

TRƯỜNG ĐẠI HỌC SƯ PHẠM TP. HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN
PHÒNG SAU ĐẠI HỌC

TIỂU LUẬN KẾT THÚC HỌC PHẦN
MÔN HỌC MÁY NÂNG CAO

BÀI TOÁN PHÂN LOẠI HAI LỚP VÀ
NHIỀU LỚP VỚI THUẬT TOÁN
SUPPORT VECTOR MACHINE

Giáo viên hướng dẫn:

TS. Lê Minh Trung

Người thực hiện:

Nguyễn Thị Hồng Nhiên

Võ Thị Hồng Thắm

Học viên cao học Khóa 26

Chuyên ngành : Khoa học máy tính

Thành phố Hồ Chí Minh - 2016

MỤC LỤC

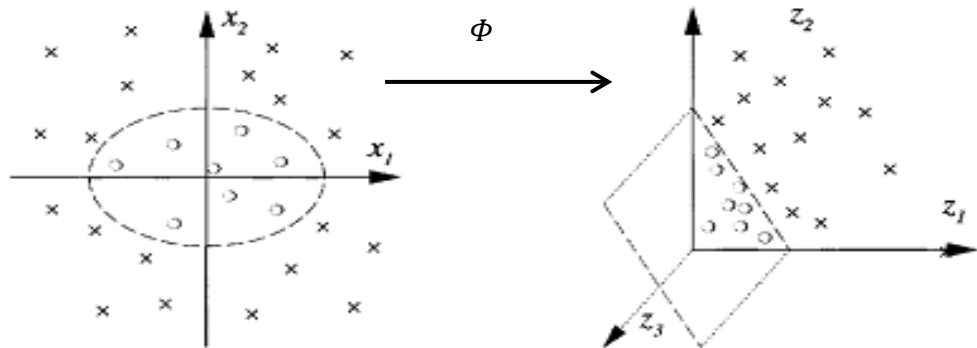
PHẦN 1: NỘI DUNG LÝ THUYẾT.....	2
1/ Lý thuyết của SVM cho bài toán phân loại hai lớp.....	2
2/ Trình bày chiến lược pairwise và one-against-all để kết hợp nhiều máy phân loại hai lớp cho bài toán phân loại nhiều lớp	8
3/ Lý thuyết thuật toán KNN (K người láng giềng gần nhất)	9
PHẦN 2: THỬ NGHIỆM TRÊN 5 DATASET NHIỀU LỚP	12
1/ Dữ liệu thử nghiệm.....	12
2/ Kết quả thử nghiệm.....	13

PHẦN 1: NỘI DUNG LÝ THUYẾT

1/ Lý thuyết của SVM cho bài toán phân loại hai lớp

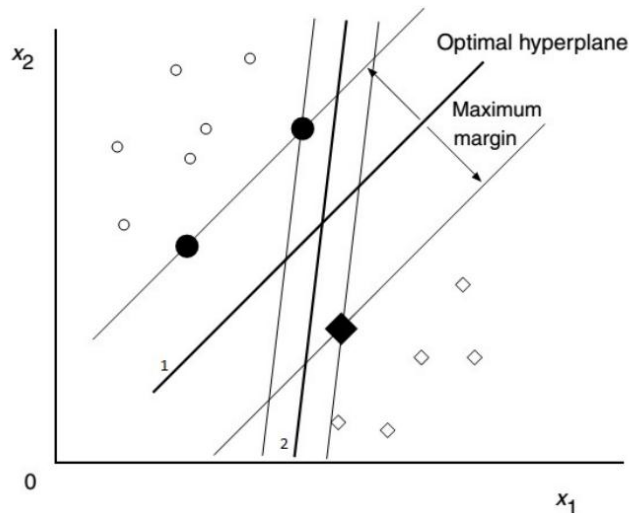
Ý tưởng của SVM

- Cho trước tập huấn luyện $D = \{(x_i, y_i)\}_{i=1}^N$ trong đó $x_i \in R^d$ và $y_i \in \{-1, 1\}$
- Dùng ánh xạ Φ biến đổi dữ liệu từ không gian input vào không gian feature



Ánh xạ Φ từ không gian Input vào không gian feature.

