Chương 2:

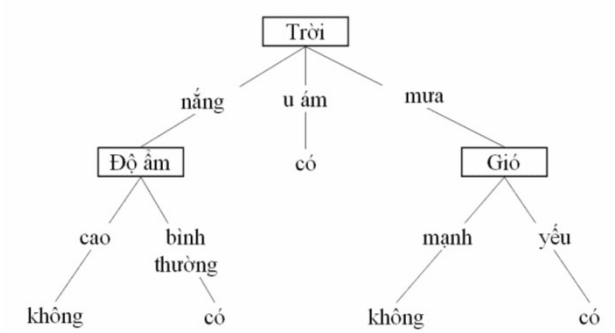
Học máy và Hệ thống thông minh

Nội dung

- Học máy cổ điển
- Học sâu
- Đánh giá mô hình
- Hệ thống dựa trên học máy
- Notebook và thư viện Scikit-learn

Học máy cổ điển

- Cây quyết định



Học máy cổ điển

- Cây quyết định
  - Các độ đo dùng trong phân lớp bằng cây quyết định:
  - Entropy: Entropy dùng trong thông tin là một khái niệm mở rộng của entropy trong Nhiệt động lực học và Cơ học thống kê. Entropy mô tả mức độ hỗn loạn trong một tín hiệu lấy từ một sự kiện ngẫu nhiên.

$$Entropy = \sum_j -p_j log_2 p_j$$

Trong đó: p_i là xác suất xuất hiện một thông tin trong tập dữ liệu.

Học máy cổ điển

- Cây quyết định
  - Các độ đo dùng trong phân lớp bằng cây quyết định:
  - Information Gain (Độ lợi thông tin): Là độ sai biệt giữa trị thông tin trước phân hoạch (Info(D)) và trị thông tin sau phân hoạch với A ($Info_A(D)$).

    $Gain(A) = Info(D) - Info_A(D)$

Học máy cổ điển

- Cây quyết định

```
procedure Build_tree(Records, Attributes)
begin
    Tạo nút N;
    if (tất cả các bản ghi thuộc về một lớp Ci nào đó) then
    begin
        N.Label = Ci;
        return N;
    end;
    if (Attributes = ∅) then
    begin
        Tìm lớp Cj mà phần lớn các bản ghi r ∈ Records thuộc về lớp đó.
        N.Label = Cj;
        return N;
    end;
    Chọn Ai ∈ Attribute sao cho Gain(Ai)→max;
    N.Label = Ai;
    for each giá trị vj đã biết của Ai do
    begin
        Thêm một nhánh mới vào nút N ứng với Ai = vj;
        Sj = Tập con của Records có Ai = vj;
        if (Sj = ∅) then
                Thêm một nút lá L với nhãn là lớp mà phần lớn các bản ghi r ∈ Records thuộc về lớp đó;
                Return L;
        else
                Thêm vào nút được trả về bởi Build_Tree(Sj, Attribute \{Ai});
    end ;
end;
```


Học máy cổ điển

- Cây quyết định
  - Thuộc tính liên tục
  - SplitInfomation: Thông tin tiềm ẩn được tạo ra bằng cách chia tập dữ liệu trong một số tập con nào đó.

$$SplitInformation(S,A) = \sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

  - ✓ S_i là tập con của S chứa các thể hiện của thuộc tính A mang giá trị V_i.
  - ✓ Splitinfomation thực sự chính là Entropy của S với sự liên quan trên những giá trị của thuộc tính A

Học máy cổ điển

- Cây quyết định
  - Thuộc tính liên tục:
  - GainRatio*:* Đánh giá sự thay đổi các giá trị của thuộc tính.

$$GainRatio(S,A) = \frac{Gain(S,A)}{SplitInformation(S,A)}$$

  - Tất cả các thuộc tính sẽ được tính toán độ đo tỷ lệ Gain, thuộc tính nào có độ đo tỷ lệ Gain lớn nhất sẽ được chọn làm thuộc tính phân chia.

Học máy cổ điển

- Cây quyết định
  - Các bước tính:
    1. Tính Entropy,
    2. Tính Gain,
    3. Tính SplitInformation,
    4. Tính GainRatio,
    5. Tính Entropy trung bình,
    6. So sánh các Entropy với Entropy trung bình + so sánh GainRatio để chọn thuộc tính phân tách.

