**Chapter 7. Neural Networks (mạng nơ ron)**

**Outline**

[**I. CÁC CÂU HỎI CHƯƠNG 7 CÓ TRONG CÁC ĐỀ THI 2**](#_43e9l3bvebs3)

[1. Fin\_201.pdf\_HKI\_(2020-2021) 2](#_ghd6d3hz7wos)

[2. Fin\_202.pdf\_HKII\_(2020-2021) 2](#_3tl6br8wrpcf)

[3. Fin\_12\_2021.pdf\_HKI\_(2021-2022) 2](#_j9gq0wo6z7xt)

[4. Fin\_12\_2021.pdf\_HKII\_(2021-2022) 2](#_lxbo7ly6sudw)

[**II. CÂU HỎI CHI TIẾT TRONG ĐỀ 2**](#_redpzuq8njsd)

[1. Fin\_201.pdf\_HKI\_(2020-2021) 2](#_3xx3b76v0id3)

[2. Fin\_202.pdf\_HKII\_(2020-2021) 2](#_7ojfifuqfd31)

[3. Fin\_12\_2021.pdf\_HKI\_(2021-2022) 3](#_s1im3gd9yoi)

[4. Fin\_12\_2021.pdf\_HKII\_(2021-2022) 3](#_xr5cagvendmm)

[**II. GIẢI ĐỀ …To do … 4**](#_q2m3hf5vaw7i)

[**1. Fin\_201.pdf\_HKI\_(2020-2021) 4**](#_iyg4khtr2i3e)

[Câu 3. (0.5 điểm) 4](#_nmud2pxxn90b)

[Đáp Án: 4](#_rwa0g85dixd6)

[Câu 4 (0.75 điểm) 5](#_e8logmfz21jt)

[Đáp Án 6](#_qgzb18we5imw)

[**2. Fin\_202.pdf\_HKII\_(2020-2021) 6**](#_sjhq1t54n7zn)

[Câu 1. (0.5 điểm) (Trùng với câu 3. Fin\_201.pdf\_HKI\_(2020-2021)) 6](#_hxpjtnrpglii)

[Câu 3. (1.25 điểm) 6](#_loznwqxgr0pe)

[Đáp án 7](#_s5spfvixdnhn)

[Câu 4. (1.75 điểm) 7](#_awvrjroa4o20)

[Đáp án 8](#_iz5sx478oh2)

[**3. Fin\_12\_2021.pdf\_HKI\_(2021-2022) 10**](#_fyqv54f8dm46)

[Câu 2. (Trùng với câu 3. Fin\_202.pdf\_HKII\_(2020-2021)) 10](#_82i3mrsme9lv)

[Câu 3. (Trùng với câu 4. Fin\_202.pdf\_HKII\_(2020-2021)) 10](#_hm8ovqgwndvt)

[Câu 4. (0.5 pts) 10](#_cgozu7mnyg7k)

[Đáp án 10](#_3v2pdmqgt20w)

[**Câu 5. (0.75 điểm) 11**](#_utk9zt4aqilh)

[**4. Fin\_12\_2021.pdf\_HKII\_(2021-2022) 11**](#_t5n72694hn1u)

[Câu 1. 11](#_erusbwv9en)

[Đáp Án 11](#_fx8q5opsrnhv)

## **I. CÁC CÂU HỎI CHƯƠNG 7 CÓ TRONG CÁC ĐỀ THI**

##### [Fin\_201.pdf\_HKI\_(2020-2021)](https://docs.google.com/document/d/1O30YVMcYqIh3FvXyK30TFXqQB4cgPsMO/edit#heading=h.k859l5ku3md)

* + Câu 3 - **Lân**
  + Câu 4 - **Lân**

##### [Fin\_202.pdf\_HKII\_(2020-2021)](https://docs.google.com/document/d/1DkbvgkUJOOQcx9eNk-DsMT5MwyhNMvLE/edit#)

* + Câu 1 ***(Trùng với câu 3.*** [***Fin\_201.pdf\_HKI\_(2020-2021)***](https://docs.google.com/document/d/1O30YVMcYqIh3FvXyK30TFXqQB4cgPsMO/edit#heading=h.k859l5ku3md)***)***
  + Câu 3 - **Lân**
  + Câu 4 - **Lân** *Khác kết quả với Toàn*

##### [Fin\_12\_2021.pdf\_HKI\_(2021-2022)](https://docs.google.com/document/d/1FBqWzRhwBudjJDBuK2_z72pM98EndxA2/edit#heading=h.t37itj2wog4b)

* + Câu 2 ***(Trùng với câu 3.*** [**Fin\_202.pdf\_HKII\_(2020-2021)**](https://docs.google.com/document/d/1DkbvgkUJOOQcx9eNk-DsMT5MwyhNMvLE/edit#)**)**
  + Câu 3 ***(Trùng với câu 4.*** [**Fin\_202.pdf\_HKII\_(2020-2021)**](https://docs.google.com/document/d/1DkbvgkUJOOQcx9eNk-DsMT5MwyhNMvLE/edit#)**)**
  + Câu 4 - **Lân**
  + Câu 5. a

##### [Fin\_12\_2021.pdf\_HKII\_(2021-2022)](https://docs.google.com/document/d/1BLSqsCee6LnqI5rh2J_ZeN3whI-TOJl-VinGDjto97A/edit#)

* + Câu 1 - **Lân**
  + Câu 2
  + Câu 3 ***(Trùng với câu 1.*** [**Fin\_202.pdf\_HKII\_(2020-2021)**](https://docs.google.com/document/d/1DkbvgkUJOOQcx9eNk-DsMT5MwyhNMvLE/edit#)**)**
  + Câu 4 **Lân**

## **II. CÂU HỎI CHI TIẾT TRONG ĐỀ**

#### **1.** [**Fin\_201.pdf\_HKI\_(2020-2021)**](https://docs.google.com/document/d/1O30YVMcYqIh3FvXyK30TFXqQB4cgPsMO/edit#heading=h.k859l5ku3md)

**Câu 3. (0.5 điểm)** Nêu điểm khác biệt giữa giải thuật *suy giảm độ dốc* (gradient descent) và giải thuật *suy giảm độ dốc tăng dần* (incremental gradient descent) khi huấn luyện một đơn vị perceptron tuyến tính.  
**→ Chương 7.**

**Câu 4 (0.75 điểm)** Khi xác định cấu hình của mạng nơ ron cho một bài toán phân lớp, căn cứ vào đâu để xác định số nút của tầng nhập và số nút của tầng xuất? Và bằng cách nào xác định số nút thích hợp cho tầng ẩn ?  
**→ Chương 7  
*(Trùng với câu 9.*** [**Fin\_12\_2021.pdf\_HKI\_(2021-2022)**](https://docs.google.com/document/d/1FBqWzRhwBudjJDBuK2_z72pM98EndxA2/edit#heading=h.t37itj2wog4b)**)**

#### **2.** [**Fin\_202.pdf\_HKII\_(2020-2021)**](https://docs.google.com/document/d/1DkbvgkUJOOQcx9eNk-DsMT5MwyhNMvLE/edit#)

**Câu 1. (0.5 điểm)** Nêu điểm khác biệt giữa giải thuật *suy giảm độ dốc* (gradient descent) và giải thuật *suy giảm độ dốc tăng dần* (incremental gradient descent) khi huấn luyện một đơn vị perceptron tuyến tính.  
**→ Chương 7. *(Trùng với câu 3.*** [***Fin\_201.pdf\_HKI\_(2020-2021)***](https://docs.google.com/document/d/1O30YVMcYqIh3FvXyK30TFXqQB4cgPsMO/edit#heading=h.k859l5ku3md)***)***

**Câu 3. (1.25 điểm)**

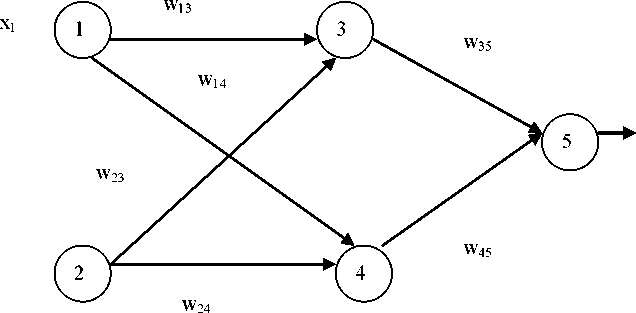
Điều chỉnh lại giải thuật *lan truyền ngược* để giải thuật này có thể làm việc được với các nút sử dụng hàm kích hoạt *tanh* thay vì hàm sigmoid. Hàm *tanh* được định nghĩa như sau:

*tanh*(*x*) = (*ex e-x*)/(*ex* + *e-x*) *a^^Hỉỵ^e e* v*e e*

Tức là điều chỉnh lại qui tắc cập nhật trọng số tại tầng xuất và tầng ẩn. Lưu ý đạo hàm hàm *tanh* có tính chất: *tanh’*(*x*) = 1 - [*tanh*(*x*)]2.  
**→ Chương 7**

**Câu 4. (1.75 điểm)**

Cho mạng nơ ron có cấu hình như sau:



Cho các giá trị trọng số và các độ lệch (bias) khởi đầu như trong bảng sau:

w13 w14 w23 w24 w35 w45 w03 w04 w05

0.2 -0.3 0.4 0.1 -0.3 -0.2 -0.4 0.2 0.1

Giả sử hệ số học *7* = 0.8, hàm sigmoid được dùng ở tầng ẩn và tầng xuất, và giải thuật lan truyền ngược được dùng để huấn luyện mạng nơ ron. Nếu mẫu huấn luyện <x1 = **0**, x2 = **1** và output = **1**> được đưa vào giải thuật huấn luyện thì các trọng số và các độ lệch (w03, w04, w05) sẽ được điều chỉnh lại như thế nào?