- Trong không gian feature, tập huấn luyện có dạng $D_\Phi = \{(\Phi(x_i), y_i)\}_{i=1}^N$ tách được tuyến tính, tức tồn tại một siêu phẳng tối ưu có thể tách dữ liệu
- $(H_0): w_0^T \Phi(x) + b_0 = 0$ hay nói cách khác $w_0 y_i (\Phi(x_i) + b_0) \geq 0$ với $\forall i = 1, 2, \dots, N$



Học siêu phẳng tối ưu trong không gian feature

- Siêu phẳng này khi ánh xạ trở lại không gian input có thể tạo ra một biên quyết định có hình dạng không tuyến tính.

Cách chọn siêu phẳng tối ưu

Trong không gian feature, có nhiều siêu phẳng có thể tách được dữ liệu D_Φ . Cần chọn một siêu phẳng tối ưu. Siêu phẳng tối ưu là siêu phẳng có lề lớn nhất. Lề của một siêu phẳng đối với tập huấn luyện D_Φ là khoảng cách từ điểm gần nhất tới siêu phẳng đó. Như vậy có thể định nghĩa lề của tập D_Φ tới siêu phẳng $(H): w^T \Phi(x) + b = 0$ như sau:

$$M(D_\Phi, H) = \min_{1 \leq i \leq N} \frac{|w^T \Phi(x_i) + b|}{\|w\|}$$

Bài toán học siêu phẳng tối ưu được chuyển thành

$$\max_{w, b} \min_{1 \leq i \leq N} \frac{y_i (w^T \Phi(x_i) + b)}{\|w\|}$$

$$\text{s.t.: } y_i (w^T \Phi(x_i) + b) \geq 0, \forall i = 1, \dots, N$$

Mô hình cứng của Support Vector Machine

Chúng ta cần giải bài toán tối ưu sau để tìm siêu phẳng có lề lớn nhất

$$\max_{w,b} \min_{1 \leq i \leq N} \left(\frac{y_i (w^T \Phi(x_i) + b)}{\|w\|} \right)$$

$$\text{s.t.: } y_i (w^T \Phi(x_i) + b) \geq 0, \forall i = 1, \dots, N$$

Bởi vì lẽ của siêu phẳng (w, b) bằng với lẽ của siêu phẳng (kw, kb) trong đó $k > 0$. Do đó, chúng ta có thể giả sử dữ liệu có thể tách rời tuyến tính, tức là $\min_{1 \leq i \leq N} y_i w^T \Phi(x_i) + b = 1$ và bài toán tối ưu ở trên trở thành:

$$\max_{w,b} \frac{1}{\|w\|}$$

$$\text{s.t.: } y_i (w^T \Phi(x_i) + b) \geq 0, \forall i = 1, \dots, N$$

hay chuyển thành bài toán cực tiểu hóa:

$$\min_{w,b} \frac{1}{2} \|w\|^2$$

$$\text{s.t.: } y_i (w^T \Phi(x_i) + b) \geq 0, \forall i = 1, \dots, N$$

Mô hình mềm của Support Vector Machine

Điều kiện của Mô hình cứng của Support Vector Machine không phù hợp với dữ liệu thực tế vì dữ liệu thực tế có thể chứa các điểm ngoại lệ (outlier) hay nhiễu (noise). Vì vậy, mô hình mềm (soft model) của SVM sử dụng các biến slack với $[\xi_i]_{i=1}^N$ với $\xi_i \geq 0$ để cho phép dữ liệu nằm trong margin. Bài toán tối ưu mô hình mềm của SVM như sau:

$$\min_{w,b} \left(\frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i \right)$$

$$\text{s.t.: } y_i (w^T \Phi(x_i) + b) \geq 1 - \xi_i, \forall i = 1, \dots, N$$

$$\xi_i \geq 0, \forall i = 1, \dots, N$$

Trong đó $C > 0$ được gọi là *tham số bù trừ* (trade-off parameter) để cân bằng giữa lỗi tổng quát (tỉ lệ với $1/\|w\|$) và lỗi thực nghiệm (tỉ lệ với $\sum_{i=1}^N \xi_i$).