9

Học máy cổ điển

- Cây quyết định
  - Vấn đề quá vừa dữ liệu (Overfitting):
    - Giải pháp hạn chế overfitting:
    - ✓ Dừng việc dựng cây quyết định sớm.
    - ✓ Xây dựng cây đầy đủ sau đó tỉa để cây trở nên đơn giản.
    - Tỉa cây – thuật toán ID3:
    - ✓ Dùng tập huấn luyện để dựng cây đầy đủ. Sâu đó xem xét tỉa dần các nút.
    - ✓ Với nút được tỉa:
    - Toàn bộ các nhánh bên dưới sẽ bị bỏ đi.
    - Nút này trở thành lá.
    - Nhã của nút lá mới lấy theo đa số nhãn của các mẫu tại nút đó
    - ✓ Nút sẽ được tỉa nếu độ chính xác sau khi tỉa không giảm.
    - ✓ Dừng tỉa khi không tìm được nút nếu tỉa sẽ tăng chính xác

10

Học máy cổ điển

- Cây quyết định
  - Vấn đề quá vừa dữ liệu (Overfitting):
    - Tỉa luật:
    - ✓ Thuật toán C4.5.
    - ✓ Xây dựng cây cho phép phân loại đúng tối đa tập huấn luyện.
    - ✓ Biến đổi cây thành luật suy diễn sao cho mỗi nhánh từ gốc đến lá tương ứng một luật
    - ✓ Tỉa từng luật bằng cách bỏ bớt điều kiện thành phần nếu sau khi bỏ độ chính xác tăng lên.
    - ✓ Sắp xếp các luật đã được tỉa theo độ chính xác trên tập kiểm tra.
    - ✓ Sử dụng luật theo thứ tự đó để phân loại mẫu mới.

11

Học máy cổ điển

- Phương pháp Naïve Bayes
  - ✓ Dùng trong phân loại dữ liệu,
  - ✓ Là trường hợp riêng của kỹ thuật học máy Bayes, với việc giả thiết về sự độc lập xác suất để đơn giản hóa việc tính xác suất.
  - ✓ Do dựa trên xác suất nên dữ liệu huấn luyện cần một số lượng tương đối lớn
  - ✓ Bộ phân loại Naïve Bayes như sau:

$$classify(f_1,...,f_n) = \arg\max_c P(C=c)\prod_{i=1}^{n} P(F_i = f_i \mid C=c)$$

  - f_1, f_2 … f_n: các mẫu cần phân lớp

12

Học máy cổ điển

- Phương pháp Naïve Bayes
  - ✓ Dùng trong phân loại dữ liệu,
  - ✓ Là trường hợp riêng của kỹ thuật học máy Bayes, với việc giả thiết về sự độc lập xác suất để đơn giản hóa việc tính xác suất.
  - ✓ Do dựa trên xác suất nên dữ liệu huấn luyện cần một số lượng tương đối lớn
  - ✓ Bộ phân loại Naïve Bayes như sau:

$$classify(f_1,...,f_n) = \arg\max_c P(C=c)\prod_{i=1}^{n} P(F_i = f_i \mid C = c)$$

  - f_1, f_2 … f_n: các mẫu cần phân lớp

13

Học máy cổ điển

- **Phương pháp Naïve Bayes**:
  - ✓ Trong bộ phân loai Naïve Bayes, nếu một thành phần $P(x_i|c_j) = 0$ thì xác suất điều kiện cuố cùng sẽ bằng không.
  - ✓ Ta có: $P(x_i|c_j) = \frac{n_c}{n}$
    - n_c là số lần x_i và c_j xuất hiện đồng thời trong tập mẫu
    - n là số lần c_j xuất hiện
  - ➢ Để $P(x_i|c_j) \neq 0$, với n rất lớn, ta có thể tính $P(x_i|c_j)$ như sau: $P(x_i|c_j) = \frac{n_c+1}{n+1}$

14

Học máy cổ điển

- Phương pháp Naïve Bayes
  - ✓ Tổng quát $P(x_i|c_j) = 0$ được xác định như sau:

$$P(x_i|c_j) = \frac{n_c + m\rho}{n + m}$$

    - ρ là xác suất tiền nhiệm của x_i, ρ = 1/k, k là số thuộc tính của x_i.
    - m là tham số cho phép xác định ảnh hưởng của ρ tới công thức

