Đề cương ôn tập Machine Learning

1. Thuật toán ID3 tìm cây quyết định.

ID3(*TrainingSet, Class Labels, Attributes*):
Tạo nốt *Root* cho cây

Nếu tất cả dữ liệu trong *TrainingSet* thuộc cùng một lớp c:
Trả về nốt *Root* có nhãn là lớp c

Nếu tập thuộc tính *Attributes* hết thuộc tính:
Trả về nốt *Root* có nhãn là lớp xuất hiện nhiều nhất trong *TrainingSet*

Tìm thuộc tính A có khả năng phân loại tốt nhất
Root $\leftarrow A$

Với từng giá trị v của thuộc tính A:
Tạo nhánh mới cho nốt *Root*
Định nghĩa $TrainningSet_v$ là tập dữ liệu mới mà tất cả dữ liệu có giá trị của thuộc tính A là v
Nếu $TrainningSet_v$ rỗng:
Tạo nốt lá với nhãn là lớp xuất hiện nhiều nhất trong *TrainingSet*
Gán nốt lá đó với nhánh mới vừa tạo
Ngược lại gán nhánh mới với cây con tạo bởi **ID3**($TrainningSet_v$, *Class Labels, Attributes*)

2. Thuật toán phân lớp Naive Bayes.

Cho tập dữ liệu huấn luyện $D = \{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \ldots, (\boldsymbol{x}^{(i)}, y^{(i)}), \ldots, (\boldsymbol{x}^{N}, y^{N})\}, \boldsymbol{x}^{(i)} \in \mathbb{R}^d, y \in \{c_1, c_2, \ldots, c_m\}$.

$\boldsymbol{x}^{(i)}$ được biểu diễn dưới dạng vectơ như sau:

$$\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_d \end{bmatrix}$$

d: Số lượng thuộc tính của tập dữ liệu.
m: Số lớp cần phân loại.
N: Tổng số dữ liệu của tập dữ liệu.

Công thức Bayes đối với bài toán phân loại:

$$P(y = c|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y = c) \times P(y = c)}{P(\boldsymbol{x})}$$

Với công thức trên, mục tiêu bài toán là tìm ra lớp thứ i nào làm cho $P(c_i|\boldsymbol{x})$ lớn nhất, từ đó kết luận đầu ra:

$$\begin{aligned} y &= \underset{c_i}{\text{argmax}}\ P(y = c_i|\boldsymbol{x}) \\ &= \underset{c_i}{\text{argmax}}\ \frac{P(\boldsymbol{x}|y = c_i) \times P(y = c_i)}{P(\boldsymbol{x})} \\ &= \underset{c_i}{\text{argmax}}\ P(\boldsymbol{x}|y = c_i) \times P(y = c_i) \end{aligned}$$

Ta giả thiết rằng các chiều của $\boldsymbol{x}$ là độc lập với nhau, nên:

$$y = \underset{c_i}{\text{argmax}} \prod_{j=1}^{d} P(x_j|y = c_i) \times P(y = c_i)$$

Ta cần tính các giá trị $P(x_j|y = c_i)$ và $P(y = c_i)$.

Dạng Naive Bayes được học trên lớp là Categorical Naive Bayes, nên ta có những giả thiết sau:
+ Ta chia ra làm m tập dữ liệu con tương ứng với m lớp, tập dữ liệu con thứ i sẽ chứa n_i dữ liệu có cùng lớp c_i.
+ x_j tuân theo phân phối danh mục (Categorical Distribution) với tham số θ_{ji} (Với mỗi thuộc tính thứ j, ta có m tham số θ tương ứng. Vì là phân phối danh mục nên θ sẽ có số lượng phần tử bằng với số giá trị của thuộc tính thứ j là $|A_j|$):

$$x_j \sim Cat(\theta_{ji}) \Rightarrow P(x_j|y = c_i) = \prod_{k=1}^{|A_j|} \theta_k^{\mathbb{I}(x_j=k)}$$

$$\theta_{ji} = \begin{bmatrix} \theta_1 & \theta_2 & \cdots & \theta_{|A_j|} \end{bmatrix}$$

$$\mathbb{I}(x_j = k) = \begin{cases} 1 & x_j = k \\ 0 & x_j \neq k \end{cases}$$