Để giải quyết bài toán tối ưu trên, cần áp dụng định lý Karush-Kuhn-Tucker, hàm Lagrange có dạng:

$$\mathcal{L}(w, b, \xi, \alpha, \beta) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i [y_i(w^T \Phi(x_i) + b) - 1 + \xi_i] - \sum_{i=1}^N \beta_i \xi_i$$

- Thiết lập đạo hàm của hàm Lagrange theo các biến chính w, b, ξ bằng 0

$$0 = \nabla_w \mathcal{L} = w - \sum_{i=1}^N y_i \alpha_i \Phi(x_i) \rightarrow w = \sum_{i=1}^N y_i \alpha_i \Phi(x_i)$$

$$0 = \nabla_b \mathcal{L} = \sum_{i=1}^N y_i \alpha_i$$

$$0 = \nabla_{\xi} \mathcal{L} = C - \alpha_i - \beta_i, \forall i = 1, \dots, N \rightarrow \alpha_i + \beta_i = C, \forall i = 1, \dots, N$$

Điều kiện KKT có dạng sau:

$$\alpha_i \geq 0, y_i(w^T \Phi(x_i) + b) \geq 1 - \xi_i, \alpha_i (y_i(w^T \Phi(x_i) + b) - 1 + \xi_i) = 0$$

$$\beta_i \geq 0, \xi_i \geq 0, \beta_i \xi_i = 0$$

- Bài toán tối ưu dạng toàn phương bậc 2:

$$\min_{\alpha} \left(\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j K(x_i, x_j) \alpha_i \alpha_j - \sum_{i=1}^N x_i \right)$$

$$s. t. : \sum_{i=1}^N \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C, \forall i = 1, \dots, N$$

Trong đó $K(x, x') = \Phi(x)^T \Phi(x')$ là hàm hạt nhân.

Để giải quyết bài toán tối ưu hóa dạng toàn phương bậc hai, có rất nhiều hướng tiếp cận được đề xuất như: phương pháp điểm trong và các phương pháp phân rã với độ phức tạp tuyến tính với số lượng dữ liệu.

Điều kiện Karush-Kuhn-Tucker (KKT) cho phép tìm ra các tính chất của các vector hỗ trợ (support vector)

- $\alpha_i = 0: \beta_i = C \rightarrow \xi_i = 0 \rightarrow y_i (w^T \Phi(x_i) + b) \geq 1$, khi này các vector x_i sẽ nằm ngoài lề và được phân lớp đúng.
- $0 \leq \alpha_i \leq C: \beta_i \geq 0 \rightarrow \xi_i = 0 \rightarrow y_i (w^T \Phi(x_i) + b) = 1$, các vector x_i sẽ nằm trên lề và được phân lớp đúng.
- $\alpha_i = C$ khi này $y_i (w^T \Phi(x_i) + b) \geq 1 - \xi_i$, các vector x_i có thể được phân loại đúng hoặc sai.

Kí hiệu $I_{BSV} = \{i: 0 < \alpha_i < C \wedge 1 \leq i \leq N\}$ là chỉ số của tập các vector hỗ trợ được bao bởi lề lân cận, $I_{BSV} = \{i: \alpha_i > 0 \wedge 1 \leq i \leq N\}$ các chỉ số của các vector hỗ trợ. Để ước lượng giá trị của b , chúng ta dựa vào tập các vector hỗ trợ được bao. Với tất cả các chỉ số $i \in I_{BSV}$, ta có phương trình sau:

$$y_i (w^T \Phi(x_i) + b) = 1$$

Nhân y_i vào hai vế của phương trình, $y_i (w^T \Phi(x_i) + b) = y_i, \forall i \in I_{BSV}$. Tính trung bình cho tất cả $i \in I_{BSV}$, ta đạt được:

$$b = \frac{1}{|I_{BSV}|} \sum_{i \in I_{BSV}} (y_i - w^T \Phi(x_i)) = \frac{1}{|I_{BSV}|} \sum_{i \in I_{BSV}} \left(y_i - \sum_{j=1}^N y_i \alpha_i K(x_i x_j) \right)$$

Trong đó $|I_{BSV}|$ là số phần tử của tập I_{BSV} .

