# Các thuật toán có thể dùng cho dự báo thời tiết

## Thuật toán phân loại bằng Naive Bayes – phân loại dựa trên xác suất

Kỹ thuật này có thể hiểu đơn giản như sau: Với một mẫu dữ liệu cần phân lớp, ta tính xác suất có điều kiện để mẫu dữ liệu đó rơi vào từng lớp trong tập các lớp đã biết trước. Mẫu dữ liệu sẽ được phân vào lớp nào có xác suất cao nhất.

Trước tiên, ta xét bài toán phân lớp. Cho một tập dữ liệu huấn luyện *X* ∈ *Rn×(m+1)* gồm *n* mẫu dữ liệu, mỗi mẫu có *m* thuộc tính và một thuộc tính lớp. Mỗi mẫu huấn luyện *x∈X* được biểu diễn là một vectơ *m*+1 chiều *x*(*x1, x2, ..., xm, y*) trong đó gồm *m* thành phần dữ liệu và *y* là nhãn lớp. Cho một tập xác định các nhãn lớp *C* = {*c1, c2,..., cq*} gồm *q* lớp. Dễ thấy *y*∈*C*. Cho một mẫu dữ liệu mới *z∈Rm* và *z* được biểu diễn bằng: *z(z1, z2,...,zm)*. Hãy xác định lớp của *z*.

Để xác định lớp của *z*, một cách đơn giản là ta tính xác suất xảy ra khả năng *z* được phân vào từng lớp *ci*, *i*=1..*q*, tức khả năng xảy ra *ci*. Mẫu *z* sẽ được phân vào lớp nào có xác suất xảy ra cao nhất.

Tuy nhiên, mẫu *z* là xác định với các thành phần quan sát được là *z1, z2,...zm*. Do đó, xác suất để *z* thuộc vào lớp *ci* phải là xác suất có điều kiện và được ký hiệu là *P(ci|z).* Theo định lý Bayes, xác suất này được tính như sau:

.

(3.5)

Trong phương pháp *Naive Bayes*, từ *Naive* có hàm ý giả sử rằng các thuộc tính là độc lập có điều kiện đối với các thuộc tính khác. Do đó



(3.6)

và (3.5) trở thành:

.

(3.7)

Mẫu dữ liệu *z* sẽ được phân vào lớp *ck* nếu *P(ck|z)* là lớn nhất tức:



(3.8)

Vì *P(z)* là hằng số đối với các *ci* khác nhau, do vậy (3.8) tương đương với:



(3.9)

Một cách đơn giản hơn, để xác định lớp cho mẫu dữ liệu *z*, ta lần lượt tính các giá trị của biểu thức  với từng lớp *ci* ∈{*c1, c2,...,cq}*. Lớp *ci* nào cho giá trị của biểu thức lớn nhất sẽ là lớp của *z*. Quá trình phân lớp sử dụng phương pháp *Naïve Bayes* bao gồm hai bước:

**Bước 1:** Đối với mỗi lớp *ci* ∈ *C*, tính giá trị của:

Xác suất tiên nghiệm *P(ci)*. Xác suất này được tính xấp xỉ bằng tổng số mẫu thuộc lớp *ci* trên tổng số mẫu của bộ dữ liệu huấn luyện.

Đối với mỗi giá trị thuộc tính *zj*, tính *P(zj|ci)* là xác suất xảy ra của giá trị đó trong lớp *ci*. Giá trị này cũng được tính xấp xỉ bằng tỷ lệ các mẫu có giá trị trên thuộc tính thứ *j* là *zj* trong số các mẫu thuộc lớp *ci*.

