1. Thuật toán

**SVM là gì?**

SVM (Support Vector Machine) là một thuật toán học máy có giám sát được sử dụng rất phổ biến ngày nay trong các bài toán phân lớp (classification) hay hồi qui (Regression).

SVM được đề xuất bởi Vladimir N. Vapnik và các đồng nhiệp của ông vào năm 1963 tại Nga và sau đó trở nên phổ biến trong những năm 90 nhờ ứng dụng giải quyết các bài toán phi tuyến tính (nonlinear) bằng phương pháp Kernel Trick.

**SVM hay SVMs?**

Khi đọc các tài liệu về SVM các bạn thường thấy SVM và SVMs đều được nhắc đến vậy chúng khác nhau thế nào. Thực chất SVM và SVMs là một. Người ta dùng SVMs là vì muốn nói đến hai loại của thuật toán của SVM:

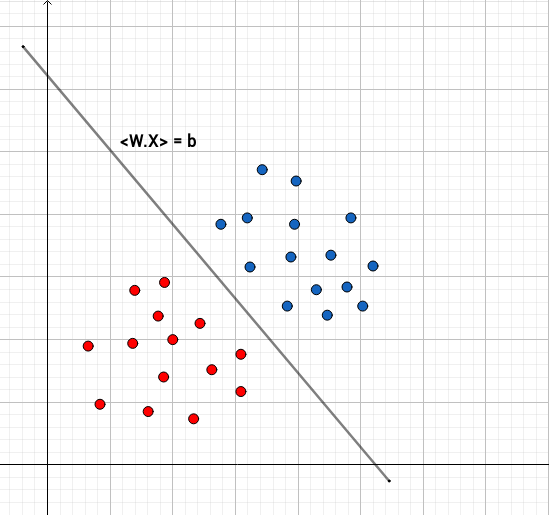
* SVM: dùng cho các bài toán phân lớp
* SVR (Support Vector Regression): dùng cho các bài toán hồi quy

Theo kinh nghiệm của mình thấy thì việc ứng dụng SVM để giải quyết các bài toán thực tế thường cho kết quả cao so vớcriterions  
i các thuật toán khác đặc biệt là các bài toán phân loại liên quan đến xử lý văn bản. Có lẽ chính vì vậy mà SVM có một nền tảng toán học và lý thuyết khá phức tạp. Trong bài này mình chỉ giới thiệu một cách tổng quan về cách thức hoạt động của SVM còn về các khía cạch toán học khác mình sẽ giải thích kỹ hơn ở một bài khác.

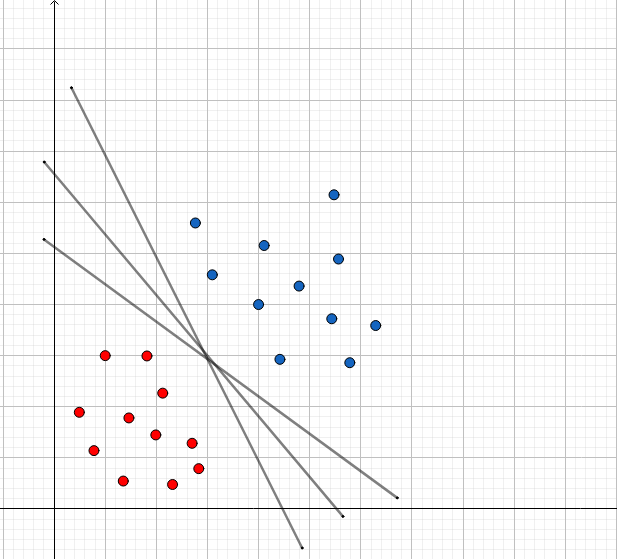
Để hiểu được một cách đầy đủ về SVM trong các bài toán thực tế chúng ta cần nắm được các bài toán con của SVM: linear, hard-margin, soft-margin, non-linear, binary-class và multi-class. Mình sẽ giải thích rõ từng vấn đề này.

**SVM làm việc như thế nào?**

Ý tưởng của SVM là tìm một siêu phẳng (hyper lane) để phân tách các điểm dữ liệu. Siêu phẳng này sẽ chia không gian thành các miền khác nhau và mỗi miền sẽ chứa một loại giữ liệu.