15

Học máy cổ điển

- Thuật giải kNN (k Nearest Neighbors):
  - ➢ Là thuật toán thuộc loại phân lớp dữ liệu
  - ➢ Với một mẫu mới cần phân lớp, kNN sẽ dựa vào các thuộc tính và lớp của các mẫu có sẵn (training data).
  - ➢ Mẫu mới được phân lớp dựa vào k láng giềng của nó
  - ➢ k là số nguyên dương được xác định trước.
  - ➢ Yếu tố "láng giềng" được xác định bằng các phép tính khoảng cách như Euclidean …

16

Học máy cổ điển

- Thuật giải kNN (k Nearest Neighbors):
  - Tính khoảng cách giữa hai vector:
    - Xét hai đối tượng dữ liệu (bản ghi) r_i và r_j , mỗi đối tượng có n thuộc tính:

$r_i = (x_{i1}, x_{i2}, \ldots, x_{in})$,

$r_j = (x_{j1}, x_{j2}, \ldots, x_{jn})$,

  - Khoảng cách Euclidean

$$d(r_i, r_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \ldots + (x_{in} - x_{jn})^2}$$

  - Khoảng cách Manhattan

$$d(r_i, r_j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{in} - x_{jn}|$$

17

Học máy cổ điển

- Thuật giải kNN (k Nearest Neighbors):
  1. Xác định tham số **K**
  2. Tính khoảng cách giữa đối tượng cần phân lớp (query point) với tất cả các đối tượng trong training data
  3. Sắp xếp khoảng cách theo thứ tự tăng dần và xác định **K** láng giềng gần nhất với query point.
  4. Lấy tất cả các lớp của **K** láng giềng gần nhất đã xác định
  5. Dựa vào phần lớn lớp của láng giềng gần nhất để xác định lớp cho query point

18

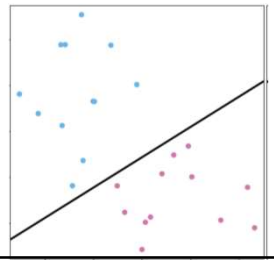
Học máy cổ điển

- Support Vector Machine (SVM)
  - Được đề xuất bởi V. Vapnik và các đồng nghiệp vào những năm 1970 ở Nga, và trở nên nổi tiếng vào những năm 1990.
  - Phương pháp học phân loại có giám sát: Bài toán phân loại 2 lớp.
  - Khái quát hóa bộ phân lớp với lề cực đại (maximal margin classifier).
  - Có thể áp dụng với dữ liệu không tách được tuyến tính.

19

Học máy cổ điển

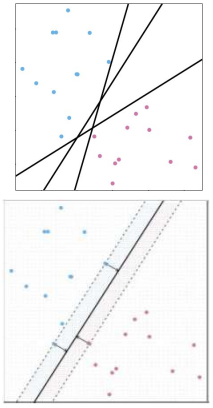
- Support Vector Machine (SVM)
  - Bộ phân lớp có lề cực đại
  - ✓ SVM là một phương pháp **phân lớp tuyến tính** (linear classifier), với mục đích xác định một siêu phẳng để phân tách **hai lớp** của dữ liệu.
  - Mặt siêu phẳng phân tách
  - ✓ Giả định: các lớp có thể tách được tuyến tính
  - ✓ Ý tưởng: Dùng mặt siêu phẳng tách hai lớp có trong dữ liệu



20

Học máy cổ điển

- Support Vector Machine (SVM)
  - Mặt siêu phẳng phân tách
  - ✓ Có thể tồn tại nhiều mặt phẳng phân tách.
  - ✓ Tiêu chí chọn mặt phẳng phân tách?
  - ✓ Chọn mặt phân tách xa nhất từ tập dữ liệu huấn luyện
  - ✓ Bộ phân lớp có lề cực đại

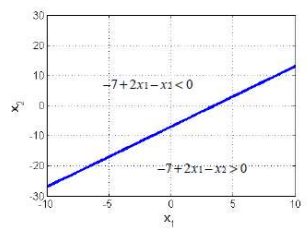


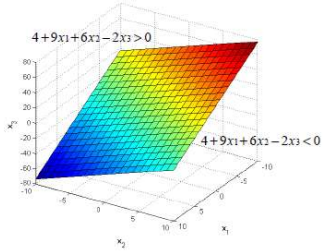
21

Học máy cổ điển

- Support Vector Machine (SVM)
  - Ví dụ về mặt siêu phẳng:

$-7+2x_1-x_2=0$ $\quad$ $4+9x_1+6x_2-2x_3=0$

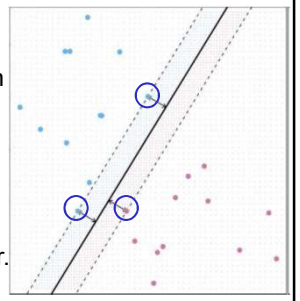




22

Học máy cổ điển

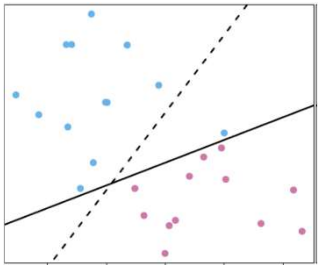
- Support Vector Machine (SVM)
  - Siêu phẳng có lề cực đại:
  - Siêu phẳng xa nhất từ tập huấn luyện cực đại lề.
  - Các vector thuộc hai đường thẳng có khoảng cách đến siêu phẳng bằng cực đại lề - siêu phẳng lề - gọi là Support Vector.



23

Học máy cổ điển

- Support Vector Machine (SVM)
  - Nhược điểm của bộ phân lớp có lề cực đại trong SVM:
    - Có thể bị overfit trên dữ liệu huấn luyện
    - Nhạy cảm với các mẫu độc lập



24

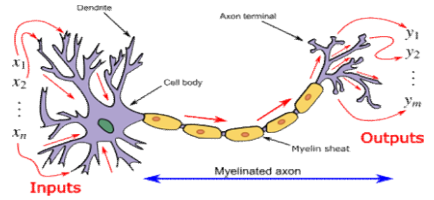
Học máy cổ điển

- Support Vector Machine (SVM)
  - Đây là phương pháp có cơ sở lý thuyết tốt dựa trên lý thuyết về khả năng khái quát hóa cả bộ phân loại.
  - Có thể làm việc tốt với dữ liệu nhiều chiều (nhiều thuộc tính)
  - Cho kết quả rất chính xác so với các phương pháp khác trong hầu hết ứng dụng

25

Neural Network

- Mạng nơ ron nhân tạo (Artificial Neural Network – ANN)
- Dựa trên nguyên lý hệ thống thần kinh (neural) của người và động vật bậc cao: sử dụng nhiều nút được nối với nhau thành một mạng lưới.

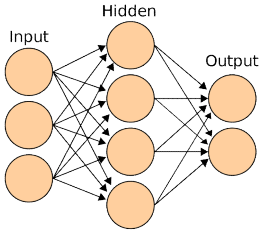


- Tín hiệu được truyền từ nút nọ sang nút kia tùy thuộc mức tín hiệu và cơ chế xử lý tại mỗi nút.

26

Neural Network

- Mạng nơ ron nhân tạo (Artificial Neural Network – ANN)
- Mô hình mạng nhân tạo

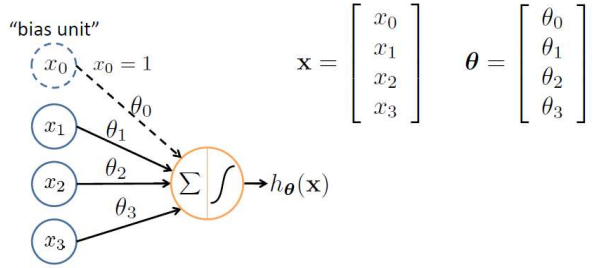


- Tại mỗi nút có một hàm cho phép trả ra kết quả theo tín hiệu đầu vào

27

Neural Network

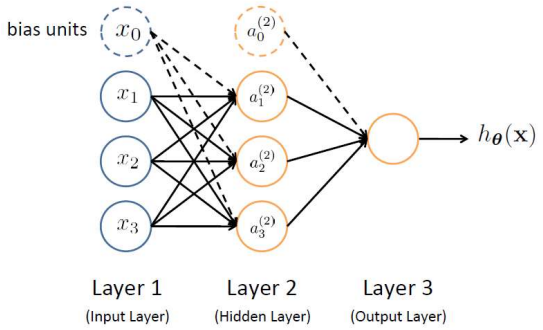
- Mạng nơ ron nhân tạo (Artificial Neural Network – ANN)



28

Neural Network

- Mạng nơ ron nhân tạo (Artificial Neural Network – ANN)



29

Neural Network

- Mạng nơ ron nhân tạo (Artificial Neural Network – ANN)
- Khi sử dụng nhiều nút với những hàm xử lý riêng tại từng nút ⇒ có thể tạo những hàm đích rất phức tạp để ánh xạ tín hiệu đầu vào thành kết quả đầu ra.
- Quá trình học là quá trình hiệu chỉnh tham số của hàm xử lý tại các nút sao cho ánh xạ đầu vào sang đầu ra phù hợp nhất với dữ liệu huấn luyện.
- Đây là phương pháp học máy được phát triển từ khá sớm.