**Bước 2:** Cần xác định lớp cho một mẫu dữ liệu mới *z*, ta thực hiện:

Đối với mỗi lớp *ci* ∈*C*, tính giá trị của biểu thức:

,

(3.10)

Xác định lớp của *z* là *ck*:

(3.11)



## Thuật toán phân loại cây quyết định]

### Thuật toán ID3

Giải thuật ID3 được phát triển đồng thời bởi Quinlan trong AI và Breiman, Friedman, Olsen và Stone trong thống kê. ID3 là một giải thuật học đơn giản nhưng tỏ ra thành công trong nhiều lĩnh vực. ID3 là một giải thuật hay vì cách biểu diễn tri thức học được của nó, tiếp cận của nó trong việc quản lý tính phức tạp, heuristic của nó dùng cho việc chọn lựa các khái niệm ứng viên, và tiềm năng của nó đối với việc xử lý dữ liệu nhiễu.

ID3 biểu diễn các khái niệm (concept) ở dạng các cây quyết định (decision tree). Biểu diễn này cho phép chúng ta xác định phân loại của một đối tượng bằng cách kiểm tra các giá trị của nó trên một số thuộc tính nào đó.

Như vậy, nhiệm vụ của giải thuật ID3 là học cây quyết định từ một tập các ví dụ rèn luyện (training example) hay còn gọi là dữ liệu rèn luyện (training data).

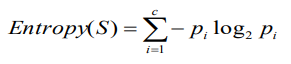
Input: Một tập hợp các ví dụ. Mỗi ví dụ bao gồm các thuộc tính mô tả một tình huống, hay một đối tượng nào đó, và một giá trị phân loại của nó.

Output: Cây quyết định có khả năng phân loại đúng đắn các ví dụ trong tập dữ liệu rèn luyện, và hy vọng là phân loại đúng cho cả các ví dụ chưa gặp trong tương lai.

Giải thuật ID3 xây dựng cây quyết định được trình bày như sau:

Lặp:  
1. Chọn A <= thuộc tính quyết định “tốt nhất” cho nút kế tiếp  
2. Gán A là thuộc tính quyết định cho nút  
3. Với mỗi giá trị của A, tạo nhánh con mới của nút  
4. Phân loại các mẫu huấn luyện cho các nút lá  
5. Nếu các mẫu huấn luyện được phân loại hoàn toàn thì NGƯNG,  
Ngược lại, lặp với các nút lá mới.

Để đánh giá chính xác thông tin thu được, dùng Entropy(S): Độ bất định (độ pha trộn/độ hỗn tạp) của S liên quan đến sự phân loại đang xét.



Trong đó pi là xác suất xuất hiện trạng thái i của hệ thống. Theo lý thuyết thông tin: mã có độ dài tối ưu là mã gán –log2p bits cho thông điệp có xác suất là p. S là một tập huấn luyện.

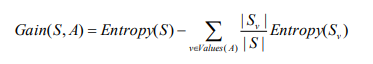
Nếu gọi p⊕ là xác suất xuất hiện các ví dụ dương trong tập S, pΘ là xác suất xuất hiện các ví dụ âm trong tập S. Entropy đo độ bất định của tập S sẽ là:



Quy định 0.log0 = 0

Chẳng hạn với tập S gồm 14 mẫu có chung một vài giá trị logic gồm 9 mẫu dương và 5 mẫu âm. Khi đó đại lượng Entropy của tập S liên quan đến sự phân loại logic này là: Entropy([9+, 5-]) = - (9/14)log2(9/14) - (5/14)log2(5/14) = 0,940 **Chú ý :**

* Đại lượng Entropy = 0 nếu tất cả thành viên của tập S cùng thuộc một lớp (vì nếu tất cả là dương (P+ = 1), do đó P- = 0, Entropy(S) = −1log2 1− 0log2 0 = 0 ).
* Đại lượng Entropy(S) = 1 khi tập S chứa tỉ lệ tập mẫu âm và mẫu dương là như nhau. Nếu tập S chứa tập mẫu âm và tâp mẫu dương có tỉ lệ P+ khác P- thì Entropy(S) ∈ (0,1).
* Dựa trên sự xác định entropy, ta tính Gain(S, A) = Lượng giảm entropy mong đợi qua việc chia các ví dụ theo thuộc tính A :



### Thuật toán C4.5

Thuật toán C4.5 là thuật toán cải tiến của ID3.

