ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN

PHÂN LỚP ẢNH CHỮ SỐ VIẾT TAY BẰNG SVM

GVHD:

Thầy: Trần Trung Kiên

Thầy: Phạm Trọng Nghĩa

Cô: Phan Thị Phương Uyên

THỰC HIỆN:

KIM ĐÌNH LỘC MSSV: 1712568

NGUYỄN HỮU THẮNG MSSV: 1712756

TP. Hồ Chí Minh, Tháng 8 Năm 2020

I. Tìm hiểu về lý thuyết mô hình SVM (Support Vector Machine):

1. SVM cho dữ liệu khả tách tuyến tính:

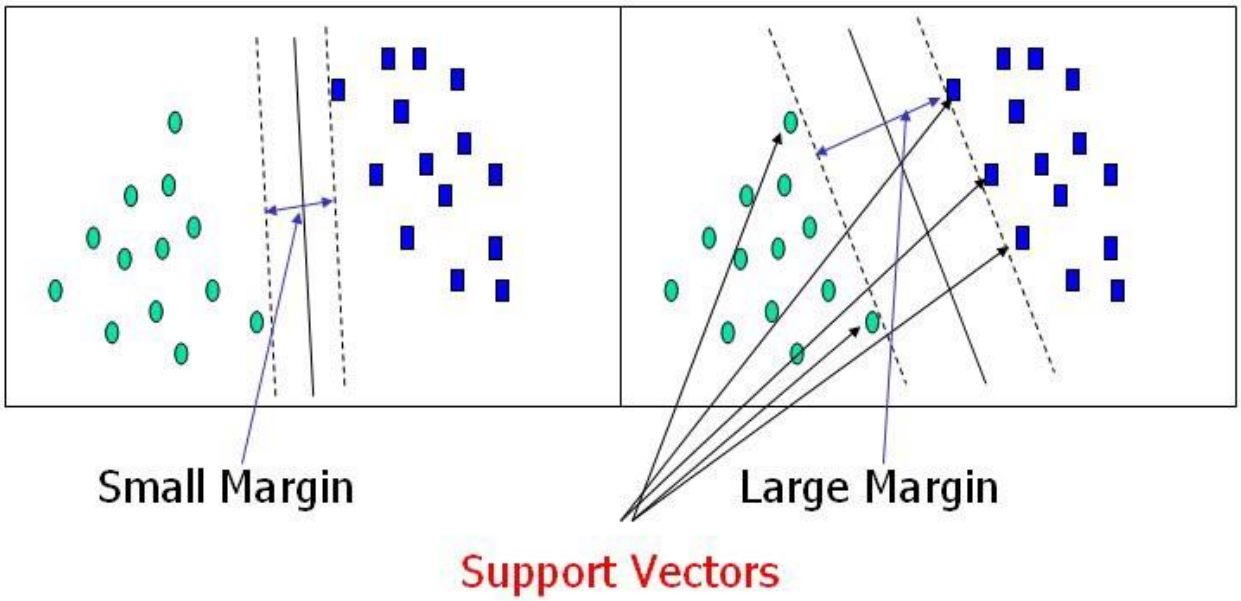
- Thuật toán SVM ban đầu được phát minh bởi Vladimir N. Vapnik và Alexey Ya. Chervonenkis vào năm 1963. Năm 1992, Bernhard Boser, Isabelle Guyon và Vladimir Vapnik đề xuất một cách tạo bộ phân loại phi tuyến bằng cách áp dụng kernel trick để ánh xạ dữ liệu đầu vào lên không gian đặc trưng cao chiều.

- Trong học máy, SVM là mô hình học có giám sát với các thuật toán học liên quan phân tích dữ liệu được sử dụng để phân loại và phân tích hồi quy. SVM tạo ra một siêu phẳng(hyper-plane) hoặc một tập hợp các siêu phẳng trong một không gian chiều cao hoặc vô hạn, có thể được sử dụng để phân loại , hồi quy hoặc các nhiệm vụ khác như phát hiện ngoại lệ(outlier). Trong SVM, sự phân tách tốt đạt được nhờ siêu phẳng có khoảng cách lớn nhất đến điểm dữ liệu huấn luyện gần nhất của bất kỳ lớp nào (được gọi là lề).