30

Neural Network

- Trọng số được điều chỉnh tại mỗi bước lặp theo nguyên tắc:

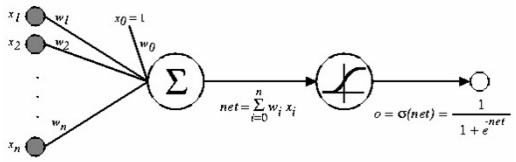
$$w_i \leftarrow w_i + \Delta w_i$$

với: $\Delta w_i = \eta(t - o)\, x_i$

- Trong đó:
  - t: đầu ra mục tiêu của hệ thống đang huấn luyện
  - o: đầu ra của Perceptron
  - η: hệ số học (*learning rate*)

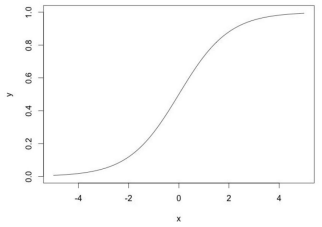
31

Neural Network



$\sigma(x)$ is the sigmoid function

$$\frac{1}{1 + e^{-x}}$$



32

Neural Network

- Các bước tính toán:

1. Với mỗi output *k*

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k) \quad (1)$$

2. Với mỗi node ẩn *h*

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{kh}\delta_k \quad (2)$$

3. Cập nhật lại trọng số w_{ji} của mạng

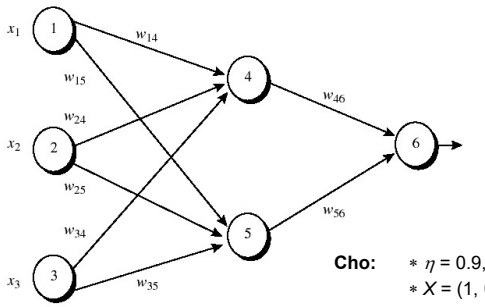
$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

với

$$\Delta w_{ji} = \eta\, \delta_j x_{ji} \quad (3)$$

33

Neural Network

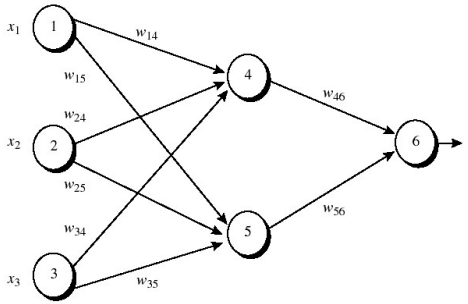


Cho:
- $\eta = 0.9$,
- $X = (1, 0, 1)$,
- Nhãn: 1.

Dùng hàm sigmoid cho tầng ẩn và đầu ra.

34

Neural Network



Bảng 1: Khởi tạo đầu vào các trọng số:

x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	w_{04}	w_{05}	w_{06}
1	0	1	0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1

35

Neural Network

Bảng 2: Tính input và output

j	Input I_j	Output O_j
4	1x0.2 + 0x0.4 + 1x(-0.5) -0.4 = -0.7	$1/(1+e^{0.7})=0.332$
5	1x(-0.3) + 0x(0.1) + 1x0.2 + 0.2 = 0.1	$1/(1+e^{-0.1})=0.525$
6	(-0.3)(0.332)+(-0.2)(0.525)+0.1 = -0.105	$1/(1+e^{0.105})=0.474$

Bảng 3: tính sai số ở mỗi node

j	δ_j
6	(0.474)(1-0.474)(1-0.474)=0.1311
5	(0.525)(1-0.525)(0.1311)(-0.2)=-0.0065
4	(0.332)(1-0.332)(0.1311)(-0.3)=-0.0087