**→ Chương 7.**

#### **3.** [**Fin\_12\_2021.pdf\_HKI\_(2021-2022)**](https://docs.google.com/document/d/1FBqWzRhwBudjJDBuK2_z72pM98EndxA2/edit#heading=h.t37itj2wog4b)

**Câu 2. *(cùng đề với câu 3.*** [**Fin\_202.pdf\_HKII\_(2020-2021)**](https://docs.google.com/document/d/1DkbvgkUJOOQcx9eNk-DsMT5MwyhNMvLE/edit#)**)**

**Câu 3. *(cùng đề với câu 4.*** [**Fin\_202.pdf\_HKII\_(2020-2021)**](https://docs.google.com/document/d/1DkbvgkUJOOQcx9eNk-DsMT5MwyhNMvLE/edit#)**)**

**Câu 4. (0.5 pts)** *Li*ệt *kê nh*ững tham số trong mạng nơ ron *RBF mà chúng ta ph*ải x*ác* định *khi hu*ấn luyện mạng nơ ron n*ày.* **→ Chương 7**

**Câu 5. (0.75 điểm)**

1. Hãy nêu công dụng của hàm kernel đối với máy véc tơ hỗ trợ trong trường hợp dữ liệu không khả tách một cách tuyến tính. (*0.25 điểm*)  
   **→ Chương 7**

#### **4.** [**Fin\_12\_2021.pdf\_HKII\_(2021-2022)**](https://docs.google.com/document/d/1BLSqsCee6LnqI5rh2J_ZeN3whI-TOJl-VinGDjto97A/edit#)

**Câu 1.**

**Phần 1.1** Giải thuật Lan Truyền Ngược (Backpropagation) làm việc theo cách tiếp cận tối ưu hóa nào sau đây:  
A. Giải thuật Newton B. Dùng nhân tử Lagrange  
C. Suy giảm độ dốc (gradient descent) D. Gia tăng độ dốc (gradient ascent)  
E. Tất cả các câu trên đều sai. (0.25 điểm)  
 **→ Chương 7**

**Phần 1.2** Phát biểu nào sau đây là không đúng  
A. Bài toán *quy hoạch toàn phương* (Quadratic Programming) liên quan đến máy vector hỗ trợ thuộc loại tối ưu hóa có ràng buộc (constrained optimization). *→ Chương 8*  
B. Trong giải thuật ADABOOST, việc xây dựng các bộ phận lớp thành phần là độc lập với nhau.*→ Chương 9*  
C. Bagging và Rừng Ngẫu nhiên sử dụng cùng chiến lược phiếu bầu: đa số phiếu (majority voting).*→ Chương 9*  
D. Quá trình huấn luyện mạng nơ ron truyền thẳng thường nhanh hơn quá trình huấn luyện mạng nơ ron RBF.*→ Chương 7*  
E. Gibb Sampling là một kỹ thuật toán học được dùng để huấn luyện mạng Deep Belief network mà dựa vào máy Boltzmann giới hạn. *→ Chương 12*  
F. Học không có giám sát không cần đến thông tin nhân lớp của các mẫu trong tập dữ liệu huấn luyện. *→ Chương 6*  (0.5 điểm)

**→ Chương 6 + Chương 7 + Chương 8 + Chương 9 + Chương 12**

**Câu 2. (0.75 điểm)** Khi xác định cấu hình của mạng nơ ron cho một bài toán phân lớp, căn cứ vào đâu để xác định số nút của tầng nhập và số nút của tầng xuất? Và bằng cách nào xác định số nút thích hợp cho tầng ẩn ?  
 **→ Chương 7**

**Câu 3. *(trùng với câu 1.*** [***Fin\_202.pdf\_HKII\_(2020-2021)***](https://docs.google.com/document/d/1DkbvgkUJOOQcx9eNk-DsMT5MwyhNMvLE/edit#)***)***

**Câu 4.** (0.5 điểm) Liệt kê những tham số trong mạng nơ ron RBF mà chúng ta phải xác định khi huấn luyện mạng nơ ron này.  
 **→ Chương 7**

## **II. GIẢI ĐỀ** …To do …

#### **1.** [**Fin\_201.pdf\_HKI\_(2020-2021)**](https://docs.google.com/document/d/1O30YVMcYqIh3FvXyK30TFXqQB4cgPsMO/edit#heading=h.k859l5ku3md)

##### **Câu 3. (0.5 điểm)**

Nêu điểm khác biệt giữa giải thuật *suy giảm độ dốc* (gradient descent) và giải thuật *suy giảm độ dốc tăng dần* (incremental gradient descent) khi huấn luyện một đơn vị perceptron tuyến tính.  
**→ Chương 7.**

###### **Đáp Án:**

**[Lân]**

[**https://datascience.stackexchange.com/questions/36450/what-is-the-difference-between-gradient-descent-and-stochastic-gradient-descent?newreg=ed7a92be3d1e4dc59e8bf0c6a7654394**](https://datascience.stackexchange.com/questions/36450/what-is-the-difference-between-gradient-descent-and-stochastic-gradient-descent?newreg=ed7a92be3d1e4dc59e8bf0c6a7654394)

In both gradient descent (GD) and stochastic gradient descent (SGD), you update a set of parameters in an iterative manner to minimize an error function.

While in GD, you have to run through ALL the samples in your training set to do a single update for a parameter in a particular iteration, in SGD, on the other hand, you use ONLY ONE or SUBSET of training sample from your training set to do the update for a parameter in a particular iteration. If you use SUBSET, it is called Minibatch Stochastic gradient Descent.

Thus, if the number of training samples are large, in fact very large, then using gradient descent may take too long because in every iteration when you are updating the values of the parameters, you are running through the complete training set. On the other hand, using SGD will be faster because you use only one training sample and it starts improving itself right away from the first sample.

SGD often converges much faster compared to GD but the error function is not as well minimized as in the case of GD. Often in most cases, the close approximation that you get in SGD for the parameter values are enough because they reach the optimal values and keep oscillating there.

*Trong cả giảm độ dốc (GD) và giảm độ dốc ngẫu nhiên (SGD), bạn cập nhật một bộ tham số theo cách lặp lại để giảm thiểu hàm lỗi.*

*Khi ở GD, bạn phải chạy qua TẤT CẢ các mẫu trong tập huấn luyện của mình để thực hiện một cập nhật duy nhất cho một tham số trong một lần lặp cụ thể, trong SGD, mặt khác, bạn CHỈ sử dụng MỘT hoặc TẬP HỢP CON mẫu huấn luyện từ tập huấn luyện của mình để thực hiện cập nhật cho một tham số trong một lần lặp cụ thể. Nếu bạn sử dụng SUBSET, nó được gọi là Minibatch Stochastic gradient Descent.*

*Do đó, nếu số lượng mẫu đào tạo lớn, trên thực tế là rất lớn, thì việc sử dụng độ dốc giảm dần có thể mất quá nhiều thời gian vì trong mỗi lần lặp lại khi bạn cập nhật giá trị của các tham số, bạn sẽ chạy qua toàn bộ tập huấn luyện. Mặt khác, sử dụng SGD sẽ nhanh hơn vì bạn chỉ sử dụng một mẫu đào tạo và nó bắt đầu tự cải thiện ngay từ mẫu đầu tiên.*