Hàm quyết định có dạng:

$$f(x) - \text{sign}(w^T \Phi(x) + b) = \text{sign}\left(\sum_{i=1}^N y_i \alpha_i K(x_i, x) + b\right)$$

Hàm quyết định của SVM được biểu thị trong phương trình, mặc dù một biên quyết định tuyến tính được học ra trong không gian đặc trưng, khi ánh xạ ngược trở lại trong không gian nguyên thủy của dữ liệu hàm quyết định này sẽ trở thành một biên quyết định phi tuyến.

Một số hàm hạt nhân thông dụng

- Linear kernel

Hàm hạt nhân tuyến tính có công thức $K(x, x') = x^T x' + c$ trong đó c là một hằng số. Không gian feature sẽ trùng với không gian nguyên thủy ban đầu của dữ liệu. Hàm hạt nhân này thường được sử dụng trong trường hợp không gian dữ liệu thưa ví dụ như trong bài toán phân loại văn bản.

- Polynomial kernel

Hàm hạt nhân đa thức có dạng $K(x, x') = (x^T x' + c)^d$ trong đó thành phần mũ d là một giá trị dương. Không gian feature gắn liền với hàm hạt nhân đa thức sẽ có số chiều rất lớn và tỉ lệ thuận với d .

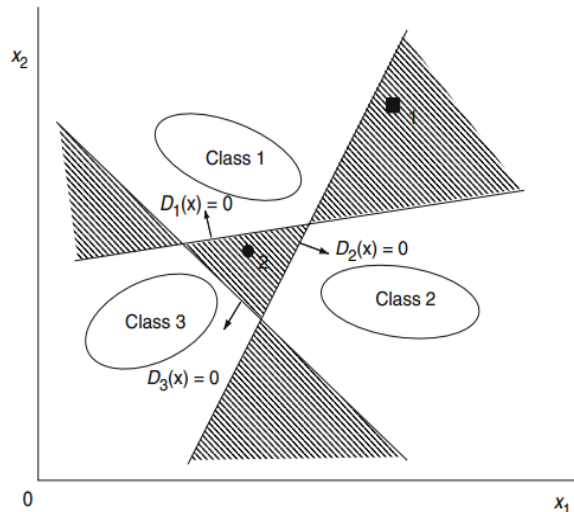
- Radial Basic Function (RBF) kernel

Hàm hạt nhân RBF có dạng $K(x, x') = e^{-\gamma \|x - x'\|^2}$ trong đó γ là một giá trị dương đại diện cho độ dày của hàm hạt nhân. Tham số điều chỉnh γ đóng một vai trò quan trọng trong việc thực hiện chức năng của hàm hạt nhân, và cần được điều chỉnh một cách cẩn thận khi áp dụng vào những bài toán cụ thể. Nếu ước lượng giá trị quá cao, các thành phần mũ sẽ hoạt động gần như tuyến tính và phép chiếu trên không gian nhiều chiều hơn sẽ bắt đầu làm mất dần khả năng phi tuyến tính của hàm hạt nhân. Trái lại, nếu ước lượng giá trị

quá thấp, hàm chức năng sẽ mất đi khả năng tổng quát hóa và biên quyết định sẽ rất nhạy cảm với nhiễu trong tập dữ liệu huấn luyện.

2/ Trình bày chiến lược pairwise và one-against-all để kết hợp nhiều máy phân loại hai lớp cho bài toán phân loại nhiều lớp

One-against-All



Xây dựng các siêu phẳng tối ưu tách lớp i ($1 \leq i \leq m$) và các lớp còn lại

- $D_i(x) = w_i^T \Phi(x_i) + b_i$
- $D_i(x)$ càng lớn, x càng ở xa siêu phẳng thứ i

Phân loại như sau $x \in C_t$ với $t = \operatorname{argmax}_i D_i(x)$

Pairwise

Với mỗi cặp lớp C_i, C_j với $i < j$, dùng dữ liệu của hai lớp này để huấn luyện siêu phẳng $D_{ij}(x) = w_{ij}^T \Phi(x) + b_{ij}$

