Báo cáo thực tập ngày 21/06/2024

Nguyễn Tiến Đạt - Thực tập sinh AI

1 SUPPORT VECTOR MACHINE - SVM

1.1 Khoảng cách từ một điểm đến một siêu phẳng

Trong không gian hai chiều, giả sử ta có một điểm x có toạ độ là (x_0, y_0) và một đường thẳng $D : ax + by + c = 0$. Khi đó, koảng cách từ điểm x đến đường thẳng D là:

$$d(x, D) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

Nếu ta bỏ dấu giá trị tuyệt đối ta có thể xác định được vị trí của điểm x sẽ nằm về phía bên nào của D:

- Tất cả những điểm (x_1, y_1) để $ax_1 + by_1 + c > 0$ sẽ nằm cùng phía, ta gọi là *phía dương*.
- Tất cả những điểm (x_2, y_2) để $ax_2 + by_2 + c < 0$ sẽ nằm cùng phía, ta gọi là *phía âm*.
- Những điểm nằm trên *đường thẳng/mặt phẳng* làm cho tử số có giá trị bằng 0 thì khoảng cách bằng 0.

Mở rộng lên không gian nhiều chiều R^d ta có $x^* = [x_1^*, x_2^*, ..., x_d^*]$ và siêu phẳng H có phương trình là:

$$w_0 + w_1x_1 + w_2x_2 + ... + w_dx_d = 0$$

hay $w_0 + w^T x^* = 0$.

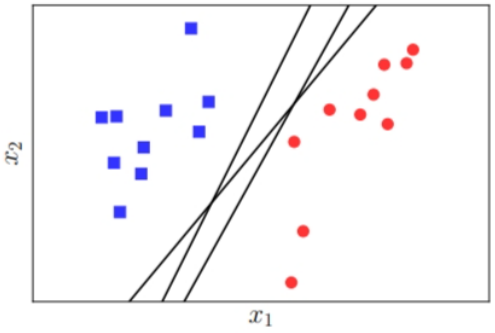
Khi đó khoảng cách từ x^* đến H là:

$$d\left(x^*, H\right) = \frac{|w_0 + w^T x^*|}{||w||_2}$$

Trong đó $d\left(x^*, H\right) = \frac{|w_0 + w^T x^*|}{||w||_2}$ với d là số chiều của không gian.

1.2 Bài toán phân loại nhị phân

Giả sử rằng có hai class khác nhau được mô tả bởi các điểm trong không gian nhiều chiều, hai lớp này tách được tuyến tính, tức tồn tại một siêu phẳng phân chia chính xác hai lớp đó. Bài toán đặt ra là tìm một siêu mặt phẳng phân chia hai lớp đó. Tuy nhiên, có một vấn đề nảy sinh khi tìm siêu phẳng là sẽ có rất nhiều siêu phẳng thỏa mãn điều kiện bài toán. Hình ảnh dưới đây là ví dụ minh họa:

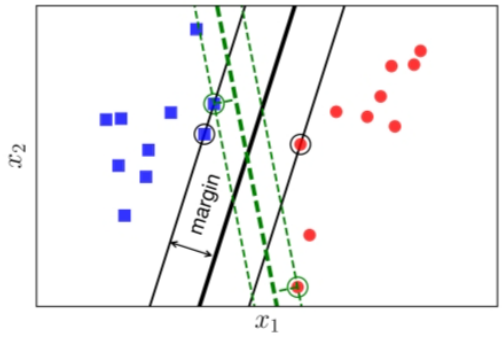


Hình 1: Hình ảnh minh họa có nhiều siêu phẳng phân tách dữ liệu thành hai lớp riêng biệt.

Do đó, bài toán mới sẽ được đặt ra là trong các siêu phẳng phân chia đó, tìm siêu phẳng phân chia tối ưu theo một tiêu chuẩn nào đó. Bởi vậy, ta sẽ định nghĩa hai tiêu chí cho siêu phẳng như sau:

- Sự cân đối - hay phân chia công bằng: Siêu phẳng P tối ưu sẽ phải "cách đều" các tập điểm của mỗi phân lớp, nghĩa là:
$$d(Class1, P) = d(Class2, P)$$
Khoảng cách này được gọi là *margin*-lề.
- Phân tách rõ ràng nhất: Khoảng cách lề là lớn nhất.