*SGD thường hội tụ nhanh hơn nhiều so với GD nhưng chức năng lỗi không được giảm thiểu tốt như trong trường hợp GD. Thông thường, trong hầu hết các trường hợp, giá trị gần đúng mà bạn nhận được bằng SGD cho các giá trị tham số là đủ vì chúng đạt đến giá trị tối ưu và tiếp tục dao động ở đó.*

|  | **Gradient Descent  (Suy giảm độ dốc)** | **Incremental Gradient Descent  (Suy giảm độ dốc tăng dần)** |
| --- | --- | --- |
| Tập dữ liệu (dataset) dùng để tính độ dốc | Sử dụng toàn bộ mẫu trong tập dữ liệu để huấn luyện | Dùng từng mẫu hoặc một nhóm mẫu dữ liệu để huấn luyện |
| Cập nhật tham số (Parameters update) | Cập nhật các thông số của mô hình 1 lần duy nhất trong 1 lần lặp trên toàn bộ tập dữ liệu, và xem xét độ dốc (gradient) của hàm chi phí trên toàn bộ tập dữ liệu | Cập nhật các thông số sau khi xử lí từng mẫu hoặc nhóm mẫu dữ liệu, và xem xét độ dốc (gradient) của hàm chi phí trên mẫu hoặc nhóm mẫu dữ liệu tương ứng |
| Hiệu suất tính toán (Performance) | Thấp hơn so với IGD do GD yêu cầu tính toán trên toàn bộ tập dữ liệu trong mỗi vòng lặp | Cao hơn so với GD vì chỉ xử lí một phần của tập dữ liệu cho mỗi lần lặp |
| Hội tụ (convergence) | GD hội tụ mượt mà hơn (smoothy) trong việc tối thiểu hàm chi phí vì xem xét độ dốc một cách tổng quan trên toàn bộ tập dữ liệu | IGD hội tụ nhanh hơn so với GD do IGD chỉ huấn luyện trên một phần mẫu của tập dữ liệu nhưng lại không tốt bằng GD trong việc tối ưu hóa hàm chi phí |

##### **Câu 4 (0.75 điểm)**

Khi xác định cấu hình của mạng nơ ron cho một bài toán phân lớp, căn cứ vào đâu để xác định số nút của tầng nhập và số nút của tầng xuất? Và bằng cách nào xác định số nút thích hợp cho tầng ẩn ?  
**→ Chương 7 *(Trùng với câu 9.*** [**Fin\_12\_2021.pdf\_HKI\_(2021-2022)**](https://docs.google.com/document/d/1FBqWzRhwBudjJDBuK2_z72pM98EndxA2/edit#heading=h.t37itj2wog4b)**)**

###### **Đáp Án**

**[Lân]**

Tầng nhập: số chiều (feature) của mẫu dữ liệu đầu vào (input)

Tầng xuất: số lượng nhãn dùng để gán cho mẫu dữ liệu.

Tầng ẩn: (quá trình thử và sai) là quá trình tinh chỉnh và điều chỉnh dựa trên kinh nghiệm và thực nghiệm, thử nghiệm và đánh giá nhiều cấu hình khác nhau giúp tìm ra một mô hình mạng nơ ron hiệu quả cho bài toán phân lớp:

1. Quá khớp (overfitting) và thiếu độ khớp (underfitting): Quan sát hiện tượng quá độ khớp và thiếu độ khớp trên mô hình mạng nơ ron có thể giúp xác định số nút thích hợp cho tầng ẩn. Nếu mô hình có quá độ khớp, có thể cần giảm số nút để tránh việc mô hình ghi nhớ quá nhiều chi tiết của dữ liệu huấn luyện và không tổng quát hóa tốt cho dữ liệu mới. Ngược lại, nếu mô hình có thiếu độ khớp, có thể cần tăng số nút để mô hình có đủ khả năng học các đặc trưng phức tạp của dữ liệu.

2. Phương pháp thử và sai: Bạn có thể thử nghiệm với các kiến trúc mạng khác nhau và số lượng nút trong tầng ẩn. Bắt đầu với một số nút nhỏ và tăng dần số lượng nút để xem hiệu suất của mô hình. Đánh giá các độ đo như độ chính xác (accuracy), độ mất mát (loss), hoặc các độ đo khác tùy thuộc vào bài toán phân lớp của bạn để xác định mô hình nào hoạt động tốt nhất trên tập dữ liệu kiểm tra hoặc tập dữ liệu chưa từng thấy.

3. Kỹ thuật chọn mô hình tổng quát (model selection techniques): Có nhiều kỹ thuật khác nhau để chọn mô hình tốt nhất, bao gồm cross-validation, grid search, và Bayesian optimization. Các kỹ thuật này giúp tổng hợp thông tin từ việc thử nghiệm các cấu hình mạng khác nhau và chọn ra mô hình có hiệu suất tốt nhất dựa trên các tiêu chí xác định trước.

4. Điều chỉnh theo quy tắc heuristic: Có một số quy tắc thông số thô sơ có thể tham khảo. Ví dụ, *quy tắc thumb rule* có thể đề xuất số nút trong mỗi tầng ẩn nằm trong khoảng từ số nút của tầng nhập và tầng xuất. Tuy nhiên, quy tắc này chỉ là một hướng dẫn chung và không phải luôn phù hợp cho tất cả các bài toán.

#### **2.** [**Fin\_202.pdf\_HKII\_(2020-2021)**](https://docs.google.com/document/d/1DkbvgkUJOOQcx9eNk-DsMT5MwyhNMvLE/edit#)

##### **Câu 1. (0.5 điểm) *(****Trùng với câu 3.* [*Fin\_201.pdf\_HKI\_(2020-2021)*](https://docs.google.com/document/d/1O30YVMcYqIh3FvXyK30TFXqQB4cgPsMO/edit#heading=h.k859l5ku3md)*)*

Nêu điểm khác biệt giữa giải thuật *suy giảm độ dốc* (gradient descent) và giải thuật *suy giảm độ dốc tăng dần* (incremental gradient descent) khi huấn luyện một đơn vị perceptron tuyến tính.  
**→ Chương 7.**

##### **Câu 3. (1.25 điểm)**

Điều chỉnh lại giải thuật *lan truyền ngược* để giải thuật này có thể làm việc được với các nút sử dụng hàm kích hoạt *tanh* thay vì hàm sigmoid. Hàm *tanh* được định nghĩa như sau:

*tanh*(*x*) = (*ex e-x*)/(*ex* + *e-x*) *a^^Hỉỵ^e e* v*e e*

Tức là điều chỉnh lại qui tắc cập nhật trọng số tại tầng xuất và tầng ẩn. Lưu ý đạo hàm hàm *tanh* có tính chất: *tanh’*(*x*) = 1 - [*tanh*(*x*)]2.  
**→ Chương 7**

###### **Đáp án**

**[Lân]**

1) Giá trị trả về của hàm *sigmoid* trả về nằm trong khoảng [0; 1].



· Khi x tiến tới -∞: lim S(x) → 0

· Khi x tiến tới +∞: lim S(x) → 1

2) Giá trị trả về của hàm *tanh* nằm trong khoảng [-1; 1]

.

· Khi x tiến tới -∞: lim *tanh*(x) → -1

· Khi x tiến tới +∞: lim *tanh*(x) → 1

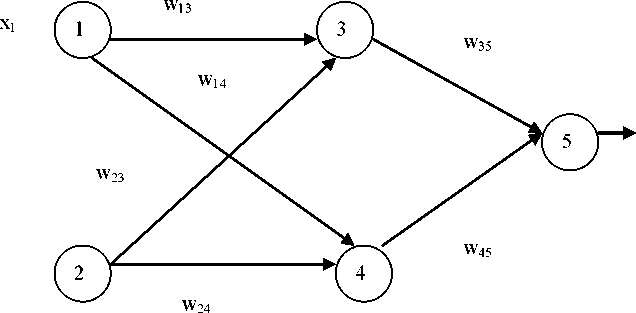
Do hàm *sigmoid* trả về giá trị nằm trong khoảng [0; 1] nên để hàm kích hoạt (activation function) *g(x)* vừa trả về kết quả cũng nằm trong khoảng [0;1] và vừa sử dụng hàm *tanh*, ta biến đổi hàm *g(x)* thành như sau:

Đạo hàm của g(x) đề dùng cho Backpropagation:



##### **Câu 4. (1.75 điểm)**

Cho mạng nơ ron có cấu hình như sau:



Cho các giá trị trọng số và các độ lệch (bias) khởi đầu như trong bảng sau:

w13 w14 w23 w24 w35 w45 w03 w04 w05

0.2 -0.3 0.4 0.1 -0.3 -0.2 -0.4 0.2 0.1

Giả sử hệ số học *7* = 0.8, hàm sigmoid được dùng ở tầng ẩn và tầng xuất, và giải thuật lan truyền ngược được dùng để huấn luyện mạng nơ ron. Nếu mẫu huấn luyện <x1 = **0**, x2 = **1** và output = **1**> được đưa vào giải thuật huấn luyện thì các trọng số và các độ lệch (w03, w04, w05) sẽ được điều chỉnh lại như thế nào?