+ Đầu ra y cũng tuân theo phân phối danh mục với tham số θ (θ có m giá trị):

$$y \sim Cat(\theta) \Rightarrow P(y = c_i) = \prod_{i=1}^{m} \theta_i^{\mathbb{I}(y=i)}$$

$$\theta = \begin{bmatrix} \theta_1 & \theta_2 & \cdots & \theta_m \end{bmatrix}$$

Ta dùng MLE(Maximum Likelihood Estimation) để ước tính các tham số θ:

$$\theta_{ji} = \begin{bmatrix} \frac{q_1}{n_i} & \frac{q_2}{n_i} & \cdots & \frac{q_{|A_j|}}{n_i} \end{bmatrix}$$

$$\theta = \begin{bmatrix} \frac{n_1}{N} & \frac{n_2}{N} & \cdots & \frac{n_m}{N} \end{bmatrix}$$

q_i: tổng số giá trị thứ i của một thuộc tính.

Để tránh việc một hoặc nhiều phần tử trong $\theta_{ji} = 0$ (do sự không xuất hiện của giá trị đó trong lớp thứ i), ta dùng phương pháp Laplace Smoothing:

$$\theta_{ji} = \begin{bmatrix} \frac{q_1 + \alpha}{n_i + \alpha|A_j|} & \frac{q_2 + \alpha}{n_i + \alpha|A_j|} & \cdots & \frac{q_{|A_j|} + \alpha}{n_i + \alpha|A_j|} \end{bmatrix}$$

α thường chọn là 1.

Bảng tham số theo thuộc tính x_j:

	c_1	c_2	$\cdots$	c_m
x_1	θ_{11}	θ_{12}	$\cdots$	θ_{1m}
x_2	θ_{21}	θ_{22}	$\cdots$	θ_{2m}
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
x_d	θ_{d1}	θ_{d2}	$\cdots$	θ_{dm}

5. Phân biệt học máy có giám sát và học máy không giám sát?

Học máy có giám sát (Supervised Learning)
- Học một hàm mà có thể dự đoán giá trị đầu ra từ giá trị đầu vào.
- Hàm này được học từ một bộ dữ liệu được gán nhãn từ trước.
- Một số thuật toán: Hồi quy tuyến tính, hồi quy Logistic, Decision Tree, SVM, Neural Network,...

Học máy không giám sát (Unsupervised Learning)
- Học một hàm từ bộ dữ liệu không được gán nhãn từ trước.
- Học cấu trúc và những điểm chung của dữ liệu để tự phân vào các cụm.
- Một số thuật toán: K-means Clustering, PCA,...

6. Overfitting.

Định nghĩa:
Một hàm mục tiêu f được gọi là **Overfitting** nếu nó tồn tại một hàm f' mà:
- Hàm f' không thích hợp để miêu tả tập dữ liệu huấn luyện so với hàm f.
- Tuy nhiên hàm f' lại cho ra độ chính xác cao hơn so với f trên toàn bộ tập dữ liệu.

Nguyên nhân:
- Tập dữ liệu quá bé, không đủ để diễn tả hết vấn đề cần giải quyết.
- Nhiễu từ tập dữ liệu (do việc thu tập dữ liệu hoặc do cấu trúc của dữ liệu).

7. Trình bày phương pháp 5-folds để đánh giá mô hình học máy.

Với phương pháp 5-folds, ta chia tập dữ liệu D ra thành 5 tập dữ liệu con có kích thước tương đối bằng nhau:
- Dữ liệu trong 5 tập con không được trùng nhau.
- Sau khi chia, ta lấy từng tập một ra để làm tập dữ liệu thử và 4 tập còn lại sẽ là tập huấn luyện.
- Sau khi hết 5 lượt như vậy, ta có 5 giá trị lỗi và tổng lỗi sẽ được tính bằng cách lấy trung bình 5 giá trị lỗi đó.