- Có $m(m+1)/2$ cặp lớp
- $D_{ji}(x) = -D_{ij}(x)$

Để quyết định nhãn cho x , sử dụng bỏ phiếu (vote)

- $v_i(x) = \sum_{j \neq i} \mathbb{I}_{D_{ij}(x) \geq 0} \rightarrow$ số lần phân loại x vào lớp i
- $x \in C_t$ với $t = \operatorname{argmax}_i v_i(x)$

3/ Lý thuyết thuật toán KNN (K người láng giềng gần nhất)

Thuật toán KNN là thuật toán có mục đích phân loại lớp cho một mẫu mới (Query point) dựa trên các thuộc tính và lớp của các mẫu sẵn có (Training Data), các mẫu này được nằm trong một hệ gọi là không gian mẫu.

Một đối tượng được phân lớp dựa vào K láng giềng của nó. K là số nguyên dương được xác định trước (thường là khoảng cách giữa các đối tượng) khi thực hiện thuật toán.

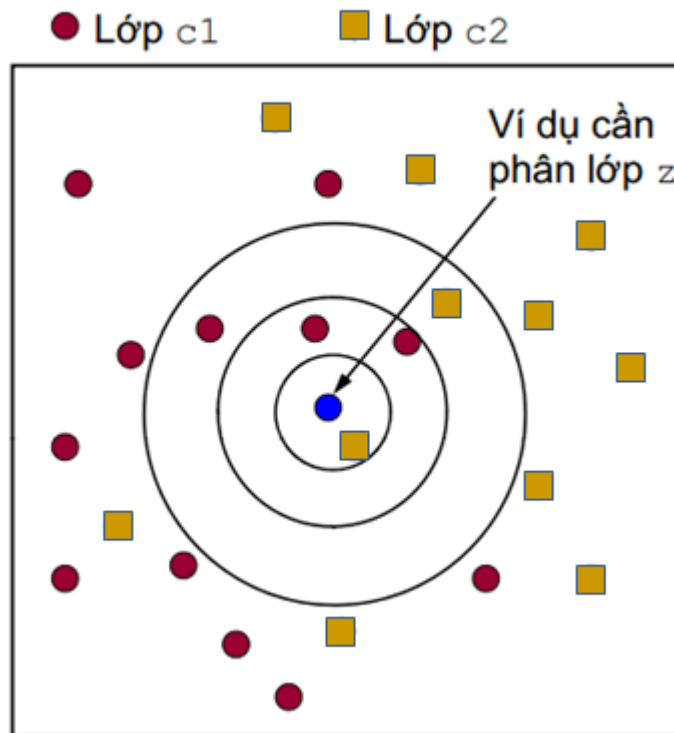
Ý tưởng thuật toán KNN

Các mẫu được mô tả bằng n chiều thuộc tính số. Mỗi mẫu đại diện cho một điểm trong một chiều không gian n chiều. Theo cách này tất cả các mẫu được lưu trữ trong một mô hình không gian n chiều.

- Xác định giá trị tham số K (số láng giềng gần nhất).
- Tính khoảng cách giữa đối tượng cần phân lớp (Query Point) với tất cả các đối tượng trong dữ liệu huấn luyện.
- Sắp xếp khoảng cách theo thứ tự tăng dần và xác định K láng giềng gần nhất với Query Point.
- Lấy tất cả các lớp của K láng giềng gần nhất đã xác định.
- Dựa vào phần lớn lớp của láng giềng gần nhất để xác định lớp cho Query Point.

Ví dụ minh họa KNN

Ví dụ cần phân lớp cho z



- Xét 1 láng giềng gần nhất $\rightarrow$ Gán z vào lớp c_2
- Xét 3 láng giềng gần gần nhất $\rightarrow$ Gán z vào lớp c_1
- Xét 5 láng giềng gần gần nhất $\rightarrow$ Gán z vào lớp c_1

Chọn một hay nhiều láng giềng gần nhất?