**→ Chương 7.**

###### **Đáp án**

**[Lân]**

*x1* = 0

*x2* = 1

*output* = 1

Tính net input và output

| **unit j** | **net input Ij** |  | **Output Oj (sigmoid)** |
| --- | --- | --- | --- |
| 3 | x1 \* w13 + x2 \* w23 + w03  = 0\*0.2 + 1\*0.4 + (-0.4) | 0 | 0.55 |
| 4 | x1 \* w14 + x2 \* w24 + w04  = 0\*(-0.3) + 1\*0.1 + 0.2 | 0.3 | 0.525 |
| 5 | o3 \* w35 + o4 \* w45 + w05  = o3 \* (-0.3) + o4 \* (-0.2) + 0.1 | 0.5 | 0.542 |

Tính sai số cho từng nốt

| **unit j** | **delta\_j = oj \* (1 - oj) \* (tj - oj)** |  |
| --- | --- | --- |
| 5 | =o5 \* ( 1 - o5) \* (output - o5) | 0.114 |
| 4 | =o4 \* ( 1 - o4) \* o4 \* w45 | -0.006 |
| 3 | =o3 \* ( 1 - o3) \* o3 \* w35 | -0.008 |

Tính toán cập nhật lại trọng số

/\* delta = learning\_rate \* delta\_j \* xji\*/

| **weight** | **new value = old\_weight + delta**  **= old\_weight + learning\_rate \* delta\_j \* xji** |  |
| --- | --- | --- |
| w35 | = w35 + learning\_rate \* delta\_5 + o3 | -0.263 |
| w45 | = w45 + learning\_rate \* delta\_5 + o4 | -0.164 |
| w13 | = w13 + learning\_rate \* delta\_3 + x1 | 0.2 |
| w14 | = w14 + learning\_rate \* delta\_4 + x1 | -0.3 |
| w23 | = w23 + learning\_rate \* delta\_3 + x2 | 0.395 |
| w24 | = w24 + learning\_rate \* delta\_4 + x2 | 0.096 |
| w05 | = w05 + learning\_rate \* delta\_5 + output | 0.168 |
| w04 | = w04 + learning\_rate \* delta\_4 + output | 0.196 |
| w03 | = w03 + learning\_rate \* delta\_3 + output | -0.405 |

**[Toàn]**

**Áp dụng cùng công thức nhưng khác nhau ở bước 1. Lưu ý ở các bước làm.**

**B1:** Tính net input và output

| **Unit j** | **Net input Ij** | **Output Oj (hàm sigmoid 1/(1+e^-x)** |
| --- | --- | --- |
| 3 | x1 \* w13 + x2 \* w23 + w03 = 0\*0.2 + 1\*0.4 + (-0.4) = 0 | 1/(1+e^0)=0.5 |
| 4 | x1 \* w14 + x2 \* w24 + w04= 0\*(-0.3) + 1\*0.1 + 0.2  = 0.3 | 1/(1+e^-0.3)=0.574 |
| 5 | o3 \* w35 + o4 \* w45 + w05 = 0.5 \* (-0.3) + 0.574 \* (-0.2) + 0.1 = -0.265  **[Lâm] chỗ này -0.164** | 1/(1+e^0.265)=0.434  **0.46** |

**B2:** Tính sai số cho từng nốt

| Unit j | delta\_j = oj \* (1 - oj) \* (tj - oj) |
| --- | --- |
| 5 (Công thức 14) | 0.434\*(1-0.434)\*(1-0.434) = 0.139 |
| 4 (Công thức 15) | 0.574\*(1-0.574)\*(0.139)\*(-0.2) = -0.0068 |
| 3 (Công thức 15) | 0.5\*(1-0.5)\*0.139(-0.3) = -0.0104 |

**B3**: Tính toán cập nhật lại trọng số

| weight | New value (Công thức 16) |
| --- | --- |
| w35 | = w35 + learning\_rate \* delta\_5\*O3 = -0.3+0.6\*0.139\*0.5=-0.258 |
| w45 | = w45 + learning\_rate \* delta\_5\*O4 =-0.2+0.6\*0.139\*0.574=-0.152 |
| w13 | = w13 + learning\_rate \* delta\_3\*x1=0.2+0.6\*(-0.0104)\*0=0.2 |
| w14 | = w14 + learning\_rate \* delta\_4\*x1=-0.3+0.6\*(-0.0068)\*0=-0.3 |
| w23 | = w23 + learning\_rate \* delta\_3\*x2=0.4+0.6\*(-0.0104)\*1=0.394 |
| w24 | = w24 + learning\_rate \* delta\_4\*x2=0.1+0.6\*(-0.0068)\*1=0.096 |
| w05 | = w05 + learning\_rate \* delta\_5 = 0.1+0.6\*0.139 =0.1834 |
| w04 | = w04 + learning\_rate \* delta\_4 = 0.2+0.6\*-0.0068=0.196 |
| w03 | = w03 + learning\_rate \* delta\_3 = -0.4+0.6\*-0.0104=-0.406 |

#### **3.** [**Fin\_12\_2021.pdf\_HKI\_(2021-2022)**](https://docs.google.com/document/d/1FBqWzRhwBudjJDBuK2_z72pM98EndxA2/edit#heading=h.t37itj2wog4b)

##### **Câu 2.** *(Trùng với câu 3.* [Fin\_202.pdf\_HKII\_(2020-2021)](https://docs.google.com/document/d/1DkbvgkUJOOQcx9eNk-DsMT5MwyhNMvLE/edit#))

##### **Câu 3.** *(Trùng với câu 4.* [Fin\_202.pdf\_HKII\_(2020-2021)](https://docs.google.com/document/d/1DkbvgkUJOOQcx9eNk-DsMT5MwyhNMvLE/edit#))

##### **Câu 4. (0.5 pts)**

*Li*ệt *kê nh*ững tham số trong mạng nơ ron *RBF mà chúng ta ph*ải x*ác* định *khi hu*ấn luyện mạng nơ ron n*ày.* **→ Chương 7**

###### **Đáp án**

**[Lân]**

<https://www.mathworks.com/matlabcentral/fileexchange/52580-radial-basis-function-neural-networks-with-parameter-selection-using-k-means>

RBFNN have 5 parameters for optimization:

1) The weights between the hidden layer and the output layer.

2) The activation function.

3) The center of activation functions.

4) The distribution of the center of activation functions.

5) The number of hidden neurons.

*(RBFNN có 5 tham số để tối ưu hóa:*

*1) Trọng số giữa lớp ẩn và lớp đầu ra.*

*2) Chức năng kích hoạt.*

*3) Trung tâm của các chức năng kích hoạt.*

*4) Sự phân bố của tâm hàm kích hoạt.*

*5) Số lượng tế bào thần kinh ẩn.)*

Chapter 7

1) M: số lượng *hàm cơ sở xuyên tâm* (radial basis function)

2) *Hình dạng* (shape) của hàm radial basis

3) Hidden Nodes: số lượng nốt ở tầng ẩn

4) Center: điểm trung tâm của *nốt ẩn* (hidden node)

5) Spread: bán kính của *điểm trung tâm* (centroid)

6) Learning Rate: hệ số học

**[Trâm]**

Để huấn luyện mạng nơ ron RBF (Radial Basis Function), chúng ta cần xác định các tham số sau:

* **Số lượng các RBF neurons(Number of RBF neurons):** là số lượng các hạt nhân RBF được sử dụng trong mạng. Mỗi hạt nhân RBF tương ứng với một tâm và một hàm RBF.
* **Tọa độ của các tâm RBF (Centers of RBF neurons):** là vị trí của các tâm RBF trong không gian đầu vào. Thường thì chúng ta cần xác định các tâm này trước khi huấn luyện mạng. Một cách phổ biến để xác định các tâm là sử dụng phương pháp gom cụm (clustering) như K-means.
* **Tham số độ rộng của các hạt nhân RBF (Width parameter of RBF neurons):** là một tham số quan trọng để xác định độ phân tán của các hạt nhân RBF. Nó ảnh hưởng đến độ ảnh hưởng của mỗi hạt nhân RBF lên dữ liệu đầu vào.
* **Trọng số (weight) của các hạt nhân RBF (Weights of RBF neurons):** là các trọng số được gán cho mỗi hạt nhân RBF. Trọng số này quyết định độ ảnh hưởng của mỗi hạt nhân đối với đầu vào và kết quả dự đoán của mạng.
* **Trọng số (weight) của các kết nối giữa RBF layer và output layer (nếu có)(Weights of connections between the RBF layer and the output layer)**: Nếu mạng RBF có một lớp output để dự đoán đầu ra, chúng ta cần xác định các trọng số kết nối từ lớp RBF tới lớp output.
* **Tham số điều chuẩn (regularization parameter):** là một tham số quan trọng để kiểm soát độ phức tạp của mạng RBF và tránh overfitting. Tham số này ảnh hưởng đến quá trình huấn luyện và việc đánh đổi giữa việc khớp dữ liệu huấn luyện và khả năng tổng quát hóa của mô hình.

Những tham số này cần được xác định và điều chỉnh phù hợp trong quá trình huấn luyện mạng nơ ron RBF để đạt được kết quả tốt nhất.