9. Tính giá trị Gain Ratio đối với các thuộc tính trong 1 tập.

Cho tập dữ liệu huấn luyện $D = \{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \ldots, (\boldsymbol{x}^{(i)}, y^{(i)}), \ldots, (\boldsymbol{x}^{N}, y^{N})\}, \boldsymbol{x}^{(i)} \in \mathbb{R}^d, y \in \{c_1, c_2, \ldots, c_m\}$.
Tập thuộc tính $A = \{A_1, A_2, \cdots, A_d\}$.
Tập giá trị của thuộc tính $A_j = \{v_1, v_2, \cdots, v_{|A_j|}\}$

Công thức Entropy cho tập dữ liệu D:

$$Entropy(D) = \sum_{i=1}^{m} -p_i log_2(p_i)$$

p_i: tỉ lệ của lớp thứ i trong tập dữ liệu huấn luyện D.

Ta có độ tăng thông tin của thuộc tính A_j:

$$InformationGain(D, A_j) = Entropy(D) - \sum_{v \in A_j} \frac{|D_v|}{|D|} Entropy(D_v)$$

D_v: tập dữ liệu con mà tất cả dữ liệu có giá trị của thuộc tính A_j là v.

Độ chia thông tin của thuộc tính A_j:

$$SplitInformation(D, A_j) = -\sum_{v \in A_j} \frac{|D_v|}{|D|} log_2 \frac{|D_v|}{|D|}$$

Tỉ lệ tăng thông tin của thuộc tính A_j:

$$GainRation(D, A_j) = \frac{InformationGain(D, A_j)}{SplitInformation(D, A_j)}$$

Tỉ lệ tăng thông tin *GainRatio* thường được dùng để thay thế cho độ tăng thông tin *InformationGain* để giảm độ ảnh hưởng của các thuộc tính có nhiều giá trị.

10. Nêu 2 phương pháp cắt tỉa cây để tránh Overfitting.

2 phương pháp cắt tỉa cây ở đây được thực hiện khi đã học được một cây hoàn chỉnh.

Reduced-error Pruning
- Từng nốt một của một cây hoàn chỉnh sẽ được kiểm tra để cắt.
- Một nốt sẽ được cắt nếu sau khi cắt nốt đó đi, cây mới có hiệu suất không tệ hơn cây gốc khi đem thử trên tập đánh giá.
- Quá trình cắt nốt bao gồm:
  + Xóa toàn bộ cây con có gốc là nốt bị cắt.
  + Chuyển nốt bị cắt thành nốt lá. Nốt lá này có lớp đầu ra là lớp chiếm nhiều nhất trong tập dữ liệu chia theo nốt đó.
- Lặp lại quá trình cắt:
  + Luôn luôn chọn nốt để cắt sao cho tối đa hóa khả năng phân loại của cây trên tập đánh giá.
  + Dừng cắt khi cây bị giảm khả năng phân loại trên tập đánh giá.

Rule Post-pruning
- Chuyển cây quyết định đã được học thành một tập các luật.
- Ta xóa những điều kiện trong luật mà điều kiện đó không mang lại sự cải thiện cho

hiệu quả phân loại.
- Sắp xếp lại các luật dựa trên khả năng phân loại của cây, rồi dùng cây đó cho các dữ liệu tương lai.

11. Thuật toán SVMs phân lớp nhị phân dựa trên ý tưởng nào?

- Với bài toán phân lớp nhị phân (điểm dữ liệu phân tách tuyến tính), ta luôn tìm được các siêu phẳng để phân chia các điểm dữ liệu thành 2 lớp.
- Tuy nhiên, không phải siêu phẳng nào cũng mang lại hiệu suất tốt đối với những dữ liệu sau này.
- Cho nên, thuật toán SVMs sẽ chọn ra một siêu phẳng mà có lề lớn nhất làm đường quyết định cho toàn bộ bài toán.