Siêu phẳng được biểu diễn bằng hàm số <W.X> = b ( W và X là các vector <W.X> là tích vô  ) Hay W^T=b ( W^T  là ma trận chuyễn vị)

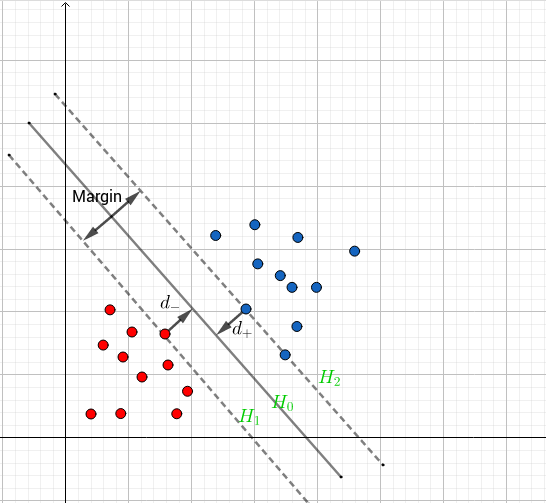


Vấn đề là có rất nhiều siêu phẳng, chúng ta phải chon cái nào để tối ưu nhất ?

**Cách chọn siêu phẳng tối ưu:**

Giả sử chúng ta phải phân loại tập dữ liệu các lớp dương  (màu xanh) nhãn là 1 và các dữ liệu lớp âm (màu đỏ) nhãn là -1 (tập dữ liệu có thể phân tách tuyến tính).

Siêu phẳng phân tách hai lớp giữ liệu H_{0}  thỏa mã <W.X> + b =0. Siêu phẳng này tạo ra hai nữa không gian (half space) dữ liệu:  
Không gian các dữ liệu lớp âm X_{i}  thỏa mãn <W.X_{i}> + b \leq -1  và không gian dữ liệu lớp dương X_{j}  thỏa mãn <W.X_{j}> + b \geq 1 



Tiếp theo ta chọn hai siêu phẳng lề H_{1}  đi qua điểm thuộc lớp âm và H_{2}  đi qua điểm thuộc lớp dương đều song song với H_{0} 

* H_{1} :  <W.X> + b =-1
* H_{2} :  <W.X> + b =1

Khoảng cách từ H_{1}  đến H_{0}  là d_{-}   
Khoảng cách từ H_{2}  đến H_{0}  là d_{+}   
m = d_{-}  + d_{+}  được gọi là mức lề

Siêu phẳng tối ưu mà chúng ta cần chọn là siêu phẳng phân tách có lề lớn nhất. Lý thuyết học máy đã chỉ ra rằng một siêu phẳng như vậy sẽ cực tiểu hóa giới hạn lỗi mắc phải.

**Tính m (margin) như thế nào ?**

Khoảng cách từ một điểm X_{k}  đến siêu phẳng H_{0}  là: \frac{|<W.X_{k}>+b|}{||W||} Trong đó ||W|| là độ dài của vector W:  ||W||= <W.W> = \sqrt{w_{1}^{2}+w_{2}^{2}+...w_{n}^{2}} 

Khoảng cách từ một điểm X_{i}  nằm trên H_{1} đến H_{0}  d_{-} = \frac{|<W.X_{i}>+b|}{||W||}  = \frac{1}{||W||}  
Khoảng cách từ một điểm X_{j}  nằm trên H_{2} đến H_{0}  d_{+} = \frac{|<W.X_{j}>+b|}{||W||}  = \frac{1}{||W||}  
Từ đó ta có thể tính được mức lề m=d_{-} + d_{+} = \frac{2}{||W||}

Giờ thì bạn đã hiểu vì sao các điểm nằm trên hai siêu phẳng H_{1}  và H_{2}  được gọi là các Support Vector.