- Thuật toán học của SVM có tập hypothesis :

$$\mathrm{H_d} = \{\mathrm{h(x)} = \mathrm{sign}(\mathrm{w^T\ x} + \mathrm{b}) \mid \mathrm{w} \in \mathrm{R^d}, \mathrm{b} \in \mathrm{R}\}$$

- Dựa trên tập hypothesis trên, SVM sẽ cố tìm (w, b) của mặt siêu phẳng phân chia dữ liệu có $E_{in} = 0$ và lề(margin) có độ lớn lớn nhất có thể.



Ảnh minh hoạ cho việc tìm mặt siêu phẳng

- Với điều kiện $E_{in} = 0$ nghĩa là ta cần phải tính:

$$\mathrm{y^{(n)}}(\mathrm{w^T x^{(n)}} + \mathrm{b}) > 0\ \forall \mathrm{n} = 1, 2.., \mathrm{N}(1)$$

+ Để đơn giản cho các bước sau ta cần chuẩn hoá (w, b) trong (1) và ta cần tính:

$$\min_{n=1,\dots,N} y^{(n)}\left(w^T x^{(n)} + b\right) = 1$$

- Tiếp đó, để thoả điều kiện có lề lớn nhất, ta cần tính khoảng cách từ 1 điểm $x^{(n)}$ đến mặt phẳng $w^T x^{(n)} + b$:

$$dist = \frac{1}{\|w\|}(w^T x^{(n)} + b)$$

- Do đó, lề cần tìm có giá trị là:

$$margin = \min_{n=1,\dots,N} \frac{1}{\|w\|}(w^T x^{(n)} + b) = \frac{1}{\|w\|}$$

- Cuối cùng, ta cần tính(w, b) thoả :

$$\max(margin) = \max_{w,b} \frac{1}{\|w\|}$$

với điều kiện: $\min_{n=1,\dots,N} y^{(n)}\left(w^T x^{(n)} + b\right) = 1$

Suy ra, ta tiếp tục tính : $\min_{w,b} \frac{1}{2} w^T w$

với điều kiện: $\min_{n=1,\dots,N} y^{(n)}\left(w^T x^{(n)} + b\right) = 1$

Suy ra, ta tiếp tục tính : $\min_{w,b} \frac{1}{2} w^T w$

với điều kiện: $y^{(n)}(w^T x^{(n)} + b) \geq 1 \; \forall n = 1,..,N$

- Để giải tiếp tục bài toán tối ưu hoá trên ta cần dùng Quadratic Programming/ SMO.Qua đó, ta sẽ tính ra được giá trị của $\boldsymbol{\alpha} = \alpha_1, \cdots, \alpha_N$ và w :

$$w = \sum_{n=1}^{N} \alpha_n y^{(n)} x^{(n)} = \sum_{\alpha_n > 0} \alpha_n y^{(n)} x^{(n)}$$

- Kết hợp với điều kiện KKT, ta tính được:

$$\alpha_n\left(y^{(n)}\left(w^T x^{(n)} + b\right) - 1\right) = 0 \; \forall n = 1,..,N$$

$$\alpha_n > 0 => y^{(n)}\left(w^T x^{(n)} + b\right) = 1$$

=> $x^{(n)}$ nằm trên lề hay $x^{(n)}$ là support vector

- Cuối cùng, ta tính được b khi dùng 1 số support vector và w đã tính thế vào biểu thức:

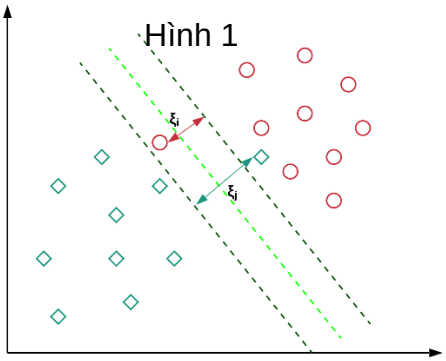
$$y^{(n)}\left(w^T x^{(n)} + b\right) = 1$$

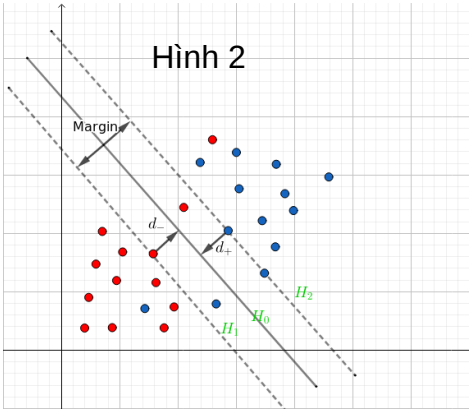
- Qua các bước trên, ta thấy các vectơ hỗ trợ là các điểm dữ liệu nằm gần mặt siêu phẳng cần tìm(nằm trên lề) ảnh hưởng đến vị trí và hướng của mặt siêu phẳng đó. Độ phức tạp của SVM phụ thuộc vào số lượng vectơ hỗ trợ hơn là số

chiều của không gian đầu vào vì chúng ảnh hưởng trực tiếp đến sự tối đa hoá lề. Các vectơ hỗ trợ cuối cùng được giữ lại từ tập dữ liệu ban đầu để góp phần dự đoán các kết quả sau này và số lượng của chúng không nhiều(tuỳ theo dữ liệu đầu vào).

2. SVM cho dữ liệu không khả tách tuyến tính:

2.1 Soft-margin:





- Thông thường việc xác định siêu phẳng phân tách giữa 2 lớp dữ liệu trong điều kiện lý tưởng như dữ liệu có thể phân tách tuyến tính (hình 1), thì ta có thể thể dễ dàng tìm được 2 siêu phẳng mà không có điểm nào giữa chúng. Nhưng đối với hình 2 có nhiều tập gây nhiễu thì việc sử dụng soft margin với siêu tham số C để xác định penalty degree đối với các lỗi này. Một margin lớn hơn trong Soft Margin SVM, chúng ta cần hy sinh một vài điểm dữ liệu bằng cách chấp nhận cho chúng rơi vào vùng không an toàn. Tất nhiên, chúng ta phải hạn chế sự hy sinh này, nếu không, chúng ta có thể tạo ra một biên cực lớn bằng cách hy sinh hầu hết các điểm. Vậy hàm mục tiêu nên là một sự kết hợp để tối đa margin và tối thiểu sự hy sinh.

- Phương pháp giải của bài toán này chỉ khác với SVM cho dữ liệu khả tách tuyến tính tại phần tối ưu hoá giá trị lề như sau:

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_{n=1}^{N} \xi_n$$

với điều kiện: $y^{(n)}\left(w^T x^{(n)} + b \right) \geq 1 - \xi_n \; \forall n = 1, .., N; \; \xi_n \geq 0$