12. Hãy nêu định nghĩa 2 mặt phẳng lề trong thuật toán và từ đó tính mức lề, và đưa ra bài toán tối ưu tương đương.

Ta có phương trình của một siêu phẳng:

$$\begin{aligned} f(\boldsymbol{x}) &= < \boldsymbol{w}, \boldsymbol{x} > + b \\ &= \boldsymbol{w}^T \boldsymbol{x} + b \end{aligned}$$

Đầu ra y với đầu vào $\boldsymbol{x}$ tương ứng:

$$y = \begin{cases} 1 & f(\boldsymbol{x}) \geq 0 \\ -1 & f(\boldsymbol{x}) < 0 \end{cases}$$

Khoảng cách từ một điểm $\boldsymbol{x}$ đến một siêu phẳng f:

$$d(\boldsymbol{x}, f) = \frac{| < \boldsymbol{w}, \boldsymbol{x} > + b|}{\|\boldsymbol{w}\|_2}$$

2 mặt phẳng lề
Giả sử có 2 điểm dữ liệu: x^+ là điểm dữ liệu ở lớp (+), x^- là điểm dữ liệu ở lớp (-). 2 điểm dữ liệu này là 2 điểm gần siêu phẳng f nhất, ta có:
- 2 mặt phẳng lề f^+ và f^- là 2 mặt phẳng song song với nhau, sao cho:
  + x^+ nằm trên f^+ và song song với f.
  + x^- nằm trên f^- và song song với f.
- Điểm trên 2 mặt phẳng này sẽ có khoảng cách đến siêu phẳng f lần lượt là $d_+(\boldsymbol{x}^+, f)$ và $d_-(\boldsymbol{x}^-, f)$.
- Dựa vào công thức khoảng cách phía trên ta thấy, nếu ta điều chỉnh vectơ pháp

tuyến $\boldsymbol{w}$ thì khoảng cách từ một điểm đến siêu phẳng f là không đổi. Nên ta có thể điều chỉnh $\boldsymbol{w}$ sao cho những điểm $\boldsymbol{x}$ gần siêu phẳng nhất có $| < \boldsymbol{w}, \boldsymbol{x} > +b| = 1$.
- Vì vậy, ta có công thức cho 2 mặt phẳng lề:

$$f^+(\boldsymbol{x}^+) =< \boldsymbol{w}, \boldsymbol{x}^+ > +b = 1$$
$$f^-(\boldsymbol{x}^-) =< \boldsymbol{w}, \boldsymbol{x}^- > +b = -1$$

- Những điểm có đầu ra $y = 1$ thì $f(\boldsymbol{x}) \geq 1$.
- Những điểm có đầu ra $y = -1$ thì $f(\boldsymbol{x}) \leq -1$.
- Với bất kì một điểm dữ liệu nào, $yf(\boldsymbol{x}) \geq 1$.

Mức lề
Ta có khoảng cách của 2 điểm $\boldsymbol{x}^+$ và x^- đến siêu phẳng f lần lượt là:

$$d_+(\boldsymbol{x}^+, f) = \frac{| < \boldsymbol{w}, \boldsymbol{x}^+ > +b|}{\|\boldsymbol{w}\|_2} = \frac{1}{\|\boldsymbol{w}\|_2}$$
$$d_-(\boldsymbol{x}^-, f) = \frac{| < \boldsymbol{w}, \boldsymbol{x}^- > +b|}{\|\boldsymbol{w}\|_2} = \frac{1}{\|\boldsymbol{w}\|_2}$$

Mức lề:

$$margin = d_+ + d_- = \frac{2}{\|\boldsymbol{w}\|_2}$$

Bài toán tối ưu tương đương
Bài toán đặt ra là tìm $\boldsymbol{w}$ và b sao cho siêu phẳng f có *margin* lớn nhất có thể và phân tách hoàn toàn 2 lớp:

$$\begin{aligned} \text{cực đại hóa} \quad & \frac{2}{\|\boldsymbol{w}\|_2} \\ \text{sao cho} \quad & y^{(i)} f(\boldsymbol{x}^{(i)}) \geq 1, \quad i = 1, \ldots, N \end{aligned}$$

Cực đại hóa $\frac{2}{\|\boldsymbol{w}\|_2}$ đồng nghĩa với việc cực tiểu hóa $\frac{\|\boldsymbol{w}\|_2}{2}$. Vì $\|\boldsymbol{w}\|_2$ không khả vi tại mọi điểm tuy nhiên lại là một hàm dương, nên ta có thể viết lại $\frac{\|\boldsymbol{w}\|_2}{2}$ thành $\frac{\|\boldsymbol{w}\|_2^2}{2}$.