Vậy việc training trong giải thuật SVM tương được với bài toán cực tiểu hóa có ràng buộc sau đây :

Cực tiểu hóa : \frac{<W.W>}{2} 

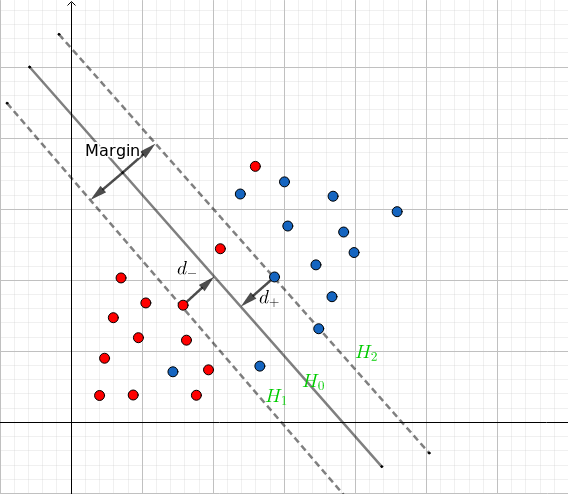
Với điều kiện:  
Nhân hai vế bất đẳng thức của (1) và (2) với y_{i} ta được điều kiện thu gọn y_{i}.(<W.X_{i}>) \geq 1 \forall i=1..n 

Bài toán tương đương với  
***Minimize****: ||W||*  
***Subject to****: y_{i}.(<W.X_{i}>) \geq 1 \forall i=1..n  (3)*

Với điều kiện này, đây chính là bài toán***hard-margin*** của SVM

Việc giải bài toán trên liên quan đến một số lý thuyết toán học tương đối phức tạp như điều kiện Karush-Kuhn-Tucker, hàm đối ngẫu Lagrange, Convex optimization … Mình sẽ không nói ở bài này.

Việc xác định siêu phẳng H_{0}  được giả sử trong điều kiện lý tưởng tập dữ liệu có thể phân tách tuyến tính, tìm được hai siêu phẳng lề H_{1}  và H_{2}  mà không có điểm dữ liệu nào nằm giữa chúng. Vậy trong trường hợp tập dữ liệu có nhiều điểm gây nhiễu, các điểm này không thỏa mãn điều kiện (3), bài toán không tìm được lời giải.

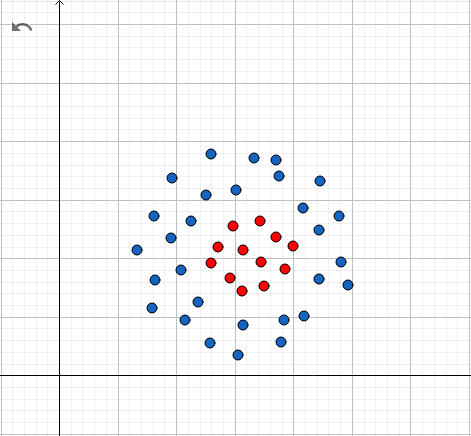


Đối với các trường hợp này chúng ta cần nới lỏng các điều kiện lề bằng việc sử dụng các biến **slack** \varepsilon_{i} \geq 0

<W.X_{i}> + b \leq -1 + \varepsilon_{i},\:if\:y_{i}=-1   
<W.X_{i}> + b \geq 1 - \varepsilon_{i},\:if\:y_{i}=-1 

Bài toán trong trường hợp này trở thành:  
***Minimize****: ||W|| +C\sum_{i=1}^{n} \varepsilon_{i} *  
***Subject to****:*

Trong đó C là tham số xác định mức độ phạt của (**penalty degree**) đối với các lỗi (đây là mà một tham số khá quan trọng mà các bạn cần phải tối ưu trong quá trình huấn luyện SVM)  
Đây chính là bài toán **Soft-margin** của SVM.

Một vấn đề cuối cùng mà mình muốn nhắc đến trong bài này là đối với các bài toán có không gian dữ liệu là phi tuyến tính (non-linear) chúng ta không thể tìm được một siêu phẳng H_{0}  thỏa mãn bài toán. 