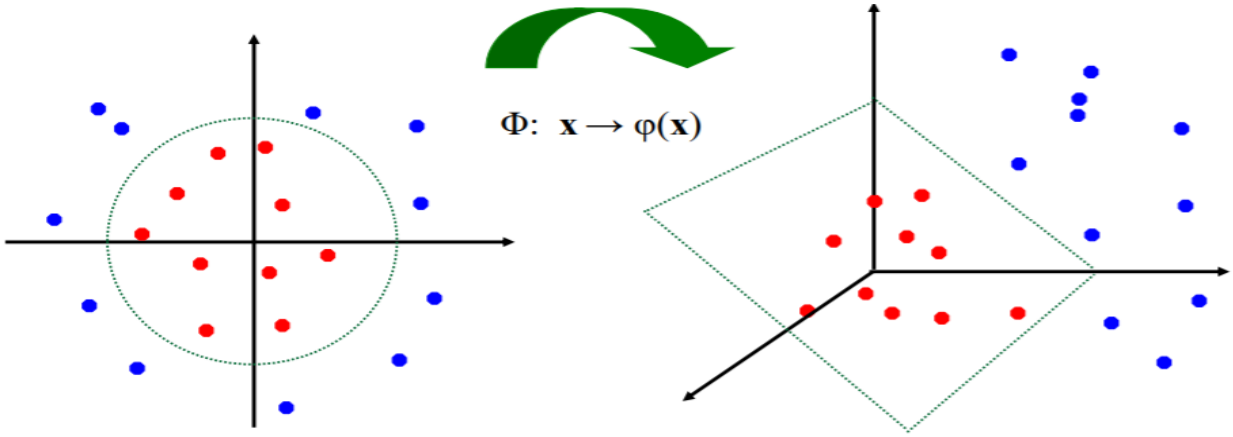
trong đó: ξ_n là độ đo sự hi sinh(slack variable) và

ξ_n=0 tương ứng với những điểm dữ liệu nằm trong vùng an toàn.

$0<\xi_n \leq 1$tương ứng với những điểm nằm trong vùng không an toàn những vẫn được phân loại đúng, tức vẫn nằm về đúng phía so với đường phân chia.

ξ_n>tương ứng với các điểm bị phân loại sai.

2.2. Kernel:



Đối với các bài toán có không gian dữ liệu là phi tuyến tính. giải quyết bài toán trong trường hợp này chúng ra cần biểu diễn (ánh xạ) dữ liệu từ không gian ban đầu X sang không gian F bằng một hàm ánh xạ phi tuyến

$$\phi : X \rightarrow F$$
$$x \rightarrow \phi(x)$$

Trong không gian F tập dữ liệu có thể phân tách tuyến tính. Nhưng nãy sinh một vấn đề lớn đó là trong không gian mới này số chiều của giữ liệu tăng lên rất nhiều so với không gian ban đầu làm cho chi phí tính toán vô cùng tốn kém. Ta có thể sử dụng kernel Trick

$$K(x,z) =< \phi(x).\phi(z) >$$

Một số hàm kernel thường dùng là:

Linear: $K(\boldsymbol{x},\boldsymbol{z}) = \boldsymbol{x}^T\boldsymbol{z}$

Polynomial: $K(\boldsymbol{x},\boldsymbol{z}) = (1+\boldsymbol{x}^T\boldsymbol{z})^Q$

Gaussan RBF: $K(\boldsymbol{x},\boldsymbol{z}) = \exp\left(-\gamma||\mathbf{x}-\mathbf{z}||^2\right.$

Sigmoidal: $K(\boldsymbol{x},\boldsymbol{z}) = \tanh\left(\beta_0\boldsymbol{x}^T\boldsymbol{z}+\beta_1\right)$

2.3 Siêu tham số C:

- Tham số C: là tham số xác định mức độ phạt đối với các lỗi:

  + Khi C quá nhỏ: việc phân lớp có thể không hiệu quả do có quá nhiều điểm bị phân lớp sai. Trước đó, chúng ta xác định là hy sinh một số điểm dữ liệu để việc phân lớp trở nên tốt hơn, nhưng mức độ phạt quá nhỏ so với lỗi sẽ dẫn đến phân lớp sai nhiều hơn.

  + Khi C quá lớn: Khi C có độ phạt lớn, bộ phân loại sẽ bị phạt nặng vì dữ liệu được phân loại sai và do đó đường phân loại bị bẻ cong để tránh bất kỳ điểm dữ liệu bị phân loại sai. Nhưng khi C quá lớn làm cho đường cong ngoằng ngèo khiến cho bộ phân lớp làm việc tốt đến mức không có một điểm nào phân lớp bị sai, điều đó dẫn tới việc có dấu hiệu overfitting và cũng khiến cho margin quá nhỏ.