Việc phân lớp chỉ dựa trên duy nhất một láng giềng gần nhất là không chính xác. Thường xét $k > 1$ các láng giềng gần nhất. Với bài toán phân loại hai lớp, k thường được chọn là một số lẻ để tránh cân bằng tỉ lệ giữa hai lớp

Hàm tính khoảng cách d

Đóng vai trò quan trọng trong phương pháp học dựa trên các láng giềng gần nhất, và d được xác định trước và không thay đổi trong quá trình học.

Các loại hàm tính khoảng cách d

- Hàm khoảng cách hình học: dành cho các bài toán có thuộc tính đầu vào kiểu số thực. Một số hàm tính khoảng cách hình học:

Euclid:

$$d(x, z) = \sqrt{\sum_{i=1}^n (x_i - z_i)^2}$$

Manhattan:

$$d(x, z) = \sum_{i=1}^n |x_i - z_i|$$

Minkowski

$$d(x, z) = \left(\sum_{i=1}^n |x_i - z_i|^p \right)^{1/p}$$

- Hàm Hamming: dành cho các bài toán có các thuộc tính đầu vào kiểu nhị phân.

$$d(x, z) = \sum_{i=1}^n \text{Difference}(x_i, z_i)$$

$$\text{Với } \text{Difference}(a, b) = \begin{cases} 1, & \text{nếu } a \neq b \\ 0, & \text{nếu } a = b \end{cases}$$

- Hàm Cosine: dành cho các bài toán phân lớp văn bản

$$d(x, z) = \frac{x \cdot z}{\|x\| \|z\|} = \frac{\sum_{i=1}^n x_i z_i}{\sum_{i=1}^n x_i^2 \sum_{i=1}^n z_i^2}$$

PHẦN 2: THỬ NGHIỆM TRÊN 5 DATASET NHIỀU LỚP

1/ Dữ liệu thử nghiệm

Phần thực nghiệm được thực hiện trên python với 3 chương trình SVM Pairwise, SVM one against all, và KNN. Mỗi chương trình chạy trên 5 dataset nhiều lớp. Mỗi dataset chia thành hai tập dữ liệu: 90% cho huấn luyện và 10% cho thử nghiệm. Với dữ liệu 90% huấn luyện thực hiện cross validation với 5 fold để tìm mô hình tốt nhất và thử nghiệm mô hình tốt nhất trên tập thử nghiệm 10%. Các dataset được thử nghiệm được lưu trên thư mục Dataset là: **iris, wine, dna, segment, vehicle, letter**. Các dataset có các lớp lần lượt là 1, 2, ..., số lớp. Thông tin về các tập dữ liệu

- **Iris:** 4 dimensions, 150 vectors, 3 class, 50 positive, 100 negative. Các lớp tương ứng là: 1, 2, 3
- **Wine:** 13 dimensions, 178 vectors, 3 class, 59 positive, 119 negative. Các lớp tương ứng là: 1, 2, 3
- **Dna:** 180 dimensions, 2000 vectors, 3 class, 464 positive, 1536 negative. Các lớp tương ứng là: 1, 2, 3
- **Vehicle:** 18 dimensions, 846 vectors, 4 class, 212 positive, 634 negative. Các lớp tương ứng là: 1, 2, 3, 4
- **Segment:** 19 dimensions, 2310 vectors, 7 class, 330 positive, 1980 negative. Các lớp tương ứng là: 1, 2, 3, 4, 5, 6, 7

Huấn luyện dữ liệu để tìm ra mô hình tốt nhất

- **Thuật toán KNN:** được huấn luyện bằng cách với mỗi fold sẽ thay đổi số láng giềng gần nhất là từ 4 đến 10. Tìm mô hình tốt nhất trên fold. Sau đó lặp lại trên 5 fold để tìm mô hình tốt nhất.
- **SVM Pairwise:** được huấn luyện bằng cách: trên mỗi fold lần lượt huấn luyện hai lớp i, j ($i < j$) sử dụng hàm RBF kernel, và thay đổi các tham số Lamda $2^{-5}, 2^{-3}, 2^{-1}, 2^1, 2^3, 2^5$ và tham số C lần lượt là

1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0 để tìm mô hình tốt nhất của hai lớp i, j . Kết hợp các mô hình phân loại hai lớp để phân loại nhiều lớp. Việc chọn lớp để dự đoán trong phân loại nhiều lớp cho một dữ liệu bằng cách chọn lớp được dự đoán có độ chính xác cao nhất

