉ lệ thuận với khoảng cách gần nhất từ một điểm của class đó tới đường/mặt phân chia, thì ở Hình 1.6 bên trái, class tròn đỏ sẽ không được hạnh phúc cho lắm vì đường phân chia gần nó hơn class vuông xanh rất nhiều. Chúng ta cần một đường phân chia sao cho khoảng cách từ điểm gần nhất của mỗi class (các điểm được khoanh tròn) tới đường phân chia là như nhau, như thế thì mới công bằng. Khoảng cách như nhau này được gọi là margin (lề).

Đã có công bằng rồi, chúng ta cần văn minh nữa. Công bằng mà cả hai đều kém hạnh phúc như nhau thì chưa phải là văn mình cho lắm.

Chúng ta xét tiếp Hình 1.6 bên phải khi khoảng cách từ đường phân chia tới các điểm gần nhất của mỗi class là như nhau. Xét hai cách phân chia bởi đường nét liền màu đen và đường nét đứt màu lục, đường nào sẽ làm cho cả hai class hạnh phúc hơn? Rõ ràng đó phải là đường nét liền màu đen vì nó tạo ra một margin rộng hơn.

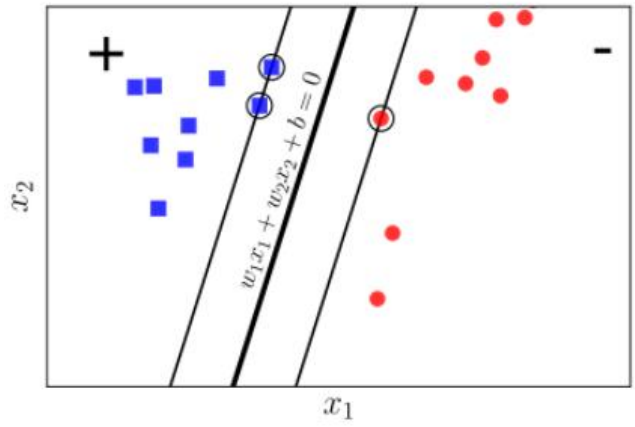
Việc margin rộng hơn sẽ mang lại hiệu ứng phân lớp tốt hơn vì sự phân chia giữa hai classes là rạch ròi hơn. Việc này, sau này các bạn sẽ thấy, là một điểm khá quan trọng giúp Support Vector Machine mang lại kết quả phân loại tốt hơn so với Neural Network với 1 layer, tức Perceptron Learning Algorithm.

Bài toán tối ưu trong Support Vector Machine (SVM) chính là bài toán đi tìm đường phân chia sao cho margin là lớn nhất. Đây cũng là lý do vì sao SVM còn được gọi là Maximum Margin Classifier. Nguồn gốc của tên gọi Support Vector Machine sẽ sớm được làm sáng tỏ.

***Xây dựng bài toán tối ưu cho SVM**

Giả sử rằng các cặp dữ liệu của training set là $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$ với vector $\mathbf{x}_i \in \mathbb{R}^d$ thể hiện đầu vào của một điểm dữ liệu và y_i à nhãn của điểm dữ liệu đó, d là số chiều dữ liệu và N là số điểm dữ liệu Giả sử rằng nhãn của mỗi điểm dữ liệu được xác định bởi $y_i = 1$ (class 1) hoặc $y_i = -1$ (class 2).

Ta cùng xét không gian 2 chiều dưới đây



Giả sử rằng các điểm vuông xanh thuộc class 1, các điểm tròn đỏ thuộc class -1 và mặt $\mathbf{w}^T\mathbf{x} + b = w_1x_1 + w_2x_2 + b = 0$ là mặt phân chia giữa hai classes (Hình 3). Hơn nữa, class 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. Chú ý rằng chúng ta cần đi tìm các hệ số w và b. Ta quan sát thấy một điểm quan trọng sau đây: với cặp dữ liệu $(\mathbf{x}_n, y_n)$ bất kỳ, khoảng cách từ điểm đó tới mặt phân chia là:

$$\frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{||\mathbf{w}||_2}$$

Điều này có thể dễ nhận thấy vì theo giả sử ở trên, y_n luôn cùng dấu với phía của $\mathbf{x}_n$. Từ đó suy ra $y_n$cùng đấu với $(\mathbf{w}^T\mathbf{x}_n + b)$, và tử số luôn là một số không âm.