Để 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 \mapsto F   
x \mapsto \phi(x) 

Trong không gian F tập dữ liệu có thể phân tách tuyến tính. Nhưng nãy sinh một vẫn 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.  Rất may trong bài toán SVM người ta đã tìm ra một cách không cần phải tính \phi(x) , \phi(z)  và hàm ánh xạ \phi mà vẫn tính được <\phi(x).\phi(z)> .  Phương pháp này gọi là Kernel Trick

K(x,z)=<\phi(x).\phi(z)>  K(x,z) là một hàm nhân (Kernel functions)

Một số hàm nhân thường dùng:

* Polynomial: K(x,z)=(<x.z>+\theta )^{d}
* Gaussian RBF: K(x,z)=e^{\frac{||x-z||^{2}}{2\sigma}}; \sigma > 0
* Sigmoidal: tanh(\beta <x.z>-\lambda )=\frac{1}{1+e^{-(\beta <x.z>-\lambda)}}; \beta ,\lambda \in R

1. Thuật toán SVM phân loại hoa

**Tập dữ liệu Iris Flowers**

Tập dữ liệu này còn có tên gọi khác là **Fisher’s Iris**vì nó do [Ronald Fisher](https://en.wikipedia.org/wiki/Ronald_Fisher) thu thập và tổng hợp. Tập dữ liệu này gồm 50 mẫu về 3 loài hoa khác nhau của họ Iris là **(Iris setosa, Iris virginica và Iris versicolor)**. Cho cái ảnh cho các bạn dễ hình dung



Với mỗi một mẫu hoa này [hắn](https://en.wikipedia.org/wiki/Ronald_Fisher) thu thập bốn thuộc tính là **chiều dài** và **chiều rộng** của **đài hoa** và **cánh hoa** với đơn vị centimet. Để có thể sử dụng tập dữ liệu này chúng ta sẽ sử dụng thư viện ***datasets*** trong ***sklearn***.



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | from sklearn import datasets    # import iris flowers dataset  iris = datasets.load\_iris() |

Sau khi đã có dữ liệu rồi, chúng ta hoàn toàn có thể chạy thuật toán **SVM** ngay để tiến hành phân loại. Tuy nhiên, mình nghĩ nên giúp các bạn có một cái nhìn trực quan hơn về tập dữ liệu này. Mà để hiển thị trực quan nhất thì không gì bằng **biểu đồ** với thư viện ***matplotlib*** thần thánh. Chúng ta cùng thử biểu diễn trên đồ thị xem tập dữ liệu mà chúng ta nhét vào máy tính nó là cái gì nhé.

**Biểu diễn tập dữ liệu bằng đồ thị 2D**

Chúng ta tưởng tượng tập dữ liệu của ta là một tập hợp của **150 điểm dữ liệu** tương ứng với **150 bông hoa.**. Chúng ta sẽ lấy chúng ta từ datasets bằng hàm sau:



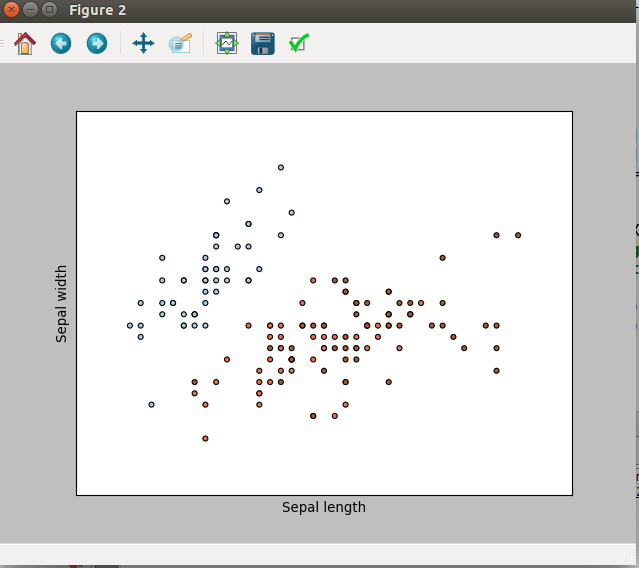
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | def getData():      # Get iris data from datasets      iris = datasets.load\_iris()        return iris |

