HỌC MÁY

I. Thuật toán học có giám sát-Supervised Learning *(dữ liệu học gồm đầu vào và đầu ra tương ứng)*

1. Hồi quy tuyến tính *(nhãn có giá trị liên tục):*

- là một phương pháp thống kê để hồi quy dữ liệu với *biến phụ thuộc (Y) có giá trị* ***liên tục*** trong khi các biến độc lập có thể có một trong hai giá trị liên tục hoặc là giá trị phân loại. Nói cách khác "Hồi quy tuyến tính" là một phương pháp để dự đoán *biến phụ thuộc (Y)* dựa trên giá trị của biến độc lập (X). Nó có thể được sử dụng cho các trường hợp chúng ta muốn dự đoán một số lượng liên tục.
- Input: tập dữ liệu được gán nhãn.
- Output: tìm ra các hệ số của đg tuyến tính (hệ số w là số thực không bị chặn)
- Cách thực hiện:

$$x = \begin{bmatrix} x1 \\ x2 \\ \dots \\ xn \end{bmatrix}$$: vector đặc trưng chứa dữ liệu đầu vào.

$$w = \begin{bmatrix} w1 \\ w2 \\ \dots \\ wn \end{bmatrix}$$: vector trọng số.

+ Xây dựng hàm mất mát:
  - Với tất cả các cặp dữ liệu quan sát được (xi,yi), i = 1, 2, …, N, chúng t muốn trung bình sai số là nhỏ nhất.
  - N: tổng số mẫu dữ liệu
  - x_i : vecter dữ liệu
  - y_i: nhãn tương ứng với từng vecter dữ liệu.

  ⇨ Tìm w để hàm mất mát nhỏ nhất:

$$\mathcal{L}(w) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - x_i^T w)^2$$

  ⇨ Cần tối thiểu hàm mất mát theo w. Nghiệm cần tìm của bài toán là:

$$w^* = \underset{w}{\operatorname{argmin}}\, \mathcal{L}(w)$$

- Đặt $y = [y1,y2,\ldots,yn]^T$, $X = [x1,x2, \ldots, xn]$, $x = = \begin{bmatrix} x1 \\ x2 \\ \ldots \\ xm \end{bmatrix}$
- Hàm mất mát được viết lại như sau:

$$\mathcal{L}(w) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - x_i^T w)^2 = \frac{1}{2N}\left\| \begin{bmatrix} y_1 \\ y_2 \\ \ldots \\ y_N \end{bmatrix} - \begin{bmatrix} x_1^T \\ x_2^T \\ \ldots \\ x_N^T \end{bmatrix} w \right\|^2 = \frac{1}{2N}\|y - X^T w\|_2^2$$

+ Tìm nghiệm của bài toán tối ưu:
Tìm giá trị tối ưu của w bằng cách giải phương trình đạo hàm của hàm mất mát theo w bằng 0.

Đạo hàm theo w của hàm $\mathcal{L}(w)$ là:

$$\frac{\nabla \mathcal{L}(w)}{\nabla w} = \frac{1}{N} X(X^T w - y)$$

Giải phương trình đạo hàm bằng không:

$$\frac{\nabla \mathcal{L}(w)}{\nabla w} = 0 \Leftrightarrow XX^T w = Xy$$

Nếu ma trận XX^T khả nghịch thì phương trình trên có nghiệm duy nhất là

$$w = (XX^T)^{-1}Xy$$

Nếu ma trận XX^T không khả nghịch thì nghiệm của phương trình có thể xác định dựa vào giả nghịch đảo

$$w = (XX^T)^{\dagger}Xy$$

2. Phân loại – Classification *(nhãn có giá trị không liên tục)*

2.1 Perceptron *(phân loại trong TH nhãn chỉ có 2 lớp)*

- Là thuật toán phân lớp đơn giản giúp tìm ra siêu phẳng cho bài toán phân lớp nhị phân (chỉ có 2 lớp dữ liệu).
- Input: Là một tập dữ liệu đã được gán nhãn
- Output: là 1 siêu phẳng phân tách 2 lớp
- Cách xây dựng hàm mất mát, và tìm nghiệm bài toán:
  - PT đường boundary:

$$f_w(x) = w_1.x_1 + w_2.x_2 + \ldots + w_d.x_d + w_0 = 0$$
$$f_w(x) = w^T.x = 0$$

