XÂY DỰNG DỮ LIỆU VÀ MÔ HÌNH PHÂN LOẠI BÀI HÁT THEO CHỦ ĐỀ*

1st NGUYỄN QUỐC THÁI
KHOA : KHOA HỌC VÀ KỸ THUẬT THÔNG TIN
EMAIL: 19520936@gm.uit.edu.vn
Thành phố Hồ Chí Minh

2nd NGUYỄN ANH TUẤN
KHOA : KHOA HỌC MÁY TÍNH
EMAIL: 20522114@gm.uit.edu.vn
Thành phố Hồ Chí Minh

***Tóm tắt nội dung*—Âm nhạc đã và đang là thứ không thể nào thiếu trong nhu cầu giải trí của con người, nhu cầu này không ngừng tăng do áp lực của công việc và cuộc sống khiến nhiều người tìm đến âm nhạc như là một liều thuốc an thần giải tỏa căng thẳng mệt mỏi. Cho nên, nhóm chúng tôi quyết định sẽ thu thập dữ liệu những bài hát và xây dựng phân loại chúng theo chủ đề phục vụ cho người dùng một trải nghiệm nghe nhạc tối ưu nhất có thể.**

I. Giới thiệu

Những dữ liệu của chúng tôi đều lấy từ trang https://www.nhaccuatui.com/ Chúng tôi chọn lọc những model có độ chính xác cao và những phương pháp dễ xài nhất để tối ưu hóa cho việc xử lý dữ liệu. Những kết quả của chúng tôi được thực hiện trên tập dataset gồm 3000 bài hát bao hàm cả tên bài hát và tên ca sỹ.

II. Xử lý dữ liệu sau khi crawl

Tiền xử lý dữ liệu là một bước rất quan trọng trong việc giải quyết bất kỳ vấn đề nào trong lĩnh vực Học Máy. Hầu hết các bộ dữ liệu được sử dụng trong các vấn đề liên quan đến Học Máy cần được xử lý, làm sạch và biến đổi trước khi một thuật toán Học Máy có thể được huấn luyện trên những bộ dữ liệu này. Các kỹ thuật tiền xử lý dữ liệu phổ biến hiện nay bao gồm: xử lý dữ liệu bị khuyết (missing data), mã hóa các biến nhóm (encoding categorical variables), chuẩn hóa dữ liệu (standardizing data), co giãn dữ liệu (scaling data),... Bởi vậy mà việc làm sạch dữ liệu trước khi bước vào bài toán là một điều hết sức quan trọng

A. Xóa đi những dữ liệu bị khuyết

Khi đối mặt với trường hợp trong bộ dữ liệu thu thập được có các dữ liệu bị khuyết, cách đơn giản nhất mà chúng tôi nghĩ tới là xóa chúng đi. Điều này có một lợi ích thấy rõ là giúp cho bộ dữ liệu được giảm đi những điểm dữ liệu mập mờ và những điểm dữ liệu nhiễu. Sau đó, chúng tôi tìm thêm những dataset khác để điền vào chỗ dữ liệu bị khuyết.

Identify applicable funding agency here. If none, delete this.

B. Tiến hành xóa thêm rác trong dataset

Để phục vụ cho việc làm sạch được dataset, chúng tôi tiến hành thiết lập những hàm có khả năng xóa đi rác trong dataset, cụ thể là những ký tự có dấu, những dấu () không cần thiết, xóa những ký tự đặc biệt và những stop word,việc này sẽ giúp cho dataset của chúng tôi không còn là một mớ hỗn độn.

C. Xử lý những ký tự viết tắt

Việc có những ký tự viết tắt là một lỗ hổng lớn trong dataset, nó sẽ ảnh hưởng rất nhiều trong quá trình training model. Cho nên, chúng tôi đã tìm và xử lý những từ viết tắt này một cách tốt nhất có thể để cho dataset đạt được hiệu quả cao nhất. VD: nsnd = nghệ sỹ nhân dân, nsưt = nghệ sỹ ưu tú,...

III. Tiến hành gán nhãn cho dataset

Tiếp theo, chúng tôi tiến hành gán nhãn cho dataset, các nhãn được chia thành 0, 1, 2, 3. Cụ thể là:

0: Nhạc cách mạng
1: Nhạc trẻ
2: Nhạc rock việt
3: Nhạc Trịnh

Tiếp theo đó, chúng tôi chia nhau ra để gán nhãn cho tập dataset. Mỗi người chúng tôi đều có kết quả riêng biệt. Việc gán nhãn diễn ra cho tới khi độ chính xác của tập dataset đạt đủ chỉ tiêu thì dừng lại.