36

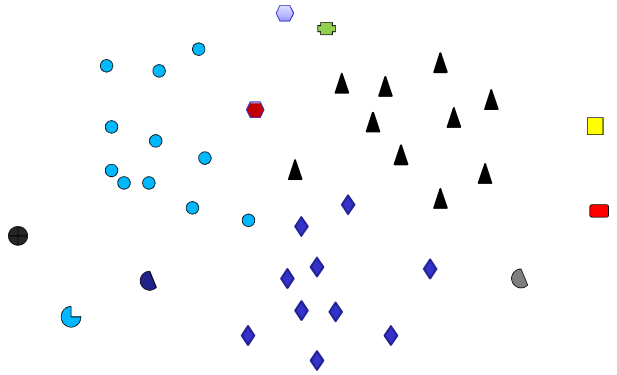
Neural Network

Bảng 4: tính trọng số cập nhật cho các node

Weight	New value
w_{46}	-0.3+(0.9)(0.1311)(0.332)= -0.261
w_{56}	-0.2+(0.9)(0.1311)(0.525)= -0.138
w_{14}	0.2 +(0.9)(-0.0087)(1) = 0.192
w_{15}	-0.3 +(0.9)(-0.0065)(1) = -0.306
w_{24}	0.4+ (0.9)(-0.0087)(0) = 0.4
w_{25}	0.1+ (0.9)(-0.0065)(0) = 0.1
w_{34}	-0.5+ (0.9)(-0.0087)(1) = -0.508
w_{35}	0.2 + (0.9)(-0.0065)(1) = 0.194
w_{06}	0.1 + (0.9)(0.1311) = 0.218
w_{05}	0.2 + (0.9)(-0.0065)=0.194
w_{04}	-0.4 +(0.9)(-0.0087) = -0.408

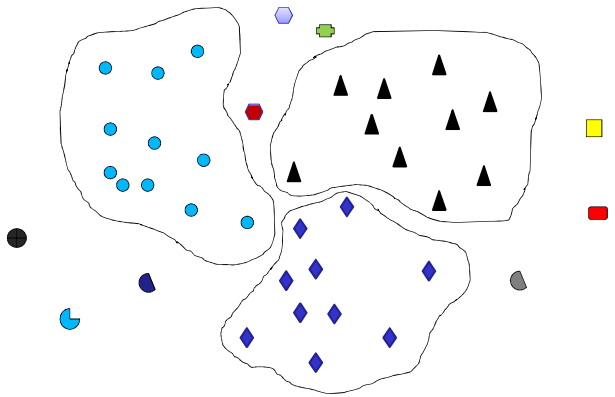
37

Gom cụm dữ liệu



38

Gom cụm dữ liệu



39

Độ đo trong gom cụm dữ liệu

Xét hai đối tượng dữ liệu (bản ghi) r_i và r_j , mỗi đối tượng có n thuộc tính:

$r_i = (x_{i1}, x_{i2}, \ldots, x_{in})$,

$r_j = (x_{j1}, x_{j2}, \ldots, x_{jn})$,

- Khoảng cách Euclidean

$$d(r_i, r_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \ldots + (x_{in} - x_{jn})^2}$$

- Khoảng cách Manhattan

$$d(r_i, r_j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{in} - x_{jn}|$$

40

Độ đo trong gom cụm dữ liệu

Trọng tâm cụm (mean/centroid):

Cụm *C* có m phần tử; mỗi phần tử có n thuộc tính:

$C = \{r_1, r_2, \ldots, r_m\}$,

$R_i = (x_{i1}, x_{i2}, \ldots, x_{in})$.

Trọng tâm *m* của cụm *C* xác định như sau:

$$m_j = \frac{1}{n}\left(\sum_{i=1}^{m} x_{i1}, \sum_{i=1}^{m} x_{i2}, \ldots, \sum_{i=1}^{m} x_{in}\right)$$

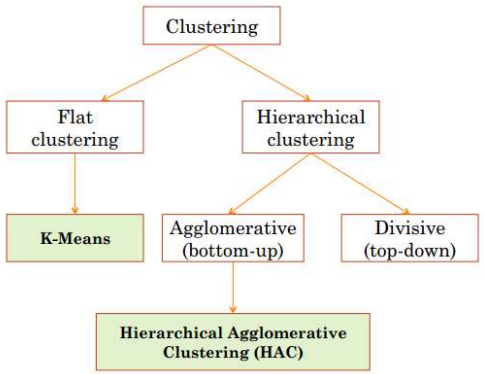
41

Một số thuật giải gom cụm dữ liệu

- Hierarchical Agglomerative Clustering (HAC)
  - Single Link
  - Complete Link
  - Centroid
  - Group Average
- K-means