    - $w = (w_0, w_1,\ldots, w_d)$: vector hệ số
    - w_0 : số hạng tự do
    - $x = (x_0, x_1,\ldots, x_d)$: vector đặc trưng

- Hàm mất mát:

$$J(\mathrm{w}) = \sum_{\mathrm{x}_i \in M} (-y_i \ \mathrm{w}^T \mathrm{x}_i)$$

  - M: số điểm phân lớp lỗi

- Nghiệm của bài toán w: tìm w sao cho hàm mất mát đạt giá trị nhỏ nhất là 0 (không có điểm dữ liệu nào bị phân lớp sai)
  - Với mỗi điểm phân lớp đúng -> kết thúc thuật toán
  - Với mỗi điểm phân lớp sai nghiệm w được cập nhật lại:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + y_i \mathbf{x}_i$$

- Phương pháp:
  B1: tại thời điểm t=0, chọn ngẫu nhiên 1 vector hệ số w_0
  B2:
  - tại thời điểm t:
    - nếu không có điểm dữ liệu nào bị phân lớp lỗi thì dừng thuật toán.
    - nếu có điểm bị phân lớp lỗi thì cập nhật lại giá trị của w.
      $w_{t+1} = w_t + y_i.x_i$

  B3: thay đổi t = t+1 quay lại bc 2

2.2 ID3

- ID3 là một thuật toán decision tree được áp dụng cho các bài toán classification mà tất cả các thuộc tính đều ở dạng categorical.
- Innput: Là một tập dữ liệu đã được gán nhãn.
- Output: là cây quyết định
- Công việc thực hiện của bài toán:
  + Sử dụng hàm "entropy" làm thước đo trong khi tạo cây quyết định (decision tree) trong thuật toán ID3 (Classification and Regression Tree)
- Các bước thực hiện:

B1: tính entropy trên toàn bộ tập S

- H(S) : entropy trên toàn bộ tập dữ liệu S
- S: tập dữ liệu huấn luyện(gồm cả x và y)
- N: tổng số mẫu dữ liệu huấn luyện
- Nc: số các mẫu dữ liệu trong tập S đc gán nhãn c
- C: tổng số nhãn khác nhau trong tập S
- c: chỉ số của nhãn dữ liệu

$$H(S) = -\sum_{c=1}^{C} \frac{N_c}{N} \log\left(\frac{N_c}{N}\right)$$

B2: Tính entropy của thuộc tính x trên tập S

- x: là thuộc tính đang xét
- K: số giá trị của thuộc tính đó
- m_k: số mẫu có thuộc tính x và có giá trị là k
- S_k: tập các mẫu sao cho thuộc tính x có giá trị là k
- $H(S_k)$: entropy trên tập dữ liệu trong S có thuộc tính là x và có giá trị k

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

B3: tính *information gain* trên thuộc tính x

$$G(x,\mathcal{S}) = H(\mathcal{S}) - H(x,\mathcal{S})$$

- Thay vì tìm thuộc tính có information gain cao nhất thì ta tìm thuộc tính có entropy nhỏ nhất.

$$x^* = \arg\max_x G(x,\mathcal{S}) = \arg\min_x H(x,\mathcal{S})$$

B4: Chia dữ liệu vào các nút con tương ứng với các giá trị của thuộc tính, thực hiện lặp tìm ra thuộc tính tốt nhất trong tập dữ liệu mới.