- **SVM One Against All:** được huấn luyện bằng cách: trên mỗi fold lần lượt huấn luyện lớp i với dữ liệu còn lại sử dụng hàm RBF kernel, và thay đổi các tham số Lamda $2^{-5}, 2^{-3}, 2^{-1}, 2^1, 2^3, 2^5$ và tham số C lần lượt là 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0 để tìm mô hình tốt nhất của hai lớp i . Kết hợp các mô hình phân loại hai lớp để phân loại nhiều lớp. Việc chọn lớp để dự đoán trong phân loại nhiều lớp cho một dữ liệu bằng cách chọn lớp được dự đoán nhiều lần nhất cho dữ liệu đó

2/ Kết quả thử nghiệm

Kết quả mỗi chương trình được ghi ra 2 file được lưu trong thư mục result: **file detail** thứ nhất ghi lại cụ thể quá trình huấn luyện, chọn tham số để tìm được mô hình tốt nhất và kết quả độ chính xác test trên 10% dữ liệu, **file final** thứ hai ghi độ chính xác, thời gian huấn luyện của mô hình tốt nhất và kết quả độ chính xác dùng mô hình tốt nhất test trên 10% dữ liệu của 5 dataset

- *Kết quả thử nghiệm với KNN*

Dataset	Time Training	Acc Training (%) Training 90%	Accuracy (%) Test 10%
Iris	0.0	100	100
Wine	0.0	100	94
Dna	0.016	85	84
Vehicle	0.0	75	69
Segment	0.015	97	96

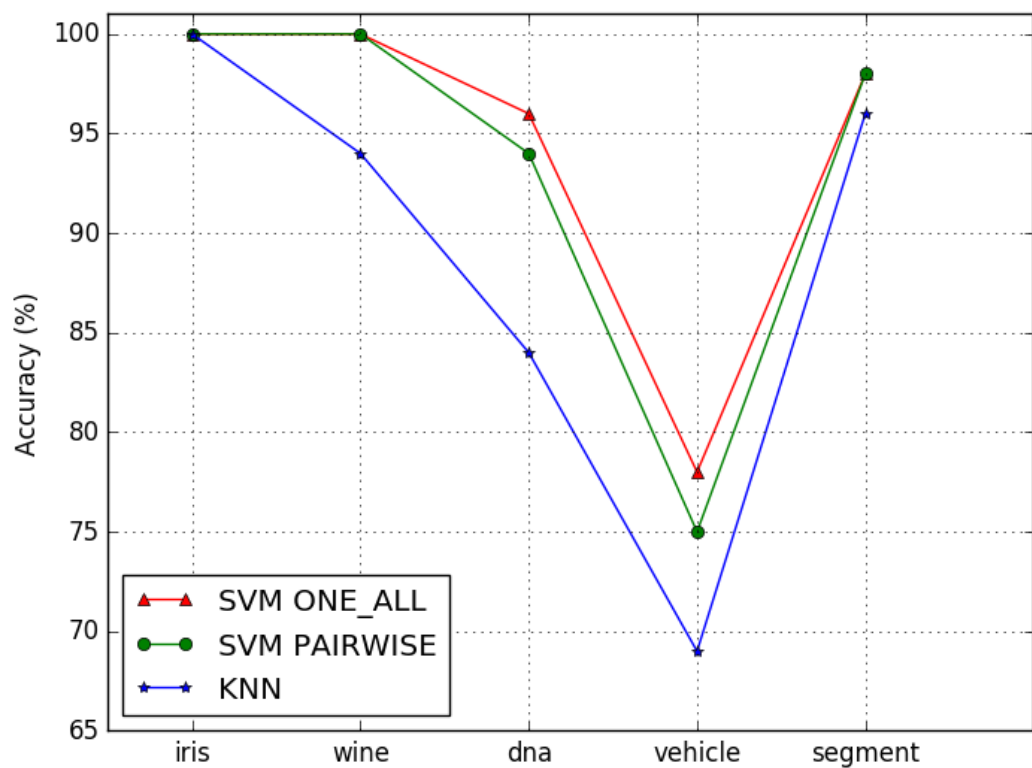
- *Kết quả thử nghiệm với SVM Pairwise*

Dataset	Time Training	Acc Training (%) Training 90%	Accuracy (%) Test 10%
Iris	0.0	100	100
Wine	0.0	100	100
Dna	0.016	85	84
Vehicle	0.0	75	69
Segment	0.015	97	96