Margin được tính là khoảng cách từ một điểm gần nhất từ một điểm tới mặt đó

$$\text{margin} = \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{||\mathbf{w}||_2}$$

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

$$(\mathbf{w}, b) = \arg\max_{\mathbf{w},b} \left\{ \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{||\mathbf{w}||_2} \right\} = \arg\max_{\mathbf{w},b} \left\{ \frac{1}{||\mathbf{w}||_2} \min_n y_n(\mathbf{w}^T\mathbf{x}_n + b) \right\}$$

Phương trình tuyến tính của SVM có dạng:

$$\mathbf{w}^T\mathbf{x} + b = 0$$

=> Trong đó w thuộc Rn là vector hệ số ứng với các chiều của vector $\mathbf{x} = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^n$, b là hệ số tự do trong không gian hai chiều thì được gọi là đường thẳng,không gian 3 chiều là mặt phẳng.

Công thức của sai số dự đoán:

$$\frac{1}{2}e^2 = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}(y - \bar{\mathbf{x}}\mathbf{w})^2$$

=> Trong đó, e là sai số dự đoán, y là giá trị thực và ŷ là giá trị dự đoán (hay còn gọi là y_pred). Hàm bình phương để tránh phương trình có thể ra kết quả âm và vì e là sai số, nên giá trị này càng nhỏ càng tốt.

Hàm mất mát:

$$J(\mathbf{w}, b) = \sum_{n=1}^{N} \max(0, 1 - y_n(\mathbf{w}^T\mathbf{x}_n + b)) + \frac{\lambda}{2}||\mathbf{w}||_2^2$$

***Các bước thực hiện**

Phương pháp SVM yêu cầu dữ liệu được diễn tả như các vector của các số thực. Như vậy nếu đầu vào chưa phải là số thì ta cần phải tìm cách chuyển chúng về dạng số của SVM.

Tiền xử lý dữ liệu: Thực hiện biến đổi dữ liệu phù hợp cho quá trình tính toán, tránh các số quá lớn mô tả các thuộc tính. Thường nên co giãn (*scaling*) dữ liệu để chuyển về đoạn [-1, 1] hoặc [0, 1].

Chọn hàm hạt nhân: Lựa chọn hàm hạt nhân phù hợp tương ứng cho từng bài toán cụ thể để đạt được độ chính xác cao trong quá trình phân lớp.

Thực hiện việc kiểm tra chéo để xác định các thám số cho ứng đụng. Điều này cũng quyết định đến tính chính xác của quá trình phân lớp.

Sử dụng các tham số cho việc huấn luyện với tập mẫu. Trong quá trình huấn luyện sử dụng thuật toán tối ưu hóa khoảng cách giữa các siêu phẳng trong quá trình phân lớp, xác định hàm phân lớp trong không gian đặc trưng nhờ việc ánh xạ dữ liệu vào không gian đặc trưng bằng cách mô tả hạt nhân, giải quyết cho cả hai trường hợp dữ liệu là phân tách và không phân tách tuyến tính trong không gian đặc trưng.

Kiểm thử tập dữ liệu Test.