Bài toán tối ưu của ta sẽ là:

$$\begin{aligned} \text{cực tiểu hóa} \quad & \frac{\|\boldsymbol{w}\|_2^2}{2} \\ \text{sao cho} \quad & 1 - y^{(i)} f(\boldsymbol{x}^{(i)}) \leq 0, \quad i = 1, \ldots, N \end{aligned}$$

13. Thuật toán SVMs dùng kỹ thuật nào để giải quyết vấn đề dữ liệu không thể phân lớp tuyến tính? Hãy nêu ý tưởng của kỹ thuật đó? Cho ví dụ về một hàm Kernel mà bạn biết?

Thuật toán SVMs sử dụng kĩ thuật *lề mềm* (*Soft Margin SVM*) để giải quyết vấn đề dữ liệu không thể phân lớp tuyến tính:
- Tập dữ liệu chứa dữ liệu nhiễu.
- Nếu cứ tiếp tục tìm đường quyết định sao cho cố gắng phân chia cả dữ liệu nhiễu đó thì *margin* tương ứng với đường quyết định đó sẽ rất nhỏ.
- Vì thế, ta sẽ phải tìm đường quyết định có *margin* rộng hơn, tuy nhiên phải chấp nhận sẽ xuất hiện các điểm dữ liệu nhiễu trong khoảng từ đường quyết định đến 2 mặt phẳng lề.
- Với điều kiện của mặt phẳng lề cũ, ta sẽ nới lỏng điều kiện này bằng cách thêm một biến mới ξ_i (biến này thể hiện độ hy sinh của điểm dữ liệu $\boldsymbol{x}^{(i)}$):

$$y^{(i)} f(\boldsymbol{x}^{(i)}) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

- Ta cũng sẽ phải tìm các ξ_i này sao cho sự hy sinh là ít nhất (vì nếu cứ chấp nhận hết các điểm nhiễu thì nó sẽ tạo ra 2 mặt phẳng lề có *margin* rất lớn), bài toán tối ưu mới sẽ là:

$$\begin{aligned} \text{cực tiểu hóa} \quad & \frac{\|\boldsymbol{w}\|_2^2}{2} + C \sum_{i=1}^{N} \xi_i \\ \text{sao cho} \quad & 1 - \xi_i - y^{(i)} f(\boldsymbol{x}^{(i)}) \leq 0, \quad i = 1, \ldots, N \\ & -\xi_i \leq 0 \end{aligned}$$

- C là hằng số thể hiện sự muốn hy sinh ít hay nhiều: C càng lớn thì sẽ tập trung vào việc giảm ξ_i nhiều hơn, ít điểm phải hy sinh hơn:
  + $\xi_i = 0$, điểm dữ liệu $\boldsymbol{x}^{(i)}$ nằm đúng bên và không phải hy sinh.
  + $\xi_i \leq 1$, điểm dữ liệu $\boldsymbol{x}^{(i)}$ nằm đúng bên tuy nhiên phải hy sinh.
  + $\xi_i \geq 1$, điểm dữ liệu $\boldsymbol{x}^{(i)}$ nằm sai bên và phải hy sinh.

Ta có thể giải bài toán tối ưu trên bằng cách sử dụng phương pháp nhân tử Lagrange:

$$L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \frac{\|\boldsymbol{w}\|_2^2}{2} + C \sum_{i=1}^{N} \xi_i + \sum_{i=1}^{N} \lambda_i (1 - \xi_i - y^{(i)} f(\boldsymbol{x}^{(i)})) - \sum_{i=1}^{N} \mu_i \xi_i$$

Xét đạo hàm của hàm L đối với các biến $\boldsymbol{w}, b, \boldsymbol{\xi}$ bằng 0, ta được:

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{w}} = 0 \quad &\Rightarrow \boldsymbol{w} = \sum_{i=1}^{N} \lambda_i y^{(i)} \boldsymbol{x}^{(i)} \\
\frac{\partial L}{\partial b} = 0 \quad &\Rightarrow -\sum_{i=1}^{N} \lambda_i y^{(i)} = 0 \\
\frac{\partial L}{\partial \boldsymbol{\xi}} = 0 \quad &\Rightarrow \boldsymbol{\lambda} = C - \boldsymbol{\mu}
\end{aligned}$$