❖ **Điều kiện dừng:**

– TH1: nếu tất cả các mẫu trong node con thuộc cùng 1 lớp C (entropy=0) thì node đó đc gán nhãn C

– TH2: nếu node con là rỗng (không có dữ liệu) thì node đó sẽ đc gán = nhãn phổ biến nhất trong tập S

– TH3: nếu không còn thuộc tính nào để phân chia (các thuộc tính đều đc xét hết) -> node lá đó sẽ đc gán = nhãn phổ biến nhất

2.3 Cart *(sd được cho cả bài toán hồi quy tuyến tính và phân loại)*

- Là một kỹ thuật phi tham số hữu ích có thể được sử dụng để giải thích một *biến phụ thuộc liên tục* hoặc *phân loại* dưới dạng nhiều biến độc lập. Các biến độc lập có thể liên tục hoặc phân loại. CART sử dụng phương pháp phân vùng thường được gọi là "**chia để trị**". Thuật toán CART hoạt động để tìm biến độc lập tạo ra nhóm đồng nhất tốt nhất khi tách dữ liệu.
- Innput: Là một tập dữ liệu đã được gán nhãn
- Output: là cây quyết định
- Công việc thực hiện của bài toán:
  + Sử dụng chỉ số "gini index" làm thước đo trong khi tạo cây quyết định (decision tree) trong thuật toán CART (Classification and Regression Tree), chỉ số "gini index" càng cao càng tốt.

- Các bước thực hiện:

B1: Tính giá trị Gini

$$\text{Gini} = 1 - \sum_{i=1}^{C}(p_i)^2$$

+ C: số lớp cần phân loại (số nhãn trong 1 thuộc tính)
+ $p_i = \frac{n_i}{N}$
+ n_i: là số lượng phần tử ở lớp thứ i
+ N: là tổng số lượng phần tử ở node đó

B2: Tính chỉ số gini_index.

$$\text{Gini\_index} = \text{gini(p)} - \sum_{i=1}^{K} \frac{m_i}{M}\text{gini}(c_i)$$

+ gini(p): chỉ số gini ở node cha
+ K: số node con được tách ra
+ gini(c_k): chỉ số gini ở node con thứ k
+ M: số phần tử ở node p
+ m_i : là số phần tử ở node con thứ i

⇨ chọn ra thuộc tính có gini_index lớn nhất → $\text{Gini}_{\text{split}}$ nhỏ nhất

$$\text{Gini}_{\text{split}} = \sum_{i=1}^{K} \frac{m_i}{M}\text{gini}(c_i)$$

B3: Chia dữ liệu vào các nút con tương ứng với các giá trị của thuộc tính, thực hiện lặp tìm ra thuộc tính tốt nhất trong tập dữ liệu mới.

❖ **Điều kiện dừng:**

– TH1: nếu tất cả các mẫu trong node con thuộc cùng 1 lớp C (entropy=0) thì node đó đc gán nhãn C

– TH2: nếu node con là rỗng (không có dữ liệu) thì node đó sẽ đc gán = nhãn phổ biến nhất trong tập S

– TH3: nếu không còn thuộc tính nào để phân chia (các thuộc tính đều đc xét hết) -> node lá đó sẽ đc gán = nhãn phổ biến nhất

2.4 Logistic regression

- Input:
- Output: 1 số thực trong khoảng [0,1]
- Hàm mất mát:

$$J(\mathbf{w}) = -\log P(\mathbf{y}|\mathbf{X};\mathbf{w})$$

$$= -\sum_{i=1}^{N}(y_i \log z_i + (1-y_i)\log(1-z_i))$$

Ký hiệu $z_i = f(\mathbf{w}^T\mathbf{x}_i)$

- Nghiệm w:
  - Chúng ta lại sử dụng phương pháp GD để tìm **w**
  - Công thức cập nhật (theo thuật toán GD) cho logistic regression là:

$$\mathbf{w} = \mathbf{w} + \eta(y_i - z_i)\mathbf{x}_i$$

2.5 SVM

- Tìm ra siêu phẳng sao cho margin bằng nhau và phải là lớn nhất. (margin: khoảng cách từ điểm gần nhất của mỗi lớp tới đường phân chia).
- Input: Là một tập dữ liệu đã được gán nhãn.
- Output: là siêu phẳng sao cho margin bằng nhau và lớn nhất
- Các bước thực hiện:
  + B1: tính khoảng cách từ 1 điểm đến mặt phân chia, điểm nào có khoảng cách nhỏ nhất thì lấy làm margin.