CHƯƠNG 2: CODE ỨNG DỤNG SVM TRÊN GOOGLE COLAB

2.1. Import thư viện

Import những thư viện cần thiết

```
import pandas as pd
from sklearn import svm
from sklearn.model_selection import GridSearchCV
import os
import matplotlib.pyplot as plt
from skimage.transform import resize
from skimage.io import imread
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
import pickle
```

Hình 2.1: Import các thư viện cần thiết

2.2. Load và tiền xử lý dữ liệu

```
Categories=['Car','Moto']
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[3] !unzip /content/drive/MyDrive/SVM/Car-Moto-Dataset.zip
```

```
[4] !mv /content/Car-Moto-Dataset /content/drive/MyDrive/SVM
```

```
flat_data_arr=[]
target_arr=[]
datadir='/content/drive/MyDrive/SVM/Car-Moto-Dataset'
for i in Categories:
  print(f'loading... category : {i}')
  path=os.path.join(datadir,i)
  for img in os.listdir(path):
    img_array=imread(os.path.join(path,img))
    img_resized=resize(img_array,(150,150,3))
    flat_data_arr.append(img_resized.flatten())
    target_arr.append(Categories.index(i))
    print(i)
  print(f'loaded category:{i} successfully')
flat_data=np.array(flat_data_arr)
target=np.array(target_arr)
print("t",target)
df=pd.DataFrame(flat_data)
df['Target']=target
df
```

Hình 2.2: Load và tiền xử lý dữ liệu

*Load data từ drive

*Tiền xử lý dữ liệu

- Resize ảnh về kích thước 150x150 px với cả 3 kênh rgb.
- Chuyển ảnh rgb ma trận mxn thành mảng 1 chiều.
- Gộp tất cả mảng 1 chiều ở trên thành 1 danh sách.

*Gán nhãn từng ảnh qua mảng 1 chiều ở trên.

2.3. Chia dữ liệu ra tập train và test

```
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=77,stratify=y)
print('Splitted Successfully')
```

Hình 2.3: Chia tập dữ liệu

Tập train 80% và tập test 20%

2.4. Train model

```
[12] param_grid={'C':[0.1,1,10,100],'gamma':[0.0001,0.001,0.1,1],'kernel':['rbf','poly']}
     svc=svm.SVC(probability=True)
     print("The training of the model is started, please wait for while as it may take few minutes to complete")
     model=GridSearchCV(svc,param_grid)
     model.fit(x_train,y_train)
     print('The Model is trained well with the given images')
     model.best_params_
```

Hình 2.4: Train model

* C, kernel và gamma cho Bộ phân loại vectơ hỗ trợ.

C *float, mặc định = 1.0*

Tham số điều chỉnh. Độ mạnh của chính quy tỷ lệ nghịch với C. Phải là cực dương. Hình phạt là hình phạt l2 bình phương.

kernel *{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'} hoặc có thể gọi, default = 'rbf'*

Chỉ định kiểu hạt nhân sẽ được sử dụng trong thuật toán. Nếu không có giá trị nào được đưa ra, 'rbf' sẽ được sử dụng. Nếu một có thể gọi được cung cấp, nó được sử dụng để tính toán trước ma trận hạt nhân từ ma trận dữ liệu; ma trận đó phải là một mảng hình dạng. (n_samples, n_samples)

int độ , mặc định = 3

Mức độ của hàm nhân đa thức ('poly'). Bị bỏ qua bởi tất cả các hạt nhân khác.

gamma *{'scale', 'auto'} hoặc float, default = 'scale'*

Hệ số nhân cho 'rbf', 'poly' và 'sigmoid'.

- Nếu gamma = ‘scale’ (mặc định) được truyền thì nó sử dụng 1 / (n_features * X.var ()) làm giá trị của gamma
- Nếu 'auto', sử dụng 1 / n_features.

- Phân loại véc tơ hỗ trợ C.
- Sử dụng GridSearch để tìm ra tham số có tỷ lệ chính xác cao.
- Train mode với model.fit.