Thay các kết quả vào hàm L, ta được hàm đối ngẫu:

$$g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y^{(i)} y^{(j)} \boldsymbol{x}^{(i)T} \boldsymbol{x}^{(j)} + \sum_{i=1}^{N} \lambda_i$$

Để tìm $\boldsymbol{\lambda}$, ta giải bài toán tối ưu:

$$\begin{aligned}
\text{cực đại hóa} \quad & g(\boldsymbol{\lambda}, \boldsymbol{\mu}) \\
\text{sao cho} \quad & \sum_{i=1}^{N} \lambda_i y^{(i)} = 0 \\
& 0 \leq \lambda_i \leq C
\end{aligned}$$

Sau khi tìm được $\boldsymbol{\lambda}$, kết hợp với điều kiện KKT của bài toán gốc, ta sẽ tìm được các tham số $\boldsymbol{w}, b, \boldsymbol{\xi}$:

$$\begin{aligned}
-\xi_i &\leq 0 \\
\lambda_i &\geq 0 \\
\mu_i &\geq 0 \\
\mu_i \xi_i &= 0 \\
\lambda_i (1 - \xi_i - y^{(i)} f(\boldsymbol{x}^{(i)})) &= 0 \\
1 - \xi_i - y^{(i)} f(\boldsymbol{x}^{(i)}) &\leq 0 \\
\boldsymbol{w} &= \sum_{i=1}^{N} \lambda_i y^{(i)} \boldsymbol{x}^{(i)} \\
\sum_{i=1}^{N} \lambda_i y^{(i)} &= 0 \\
\boldsymbol{\lambda} &= C - \boldsymbol{\mu}
\end{aligned}$$

Đọc thêm: `https://machinelearningcoban.com/2017/04/13/softmarginsmv/`

Khi dữ liệu trở nên phi tuyến tính, việc sử dụng SVM trở nên không khả quan. Vì thế, để dùng được, ta phải dùng một hàm $\Phi(\boldsymbol{x})$ để chuyển từ dạng phi tuyến giữa các điểm $\boldsymbol{x}$ thành tuyến tính giữa các điểm $\Phi(\boldsymbol{x})$.

Để ý thấy trong công thức của $g(\boldsymbol{\lambda}, \boldsymbol{\mu})$, ta cần tính tích vô hướng của $\boldsymbol{x}^{(i)}$ và $\boldsymbol{x}^{(j)}$ ($< \boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} >$). Cho nên, với dữ liệu phi tuyến, khi đã biết $\Phi(\boldsymbol{x})$ của chúng thì ta chỉ cần tính $< \Phi(\boldsymbol{x}^{(i)}), \Phi(\boldsymbol{x}^{(j)}) >$.

Và ta gọi đó là một hàm *Kernel*:

$$K(\boldsymbol{x}, \boldsymbol{z}) =< \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) >$$

Với việc chọn các hàm $\Phi(\boldsymbol{x})$ khác nhau, ta có một số hàm *Kernel* như sau:
+ Linear Kernel:

$$K(\boldsymbol{x}, \boldsymbol{z}) =< \boldsymbol{x}, \boldsymbol{z} >$$

+ Polynomial Kernel :

$$K(\boldsymbol{x}, \boldsymbol{z}) = (< \boldsymbol{x}, \boldsymbol{z} > +\theta)^d$$

+ Gaussian Radial Basis Function:

$$K(\boldsymbol{x}, \boldsymbol{z}) = e^{-\dfrac{\|\boldsymbol{x} - \boldsymbol{z}\|_2^2}{2\sigma}}$$

+ Sigmoid:

$$K(\boldsymbol{x}, \boldsymbol{z}) = tanh(\beta < \boldsymbol{x}, \boldsymbol{z} - \gamma >)$$