IV. Vector hóa dữ liệu bằng Tfidf Vectorizer

TF-IDF có nghĩa là Tần suất kỳ hạn - Tần suất tài liệu nghịch đảo. Đây là một thống kê dựa trên tần suất xuất hiện của một từ trong ngữ liệu nhưng nó cũng cung cấp một đại diện bằng số về mức độ quan trọng của một từ đối với phân tích thống kê.

Sở dĩ chúng tôi chọn phương pháp này vì chúng tôi cho rằng TF-IDF tốt hơn Count Vectorizers vì nó không chỉ tập trung vào tần suất xuất hiện của các từ trong ngữ liệu mà còn cung cấp mức độ quan trọng của các từ. Sau đó, chúng tôi có thể loại bỏ các từ ít quan trọng hơn để phân tích, do đó làm cho việc xây dựng mô hình bớt phức tạp hơn bằng cách giảm kích thước đầu vào. Về mặt toán học, TFIDIF có thể được định nghĩa như sau:

For a term i in document j:

$w_{ij} = tf_{ij}$ x log($\frac{N}{df_i}$)

trong đó:
tf_{ij}: số lượng biến số trong j
df_i: những văn bản trong i
N: tổng số văn bản

Ngoài ra, vì trong công thức df có mặt trong mẫu số của (N/df) nên nó được gọi là tần số tài liệu nghịch đảo. Do đó tên tf-idf.

A. Cách thức hoạt động

TFIDF dựa trên logic rằng các từ quá nhiều trong kho văn bản và các từ quá hiếm đều không quan trọng về mặt thống kê để tìm ra một mẫu. Hệ số logarit trong tfidf có thể tìm các lỗi từ như các từ quá nhiều hoặc quá hiếm trong kho văn bản bằng cách cho chúng điểm tfidf thấp.

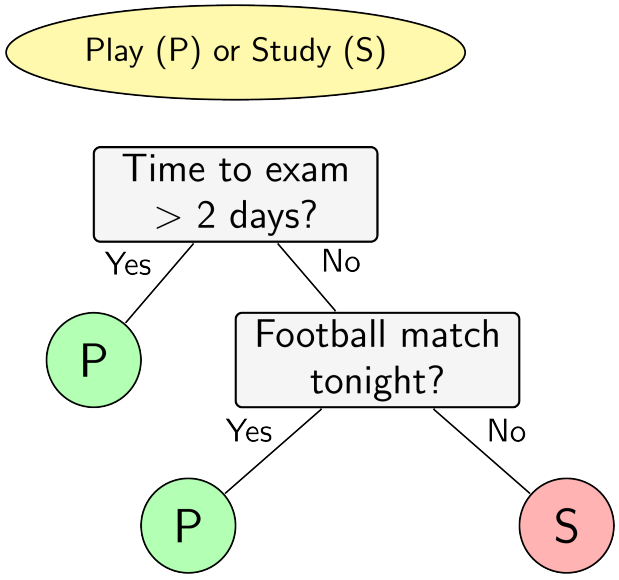
Giá trị cao hơn của tfidf biểu thị tầm quan trọng cao hơn của các từ trong tài liệu trong khi giá trị thấp hơn biểu thị tầm quan trọng thấp hơn.

V. Thử những training model để tìm ra cái tốt nhất

A. Decision Tree

Chúng tôi xin giới thiệu tóm gọn decison tree qua ví dụ dưới đây:

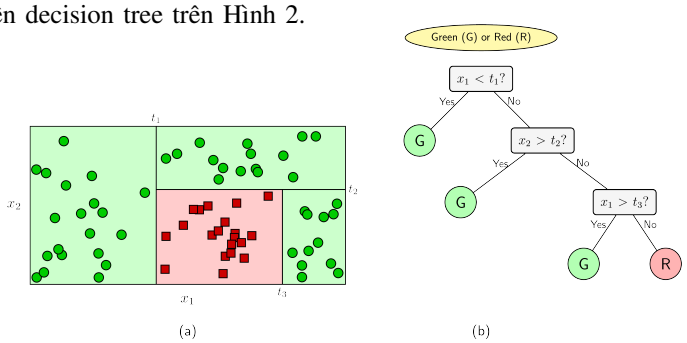
Sắp đến kỳ thi, một cậu sinh viên tự đặt ra quy tắc học hay chơi của mình như sau. Nếu còn nhiều hơn hai ngày tới ngày thi, cậu ra sẽ đi chơi. Nếu còn không quá hai ngày và đêm hôm đó có một trận bóng đá, cậu sẽ sang nhà bạn chơi và cùng xem bóng đêm đó. Cậu sẽ chỉ học trong các trường hợp còn lại. Việc ra quyết định của cậu sinh viên này có thể được mô tả trên sơ đồ trong Hình 1. Hình ellipse nền vàng thể hiện quyết định cần được đưa ra. Quyết định này phụ thuộc vào các câu trả lời của các câu hỏi trong các ô hình chữ nhật màu xám. Dựa trên các câu trả lời, quyết định cuối cùng được cho trong các hình tròn màu lục (chơi) và đỏ (học). Sơ đồ trong Hình 1 còn được gọi là một cây quyết định.