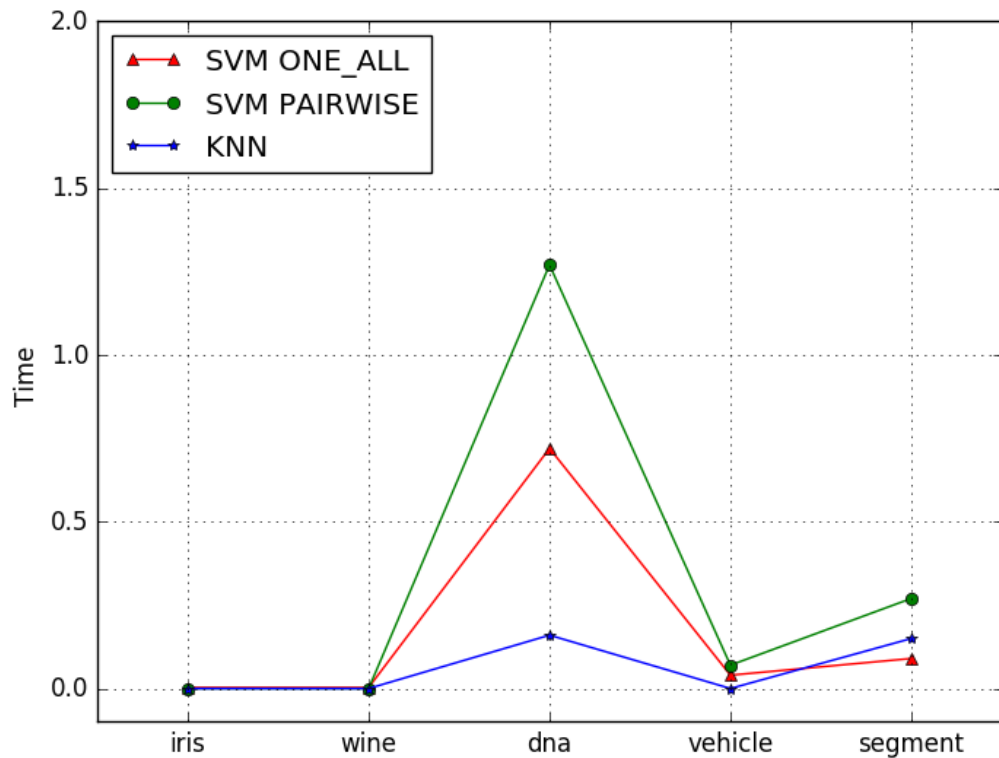
- **Kết quả thử nghiệm với SVM One against all**

Dataset	Acc Training (90%)	Acc Training (%) Training 90%	Accuracy (%) Test 10%
Iris	0.0	100	100
Wine	0.0	100	100
Dna	1.28	96	94
Vehicle	0.65	86	75
Segment	0.27	97	98

Biểu đồ so sánh độ chính xác và thời gian huấn luyện mô hình tốt nhất của các



Biểu đồ so sánh độ chính xác trên các tập dữ liệu



Biểu đồ so sánh thời gian huấn luyện mô hình tốt nhất trên các tập dữ liệu

Nhận xét

- Độ chính xác: Với các tập dữ liệu nhỏ thì KNN, SVM Pairwise và SVM One against All có độ chính xác gần như tương nhau. Nhưng với các tập dữ liệu lớn thì SVM có độ chính xác cao hơn
- Thời gian huấn luyện cho mô hình tốt nhất: Với các tập dữ liệu nhỏ, thời gian huấn luyện của của KNN, SVM Pairwise và SVM One against All tương nhau. Với các tập dữ liệu lớn, KNN có thời gian huấn luyện tốt hơn