42

Một số thuật giải gom cụm dữ liệu



43

Một số thuật giải gom cụm dữ liệu

- Giải thuật K-means
  - Là phương pháp thuộc nhóm phân cụm phân vùng (phân cụm phẳng).
  - Xây dựng từng bước phân hoạch các cụm và đánh giá chúng theo các tiêu chí tương ứng.
  - Các tính toán dựa trên độ đo tương tự / khoảng cách

44

Một số thuật giải gom cụm dữ liệu

- Giải thuật K-means

Input: Tập dữ liệu D gồm m đối tượng dữ liệu (bản ghi): r_1, r_2,…, r_m. Số lượng cụm k.

Output: k cụm dữ liệu.

Begin

Chọn ngẫu nhiên k đối tượng làm trọng tâm cho k cụm;

Repeat

- ✓ Gán mỗi đối tượng ri cho cụm mà khoảng cách từ đối tượng đến trọng tâm cụm là nhỏ nhất trong số k cụm;
- ✓ Xác định lại trọng tâm cho mỗi cụm dựa trên các đối tượng được gán cho cụm;

Until Hội tụ (không còn sự thay đổi);

End;

Một số thuật giải gom cụm dữ liệu

- Giải thuật K-means – Điều kiện dừng:
  - Giải thuật hội tụ: không còn sự phân chia lại các đối tượng giữa các cụm, hay trọng tâm các cụm là không đổi. Lúc đó tổng các tổng khoảng cách từ các đối tượng thuộc cụm đến trọng tâm cụm là cực tiểu:

$$J = \sum_{j=1}^{k} \sum_{r_i \in C_j} d(r_i, m_j) \rightarrow \min$$

Một số thuật giải gom cụm dữ liệu

- Giải thuật K-means – Điều kiện dừng:
  - Giải thuật không hội tụ: trọng tâm của các cụm liên tục thay đổi. Khi này có các lựa chọn:
    - ✓ Dừng giải thuật khi số lượng vòng lặp vượt quá một ngưỡng nào đó định trước.
    - ✓ Dừng giải thuật khi giá trị J nhỏ hơn một ngưỡng nào đó định trước.
    - ✓ Dừng giải thuật khi hiệu giá trị của J trong hai vòng lặp liên tiếp nhỏ hơn một ngưỡng nào đó định trước: $|J_{n+1} - J_n| < \varepsilon$

Thuật giải K-means

- Phân dữ liệu sau thành 2 cụm (K=2).

Medicine	Weight	pH-Index
A	1	1
B	2	1
C	4	3
D	5	4

Thuật giải K-means

Bước 1: Chọn tâm ban đầu $c_1 = A, c_2 = B$

- ✓ Dùng công thức tính khoảng cách (Euclidean) để lần lượt tính khoảng cách từ các tâm đến từng đối tượng.
- ✓ Gán đối tượng vào cụm mà khoảng cách từ đối tượng đến tâm là gần hơn

$$d(D,c_1) = \sqrt{(5-1)^2 + (4-1)^2} = 5$$
$$d(D,c_2) = \sqrt{(5-2)^2 + (4-1)^2} = \sqrt{18}$$
$$\rightarrow D \in \{B\}$$
$$d(C,c_1) = \sqrt{(4-1)^2 + (3-1)^2} = \sqrt{13}$$
$$d(C,c_2) = \sqrt{(4-2)^2 + (3-1)^2} = \sqrt{8}$$
$$\rightarrow C \in \{B\}$$

Bước 2: Tính lại tâm mới của cụm

Bước 3: Lặp lại các Bước 1 và Bước 2

Thuật giải HAC (Hierarchical Agglomerative Clustering)

- ➢ Là giải thuật phân cụm theo mô hình phân cấp
- ➢ Xây dựng hợp (tách) dần các cụm tạo cấu trúc phân cấp và đánh giá theo các tiêu chí tương ứng
- ➢ Các tính toán dựa trên độ đo tương tự / khoảng cách.
- ➢ Không cho trước số lượng cụm k, cho phép đưa ra các phương án phân cụm theo các giá trị k khác nhau

Thuật giải HAC (Hierarchical Agglomerative Clustering)

Ý tưởng: tích lũy từ dưới lên

1. Ban đầu, mỗi đối tượng (bản ghi) dữ liệu được coi là một cụm.
2. Từng bước kết hợp các cụm đã có thành các cụm lớn hơn với yêu cầu là khoảng cách giữa các đối tượng trong nội bộ cụm là nhỏ.
3. Dừng thuật toán khi đã đạt số lượng cụm mong muốn, hoặc chỉ còn một cụm duy nhất chứa tất cả các đối tượng hoặc thỏa mãn điều kiện dừng nào đó.