Hình 1: ví dụ về việc quyết định dựa trên các câu hỏi

Việc quan sát, suy nghĩ và ra các quyết định của con người thường được bắt đầu từ các câu hỏi. Machine learning cũng có một mô hình ra quyết định dựa trên các câu hỏi. Mô hình này có tên là cây quyết định (decision tree).

Xét ví dụ trên với hai class màu lục và đỏ trên không gian hai chiều. Nhiệm vụ là đi tìm ranh giới đơn giản giúp phân chia hai class này. Hay nói cách khác, đây là một bài toán classification, ta cần xây dựng một bộ phân lớp để quyết định việc một điểm dữ liệu mới thuộc vào class nào. Quan sát hình ta thấy rằng ranh giới cho hai class trong bài toán này khá đơn giản–chúng là các đường song song với các trục toạ độ. Nếu một điểm có thành phần thứ nhất, x_1 nhỏ hơn ngưỡng t_1 ta quyết định ngay được rằng nó thuộc class lục. Ngoài ra, nếu thành phần thứ hai,x_2 lớn hơn ngưỡng t_2 ta quyết định nó cũng thuộc vào class lục. Xét tiếp, nếu thành phần thứ nhất,x_1, lớn hơn ngưỡng t_3 ta quyết định nó thuộc vào class lục. Các điểm không thoả mãn các điều kiện trên được xếp vào class đỏ. Việc ra quyết định một điểm thuộc class nào được mô tả trên decision tree trên Hình 2.



Hình 2: Ví dụ về bài toán phân lớp sử dụng decision tree.

Trong decision tree, các ô màu xám, lục, đỏ trên Hình 2 được gọi là các node. Các node thể hiện đầu ra (màu lục và đỏ) được gọi là node lá (leaf node hoặc terminal node). Các node thể hiện câu hỏi là các non-leaf node. Non-leaf node trên cùng (câu hỏi đầu tiên) được gọi là node gốc (root node). Các non-leaf node thường có hai hoặc nhiều node con (child node). Các child node này có thể là một leaf node hoặc một non-leaf node khác. Các child node có cùng bố mẹ được gọi là sibling node. Nếu tất cả các non-leaf node chỉ có hai child node, ta nói rằng đó là một binary decision tree (cây quyết định nhị phân). Các câu hỏi trong binary decision tree đều có thể đưa được về dạng câu hỏi đúng hay sai. Các decision tree mà một leaf node có nhiều child node cũng có thể được đưa về dạng một binary decision tree. Điều này có thể đạt được vì hầu hết các câu hỏi đều có thể được đưa về dạng câu hỏi đúng sai.

Decision tree là một mô hình supervised learning, có thể được áp dụng vào cả hai bài toán classification và regression. Việc xây dựng một decision tree trên dữ liệu huấn luyện cho trước là việc đi xác định các câu hỏi và thứ tự của chúng. Một điểm đáng lưu ý của decision tree là nó có thể làm việc với các đặc trưng (trong các tài liệu về decision tree, các đặc trưng thường được gọi là thuộc tính – attribute) dạng categorical, thường là rời rạc và không có thứ tự. Ví dụ, mưa, nắng hay xanh, đỏ, v.v. Decision tree cũng làm việc với dữ liệu có vector đặc trưng bao gồm cả thuộc tính dạng categorical và liên tục

(numeric). Một điểm đáng lưu ý nữa là decision tree ít yêu cầu việc chuẩn hoá dữ liệu.

Việc mô hình này được chúng tôi tin tưởng không những là do những yếu tố kể trên, mà còn là do thuật toán của nó. Chúng tôi tin rằng thuật toán của Decision Tree rất thích hợp cho những bài toán kiểu như vậy, cụ thể như sau:

1) ID3: Trong ID3, chúng ta cần xác định thứ tự của thuộc tính cần được xem xét tại mỗi bước. Với các bài toán có nhiều thuộc tính và mỗi thuộc tính có nhiều giá trị khác nhau, việc tìm được nghiệm tối ưu thường là không khả thi. Thay vào đó, một phương pháp đơn giản thường được sử dụng là tại mỗi bước, một thuộc tính tốt nhất sẽ được chọn ra dựa trên một tiêu chuẩn nào đó (chúng ta sẽ bàn sớm). Với mỗi thuộc tính được chọn, ta chia dữ liệu vào các child node tương ứng với các giá trị của thuộc tính đó rồi tiếp tục áp dụng phương pháp này cho mỗi child node. Việc chọn ra thuộc tính tốt nhất ở mỗi bước như thế này được gọi là cách chọn greedy (tham lam). Cách chọn này có thể không phải là tối ưu, nhưng trực giác cho chúng ta thấy rằng cách làm này sẽ gần với cách làm tối ưu. Ngoài ra, cách làm này khiến cho bài toán cần giải quyết trở nên đơn giản hơn.