$$\text{margin} = \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{||\mathbf{w}||_2}$$

  + B2: Tìm ra siêu phẳng sao cho margin là lớn nhất <=> tìm w và b sao cho margin lớn nhất.

$$(\mathbf{w}, b) = \arg\max_{\mathbf{w},b}\left\{\min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{||\mathbf{w}||_2}\right\} = \arg\max_{\mathbf{w},b}\left\{\frac{1}{||\mathbf{w}||_2}\min_n y_n(\mathbf{w}^T\mathbf{x}_n + b)\right\}$$

II. Thuật toán học không giám sát *(dữ liệu học chỉ có đầu vào không có đầu ra)*

1. Phân cụm

1.1 Clustering

- Input: Dữ liệu X và số cụm muốn tìm
- Output: tâm của mỗi cụm và phân dữ liệu vào các cụm
- Hàm mất mát:

Sai số cho toàn bộ dữ liệu sẽ là:

$$\mathcal{L}(Y, M) = \sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij}\|x_i - m_j\|_2^2$$

  + Tìm Y và M để hàm mất mát là nhỏ nhất:
    - $Y = [y_1; y_2; \dots; y_N]$ là ma trận véc tơ nhãn
    - $M = [m_1, m_2, \dots, m_K]$ là ma trận tâm của mỗi cụm
- Các bước thực hiện:
  + B1: Chọn K điểm bất kỳ làm các center ban đầu.
  + B2: Phân mỗi điểm dữ liệu vào cụm có tâm gần nó nhất.
    (Nếu việc phân dữ liệu vào từng cụm *không thay đổi* so với vòng lặp trước đó thì *thuật toán dừng*)
  + B3: Cập nhật tâm cho từng cụm bằng cách lấy trung bình cộng của tất cả các điểm dữ liệu trong cụm đó.
  + B4: thực hiện vòng lặp tiếp theo -> quay lại bước 2.

III. Phương pháp chọn ra mô hình tốt nhất.

1. CROSS-VALIDATIO

- Tránh trường hợp overfitting (mô hình tìm đc quá khớp với dữ liệu training set, nhưng k thực sự tốt mới dữ liệu test).
- Input: vector đặc trưng.
- Output: số thực dương.
- Cách thực hiện:
  + **Bước 1:** chia tập dữ liệu thành 2 phần: tập dữ liệu huấn luyện và tập dữ liệu test.
  + **Bước 2:** Chia toàn bộ tập dữ liệu huấn luyện thành k phần (phương pháp k-fold cross validation).
  + **Bước 3:**
    - Chọn ngẫu nhiên k-1 phần làm training data, 1 phần còn lại làm validation data.
    - Sử dụng phương pháp học máy đã lựa chọn trên tập training data và validation data để xây dựng và đánh giá mô hình (bước này được làm k lần)
  + **Bước 4:** Chọn mô hình có (train error + validation error) là nhỏ nhất.

2. Phân tích thành phần chính (PCA)

- Tìm một hệ trực chuẩn mới sao cho trong hệ này, các thành phần quan trọng nhất nằm trong K thành phần đầu tiên.
- Input: tập thuộc tính, số thuộc tính quan trọng muốn dữ lại
- Output: tập thuộc tính mới sau khi loại bỏ các thuộc tính ít quan trọng
- Các bước thực hiện:
  + B1: Tính vector kỳ vọng của toàn bộ dữ liệu:

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

  + B2: Trừ mỗi điểm dữ liệu đi vector kỳ vọng của toàn bộ dữ liệu:

$$\hat{\mathbf{x}}_n = \mathbf{x}_n - \bar{\mathbf{x}}$$

  + B3: Tính ma trận hiệp phương sai S:

$$\mathbf{S} = \frac{1}{N}\hat{\mathbf{X}}\hat{\mathbf{X}}^T$$

  + B4: Tính các trị riêng và vector riêng của S , sắp xếp chúng theo thứ tự giảm dần của trị riêng.
  + B5: Chọn K vector riêng ứng với K trị riêng lớn nhất để xây dựng ma trận $\mathbf{U}_K$ có các cột tạo thành một hệ trực giao. K vectors này là các thành phần chính.
  + B6: Chiếu dữ liệu ban đầu đã chuẩn hoá $\hat{X}$ xuống không gian con tìm được.
  + B7: Dữ liệu mới chính là toạ độ của các điểm dữ liệu trên không gian mới.