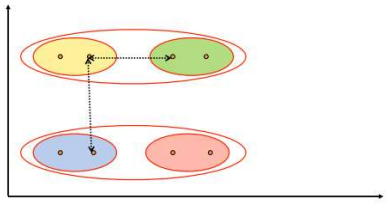
Thuật giải HAC (Hierarchical Agglomerative Clustering)

G: tập các cụm.
D: tập các đối tượng (bản ghi) dữ liệu cần phân cụm.
k: số lượng cụm mong muốn.
d_0: ngưỡng khoảng cách giữa 2 cụm.

1. G = {{r} | r ∈ D}; *//Khởi tạo G là tập các cụm chỉ gồm 1 đối tượng*
2. Nếu |G| = k thì dừng thuật toán; *//Đạt số lượng cụm mong muốn*
3. Tìm hai cụm S_i , $S_j \in G$ có khoảng cách $d(S_i, S_j)$ là nhỏ nhất;
4. Nếu $d(S_i, S_j) > d_0$ thì dừng thuật toán; *//Khoảng cách 2 cụm gần nhất đã lớn hơn ngưỡng cho phép*
5. $G = G \setminus \{S_i, S_j\}$; *//Loại bỏ 2 cụm S_i ,S_j khỏi tập các cụm*
6. $S = S_i \cup S_j$; *//Ghép S_i, S_j thành cụm mới S*
7. $G = G \cup \{S\}$; *//Kết nạp cụm mới vào G*
8. Quay về bước 2.

Thuật giải HAC (Hierarchical Agglomerative Clustering)

- Single Link (đo khoảng cách gần nhất):
  - Khoảng cách giữa hai cụm được xác định là khoảng cách giữa hai phần tử "gần" nhau nhất của hai cụm đó.

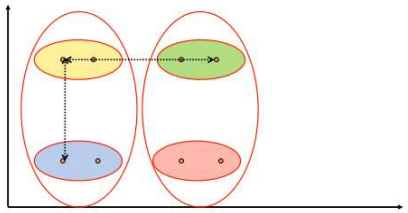


$$d(S_1, S_2) = \min_{r_i \in S_1, r_j \in S_2} d(r_i, r_j)$$

53

Thuật giải HAC (Hierarchical Agglomerative Clustering)

- Complete Link (đo khoảng cách xa nhất):
  - Khoảng cách giữa hai cụm được xác định là khoảng cách giữa hai phần tử "xa" nhau nhất của hai cụm đó

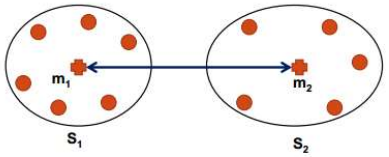


$$d(S_1, S_2) = \max_{r_i \in S_1, r_j \in S_2} d(r_i, r_j)$$

54

Thuật giải HAC (Hierarchical Agglomerative Clustering)

- Centroid Link (đo khoảng cách trọng tâm):
  - Khoảng cách giữa hai cụm được xác định là khoảng cách giữa hai trọng tâm của hai cụm đó

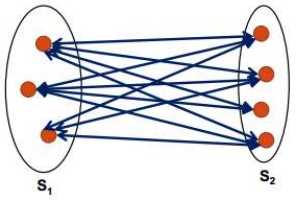


$$d(S_1, S_2) = d(m_i, m_j)$$

55

Thuật giải HAC (Hierarchical Agglomerative Clustering)

- Group Average Link (đo khoảng cách trung bình nhóm):
  - Khoảng cách giữa hai cụm được xác định là khoảng cách trung bình giữa các phần tử thuộc về hai cụm đó



$$d(S_1, S_2) = \frac{1}{|S_1| \, |S_2|} \sum_{r_i \in S_1, r_j \in S_2} d(r_i, r_j)$$

56