Sau mỗi câu hỏi, dữ liệu được phân chia vào từng child node tương ứng với các câu trả lời cho câu hỏi đó. Câu hỏi ở đây chính là một thuộc tính, câu trả lời chính là giá trị của thuộc tính đó. Để đánh giá chất lượng của một cách phân chia, chúng ta cần đi tìm một phép đo.

Trước hết, thế nào là một phép phân chia tốt? Bằng trực giác, một phép phân chia là tốt nhất nếu dữ liệu trong mỗi child node hoàn toàn thuộc vào một class–khi đó child node này có thể được coi là một leaf node, tức ta không cần phân chia thêm nữa. Nếu dữ liệu trong các child node vẫn lẫn vào nhau theo tỉ lệ lớn, ta coi rằng phép phân chia đó chưa thực sự tốt. Từ nhận xét này, ta cần có một hàm số đo độ tinh khiết (purity), hoặc độ vẩn đục (impurity) của một phép phân chia. Hàm số này sẽ cho giá trị thấp nhất nếu dữ liệu trong mỗi child node nằm trong cùng một class (tinh khiết nhất), và cho giá trị cao nếu mỗi child node có chứa dữ liệu thuộc nhiều class khác nhau.

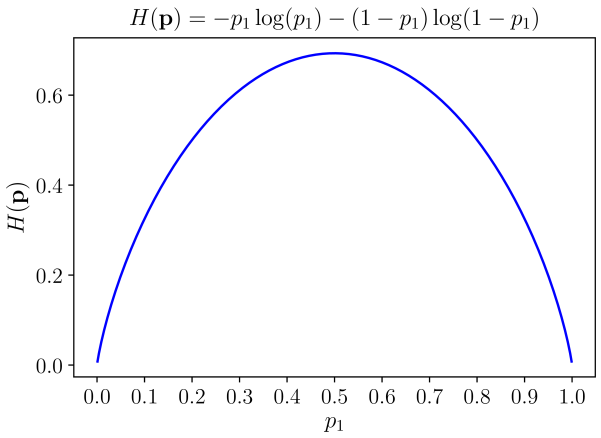
Một hàm số có các đặc điểm này và được dùng nhiều trong lý thuyết thông tin là hàm entropy.

Hàm entropy có thể được phát biểu ngắn gọn như sau:

Cho một phân phối xác suất của biến rời rạc x có thể nhận n giá trị khác nhau x_1 , x_2. Gỉa sử xác suất để x nhận các giá trị này là p_i = p (x = x_i) với 0 ≤p_i ≤ 1,$\sum_{i=1}^{n} p_i$ = 1. Ký hiệu phân phối này là p = (p_1 , p_2,....,p_n). Entroy của phân phối này được định nghĩa là:

H(p) = - $\sum_{i=1}^{n}$ $p_i$log(p_i) (1)

Xét một ví dụ với n = 2 được cho trên Hình 3. Trong trường hợp p tinh khiết nhất, tức một trong hai giá trị của giá trị p_i bằng 1, giá trị kia bằng 0 , entropy của phân phoi này là H(p) = 0 . Khi p vẩn đục nhất, tức hai giá trị p_i = 0.5, hàm entropy có giá trị cao nhất.



Hình 3: Đồ thị của hàm entropy với n = 2.

Tổng quát lên với n > 2 , hàm entropy đạt giá trị nhỏ nhất nếu có một giá trị p_i = 1, đạt giá trị lớn nhất nếu các p_i bằng nhau.

Những tính chất này của hàm entropy khiến nó được sử dụng trong việc đo độ vẩn đục của một phép phân chia của ID3. Vì lý do này, ID3 còn được gọi là entropy-based decision tree.

2) Thuật toán của ID3: Trong ID3, tổng có trọng số của entropy tại các leaf-node sau khi xây dựng decision tree được coi là hàm mất mát của decision tree đó. Các trọng số ở đây tỉ lệ với số điểm dữ liệu được phân vào mỗi node. Công việc của ID3 là tìm các cách phân chia hợp lý (thứ tự chọn thuộc tính hợp lý) sao cho hàm mất mát cuối cùng đạt giá trị càng nhỏ càng tốt. Như đã đề cập, việc này đạt được bằng cách chọn ra thuộc tính sao cho nếu dùng thuộc tính đó để phân chia, entropy tại mỗi bước giảm đi một lượng lớn nhất. Bài toán xây dựng một decision tree bằng ID3 có thể chia thành các bài toán nhỏ, trong mỗi bài toán, ta chỉ cần chọn ra thuộc tính giúp cho việc phân chia đạt kết quả tốt nhất. Mỗi bài toán nhỏ này tương ứng với việc phân chia dữ liệu trong một non-leaf node. Chúng ta sẽ xây dựng phương pháp tính toán dựa trên mỗi node này. Xét một bài toán với C class khác nhau. Giả sử ta đang làm việc với một non-leaf node với các điểm dữ liệu tạo thành một tập S với số phần tử là |S| = N. Giả sử thêm rằng trong số điểm dữ liệu này, N_C,c = 1,2,....,C điểm thuộc vào class c. Xác suất để mỗi điểm dữ liệu rơi vào một class c được xấp xỉ bằng $\frac{N_c}{N}$ (maximum likelihood estimation). Như vậy, entropy tại node này được tính bởi: H(S) = - $\sum_{c=1}^{C}$ $\frac{N_c}{N}$ log($\frac{N_c}{N}$) (2)