14. Trình bày kỹ thuật One vs All trong thuật toán SVMs, và giải thích mục đích của nó?

- Kĩ thuật One vs All được sử dụng với số lượng lớp lớn hơn 2, tổng quát là m lớp.
- Khi sử dụng kĩ thuật này, ta sẽ phải huấn luyện m bộ phân loại khác nhau dành cho m lớp.
- Khi huấn luận bộ phân lớp thứ i, ta coi đầu ra của bài toán thành 2 lớp: nếu rơi vào lớp i thì là lớp (+), còn nếu rơi vào $m - 1$ lớp còn lại thì là lớp (-).
- Khi đã huấn luyện xong, với mỗi điểm dữ liệu mới $\boldsymbol{x}$, ta cho đi qua m bộ phân lớp. Bộ phân lớp nào cho đầu ra rơi vào lớp (+) cao nhất thì ta gán đầu ra cho lớp đó.
- Mục đích: đưa tập dữ liệu về dạng nhị phân, vì thuật toán SVMs hoạt động rất tốt với tập dữ liệu chỉ có 2 lớp.

15. Giả sử có 1 nơ ron sử dụng hàm kích hoạt là hàm hl2. Hãy tính đầu ra của nơ ron.

Hàm *hl2* chính là hàm *sign*:

$$sign(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}$$

Có vectơ đầu vào $\boldsymbol{x} = \begin{bmatrix} 3 & 5 & 7 \end{bmatrix}$ và vectơ trọng số $\boldsymbol{w} = \begin{bmatrix} 1 & 1 & 0 & 1 \end{bmatrix}$, thêm 1 vào vectơ $\boldsymbol{x}$, ta có đâu ra:

$$\begin{aligned} y &= sign(< \boldsymbol{x}, \boldsymbol{w}^T >) \\ &= sign(\begin{bmatrix} 3 & 5 & 7 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}) \\ &= sign(3.1 + 5.1 + 7.0 + 1.1) \\ &= sign(9) \\ &= 1 \end{aligned}$$

16. Hãy nêu định nghĩa về mạng nơ ron đầy đủ (fully connected) hồi quy (recurrent) và vẽ ví dụ về mạng nơ ron hồi quy 2 tầng ẩn?

Mạng nơ-ron kết nôi đầy đủ tính toán thẳng là mạng tính thẳng giá trị đầu ra y từ đầu vào $\boldsymbol{x}$. Giá trị đầu ra y được đem ngược trở lại để làm đầu vào cùng với dữ liệu $\boldsymbol{x}$ mới. Giá trị đầu ra của tầng trước sẽ được đem làm giá trị đầu vào cho tầng ngay sau nó. Mạng có các ma trận trọng số $\boldsymbol{W}$ giữa các tầng là một ma trận dày đặc (các phần tử đều khác 0), thể hiện cho việc các nơ-ron của tầng trước kết nối đến tất cả các nơ-ron của tầng sau. Ma trận trọng số của y ngược lại đầu vào cũng là một ma trận đầy đủ.

17. Hãy nêu định nghĩa về mạng nơ ron đầy đủ (fully connected) tiến (feed-forward) và vẽ ví dụ về mạng nơ ron tiến 2 tầng ẩn?

Mạng nơ-ron kết nôi đầy đủ tính toán thẳng là mạng tính thẳng giá trị đầu ra y từ đầu vào $\boldsymbol{x}$. Giá trị đầu ra của tầng trước sẽ được đem làm giá trị đầu vào cho tầng

ngay sau nó. Mạng có các ma trận trọng số $\boldsymbol{W}$ giữa các tầng là một ma trận dày đặc (các phần tử đều khác 0), thể hiện cho việc các nơ-ron của tầng trước kết nối đến tất cả các nơ-ron của tầng sau.

18. Trình bày thuật toán Perceptron dưới dạng giả mã.

Cho tập dữ liệu D và tốc độ học α
Perceptron(D, α)
  Khởi tạo ngẫu nhiên các thành phần trong vectơ $\boldsymbol{w}$
  do
    for từng dữ liệu $(\boldsymbol{x}, y \in D)$
      Tính giá trị đầu ra $\hat{y}$
      if $(\hat{y} \neq y)$
        $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha(y - \hat{y})\boldsymbol{x}$
    end for
  until các dữ liệu trong tập D được phân vào đúng lớp

  return $\boldsymbol{w}$