+ Khi C vừa đủ đối với model: Bộ phân lớp sẽ phân lớp sẽ không dung nạp các điểm dữ liệu bị phân loại sai do đó ranh giới quyết định sẽ ít sai hơn và có phương sai lớn hơn, phụ thuộc nhiều hơn vào các điểm riêng lẻ.

2.4 Siêu tham số γ:

γ là một tham số của RBF kernel và có thể được coi là 'độ lan truyền' của kernel và là vùng quyết định.

Khi gamma ở mức thấp, 'đường phân lớp' và ranh giới quyết định rất thấp và do đó vùng quyết định rất rộng

Khi gamma ở mức cao, đường phân lớp' và ranh giới quyết định rất cao do đó các vùng quyết định tạo thành các đảo nhỏ.

2.5 Vấn đề K lớp(Multi – class support vector machine)

Thông thường ta có 2 lựa chọn là One vs One và One vs All

2.5.1 One vs One

Số lượng bộ phân loại cần thiết: $\frac{n(n-1)}{2}$

SVM được áp dụng trên phân loại nhị phân, chia các điểm dữ liệu theo 1 hoặc 0, Multi-class chia thành nhiều trường hợp như thế. Về cơ bản nó chia lớp x với tất cả các lớp còn lại. Theo đó, 1 lớp nhất định phải phân biệt so với tất cả các lớp còn lại.

Chiến lược One-vs-One chia phân loại nhiều lớp thành một bài toán phân loại nhị phân cho mỗi cặp lớp.

2.5.2 One vs All

Số lượng bộ phân loại cần thiết: n^2

One vs All là việc tách tập dữ liệu nhiều lớp thành nhiều bài toán phân loại nhị phân. Sau đó, một bộ phân loại nhị phân được đào tạo về mỗi bài toán phân loại nhị phân và các dự đoán được thực hiện bằng cách sử dụng mô hình đáng tin cậy nhất.

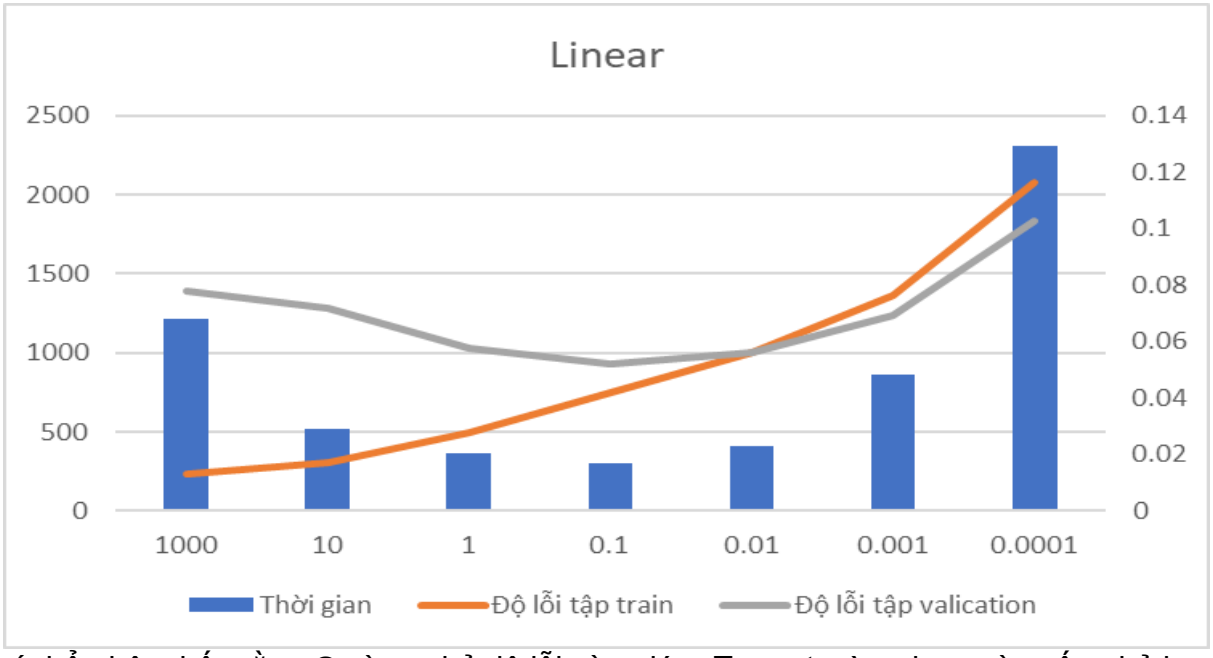
Một nhược điểm có thể có của phương pháp này là nó yêu cầu một mô hình được tạo cho mỗi lớp