Tiếp theo, giả sử thuộc tính được chọn là Dựa trên x, các điểm dữ liệu trong S được phân ra thành K child node S_1, S_2,...., S_k với số điểm trong mỗi child node lần lượt là m_1,m_2,...,m_k. Ta định nghĩa:

H(x,S) = $\sum_{k=1}^{K}$ $\frac{m_k}{N}$ H(S_k)(3)

là tổng có trọng số entroy của mỗi child node–được tính tương tự như (2). Việc lấy trọng số này là quan trọng vì các node thường có số lượng điểm khác nhau.

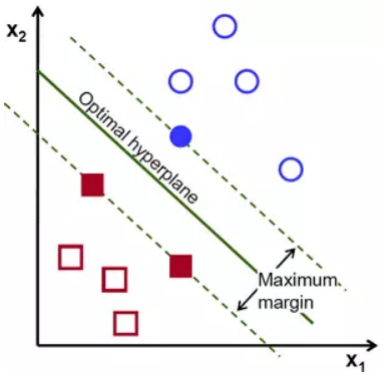
Tiếp theo, ta định nghĩa information gain dựa trên thuộc tính x: G(x,S) = H(S) - H(x,S)

Trong ID3, tại mỗi node, thuộc tính được chọn được xác định dựa trên: x^* = argmax G(x,S) = argmin H(x,S) tức thuộc

tính khiến cho information gain đạt giá trị lớn nhất.

B. Support Vector Machine (SVM)

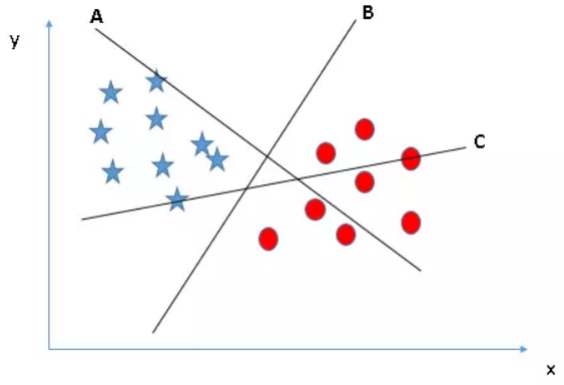
SVM là một thuật toán giám sát, nó có thể sử dụng cho cả việc phân loại hoặc đệ quy. Tuy nhiên nó được sử dụng chủ yếu cho việc phân loại. Trong thuật toán này, chúng ta vẽ đồi thị dữ liệu là các điểm trong n chiều (ở đây n là số lượng các tính năng bạn có) với giá trị của mỗi tính năng sẽ là một phần liên kết. Sau đó chúng tôi 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 hai phần riêng biệt.



Support Vectors hiểu một cách đơn giản là các đối tượng trên đồ thị tọa độ quan sát, Support Vector Machine là một biên giới để chia hai lớp tốt nhất.

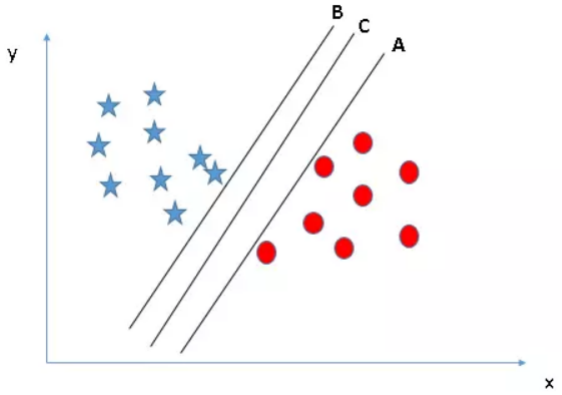
Tiếp theo sau đây sẽ là cách sử dụng của model này:

1) Identify the right hyper-plane (Scenario-1): Ở đây, có 3 đường hyper-lane (A,B and C). Bây giờ đường nào là hyper-lane đúng cho nhóm ngôi sao và hình tròn.