##### **Câu 5. (0.75 điểm)**

1. Hãy nêu công dụng của hàm kernel đối với máy véc tơ hỗ trợ trong trường hợp dữ liệu không khả tách một cách tuyến tính. (*0.25 điểm*)  
   **→ Chương 7**

#### **4.** [**Fin\_12\_2021.pdf\_HKII\_(2021-2022)**](https://docs.google.com/document/d/1BLSqsCee6LnqI5rh2J_ZeN3whI-TOJl-VinGDjto97A/edit#)

##### **Câu 1.**

**Phần 1.1** Giải thuật Lan Truyền Ngược (Backpropagation) làm việc theo cách tiếp cận tối ưu hóa nào sau đây:  
A. Giải thuật Newton B. Dùng nhân tử Lagrange  
C. Suy giảm độ dốc (gradient descent) D. Gia tăng độ dốc (gradient ascent)  
E. Tất cả các câu trên đều sai. (0.25 điểm)  
 **→ Chương 7**

###### **Đáp Án**

**[Lân]** C. Suy giảm độ dốc (gradient descent)

#### **1.** [**Fin\_201.pdf\_HKI\_(2020-2021)**](https://docs.google.com/document/d/1O30YVMcYqIh3FvXyK30TFXqQB4cgPsMO/edit#heading=h.k859l5ku3md)

##### **Câu 1. (1 điểm)**

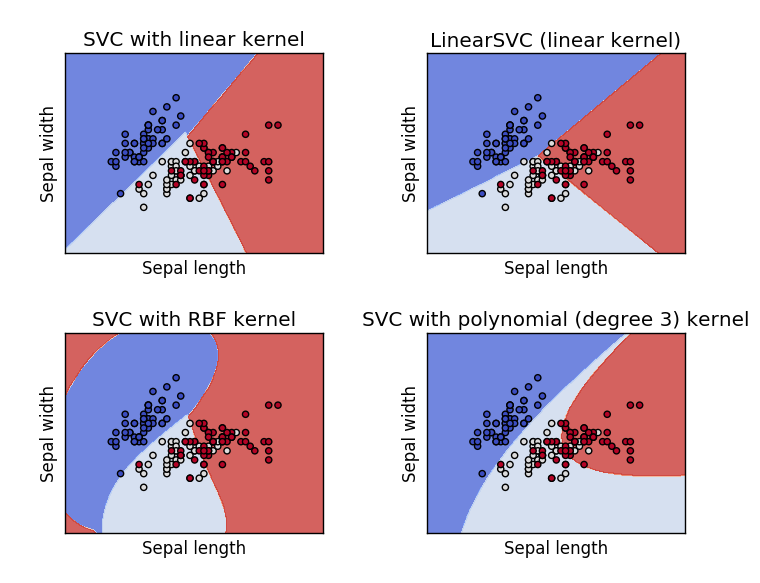
Trả lời các câu hỏi sau đây, mỗi câu 0.25 điểm

**1.1** Cho một tập dữ liệu không khả tách một cách tuyến tính, bộ phân lớp nhị phân nào sau đây có thể tách tập dữ liệu ra làm hai lớp một cách đúng đắn:

SVM với RBF kernel

###### **Đáp Án câu 1.1**

**[Tâm]**

****

**C. SVM với RBF kernel.**

Trong bài toán phân lớp, **khả tách tuyến tính (linearly separable) ám chỉ trạng thái khi tập dữ liệu của hai lớp có thể được tách ra bằng một đường thẳng, một siêu phẳng hoặc một siêu mặt phẳng tuyến tính**. Trong trường hợp này, một bộ phân lớp tuyến tính như SVM tuyến tính hay perceptron có thể được sử dụng để tìm ra đường phân chia chính xác giữa các lớp.

Tuy nhiên, khi tập dữ liệu **không khả tách tuyến tính (linearly inseparable), không tồn tại một đường thẳng, một siêu phẳng hoặc một siêu mặt phẳng tuyến tính có thể tách các điểm dữ liệu của hai lớp**. Trong trường hợp này, cần sử dụng các bộ phân lớp phi tuyến (nonlinear classifiers) hoặc các phương pháp chuyển đổi dữ liệu sang không gian cao hơn để tìm một đường phân chia không tuyến tính.

Trong trường hợp tập dữ liệu không khả tách tuyến tính, SVM với RBF kernel (Radial Basis Function) thường cho phép phân lớp một cách chính xác hơn so với SVM tuyến tính hoặc perceptron. **Kernel RBF cho phép phân lớp các dữ liệu không khả tách tuyến tính bằng cách chuyển đổi chúng vào không gian cao hơn, nơi chúng có thể được phân lớp một cách tuyến tính**.

Perceptron sử dụng perceptron training rule không phải là một phương pháp hiệu quả để xử lý tập dữ liệu không khả tách tuyến tính. SVM tuyến tính cũng không thể tách một tập dữ liệu không khả tách tuyến tính một cách chính xác.

Vì vậy, đáp án đúng là C. SVM với RBF kernel.

**1.3** (Đúng/Sai)

SVM và ANN có thể được dùng cho cả hai bài toán phân lớp và hồi quy.

###### **Đáp án câu 1.3**

**[Tâm]**

Đúng. Cả SVM (Support Vector Machines) và ANN (Artificial Neural Networks) có thể được sử dụng cho cả bài toán phân lớp và hồi quy.

Mục tiêu chính trong bài toán hồi quy là tìm một mô hình tốt nhất để ước lượng giá trị đầu ra dựa trên các đặc trưng đầu vào.

Bài toán phân lớp (classification) mục tiêu là dự đoán lớp hoặc nhãn (label) của một mẫu dữ liệu dựa trên các đặc trưng (features) của nó. Mỗi mẫu dữ liệu thường được biểu diễn bằng một vectơ đặc trưng, và mục tiêu là phân loại mẫu dữ liệu vào một trong các lớp đã được xác định trước.

[Nam] Đây là lí do mạng nơ ron có thể xấp xỉ các hàm số tốt và không cần biết trước rằng hàm số đó có dạng gì, có là đa thức hay không. Và kết quả của quá trình huấn luyện sẽ là một quan hệ được biểu diễn bằng một hàm phức tạp thể hiện bằng các tham số của mạng nơ ron. Trong thực tế, các đơn vị tính toán cơ bản nhất trong mạng nơ ron được lấy cảm hứng từ các thuật toán học máy truyền thống như SVM hoặc hồi quy logistic.

=> Dong y phuong an tra loi la True

##### **Câu 5 (1.5 điểm)**

1. Giải thích các thuật ngữ khoảng biên (margin), siêu phẳng tách với khoảng biên tối đa (maximum margin hyperplane), siêu phẳng hỗ trợ (support hyperplane), véc tơ hỗ trợ (support vector). (*1 điểm*)

###### **Đáp án câu c**

**[Tâm]**

Trong SVM (Support Vector Machines), các thuật ngữ sau đây được sử dụng để mô tả các khái niệm quan trọng:

* ***Khoảng biên (Margin):*** Là khoảng cách giữa siêu phẳng tách và các mẫu dữ liệu gần nhất của các lớp khác nhau. Khoảng biên càng lớn, mô hình SVM càng tổng quát hóa tốt hơn và có khả năng phân loại tốt hơn các mẫu dữ liệu mới. Khoảng cách giữa support vector và siêu phẳng tách
* ***Siêu phẳng tách với khoảng biên tối đa (Maximum Margin Hyperplane):*** Là siêu phẳng tách lớn nhất mà có thể tách các mẫu dữ liệu của các lớp khác nhau. Siêu phẳng này được tìm kiếm trong quá trình huấn luyện SVM và được tối ưu hóa để có khoảng biên lớn nhất.
* ***Siêu phẳng hỗ trợ (Support Hyperplane):*** Là siêu phẳng tách gần nhất với các mẫu dữ liệu của các lớp khác nhau. Siêu phẳng hỗ trợ chứa các điểm dữ liệu gần nhất đến đường phân chia, và nó quyết định vị trí của đường phân chia và khoảng biên. Là 2 siêu phẳng cách đều siêu phẳng tách với khoảng cách là margin
* ***Véc tơ hỗ trợ (Support Vector):*** Là các điểm dữ liệu nằm trên siêu phẳng hỗ trợ. Các điểm dữ liệu này là quan trọng nhất trong việc xác định siêu phẳng tách và khoảng biên. Véc tơ hỗ trợ chỉ ra các mẫu dữ liệu quan trọng nhất để xác định đường phân chia.

Trong SVM, mục tiêu là tìm một siêu phẳng tách tối ưu có khoảng biên lớn nhất và tối thiểu hóa sai số phân loại. Siêu phẳng tách và các véc tơ hỗ trợ chính là thành phần quan trọng để xác định đường phân chia và hỗ trợ quan trọng nhất cho quá trình phân loại.