2.5. Đánh giá model qua tập test

```
[ ]  y_pred=model.predict(x_test)
     print("The predicted Data is :")
     y_pred
```

```
The predicted Data is :
array([1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1,
       0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
       0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0,
       1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,
       1, 1, 0, 1, 0, 1])
```

```
[ ]  print("The actual data is:")
     np.array(y_test)
```

```
The actual data is:
array([1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1,
       0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 1, 1, 1])
```

```
[ ]  #classification_report(y_pred,y_test)
     print(f"The model is {accuracy_score(y_pred,y_test)}% accurate")
     #confusion_matrix(y_pred,y_test)
```

```
The model is 0.78125% accurate
```

Hình 2.5: Đánh giá model

- Dự đoán tập test bằng model.
- In tập test.
- Tỉ lệ chính xác của model.

2.6. Lưu model, thử nghiệm với ảnh, train lại nếu kết quả sai

```
[ ]  pickle.dump(model,open('/content/drive/MyDrive/SVM/Car-Moto-Dataset/img_model.p','wb'))
     print("Pickle is dumped successfully")
```

```
Pickle is dumped successfully
```

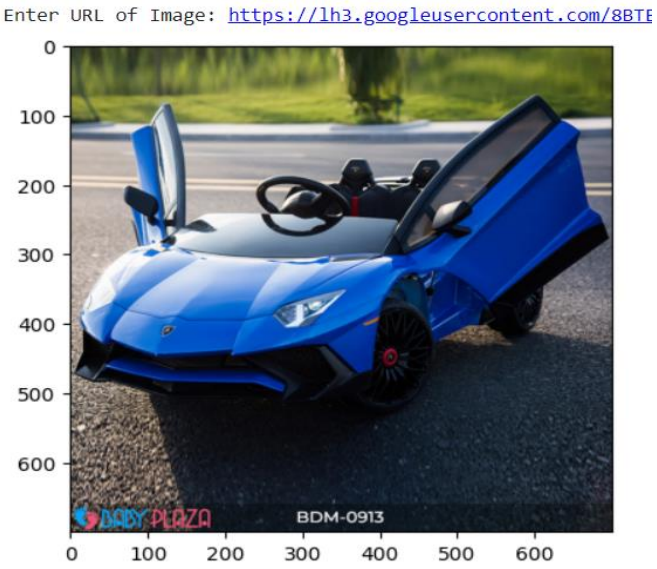
```
model=pickle.load(open('/content/drive/MyDrive/SVM/Car-Moto-Dataset/img_model.p','rb'))

url=input('Enter URL of Image: ')
img=imread(url)
plt.imshow(img)
plt.show()
img_resize=resize(img,(150,150,3))
l=[img_resize.flatten()]
probability=model.predict_proba(l)
for ind,val in enumerate(Categories):
  print(f'{val} = {probability[0][ind]*100}%')
print("The predicted image is : "+Categories[model.predict(l)[0]])
print(f'Is the image a {Categories[model.predict(l)[0]]} ?(y/n)')
while(True):
  b=input()
  if(b=="y" or b=="n"):
    break
  print("please enter either y or n")

if(b=='n'):
  print("What is the image?")
  for i in range(len(Categories)):
    print(f"Enter {i} for {Categories[i]}")
  k=int(input())
  while(k<0 or k>=len(Categories)):
    print(f"Please enter a valid number between 0-{len(Categories)-1}")
    k=int(input())
  print("Please wait for a while for the model to learn from this image :)")
```

```
  flat_arr=flat_data_arr.copy()
  tar_arr=target_arr.copy()
  tar_arr.append(k)
  flat_arr.extend(l)
  tar_arr=np.array(tar_arr)
  flat_df=np.array(flat_arr)
  df1=pd.DataFrame(flat_df)
  df1['Target']=tar_arr
  model1=GridSearchCV(svc,param_grid)
  x1=df1.iloc[:,:-1]
  y1=df1.iloc[:,-1]
  x_train1,x_test1,y_train1,y_test1=train_test_split(x1,y1,test_size=0.20,random_state=77,stratify=y1)
  d={}
  for i in model.best_params_:
    d[i]=[model.best_params_[i]]
  model1=GridSearchCV(svc,d)
  model1.fit(x_train1,y_train1)
  y_pred1=model.predict(x_test1)
  print(f"The model is now {accuracy_score(y_pred1,y_test1)*100}% accurate")
  pickle.dump(model1,open('img_model.p','wb'))
print("Thank you for your feedback")
```