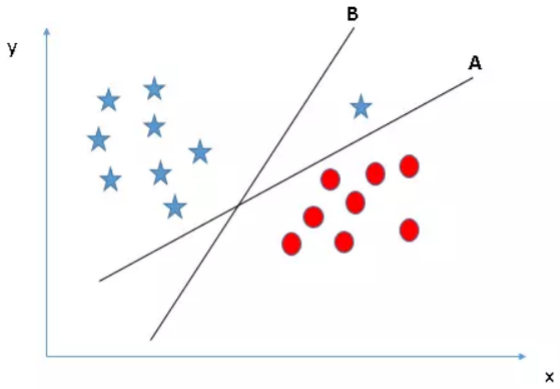
Quy tắc số một để chọn 1 hyper-lane, chọn một hyper-plane để phân chia hai lớp tốt nhất. Trong ví dụ này chính là đường B.

2) Identify the right hyper-plane (Scenario-2: Ở đây chúng ta cũng có 3 đường hyper-plane (A,B và C), theo quy tắc số 1, chúng đều thỏa mãn.



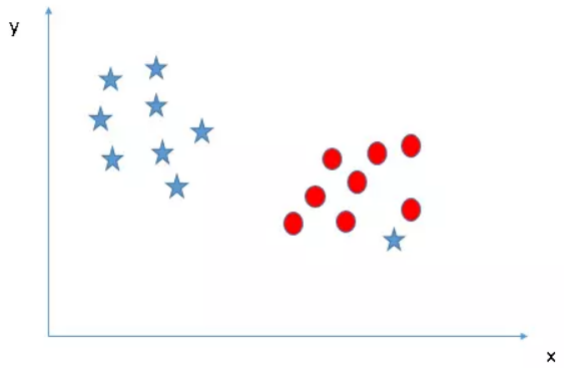
Quy tắc thứ hai chính là xác định khoảng cách lớn nhất từ điểu gần nhất của một lớp nào đó đến đường hyper-plane. Khoảng cách này được gọi là "Margin", Hãy nhìn hình bên dưới, trong đấy có thể nhìn thấy khoảng cách margin lớn nhất đấy là đường C. Cần nhớ nếu chọn lầm hyper-lane có margin thấp hơn thì sau này khi dữ liệu tăng lên thì sẽ sinh ra nguy cơ cao về việc xác định nhầm lớp cho dữ liệu.

3) Identify the right hyper-plane (Scenario-3): Sử dụng các nguyên tắc đã nêu trên để chọn ra hyper-plane cho trường hợp sau:

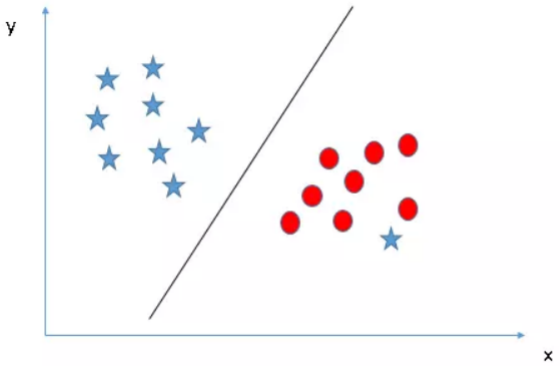


Có thể có một vài người sẽ chọn đường B bởi vì nó có margin cao hơn đường A, nhưng đấy sẽ không đúng bởi vì nguyên tắc đầu tiên sẽ là nguyên tắc số 1, chúng ta cần chọn hyper-plane để phân chia các lớp thành riêng biệt. Vì vậy đường A mới là lựa chọn chính xác.

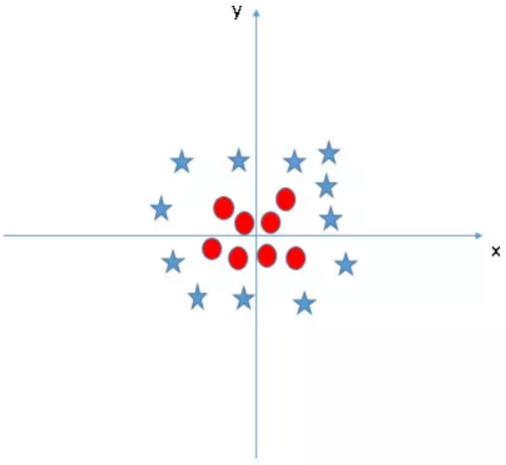
4) Can we classify two classes (Scenario-4): Tiếp the hãy xem hình bên dưới, không thể chia thành hai lớp riêng biệt với 1 đường thẳng, để tạo 1 phần chỉ có các ngôi sao và một vùng chỉ chứa các điểm tròn.