1. Giải thích ý nghĩa của các biến bù *xi* và thông số *C* trong máy vec tơ hỗ trợ với khoảng biên mềm (soft margin) (*0.5 điểm*)

###### **Đáp án câu d**

**[Tâm]**

Biến bù xi (Slack Variables): Biến bù xi được sử dụng để đo lường mức độ vi phạm của các điểm dữ liệu đến khoảng biên. Mỗi điểm dữ liệu có thể có một giá trị bù xi tương ứng, cho biết mức độ vi phạm của nó đối với siêu phẳng tách.

* Nếu giá trị bù xi là 0, điểm dữ liệu nằm chính xác trên siêu phẳng tách.
* Nếu giá trị bù xi là dương và nhỏ hơn 1, điểm dữ liệu nằm trong vùng an toàn và gần siêu phẳng tách.
* Nếu giá trị bù xi là lớn hơn 1, điểm dữ liệu vi phạm quy tắc phân loại và nằm ở phía sai lớp.

Mục tiêu của quá trình tối ưu hóa là tìm một mô hình SVM tối ưu với sự phạm vi vi phạm của các điểm dữ liệu được kiểm soát thông qua việc điều chỉnh các giá trị bù xi.

Thông số C: Thông số C là một tham số quan trọng trong SVM với khoảng biên mềm. Nó kiểm soát mức độ mà mô hình cho phép vi phạm các điểm dữ liệu.

* Khi C càng lớn, mô hình SVM càng cứng và không chấp nhận sự phạm vi vi phạm cao hơn. Điều này có nghĩa là mô hình sẽ cố gắng tìm kiếm một siêu phẳng tách chặt chẽ và giảm thiểu vi phạm.
* Khi C nhỏ, mô hình SVM trở nên linh hoạt hơn và cho phép sự phạm vi vi phạm lớn hơn. Điều này có thể dẫn đến sự chấp nhận một số điểm dữ liệu vi phạm để tạo ra một đường phân chia tổng quát hơn và phù hợp với dữ liệu huấn luyện.

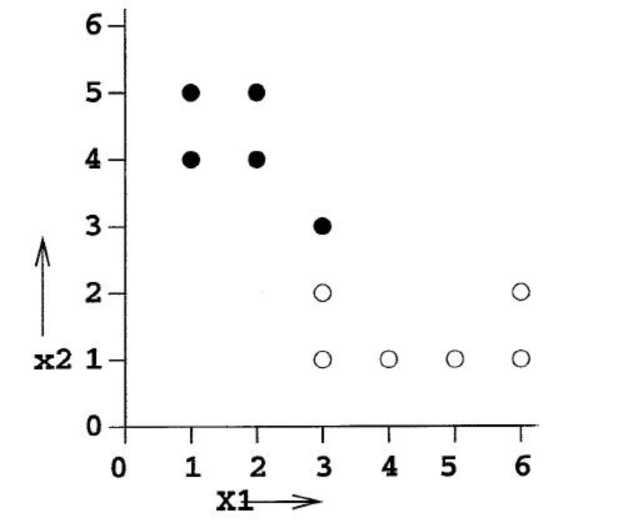
Lựa chọn giá trị C phù hợp là một quá trình cân nhắc giữa việc tối đa hóa sự phân loại chính xác và vi phạm dữ liệu. Một giá trị C lớn sẽ đánh đổi sự linh hoạt để giảm thiểu vi phạm, trong khi một giá trị C nhỏ sẽ tạo ra một siêu phẳng tách mềm hơn và cho phép vi phạm đến một mức độ nhất định.

#### **2.** [**Fin\_202.pdf\_HKII\_(2020-2021)**](https://docs.google.com/document/d/1DkbvgkUJOOQcx9eNk-DsMT5MwyhNMvLE/edit#)

##### **Câu 5. (1.75 điểm)**

1. Cho một tập dữ liệu mà mỗi điểm dữ liệu có hai thuộc tính (x1 and x2) và một nhãn lớp như trong hình vẽ sau. Các điểm thuộc lớp dương ký hiệu bằng dấu chấm đậm và các điểm thuộc lớp âm ký hiệu bằng những hình tròn nhỏ.

Giả sử chúng ta đang dùng SVM tuyến tính để phân lớp. Hãy vẽ 3 đường thẳng biểu diễn siêu phẳng tách có khoảng biên cực đại và hai siêu phẳng hỗ trợ. Tính khoảng biên cực đại. (*1 điểm*)



###### **Đáp Án**

**[Tâm]**

Với bài toán phân loại SVM tuyến tính, chúng ta cần tìm một siêu phẳng tách để phân chia hai lớp dữ liệu. Siêu phẳng này sẽ có dạng w1\*x1 + w2\*x2 + b = 0

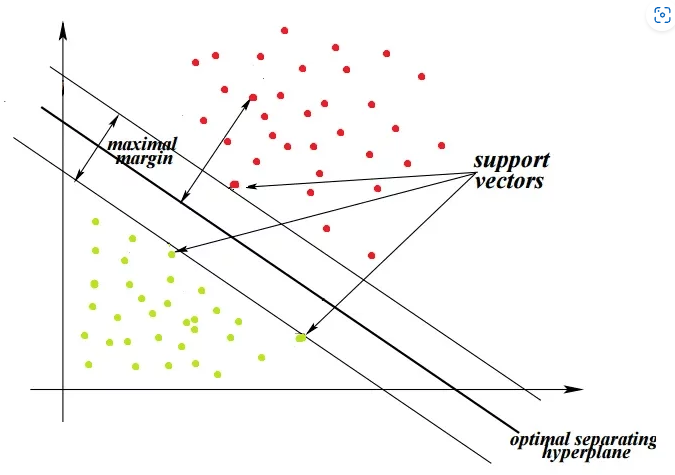
w1, w2 là tham số b là hệ số điều chỉnh

**Tìm đường biên có lề lớn nhất**

Siêu phẳng tách: w1\*x1 + w2\*x2 + b = 0 (Separating hyperplane)

Siêu phẳng hỗ trợ bên trái: w1\*x1 + w2\*x2 + b = -1 (Left support hyperplane)

Siêu phẳng hỗ trợ bên phải: w1\*x1 + w2\*x2 + b = 1 (Right support hyperplane)



ảnh minh họa

Để tìm ra đường biên có độ rộng lề là lớn nhất thì chúng ta cần **tối đa hoá khoảng cách từ các điểm thuộc tập hỗ trợ tới đường biên**. Điều này tương đương với giải bài toán tối ưu:

Để tính khoảng cách giữa hai siêu phẳng hỗtrợ, ta sử dụng công thức khoảng cách giữa một điểm và một siêu phẳng trong không gian hai chiều:

distance = |w \* x\_i + b| / ||w||

Khoảng cách giữa hai siêu phẳng hỗ trợ sẽ bằng đường kính của đường tròn tối ưu, và được tính bằng 2 / ||w||.

Với tập dữ liệu trên, ta có thể giải bài toán tối ưu hóa bằng phương pháp Lagrange và tính toán khoảng cách giữa hai siêu phẳng hỗ trợ như sau:

Bước 1: Biến đổi bài toán tối ưu hóa thành dạng tương đương:

minimize 1/2 ||w||^2 - sum(alpha\_i \* (y\_i \* (w \* x\_i + b) - 1))

subject to alpha\_i >= 0 for i = 1,2,...,n

sum(alpha\_i \* y\_i) = 0

Bước 2: Giải hệ phương trình tối đa hóa đốivới các hệ số Lagrange alpha\_i. Ta có thể sử dụng công thức sau để tính vector trọng số w và hệ số điều chỉnh bias b:

w = sum(alpha\_i \* y\_i \* x\_i)

b = y\_j - sum(alpha\_i \* y\_i \* (x\_i \* x\_j)) for any alpha\_j > 0

Trong đó, j là chỉ số của một support vector.