Lúc này đã có một tập hợp các điểm dữ liệu. Mỗi điểm dữ liệu bao gồm 4 thuộc tính như đã nói ở trên. Tuy nhiên để biểu diễn trong đồ thị hai chiều chúng ta cần phải giảm bớt số thuộc tính biểu diễn. Ở đây giả sử chọn hai thuộc tính đầu tiên là độ rộng và chiều cao của đài hoa. Chúng ta có hàm xử lý vẽ đồ thị 2D như sau:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | def get2DPlot(iris):      X = iris.data[:, :2]  # Lấy hai thuộc tính đầu tiên      Y = iris.target      X\_min, X\_max = X[:, 0].min() - .5, X[:, 0].max() + .5      Y\_min, Y\_max = X[:, 1].min() - .5, X[:, 1].max() + .5      plt.figure(2, figsize=(8, 6))      plt.clf()        # Biểu diễn tập dữ liệu huấn luyện bằng đồ      plt.scatter(X[:, 0], X[:, 1], c=Y, cmap=plt.cm.Paired)      plt.xlabel('Sepal length')      plt.ylabel('Sepal width')        plt.xlim(X\_min, X\_max)      plt.ylim(Y\_min, Y\_max)      plt.xticks(())      plt.yticks(())      plt.show() |

Và đây là thành quả



Chúng ta có thể thấy được các điểm dữ liệu với hai thuộc tính trên đồ thị hai chiều. Các điểm này được phân biệt bằng **mắt thường** với 3 màu khác nhau. Tuy nhiên muốn máy tính phân biệt được như chúng ta lại là một câu chuyện hoàn toàn khác và bạn sẵn sàng cùng tôi đi đến cuối cùng của câu chuyện này chứ. OK chúng ta tiếp tục thôi

Phân lớp sử dụng SVM với các Kernel khác nhau

Với tập dữ liệu **Iris** chúng ta cần phân loại các bông hoa thành 3 lớp dữ liệu. Sử dụng SVM với các phương pháp khác nhau sẽ cho hiệu quả phân lớp khác nhau. Cũng tương tự như trên, chúng ta chỉ xem xét đến 2 thuộc tính đầu tiên của tập dữ liệu, tức là phân lớp trong không gian 2 chiều. Chúng ta sử dụng các Kernel khác nhau bao gồm:

* **SVC with linear kernel**
* **LinearSVC (linear kernel**
* **SVC with RBF kernel**
* **SVC with polynomial (degree 3) kernel**