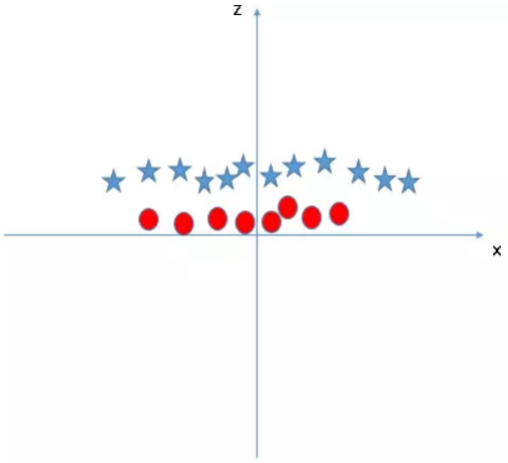
Ở đây sẽ chấp nhận, một ngôi sao ở bên ngoài cuối được xem như một ngôi sao phía ngoài hơn, SVM có tính năng cho phép bỏ qua các ngoại lệ và tìm ra hyper-plane có biên giới tối đa . Do đó có thể nói, SVM có khả năng mạnh trong việc chấp nhận ngoại lệ.



5) Find the hyper-plane to segregate to classes (Scenario-5): Trong trường hợp dưới đây, không thể tìm ra 1 đường hyper-plane tương đối để chia các lớp, vậy làm thế nào để SVM phân tách dữ liệu thành hai lớp riêng biệt? Cho đến bây giờ chúng ta chỉ nhìn vào các đường tuyến tính hyper-plane.

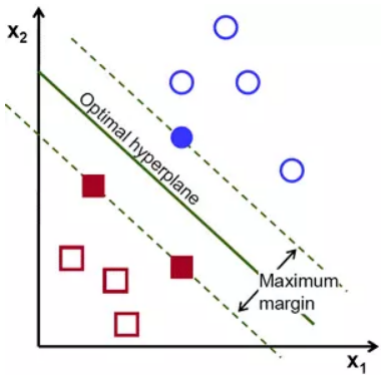


SVM có thể giải quyết vấn đề này, Khá đơn giản, nó sẽ được giải quyết bằng việc thêm một tính năng, Ở đây chúng ta sẽ thêm tính năng z = x^2+ y^2. Bây giờ dữ liệu sẽ được biến đổi theo trục x và z như sau:



Trong sơ đồ trên, các điểm cần xem xét là: • Tất cả dữ liệu trên trục z sẽ là số dương vì nó là tổng bình phương x và y • Trên biểu đồ các điểm tròn đỏ xuất hiện gần trục x và y hơn vì thế z sẽ nhỏ hơn => nằm gần trục x hơn trong đồ thị (z,x) Trong SVM, rất dễ dàng để có một siêu phẳng tuyến tính (linear hyper-plane) để chia thành hai lớp, Nhưng một câu hỏi sẽ nảy sinh đấy là, chúng ta có cần phải thêm một tính năng phân chia này bằng tay hay không. Không, bởi vì SVM có một kỹ thuật được gọi là kernel trick (kỹ thuật hạt nhân), đây là tính năng có không gian đầu vào có chiều sâu thấm và biến đổi nó thành không gian có chiều cao hơn, tức là nó không phân chia các vấn đề thành các vấn đề riêng biệt, các tính năng này được gọi là kernel. Nói một cách đơn giản nó thực hiện một số biết đổi dữ liệu phức tạp, sau đó tìm ra quá trình tách dữ liệu dựa trên các nhãn hoặc đầu ra mà chúng ra đã xác định trước.

C. Margin trong SVM



Margin là khoảng cách giữa siêu phẳng đến 2 điểm dữ liệu gần nhất tương ứng với các phân lớp. Trong ví dụ quả táo quả lê đặt trên mặt bán, margin chính là khoảng cách giữa cây que và hai quả táo và lê gần nó nhất. Điều quan trọng ở đây đó là phương pháp SVM luôn cố gắng cực đại hóa margin này, từ đó thu được một siêu phẳng tạo khoảng cách xa nhất so với 2 quả táo và lê. Nhờ vậy, SVM có thể giảm thiểu việc phân lớp sai (misclassification) đối với điểm dữ liệu mới đưa vào

VI. Logicstics Regression

Logistic Regression là 1 thuật toán phân loại được dùng để gán các đối tượng cho 1 tập hợp giá trị rời rạc (như 0, 1, 2, ...). Một ví dụ điển hình là phân loại Email, gồm có email công việc, email gia đình, email spam, ... Giao dịch trực tuyến có là an toàn hay không an toàn, khối u lành tính hay ác tình. Thuật toán trên dùng hàm sigmoid logistic để đưa ra đánh giá theo xác suất.