Bước 3: Tính khoảng cách giữa hai siêu phẳng hỗ trợ bằng công thức:

distance = 2 / ||w||

Với tập dữ liệu trên, ta có:

Chọn một support vector, ví dụ (3, 2), và tính hệ số Lagrange alpha\_j tương ứng:

alpha\_j = 1 / ||w|| = 1 / sqrt(13)

Tính vector trọng số w và hệ số điều chỉnh bias b:

w = sum(alpha\_i \* y\_i \* x\_i) = (1/13)\*(1, 3)

b = y\_j - sum(alpha\_i \* y\_i \* (x\_i \* x\_j)) = -5/13

Do đó, siêu phẳng tách có phương trình w \* x+ b = 0 là:

x2 = (3/13)\*x1 - (5/13)

Bước 4: Tìm hai siêu phẳng hỗ trợ bằng cách tìm các support vectors. Ta có:

Lớp dương: các điểm (3, 3), (1, 4), (2, 4), (1, 5), (2, 5)

Lớp âm: các điểm (3, 2), (3, 1), (4, 1), (5, 1), (6, 1), (6, 2)

Các support vectors là các điểm có hệ số Lagrange alpha\_i > 0. Ta có:

Lớp dương: các điểm (1, 4), (1, 5), (2, 5)

Lớp âm: các điểm (3, 2), (4, 1)

Do đó, hai siêu phẳng hỗ trợ là:

Siêu phẳng hỗ trợ đầu tiên: tương ứng với các điểm support vectors của lớp dương và có phương trình w \* x + b = 1:

x2 = (3/13)\*x1 - (8/13)

Siêu phẳng hỗ trợ thứ hai: tương ứng với các điểm support vectors của lớp âm và có phương trình w \* x + b = -1:

x2 = (3/13)\*x1 - (2/13)

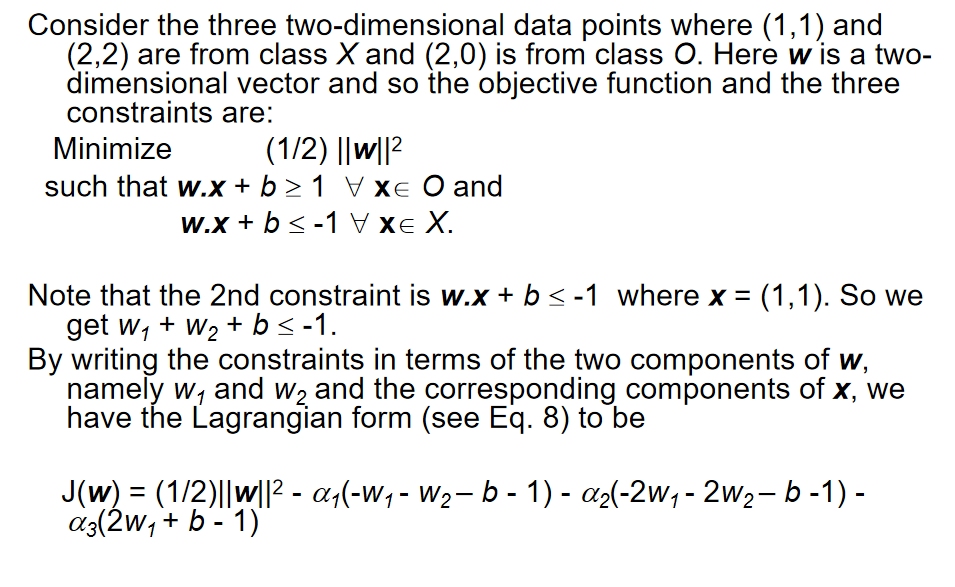
Bước 5: Tính khoảng cách giữa hai siêu phẳng hỗ trợ bằng công thức:

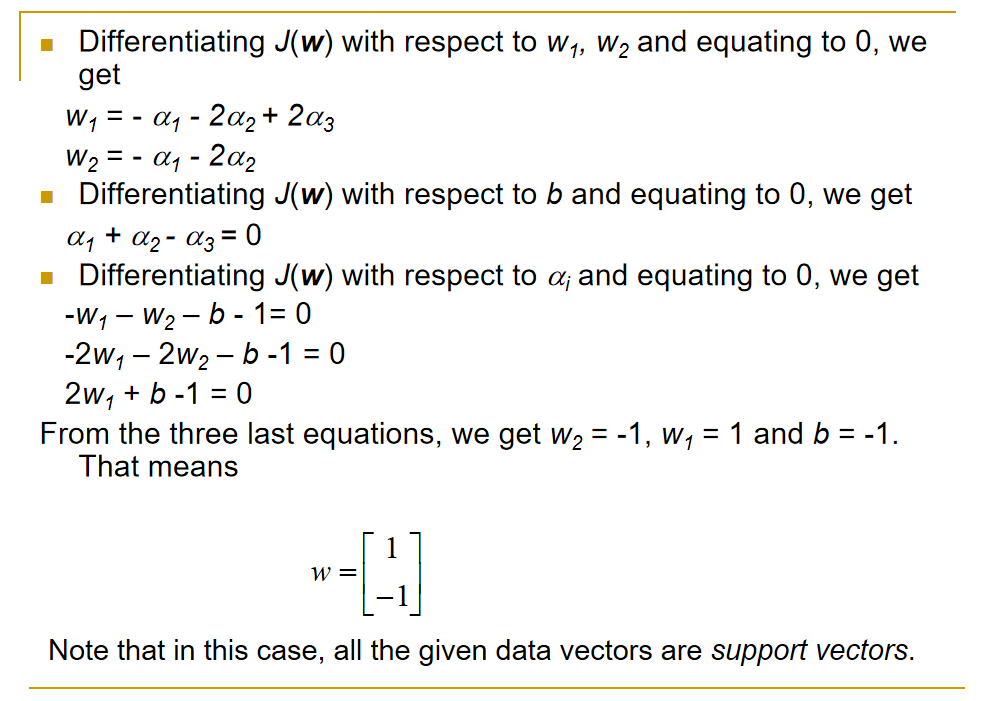
distance = 2 / ||w|| = 2 / sqrt(13)

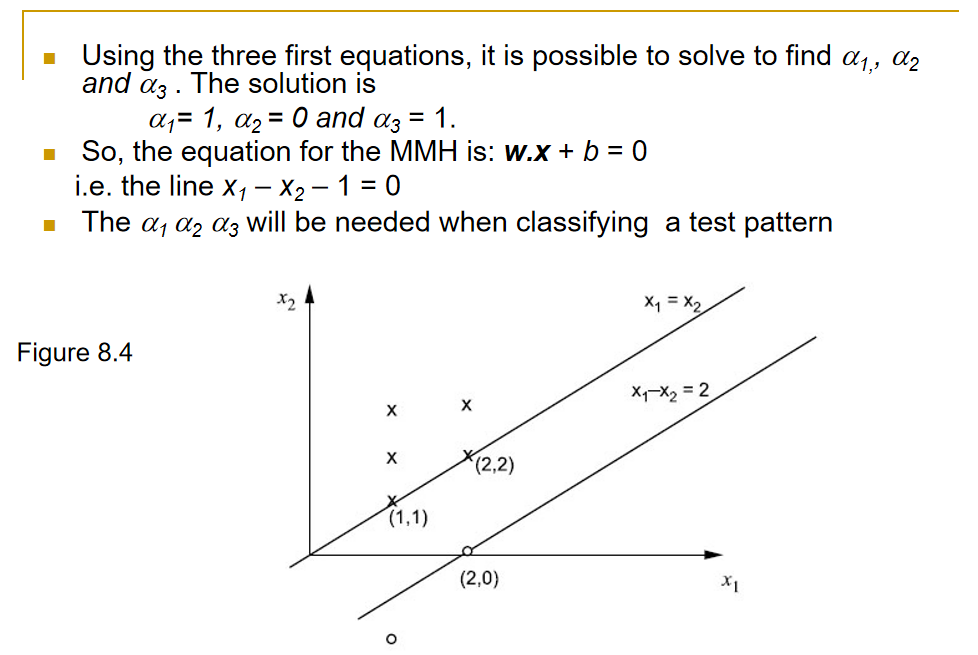
Vậy, khoảng cách giữa hai siêu phẳng hỗ trợ là 2 / sqrt(13), và đường kính của đường tròn tối ưu là 2 / sqrt(13). Các điểm dữ liệu nằm trong đường tròn tối ưu được phân lớp vào lớp dương, và các điểm dữ liệu nằm ngoài đường tròn tối ưu được phân lớp vào lớp âm..

Ở bài này. Ta cũng có thể làm giống như slide 25 chương 8. Bài tập ví dụ như sau:

Làm giống như bài này, để tìm ra phương trình của biên quyết định. Cũng như độ lớn của Margin - khoảng biên.







Sau khi tìm được được W ta sẽ có phương trình của biên quyết định

Sau đó sẽ tìm được margin - độ rộng của khoảng biên.

**b. Hãy nêu công dụng của hàm kernel đối với máy véc tơ hỗ trợ trong trường hợp dữ liệu không khả tách một cách tuyến tính. (*0.25 điểm*)**

Hàm kernel là một phương pháp được sử dụng trong máy học để giải quyết vấn đề phân loại dữ liệu không khả tách một cách tuyến tính. Trong SVM, hàm kernel được sử dụng để ánh xạ dữ liệu từ không gian dữ liệu ban đầu sang một không gian mới nơi dữ liệu có thể được phân loại tốt hơn bằng một siêu phẳng tách tuyến tính.

Các hàm kernel thường được sử dụng trong SVM bao gồm:

Hàm đa thức (Polynomial Kernel): Đây là hàm kernel đơn giản nhất và được sử dụng phổ biến nhất trong SVM. Hàm đa thức ánh xạ dữ liệu từ không gian ban đầu sang một không gian mới bằng cách sử dụng một đa thức bậc cao hơn để giúp phân loại dữ liệu tốt hơn.