Chiến lược One-vs-Rest chia sự phân loại nhiều lớp thành một bài toán phân loại nhị phân cho mỗi lớp.

II. Huấn luyện mô hình SVM cho ảnh chữ số viết tay.

1. Mô hình linear:

C	1000	10	1	0.1	0.01	0.001	0.0001
Thời gian	1212	517	366	302	412	864	2304
Độ lỗi tập train	0.013	0.01692	0.02754	0.04188	0.05594	0.0761	0.11612
Độ lỗi tập valication	0.0777	0.0716	0.0577	0.0519	0.0563	0.0691	0.1029



Ta có thể nhận thấy rằng C càng nhỏ độ lỗi càng lớn. Trong trường hợp này nếu nhỏ hơn 0.01 thì độ lỗi của tập train bắt đầu lớn hơn so với tập val. Và trong đoạn trước đó thì ngược lại.

Ta có thể chọn mô hình có C=0.1 là hàm dự đoán cuối cùng.

Score: 0.9463 => Độ lỗi: 0.0637

2. Thời gian chạy mô hình RBF Kernel:

C / γ	**100**	**10**	**1**	**0.1**	**0.01**	**0.001**	**0.0001**
0.1	6848	6907	6502	4276	5127	5383	4905
0.01	314	319	377	860	2293	5374	4893
0.001	286	364	690	1781	4552	5448	5562
0.0001	370	703	1691	4404	5411	4909	5626

3. Độ lỗi mô hình RBF Kernel:

Độ lỗi tập train:

C / γ	**100**	**10**	**1**	**0.1**	**0.01**	**0.001**	**0.0001**
0.1	0.00	0.00	0.00004	0.28952	0.7843	0.88644	0.88644
0.01	0.00	0.00058	0.01526	0.04702	0.0939	0.47486	0.88644
0.001	0.00942	0.03794	0.06422	0.09824	0.25374	0.88644	0.88644
0.0001	0.04984	0.06864	0.0997	0.23658	0.88644	0.88644	0.88644

Độ lỗi tập valication:

γ \ C	**100**	**10**	**1**	**0.1**	**0.01**	**0.001**	**0.0001**
0.1	0.0434	0.0434	0.0448	0.3125	0.7828	0.8936	0.8936
0.01	0.0168	0.0165	0.0223	0.0422	0.0822	0.4596	0.8936
0.001	0.0282	0.0408	0.0589	0.0861	0.2279	0.8936	0.8936
0.0001	0.0524	0.0631	0.0879	0.2105	0.8936	0.8936	0.8936

Ta nhận thấy rằng γ và C có ảnh hưởng đến kết quả của một mô hình.

Khi đạt một giá trị sigma nào đó thì độ lỗi trên tập valication đạt giá trị cực tiểu, từ đó tập test có một mô hình tốt nhất có thể.

Ta chọn hàm dự đoán cuối cùng là: $\mathrm{C} = 10,\ \gamma = 0.01$:

Score: 0.982 => Độ lỗi: 0.018