Trong thuật toán ID3, Information Gain được sử dụng làm độ đo. Tuy nhiên, phương pháp này lại ưu tiên những thuộc tính có số lượng lớn các giá trị mà ít xét tới những giá trị nhỏ hơn. Do vậy, để khắc phục nhược điểm trên, ta sử dụng độ đo Gain Ratio (trong thuật toán C4.5) như sau:

Đầu tiên, ta chuẩn hoá information gain với trị thông tin phân tách (split information):



Trong đó: Split Info được tính như sau:



Giả sử chúng ta phân chia biến thành n nút cón và Di đại diện cho số lượng bản ghi thuộc nút đó. Do đó, hệ số Gain Ratio sẽ xem xét được xu hướng phân phối khi chia cây.

## Thuật toán phân loại dựa trên láng giềng gần nhất (K-Nearest Neighbors)

Thuật toán KNN là một trong những phương pháp học có giám sát “Supervised Learning” tức dựa trên biến mục tiêu đã được xác định trước đó, thuật toán sẽ xem xét dữ liệu đã chứa biến mục tiêu (đã phân loại) để “học” và tìm ra những biến d có thể tác động đến biến mục tiêu.

KNN dựa trên giả định là những thứ tương tự hay có tính chất gần giống nhau sẽ nằm ở vị trí gần nhau, với giả định như vậy, KNN được xây dựng trên các công thức toán học phục vụ để tính khoảng cách giữa 2 điểm dữ liệu (gọi là Data points) để xem xét mức độ giống nhau của chúng.

KNN còn gọi là “Lazy learning method” vì tính đơn giản của nó, có nghĩa là quá trình training không quá phức tạp để hoàn thiênhj mô hình (tất cả các dữ liệu đào tạo có thể được sử dụng để kiểm tra mô hình KNN). Điều này làm cho việc xây dựng mô hình nhanh hơn nhưng giai đoạn thử nghiệm chậm hơn và tốn kém hơn về mặt thời gian và bộ nhớ lưu trữ, đặc biệt khi bộ dữ liệu lớn và phức tạp với nhiều biến khác nhau. Trong trường hợp xấu nhất, KNN cần thêm thời gian để quét tất cả các điểm dữ liệu và việc này sẽ cần nhiều không gian bộ nhớ hơn để lưu trữ dữ liệu. Ngoài ra KNN không cần dựa trên các tham số khác nhau để tiến hành phân loại dữ liệu, không đưa ra bất kỳ kết luận cụ thể nào giữa biến đầu vào và biến mục tiêu, mà chỉ dựa trên khoảng cách giữa data point cần phân loại với data point đã phân loại trước đó. Đây là một đặc điểm cực kỳ hữu ích vì hầu hết dữ liệu trong thế giới thực tại không thực sự tuân theo bất kỳ giả định lý thuyết nào ví dụ như phân phối chuẩn trong thống kê.

Cho một bộ dữ liệu huấn luyện X gồm n mẫu: X(x1, x2, …, xn). Mỗi mẫu dữ liệu xi (i=1,...,n) gồm d thuộc tính dữ liệu và một thuộc tính lớp. Một mẫu dữ liệu *y(y1, y2,…,yd) chưa* được xác định lớp. Để phân lớp mẫu *y*, ta tiến hành như sau:

**Bước 1:** Tính khoảng cách từ *y* tới *n* mẫu dữ liệu trong bộ dữ liệu huấn luyện, tức là tính *n* khoảng cách *d(y, xi), i*=1,…,*n*.