Đầu tiên chúng ta cần huấn luyện dữ liệu



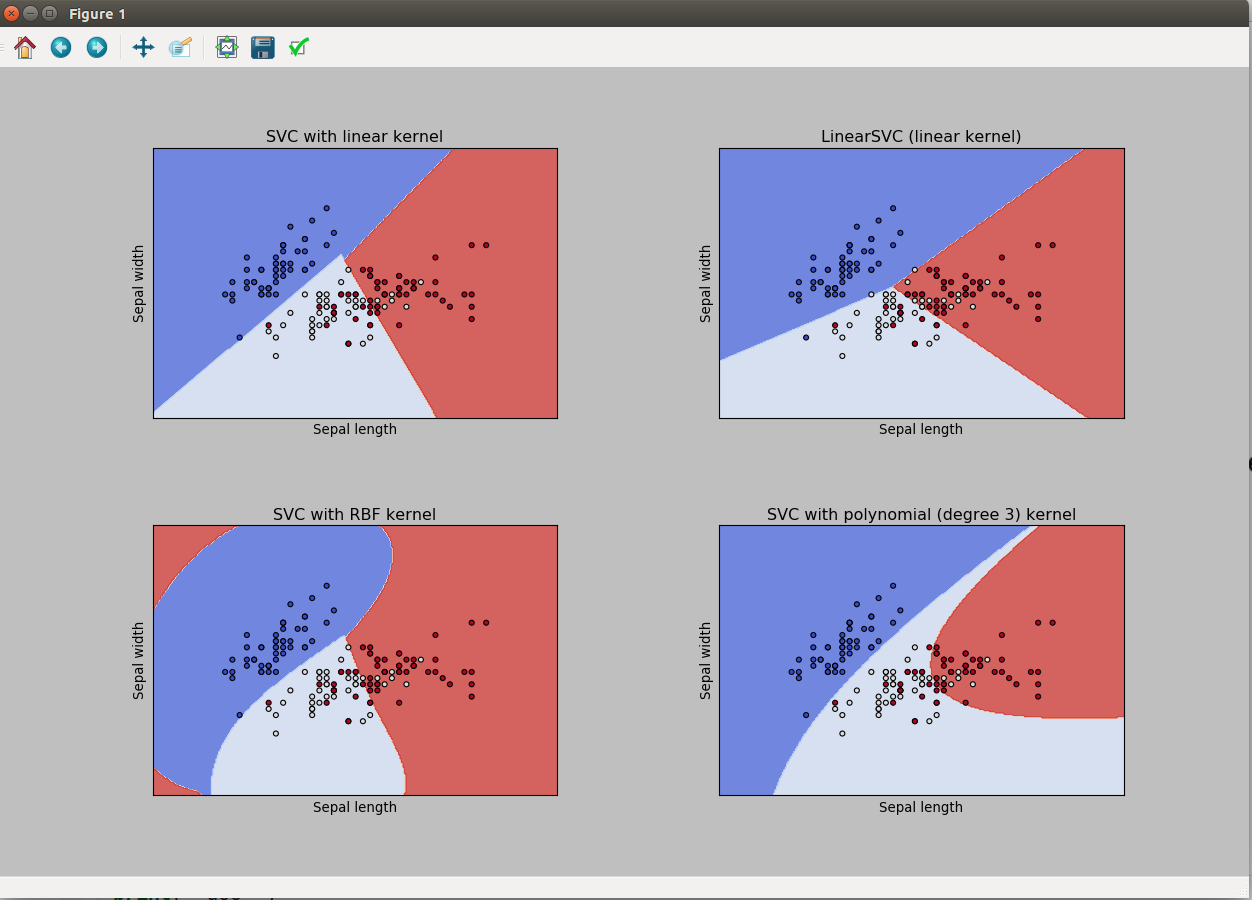
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | X = iris.data[:, :2]  # lấy hai thuộc tính đầu tiên của tập dữ liệu để tiện quan sát bằng đồ thị hai chiều  y = iris.target    h = .02  # hiệu chỉnh độ mỏng của lưới tọa độ trên đồ thị. Càng nhỏ thì càng sắc nét    # we create an instance of SVM and fit out data. We do not scale our  # data since we want to plot the support vectors  C = 1.0  # SVM regularization parameter  svc = svm.SVC(kernel='linear', C=C).fit(X, y)  rbf\_svc = svm.SVC(kernel='rbf', gamma=0.7, C=C).fit(X, y)  poly\_svc = svm.SVC(kernel='poly', degree=3, C=C).fit(X, y)  lin\_svc = svm.LinearSVC(C=C).fit(X, y) |

Tiếp theo là việc đẩy các mô hình thu được vào đồ thị



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | # title for the plots  titles = ['SVC with linear kernel',  'LinearSVC (linear kernel)',  'SVC with RBF kernel',  'SVC with polynomial (degree 3) kernel']    for i, clf in enumerate((svc, lin\_svc, rbf\_svc, poly\_svc)):  # Plot the decision boundary. For that, we will assign a color to each  # point in the mesh [x\_min, x\_max]x[y\_min, y\_max].  plt.subplot(2, 2, i + 1)  plt.subplots\_adjust(wspace=0.4, hspace=0.4)    Z = clf.predict(np.c\_[xx.ravel(), yy.ravel()])    # Put the result into a color plot  Z = Z.reshape(xx.shape)  plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)    # Plot also the training points  plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm)  plt.xlabel('Sepal length')  plt.ylabel('Sepal width')  plt.xlim(xx.min(), xx.max())  plt.ylim(yy.min(), yy.max())  plt.xticks(())  plt.yticks(())  plt.title(titles[i])    plt.show() |

Bây giờ sau khi vẽ các đồ thị trên là chúng ta có thể so sánh hiệu quả phân lớp của các Kernel khác nhau rồi đấy. Kết quả như sau:



Bằng việc phân lớp chúng ta có thể chia mặt phẳng tọa độ thành các phần khác nhau. Khi có một điểm dữ liệu mới chúng ta có thể dựa vào tọa độ của chúng để **phán đoán** xem nó thuộc lớp nào.