A. Xác định câu hỏi

Bất kỳ quá trình phân tích dữ liệu nào cũng bắt đầu bằng một câu hỏi kinh doanh. Đối với hồi quy logistic, bạn nên giới hạn phạm vi câu hỏi để có được kết quả cụ thể:

Những ngày mưa có ảnh hưởng đến doanh số hàng tháng của chúng ta không? (có hoặc không)

Khách hàng đang thực hiện loại hoạt động thẻ tín dụng nào? (ủy quyền, gian lận hoặc có khả năng gian lận)

1) Thu thập dữ liệu lịch sử: Sau khi xác định câu hỏi, bạn cần xác định các yếu tố dữ liệu có liên quan. Sau đó, chúng tôi sẽ thu thập dữ liệu trước đây cho tất cả các yếu tố. Ví dụ: để trả lời câu hỏi đầu tiên ở trên, bạn có thể thu thập dữ liệu doanh số hàng tháng và số ngày mưa mỗi tháng trong ba năm qua.

2) training model hồi quy: Chúng tôi sẽ xử lý dữ liệu lịch sử bằng phần mềm hồi quy. Phần mềm sẽ xử lý các điểm dữ liệu khác nhau và kết nối chúng theo phương thức toán học bằng cách sử dụng phương trình. Ví dụ: nếu số ngày mưa trong ba tháng là 3, 5 và 8 còn doanh số trong những tháng đó là 8, 12 và 18, thuật toán hồi quy sẽ kết nối các yếu tố này với phương trình:

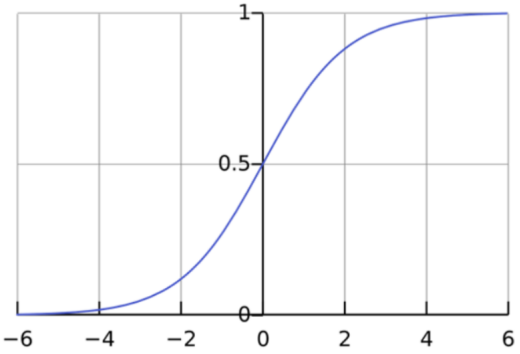
Doanh số = 2*(Số ngày mưa) + 2.

B. Model hoạt động như thế nào?

1) Hàm hồi quy logicstics: Hồi quy logistic là một mô hình thống kê sử dụng hàm logistic, hay hàm logit trong toán học làm phương trình giữa x và y. Hàm logit ánh xạ y làm hàm sigmoid của x.

f(x) = $\frac{1}{1+e^{-x}}$

Nếu vẽ phương trình hồi quy logistic này, chúng tôi sẽ có một đường cong hình chữ S như hình dưới đây.



Hàm logit chỉ trả về các giá trị giữa 0 và 1 cho biến phụ thuộc, dù giá trị của biến độc lập là gì. Đây là cách hồi quy logistic ước tính giá trị của biến phụ thuộc. Phương pháp hồi quy logistic cũng lập mô hình phương trình giữa nhiều biến độc lập và một biến phụ thuộc.

2) Phân tích hồi quy với biến độc lập: Trong nhiều trường hợp, nhiều biến giải thích ảnh hưởng đến giá trị của biến phụ thuộc. Để lập mô hình các tập dữ liệu đầu vào như vậy, công thức hồi quy logistic phải giả định mối quan hệ tuyến tính giữa các biến độc lập khác nhau. Bạn có thể sửa đổi hàm sigmoid và tính toán biến đầu ra cuối cùng như sau:

y = f(B_0 + B_1xX_1 + ... + B_nxX_n)

Ký hiệu B đại diện cho hệ số hồi quy. Mô hình logit có thể đảo ngược tính toán các giá trị hệ số này khi bạn cho nó một tập dữ liệu thực nghiệm đủ lớn có các giá trị đã xác định của cả hai biến phụ thuộc và biến độc lập.

VII. So sánh độ chính xác giữa chúng để tìm ra model tốt nhất

Sau khi đã tìm hiểu về khái niệm, tính chất và cách sử dụng của từng loại model.Chúng tôi tiến hành thực nghiệm dựa trên mẫu 3000 bài hát và đưa ra đánh giá của từng loại. Những con số được hiển thị ra được chúng tôi đánh giá thông qua các hàm tính độ chính xác của từng loại mode, cụ thể qua bảng sau:

	Decision Tree	SVM	Log	Voting Classifier
Tên bài hát	0.62	0.63	0.63	0.63
Tên bài hát và tên nhạc sĩ	0.72	0.83	0.79	0.81
Theo lời bài hát	0.63	0.8	0.78	0.78
Sử dụng data cả 3	0.65	0.84	0.79	0.81

Tài liệu

[1] Slide môn học Học Máy Thống Kê

[2] Machine Learning for Dummies (IBM Limited Edition – Hurwitz and Kirsch)

[3] Understanding Machine Learning (Shai Shalev-Shwartz and Shai Ben-David).

[4] A Programmer's Guide to Data Mining (Zacharski).