**Bước 2:** Lựa chọn ra *k* mẫu dữ liệu trong bộ dữ liệu huấn luyện “*gần*” với *y* nhất, tức có khoảng cách tới *y* là nhỏ nhất.

**Bước 3:** Xác định lớp cho *y*. Lớp của mẫu *y* được xác định là lớp xuất hiện nhiều nhất trong số các giá trị lớp của *k* mẫu gần với *y* nhất đã tìm được ở Bước 2.

Bước khó khăn nhất của thuật toán KNN, và cũng là bước đau đầu nhất, cần sự kinh nghiệm của nhà phân tích, đó chính là chọn K là bao nhiêu.

Để xác định độ “gần nhau” giữa hai mẫu dữ liệu, người ta sử dụng một độ đo khoảng cách được định nghĩa trước. Tùy theo kiểu dữ liệu của mẫu và đặc điểm của đối tượng nhận dạng mà ta sử dụng một độ đo phù hợp. Có rất nhiều độ đo khoảng cách (hay độ khác biệt) đã được định nghĩa.

Xét một mẫu dữ liệu *x* gồm *m* thuộc tính. Khi đó, mẫu *x* được xem là một véc tơ trong không gian *m* chiều (*x* có *m* thành phần). Gọi *x(x1, x2,…, xm)* và *y(y1, y2,…, ym)* là hai mẫu dữ liệu. Để tính khoảng cách giữa *x* và *y*, ký hiệu *d(x, y)* ta thường sử dụng một số độ đo sau:

Với dữ liệu kiểu số

**- Khoảng cách Euclidean:**

(3.14)



Dễ thấy, khoảng cách này chính là chuẩn 2 của *x-y*: *d(x, y)* = ||*x-y*||.

**- Khoảng cách Square Euclidean:**

(3.15)



Dễ thấy, khoảng cách này chính là bình phương của khoảng cách Euclidean: *d(x,y)*=||*x-y*||2.

**- Khoảng cách Manhattan:**



(3.16)

Khoảng cách này chính là chuẩn 1 của *x-y*: *d(x, y)* = ||*x-y*||1.

**- Khoảng cách Chebyshev:**

(3.17)



Khoảng cách này chính là chuẩn vô cùng của *x-y:* *d(x, y)* = ||*x-y*||∞.

**- Khoảng cách Cosin:**



(3.18)

Khoảng cách này được hiểu là:

Khi sử dụng phương pháp này, ta cần lưu ý một số điểm sau:

- Số láng giềng gần nhất (k) được coi như đầu vào của thuật toán, tức là cần chọn một giá trị cho k trước khi thực hiện thuật toán. Hiển nhiên là số k được chọn sẽ có thể ảnh hưởng tới chất lượng của giải thuật. Một câu hỏi đặt ra là: với một bộ dữ liệu cho trước, số k được chọn như thế nào để cho kết quả thuật toán tốt nhất? Câu hỏi này mở ra một lĩnh vực nghiên cứu sôi động và các kết quả hầu như đều chưa đưa ra được câu trả lời thỏa đáng. Nói chung, giá trị của k phụ thuộc mạnh vào bộ dữ liệu huấn luyện mà ta có. Một số phương pháp phổ biến để xác định số k có thể kể tới như cross-validation hoặc bootstrapping…Chúng thực chất là phương pháp thử với nhiều giá trị k khác nhau và lựa chọn ra một số k tối ưu đối với một bộ dữ liệu huấn luyện. Trong một số ứng dụng, người ta sử dụng số k mặc định bằng 1.

- Độ đo khoảng cách cần phải phù hợp với dữ liệu hiện có. Hiển nhiên, độ đo này phải phù hợp với kiểu của dữ liệu. Ngoài ra, với một kiểu dữ liệu cụ thể, lại có thể có nhiều độ đo khoảng cách cho kiểu dữ liệu đó. Lựa chọn độ đo nào trong số các độ đo này là tùy thuộc vào bài toán cụ thể.