Như vậy, với hai tiêu chí trên thì siêu phẳng sẽ mang lại hiệu ứng phân lớp tốt nhất vì sự phân chia giữa hai lớp là rạch ròi nhất. Bởi vậy, bài toán tối ưu trong Support Vector Machine (SVM) chính là bài toán đi tìm đường phân chia sao cho margin là lớn nhất.



Hình 2: Hình ảnh minh họa về siêu phẳng có margin lớn nhất.

1.3 Xây dựng bài toán tối ưu cho SVM

Giả sử ta có tập dữ liệu là $(x_1, y_1), ..., (x_n, y_n); x_i \in R^d$, trong đó y_i là nhãn của x_i, d là số chiều của dữ liệu và N là số điểm dữ liệu. Như đã đề cập trước đó là siêu phẳng sẽ chia không gian thành hai phía là phía âm và phía dương, cho nên, ta định nghĩa những điểm nằm ở phía dương thì thuộc lớp 1 và phía còn lại thuộc lớp -1. Khi đó, khoảng cách từ một điểm đến một siêu phẳng có thể viết như sau:

$$d(x_n, H) = \frac{y_n\left(w^T x_n + w_0\right)}{||w||_2}$$

Do đó, ta định nghĩa *margin* là khoảng cách gần nhất từ 1 điểm tới mặt đó (bất kể điểm nào trong hai lớp):

$$margin = \min_n \frac{y_n\left(w^T x_n + w_0\right)}{||w||_2}$$

Lúc này, bài toán tối ưu của SVM là tìm w và $w0$ sao cho *margin* là lớn nhất:

$$(w, w_0) = arg\max_{w,w_0}\left\{\min_n \frac{y_n\left(w^T x_n + w_0\right)}{||w||_2}\right\}$$

$$= arg\max_{w,w_0}\left\{\frac{1}{||w||_2}\min_{w,w_0} y_n\left(w^T x_n + w_0\right)\right\}$$

Ta thấy rằng nếu thay w và w_0 bằng kw và kw_0 với k là một số dương thì siêu phẳng không bị thay đổi. Do đó ta có thể coi $y_n(w^T x_n + w_0) = 1$. Như vậy với mọi n ta có:

$$y_n\left(w^T x_n + w_0\right) \geqslant 1$$

Như vậy, bài toán tối ưu của SVM có thể đưa về dạng bài toán tối ưu có ràng buộc như sau:

$$(w, w_0) = arg\ \max_{w,w_0} \frac{1}{||w||_2}$$

sao cho $y_n \left(w^T x_n + w_0\right) \geqslant 1, \forall n = 1, 2, ..., N$

Để đơn giản, ta có thể biến đổi công thức về dạng sau:

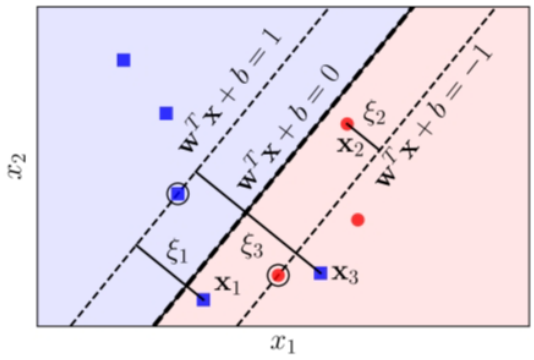
$$(w, w_0) = arg\ \min_{w,w_0} \frac{1}{2}||w||_2^2$$

sao cho $1 - y_n \left(w^T x_n + w_0\right) \leqslant 0, \forall n = 1, 2, ..., N$

Ta có thể thấy hàm mục tiêu là một hàm lồi và ràng buộc là các hàm tuyến tính nên chúng cũng là hàm lồi. Do đó, bài toán của SVM là một bài toán lồi, hơn nữa đây là một bài toán quy hoạch toàn phương. Bên cạnh đó, hàm mục tiêu là một hàm lồi chặt vì $||w||_2^2 = w^T I w$ với I là ma trận đơn vị - là một ma trận xác định dương, nên nghiệm SVM là duy nhất.