Hình 2.6: Lưu và thử nghiệm model

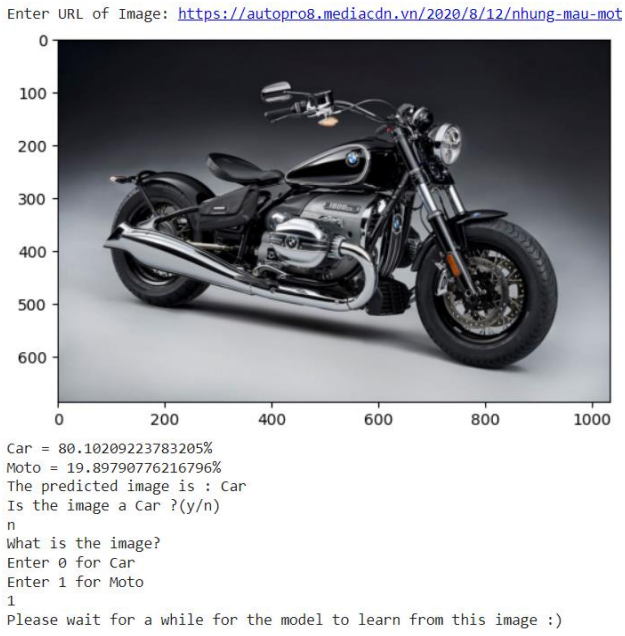
CHƯƠNG 3: KẾT QUẢ

3.1. Trường hợp kết quả trả về là đúng



Hình 3.1: Kết quả đúng

3.2. Trường hợp kết quả trả về là sai



Hình 3.2: Kết quả sai

KẾT LUẬN

Thông qua bài báo cáo này, chúng ta đã tìm hiểu về cách SVM hoạt động, cách nó phân loại và các ứng dụng thực tế của nó. Chúng ta cũng đã xem xét ví dụ và thực hành với tập dữ liệu để hiểu rõ hơn về cách áp dụng SVM trong thực tế. SVM là một trong những thuật toán học máy quan trọng nhất hiện nay, với khả năng phân loại và dự đoán rất cao. Nhưng bên cạnh đó thuật toán vẫn còn những ưu, nhược điểm:

Ưu điểm: Đây là thuật toán hoạt động hiệu quả với không gian cao chiều (high dimensional spaces). Thuật toán tiêu tốn ít bộ nhớ vì chỉ sử dụng các điểm trong tập hỗ trợ để dự báo trong hàm quyết định. Chúng ta có thể tạo ra nhiều hàm quyết định từ những hàm kernel khác nhau.

Nhược điểm: Đây là thuật toán hoạt động hiệu quả với không gian cao chiều (high dimensional spaces). Thuật toán tiêu tốn ít bộ nhớ vì chỉ sử dụng các điểm trong tập hỗ trợ để dự báo trong hàm quyết định. Chúng ta có thể tạo ra nhiều hàm quyết định từ những hàm kernel khác nhau.

Trong tương lai, SVM còn tiềm năng phát triển và ứng dụng rộng rãi trong nhiều lĩnh vực, đặc biệt là trong lĩnh vực Công nghệ Thông tin và Truyền thông. Vì vậy, chúng ta cần phải tiếp tục nghiên cứu và phát triển SVM để nó có thể đáp ứng được nhu cầu ngày càng tăng của xã hội về các bài toán phân loại và dự đoán.