$$\mathbf{Z} = \mathbf{U}_K^T\hat{\mathbf{X}}$$

IV. Đánh giá chất lượng mô hình học máy.

1. Hồi quy tuyến tính.

1. Nash–Sutcliffe efficiency (NSE)

$$NSE = 1 - \frac{\sum_{i=1}^{n}\left(H_i^{mea} - H_i^{pre}\right)^2}{\sum_{i=1}^{n}\left(H_i^{mea} - \overline{H}^{mea}\right)^2} \quad (16)$$

2. Coefficient of determination (R^2)

$$R^2 = \left[\frac{\sum_{i=1}^{n}(H_i^{mea} - H^{-mea})\left(H_i^{pre} - H^{-pre}\right)}{\sqrt{\sum_{i=1}^{n}(H_i^{mea} - H^{-mea})^2 \sum_{i=1}^{n}\left(H_i^{pre} - H^{-pre}\right)^2}}\right]^2 \quad (17)$$

3. Mean absolute error (MAE)

$$MAE = \frac{\sum_{i=1}^{n}\left|H_i^{mea} - H_i^{pre}\right|}{n} \quad (18)$$

4. Root mean square error (RMSE)

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(H_i^{mea} - H_i^{pre}\right)^2}{n}} \quad (19)$$

where H_i^{mea} and H_i^{pre} are, respectively, the measured and predicted river stages and $\overline{H}^{mea}$ and $\overline{H}^{pre}$ are, respectively, the mean of the measured and predicted river stages.

2. Mô hình phân lớp.

Lớp Ci		**Gía trị dự đoán**	
		Positive	Negative
Gía trị thực tế	Positive	True Positive(TPi)	False Negative (FNi)
	Negative	False Positive (FPi)	True Negative (TNi)

TPi: số lượng dữ liệu thuộc lớp Ci đc phân loại đúng vào lớp Ci
FPi: số lượng dữ liệu bên ngoài bị phân loại nhầm vào lớp Ci
TNi: số lượng dữ liệu không thuộc lớp Ci đc phân loại chính xác
FNi: số lượng dữ liệu thuộc lớp Ci bị phân loại nhầm (vào các lớp khác Ci)

- Độ chính xác:

$$Precision(c_i) = \frac{TP_i}{TP_i + FP_i}$$

- Độ thu hồi:

$$Recall(c_i) = \frac{TP_i}{TP_i + FN_i}$$

- Trung bình điều hòa của Precision và Recall:

$$F_1 = \frac{2.Precision.Recall}{Precision + Recall} = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

- Accuracy:

$$\text{Accuracy} = \frac{\text{số mẫu dự đoán đúng}}{\text{tổng số mẫu dự đoán}}$$

3. Mô hình phân cụm.
   a) Độ bóng (Silhouette)

Giả sử mạng lưới được chia thành k cụm.

Với mỗi node i, đặt:

- *a(i)* là khoảng cách trung bình từ i tới tất cả các node trong cùng cụm với *i*.
- *b(i)* là khoảng cách trung bình ngắn nhất từ i tới bất kỳ cụm nào không chứa *i*. Cụm tương ứng với *b(i)* này được gọi là cụm hàng xóm của *i*.

Khi đó:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$$

   b) Độ đo Davies – Bouldin.

Độ đo *Davies-Bouldin* được tính theo công thức:

$$DB = \frac{1}{n}\sum_{i=1}^{n} Max_{i \neq j} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right)$$

Trong đó :

- n là số cụm.
- c_x là trọng tâm của cụm x
- σ_x là trung bình khoảng cách của tất cả các phần tử trong cụm x tới trọng tâm c_x
- $d(c_i, c_j)$ là khoảng cách giữa hai trọng tâm của cụm i và j.

Giá trị *DB* càng nhỏ thì chất lượng phân cụm càng tốt.