1.4 Soft Margin SVM

Ý tưởng của *Soft Margin SVM* là hy sinh một vài điểm dữ liệu nằm sai phía so với các đường support (các điểm nằm trong vùng không an toàn) để đạt được hiệu quả phân chia tốt nhất. Bởi vậy, hàm mục tiêu nên là một sự kết hợp để tối đa margin và tối thiểu sự hy sinh.



Hình 3: Hình minh họa với biến ζ

Với mỗi điểm x_n trong tập toàn bộ dữ liệu huấn luyện, ta định nghĩa thêm một biến đo sự hy sinh ζ_n tương ứng. Biến này còn được gọi là *slack variable*. Với những điểm x_n nằm trong vùng an toàn thì $\zeta_n = 0$, ngược lại $\zeta_n \geqslant 0$. Với $y_i = \pm 1$ là nhãn của x_i thì $\zeta_i = |w^T + w_0 - y_i|$. Kết hợp với bài toán tối ưu của *Hard Margin SVM*, ta có hàm tối ưu sau:

$$\frac{1}{2}||w||_2^2 + \mathbf{C}\sum_{n=1}^{N} \zeta_n$$

trong đó C là một hằng số dương được dùng để điều chỉnh tầm quan trọng giữa **margin** và sự hy sinh.

Ta sẽ thay đổi điều kiện của SVM đơn thuần là $y_n(w^T x_n + w_0) \geqslant 1$ về dạng như sau:

$$y_n \left(w^T x_n + w_0\right) \geqslant 1 - \zeta_n \Leftrightarrow 1 - \zeta_n - y_n \left(w^T x_n + w_0\right) \leqslant 0;$$

với $\forall n = 1, 2, ..., N; \zeta_n \geqslant 0$

Cuối cùng ta có bài toán tối ưu cho *Soft Margin SVM*:

$$(w, w_0, \zeta) = arg\ \min_{w,w_0,\zeta} \frac{1}{2}||w||_2^2 + C\sum_{i=1}^{N} \zeta_n$$

sao cho:

$$1 - \zeta_n - y_n \left(w^T x_n + w_0\right) \leqslant 0; \forall n = 1, 2, ..., N; -\zeta_n \leqslant 0$$

Tương tự như *Hard Margin SVM*, ta có thể đưa bài toán này về dạng đối ngẫu Lagrange.Tuy nhiên, khác với SVM đơn thuần thì ta có thể đưa bài toán về dạng tối ưu không ràng buộc để giải theo phương pháp *Gradient Descent*.

1.5 Kernel SVM

Ta biết rằng có rất nhiều dạng dữ liệu không tách được tuyến tính cho nên ta khó có thể tìm được siêu phẳng đủ tốt để phân tách dữ liệu. Tuy nhiên, khi ta ánh xạ dữ liệu lên một không gian nhiều chiều hơn thì có một số trường hợp dữ liệu phân tác tuyến tính hoặc gần tuyến tình. Khi đó, ta hoàn toàn có thể tìm được một siêu phẳng tốt hơn so với siêu phẳng ở không gian cũ.

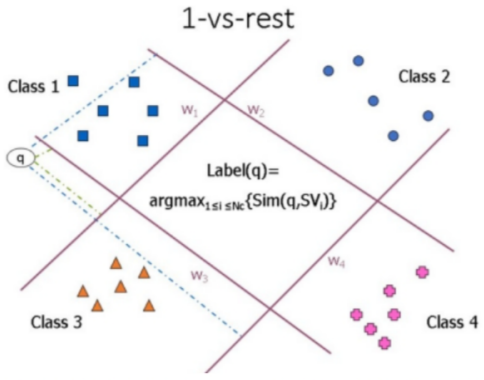
Như vậy, kernel SVM là một hàm ánh xạ dữ liệu ban đầu sang một không gian mới. Sau đây là một số kernel SVM phổ biến:

Tên	Công thức	kernel	Thiết lập hệ số
linear	$x^T z$	$'linear'$	không có hệ số
polynomial	$\left(r + \gamma x^T z\right)^d$	$'poly'$	d: degree; γ: gamma
sigmoid	$tanh\left(\gamma x^T z + r\right)$	$'sigmoid'$	r: coef
rbf	$exp\left(-\gamma\|\|x - z\|\|_2^2\right)$	$'rbf'$	$\gamma > 0$