Source demo: https://github.com/thandongtb/machine-learning-repo/blob/develop/SVM/svm\_basic.py

1. Áp dụng thuật toán

Đầu vào của mô hình là dữ liệu chuỗi thời gian quá khứ bao gồm giá đóng cửa và các chỉ số kỹ thuật trong chứng khoán. Để đảm bảo được số chiều vừa phải và tránh việc làm nhiễu dữ liệu đầu vào, chúng tôi chọn các chỉ số kĩ thuật thông dụng nhất bao gồm: trung bình trượt giản đơn (SMA) 25 ngày và 65 ngày, Aroon up, Aroon down, dải biên độ biến động giá (Bollinger bands) với Bollinger upper, middle và lower, trung bình trượt hội tụ và phân kỳ (MACD), và MACD Histogram. Giá đóng cửa cùng những chỉ số kỹ thuật tạo nên véc tơ đầu vào với số chiều là 10. Giá trị trong mỗi chiều được chuyển về đoạn [-1, 1].

Mô hình dự đoán xu hướng sẽ kết xuất ra ba giá trị những lớp ứng với xu hướng tăng, giảm và không có xu hướng. Để quyết định xu hướng của ngày hiện tại, chúng tôi dùng một định nghĩa được mô tả chặt chẽ như sau:

Thị trường được xác định có xu hướng tăng (giảm) trong ngày hiện tại khi tất cả những điều kiện sau được thỏa mãn:

Giá đóng cửa phải cao hơn (thấp hơn) chỉ số trung bình trượt 25 ngày.  
Chỉ số trung bình trượt 25 ngày phải cao hơn (thấp hơn) chỉ số trung bình trượt 65 ngày.  
Đường trung bình trượt 25 ngày phải tăng (giảm) ít nhất trong 5 ngày.  
Đường trung bình trượt 65 ngày phải tăng (giảm) ít nhất 1 ngày.  
Nếu không thể thỏa mãn tất cả điều kiện trên để được đánh nhãn lớp có xu hướng tăng và giảm thì ngày hiện tại được đánh nhãn không có xu hướng.

Bô phân lớp SVM:

Cho tập véctơ đầu vào x\_i∈R^D,∀i∈[1,N] và tập các giá trị nhãn lớp tuơng ứng y\_i∈{-1;+1} cho mô hình phân lớp nhịp hân. Hàm tuyến tính phân biệt hai lớp như sau:  
g(x)= w^T.Φ(x)+b (1)  
trong đó, w là véctơ chuẩn của siêu phẳng phân cách, b là độ lệch, và Φ(x) là hàm ánh xạ từ không gian đầu vào sang không gian đặc trưng, Φ(x):R^D→R^M (M > D).

Mục tiêu của SVM là tìm một siêu phẳng tối ưu sao cho khoảng cách lề (margin) giữa hai lớp đạt giá trị cực đại. Bên cạnh đó, để đảm bảo tính tổng quát hóa cao, slack variable đựoc đưa vào để nới lỏng điều kiện phân lớp. Bài toán đưa đến giải quyết tối ưu có ràng buộc:  
min┬(w,b,ξ)⁡〖1/2 w^T w+C∑\_(i=1)^N▒ξ\_i 〗 (2)  
sao cho: y\_i (w^T.Φ(x\_i )+b)≥1-ξ\_i  
ξ\_i≥0,∀i∈[1,N]  
trong đó, C > 0 là tham số chuẩn tắc (regularization parameter), ξ\_i là biến lỏng.

max┬α⁡〖L(α)≡∑\_(i=1)^N▒αi -1/2 ∑(i,j)▒〖α\_i α\_j y\_i y\_j 〖Φ(x\_i )〗^T.Φ(x\_j)〗〗 (3)  
Thỏa mãn: 0≤αi≤C,∀i∈[1,N] và ∑(i=1)^N▒〖α\_i y\_i 〗=0, với αi là các nhân tử Lagrange.  
Sau khi có đựoc các giá trị α\_i , ta sẽ thu đươc các giá trị tối ưu w\* và b\* của siêu phẳng.

Chỉ có các mẫu có αi≥0 mới tham gia vào các véc tơ hỗ trợ (support vector). Cuối cùng, hàm quyết định phân lớp có dạng:f(x)=sgn(∑(i=1)^N▒〖α\_i y\_i 〗 〖Φ(x\_i )〗^T.Φ(x)+b) (4)  
Gọi K(x\_i,x\_j )=〖Φ(x\_i )〗^T.Φ(x\_j ) là hàm nhân của không gian đầu vào. Theo đó, tích vô huớng trong không gian đặc trưng tuơng đương với hàm nhân K ở không gian đầu vào. Như vậy, thay vì tính trực tiếp giá trị tích vô huớng, ta thực hiện gián tiếp thông qua K.