Hàm Gauss (Gaussian Kernel): Đây là hàm kernel phổ biến khác được sử dụng để xử lý dữ liệu không khả tách một cách tuyến tính trong SVM. Hàm Gauss ánh xạ dữ liệu từ không gian ban đầu sang một không gian mới bằng cách sử dụng một phân phối Gauss để tính toán độ tương đồng giữa các điểm dữ liệu. Hàm này cho phép phân loại các điểm dữ liệu theo các siêu phẳng không tuyến tính.

Hàm Sigmoid (Sigmoid Kernel): Hàm kernel này ánh xạ dữ liệu từ không gian ban đầu sang một không gian mới bằng cách sử dụng một hàm sigmoid tuyến tính. Hàm này thường được sử dụng trong bài toán phân loại nhị phân.

**Các hàm kernel này giúp máy véc tơ hỗ trợ có thể phân loại dữ liệu không khả tách một cách tuyến tính bằng cách ánh xạ dữ liệu từ không gian ban đầu sang một không gian mới có số chiều lớn hơn. Khi dữ liệu được ánh xạ sang không gian mới, nó có thể được phân loại tốt hơn bằng một siêu phẳng tách tuyến tính. Hàm kernel không chỉ giúp giải quyết vấn đề phân loại dữ liệu không khả tách một cách tuyến tính mà còn giúp giảm chi phí tính toán và không gian lưu trữ dữ liệu khi áp dụng SVM cho các tập dữ liệu lớn.**

**c. Giải thích ý nghĩa của các biến bù *Epsilon* và thông số *C* trong máy vec tơ hỗ trợ với khoảng biên mềm (soft margin)**

 là một siêu tham số, cho phép có bao nhiêu điểm dữ liệu được phép vi phạm ràng buộc khoảng biên.

C is a parameter chosen by the user that controls the tradeoff between the margin and the misclassification errors. A larger C means that a higher penalty to misclassification errors is assigned.

C điều tiết sự đánh đổi giữa độ rộng của khoảng biên và số lỗi do phân lớp sai. C Lớn Có nghĩa là penalty cao hơn cho lỗi phân lớp.

Đôi với Epsilon:

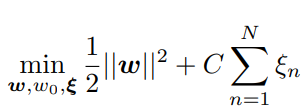
Nếu dữ liệu không khả tách tuyến tính, sẽ không có nghiệm khả thi cho ràng buộc:

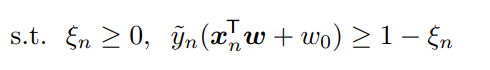


Vi vậy, Chúng ta giới thiệu biến dư - Slack Variables -

Sau đó thay thế ràng buộc cứng  bằng ràng buộc độ lớn khoảng biên mềm, 

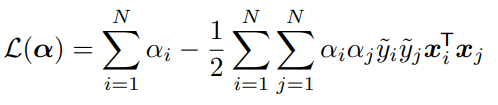
Vậy hàm mục tiêu mới là:



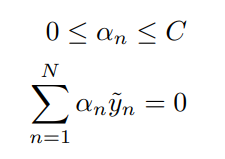


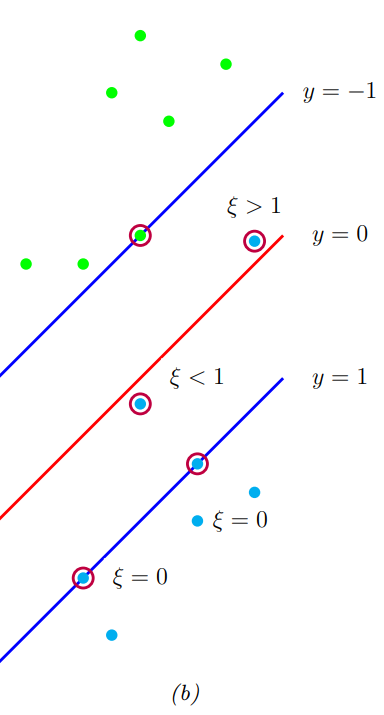


Tối ưu hóa w, w0, Epsilon ta bằng Lagrange đối ngẫu:



Mặc dù đối ngẫu giống hệt với trường hợp độ rộng khoảng biên cứng - Hard Margin. Nhưng khác nhau về KKT condition:





Nếu:

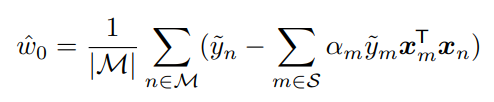
 điểm dữ liệu sẽ bị bỏ qua.

 thì,  vậy điểm dữ liệu sẽ nằm trên khoảng biên.

 thì điểm dữ liệu sẽ nằm bên trong khoảng biên. Lúc này

1.  có thể, có khả năng điểm được phân lớp đúng
2.  điểm được phân lớp sai

 là biên trên - upper bound của của số lượng điểm bị phân lớp sai



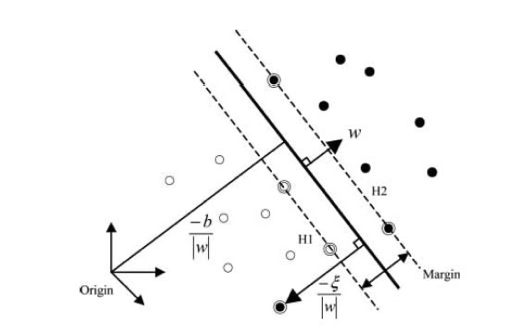
với:  là tập hợp các điểm có 

#### **3.** [**Fin\_12\_2021.pdf\_HKI\_(2021-2022)**](https://docs.google.com/document/d/1FBqWzRhwBudjJDBuK2_z72pM98EndxA2/edit#heading=h.t37itj2wog4b)

**Câu 5. (0.75 điểm)**

**b.** Giải thích ý nghĩa của các biến bù *Epsilon* và thông số *C* trong máy vec tơ hỗ trợ với khoảng biên mềm (soft margin) (*0.5 điểm*)  
**→ Chương 8**

[Nghia]

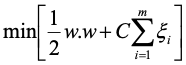


1. Biến bù (Slack variables):

- Các biến bù được sử dụng để đo lường mức độ vi phạm của mỗi điểm dữ liệu đến siêu phẳng tối ưu.

- Mỗi điểm dữ liệu có thể có một biến bù tương ứng. Giá trị của biến bù cho biết mức độ vi phạm của điểm dữ liệu đó đến siêu phẳng. Nếu biến bù có giá trị lớn, điểm dữ liệu đó được coi là vi phạm mức độ lớn hơn và có khả năng bị phân loại sai. Nếu biến bù có giá trị nhỏ hoặc bằng 0, điểm dữ liệu đó nằm đúng phía của siêu phẳng hoặc nằm trong vùng an toàn.

- Mục tiêu là tìm một giải pháp tối ưu (minimize hàm dưới) mà tổng các biến bù là nhỏ nhất để đạt được sự cân bằng giữa việc tìm siêu phẳng tối ưu và vi phạm của các điểm dữ liệu.



2. Thông số C:

- Thông số C được sử dụng để kiểm soát mức độ phân loại sai sót mà chúng ta chấp nhận trong SVM với khoảng biên mềm.

- Thông số C quyết định sự cân nhắc giữa việc tìm một siêu phẳng tối ưu và vi phạm của các điểm dữ liệu. Nếu C lớn, mức độ vi phạm sẽ nhỏ, điều này dẫn đến việc SVM sẽ cố gắng tìm một siêu phẳng chặt chẽ hơn, nhưng có khả năng overfitting.

- Ngược lại, nếu C nhỏ, SVM sẽ chấp nhận một mức độ vi phạm lớn hơn và tìm một siêu phẳng có độ rộng lớn hơn, dẫn đến tính tổng quát hóa tốt hơn trên dữ liệu mới.

- Lựa chọn thông số C phụ thuộc vào đặc tính của dữ liệu và mục tiêu của bài toán. Nếu dữ liệu nhiễu nhiều hoặc có sự chồng lấn mạnh giữa các lớp, ta có thể sử dụng C lớn để giảm thiểu phân loại sai sót. Ngược lại, nếu dữ liệu ít nhiễu và phân tách rõ ràng, ta có thể sử dụng C nhỏ để đạt được tính tổng quát hóa tốt hơn.

#### **4.** [**Fin\_12\_2021.pdf\_HKII\_(2021-2022)**](https://docs.google.com/document/d/1BLSqsCee6LnqI5rh2J_ZeN3whI-TOJl-VinGDjto97A/edit#)

Câu 5 **(*Trùng* với câu 5.** [**Fin\_202.pdf\_HKII\_(2020-2021)**](https://docs.google.com/document/d/1DkbvgkUJOOQcx9eNk-DsMT5MwyhNMvLE/edit#)**)**