Bảng 1: Bảng các kernel và công thức

1.6 Multi-class SVM

Đối với bài toán có nhiều hơn hai lớp ($C > 2$) thì chúng ta sẽ áp dụng phương pháp là One-vs-rest. Cụ thể, nếu có C classes thì ta sẽ xây dựng C mô hình SVM, mỗi mô hình SVM tương ứng với một lớp. Mô hình SVM thứ nhất giúp phân biệt lớp 1 với lớp không phải lớp 1, tức xem một điểm có thuộc lớp 1 hay không, hoặc xác suất để một điểm rơi vào lớp 1 là bao nhiêu. Tương tự như thế, mô hình SVM thứ hai sẽ phân biệt lớp 2 với không phải lớp 2,...Kết quả cuối cùng có thể được xác định bằng cách xác định lớp mà một điểm rơi vào với xác suất cao nhất. Khi ta dự đoán lớp của một điểm, nếu như điểm đấy nằm về phía dương của nhiều siêu phẳng SVM thì ta sẽ lấy nhãn của tập dữ liệu mà có khoảng cách từ điểm đó đến siêu phẳng là lớn nhất để gán nhãn cho điểm được dự đoán.



Hình 4: Hình minh họa One-vs-rest

1.7 Thực nghiệm và đánh giá

Để sử dụng được mô hình SVM cho bài toán nhận dạng chữ số viết tay, báo cáo sẽ chuyển dữ liệu ban đầu là một ma trận có kích thước 28×28 thành một vector có số chiều là 784.Tiếp đó, báo cáo sẽ sử cả hai mô hình SVM là Soft Margin SVM và Hard Margin SVM cùng với 4 kernel đã được đề cập ở trên.

Về phần thiết lập SVM, báo cáo sử dụng thư viện Sklearn để cài đặt mô hình SVM với $C = 10$ để sử dụng Softmax Margin SVM và $C = 1e5$ để sử dụng Hard Margin SVM.

Loại mô hình	Accuracy	Precision	Recall
Hard Margin SVM + linear kernel	72.8%	73.1%	72.92%
Soft Margin SVM + linear kernel	72.8%	73.1%	72.92%
Hard Margin SVM + rbf kernel	**97.92%**	97.91%	97.89%
Soft Margin SVM + rbf kernel	97.89%	97.89 %	97.86%
Hard Margin SVM + poly kernel	96.77%	96.78%	96.73%
Soft Margin SVM + poly kernel	96.95%	96.94%	96.92%
Hard Margin SVM + sigmoid kernel	56.02%	62.33%	55.59%
Soft Margin SVM + sigmoid kernel	58.58%	65.29%	58.31%

Bảng 2: Bảng thống kê kết quả của 8 mô hình SVM

Có thể thấy hầu như mô hình Hard Margin SVM đều cho kết quả tốt hơn một chút so với Soft Margin SVM. Tuy nhiên, Sigmoid kernel lại cho thấy điều ngược lại khi độ chính xác của Soft Margin SVM cao hơn Hard Margin SVM.

Mô hình Hard Margin SVM với RBF kernel cho kết quả tốt nhất với cả 3 độ đo so với 7 mô hình còn lại.

Do đó, báo cáo sẽ chỉ đưa ra ma trận nhầm lẫn của mô hình tốt nhất. Dưới đây là ma trận nhầm lẫn của Hard Margin SVM + RBF kernel:

$$\begin{bmatrix} 975 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & \\ 0 & 1131 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & \\ 5 & 1 & 1014 & 0 & 2 & 0 & 1 & 4 & 4 & 1 \\ 0 & 0 & 4 & 989 & 0 & 6 & 0 & 3 & 6 & 2 \\ 1 & 0 & 3 & 0 & 965 & 0 & 2 & 1 & 1 & 9 \\ 2 & 0 & 0 & 16 & 2 & 864 & 3 & 0 & 4 & 1 \\ 5 & 2 & 0 & 0 & 2 & 4 & 944 & 0 & 1 & 0 \\ 0 & 5 & 8 & 2 & 4 & 0 & 0 & 997 & 0 & 12 \\ 3 & 0 & 3 & 12 & 1 & 3 & 2 & 2 & 944 & 4 \\ 3 & 4 & 0 & 8 & 13 & 2 & 0 & 3 & 7 & 969 \end{bmatrix}$$