Với thừa nhận dữ liệu chứng khoán biến đổi một cách phi tuyến, ta chọn hàm nhân cho mô hình là hàm phi tuyến Gauss (RBF-Radial Basis Function):  
K(x\_i,x\_j )=exp⁡(-γ‖x\_i-x\_j ‖^2) (5)  
Ước lượng xác suất:  
Với việc áp dụng bộ phân lớp SVM, để phân k lớp, ta áp dụng cách tiếp cận một đối một.

Vì vậy, sẽ có k(k-1)/2 bộ phân lớp được xây dựng để phân biệt mỗi cặp lớp. Trong SVM, để dự đoán được nhãn lớp, ta áp dụng chiến lược voting strategy.

Nghĩa là, ta sẽ xây dựng một luật để phân biệt từng cặp lớp rồi chọn lớp thuộc về cặp chiến thắng nhờ hàm quyết định. Tuy nhiên, trong [20], Wu (2004) đã đề xuất mô hình ước lượng xác suất cho việc phân k lớp và chứng minh cả trên mặt lý thuyết và thực nghiệm, mô hình đề xuất tốt hơn chiến lược bỏ phiếu.  
Cho véc tơ đầu vào x và nhãn lớp y, mục tiêu là ước lượng  
p\_i=P(y=i│x), i=1,…,k. (6)  
Theo cách tiếp cận một, ta thừa nhận rằng đã tồn tại ước lượng xác suất cặp lớp (pairwise class probability) rij của µij:  
r\_ij≈μij≡P(y=i|y=i hoặc j,x) (7)Từ lớp thứ i và thứ j của tập huấn luyện, ta sẽ tính được rij là xấp xỉ của µij thông qua x. Platt (2000) đã đưa ra công thức xấp xỉ rij bằng hàm sigmoid:r\_ij (f)=1/(1+exp⁡(Af+B)) , (8)trong đó, f là hàm quyết định của x, A và B được xấp xỉ bằng cách cực tiểu hóa hàm log-likelihood (với N+ của giá trị yk dương, N- của yk âm):min┬(A,B)⁡〖F(A,B)=-∑(n=1)^N▒[t\_n log⁡(p\_n )+(1-t\_n)log⁡(1-p\_n ) ] 〗 (9)  
trong đó, p\_n=r\_ij (f\_n≡f(x\_n ) ),  
và t\_n={█((N\_++1)/(N\_++2) nếu y\_n=+1@1/(N\_-+2) nếu y\_n=-1)┤, n = 1, …, N.  
Ta thực hiện phương pháp đánh giá chéo (cross-validation) với bộ dữ liệu được chia 5 phần để có được giá trị hàm quyết định f, trước khi giải quyết (9). Sau khi có tất cả giá trị rij, ta áp dụng phương pháp thứ hai của Wu (2004) để tính các giá trị pi trong (6):  
min┬p⁡〖1/2 ∑(i=1)^k▒∑(j: j≠i)▒(r\_ji p\_i-r\_ij p\_j )^2 〗 (10)  
với ràng buộc p\_i≥0,∀i,∑(i=1)^k▒p\_i =1.Bài toán (10) có được nhờ vào đẳng thứcP(y=j│y=i hoặc j,x)∙P(y=i|x)=P(y=i│y=i hoặc j,x)∙P(y=j|x) (11)Ta có thể viết lại (10) theo dạng:min┬p⁡〖1/2 p^T Qp〗 , (12)với Q\_ij={█(∑(s: s≠i)▒〖r\_si〗^2 nếu i=j,[@-r\_ji](https://viblo.asia/u/-r_ji) r\_(ij ) nếu i≠j.)┤

Thuật toán giải quyết được mô tả như sau:  
Bước 1. Khởi gán p thỏa p\_i≥0,∀i,∑(i=1)^k▒p\_i =1.Bước 2. Lặp (t = 1, …, k, 1, …)p\_t←1/Q\_tt [-∑(j:j≠t)▒〖Q\_tj p\_j 〗+p^T Qp] (16)  
Chuẩn hóa p