TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CƠ KHÍ
BỘ MÔN CƠ ĐIỆN TỬ

ĐỀ THI CUỐI HỌC KỲ
MÔN ĐIỀU KHIỂN HỆ THỐNG 3
Ngày thi: 20/06/2011
Thời gian thi: 75 phút

Sinh viên không được sử dụng tài liệu

Câu 1: (2,0 đ)

Nêu cơ sở để lập thuật toán học cho mạng Perceptron và các bước lập trình mạng Perceptron

Câu 2: (2,0 đ)

Nêu sự khác nhau giữa adaline và perceptron

Câu 3: (4,0 đ)

a. Nêu sự khác nhau giữa static network và dynamic network. Nêu ứng dụng của dynamic network.
b. Nêu ứng dụng của mạng trễ hội tụ (Focused Time-Delay Neural Network) và các bước thiết lập mạng trễ hội tụ (FTDNN)

Câu 4: (2,0 đ)

Nêu nguyên tắc huấn luyện cho mạng feedforward và nguyên tắc lan truyền ngược.

Chủ nhiệm bộ môn

Giảng viên ra đề thi

TS. Nguyễn Duy Anh

ĐÁP ÁN ĐỀ THI
MÔN ĐIỀU KHIỂN HỆ THỐNG 3
Ngày thi: 20/06/2011
Thời gian: 75 phút

Câu 1: (2,0 đ)	Điểm
Cơ sở để ta lập thuật toán học cho mạng perceptron như sau: Ta có 3 trường hợp: **Case 1:** ***a = t => e = 0*** : ***w*** không thay đổi **Case 2:** ***a=0, t=1=> e=t-a =1*** : ***w = w+p*** vì lúc đó ***w*** sẽ gần ***p*** hơn và có khả năng cho kết quả ***a =1*** cao hơn **Case 3:** ***a=1,t=0=> e=t-a = -1*** : ***w = w-p*** vì lúc đó ***w*** sẽ xa ***p*** hơn và có khả năng cho kết qủa ***a = 0*** cao hơn. Từ đó suy ra nguyên tắc học thống nhất: $${}_1w^{new} = {}_1w^{old} + ep = {}_1w^{old} + (t-a)p$$ $$b^{new} = b^{old} + e$$	1,0
Các bước lập trình mạng Perceptron Bước 1 : Chọn tốc độ học Bước 2 : Khởi động - gán sai số E=0 - gán biến chạy k=1 - gán các vector trọng số $w_i(k)$ (i=1,n) bằng giá trị ngẫu nhiên nhỏ bất kì Bước 3 : Quá trình huấn luyện bắt đầu, tính : $y_i(k) = step(w_i^T(k)x(k)) = step\left(\sum_{j=1}^{m} w_{ij}(k)x_j(k)\right) \quad (i=\overline{1,n})$ Bước 4 : Cập nhận các vector trọng số $w_i(k+1) = w_i(k) + \eta(d_i(k) - y_i(k))x(k) \quad (i=\overline{1,n})$ Bước 5 : Tính sai số tích lũy $E = E + \frac{1}{2}\|d(k) - y(k)\|^2$ Bước 6 : Nếu k < K thì gán k=k+1 và trở lại Bước 3. Nếu k=K thì tiếp tục Bước 7 Bước 7 : Kết thúc một chu kì huấn luyện (epoch). Nếu E=0 thì kết thúc quá trình học. Nếu E≠0 thì gán E=0, k=1 và trở lại Bước 3 bắt đầu một chu kì huấn luyện mới.	1,0
Câu 2: (2,0 đ)	
Adaline - Mạng tuyến tính giống như perceptron, nhưng hàm truyền của nó tuyến tính hơn hard-limiting. Ngõ ra của adaline có thể là giá trị bất kỳ, trong khi ngõ ra của perceptron bị giới hạn trong khoảng 0 và 1. Mạng tuyến tính, giống như perceptron, chỉ có thể giải quyết những vấn đề tuyến tính riêng biệt.	2,0
Câu 3: (4,0 đ)	
Static network: Không có phần tử hồi tiếp và delay. Đầu ra được tính từ đầu vào thông qua các kết nối feedforward	1,0

Dynamic network: Đầu ra phụ thuộc vào đầu vào hiện tại, ngoài ra còn phụ thuộc vào các đầu vào và đầu ra trước đó và trạng thái của mạng. Dynamic network gồm hai loại Có các kết nối feedforward Các kết nối hồi tiếp hoặc recurrent. Dynamic network thường mạnh hơn Static network (mặc dù khó huấn luyện hơn). Bởi vì Dynamic network có khả năng nhớ, chúng có thể được huấn luyện để học tuần tự hoặc các đối tượng thay đổi theo thời gian.	1,0
Chúng có những ứng dụng trong những lĩnh vực khác nhau như là: dự đoán thị trường tài chính, cân bằng kênh trong hệ thống thông tin, phát hiện pha trong hệ thống năng lượng, sắp xếp, dự đoán cấu túc protein trong di truyền. Một ứng dụng chính của mạng nơron động là trong điều khiển hệ thống. Mạng động còn rất thích hợp cho việc lọc.	0,5
Mạng trễ hội tụ (Focused Time-Delay Neural Network) rất thích hợp để dự đoán theo chuỗi thời gian.	0,5
Các bước chính: Bước 1: Load dữ liệu, bình thường hóa nó, và chuyển nó thành tuần tự theo thời gian (thực hiện bởi mảng các phần tử). Bước 2: tạo một mạng FTDNN ,dùng lệnh *newfftd*. Bước 3: Sắp xếp ngõ vào và đích (target)của mạng cho việc huấn luyện. Bước 4: mô phỏng mạng và xác định sai số trong dự đoán.	1,0
Câu 4 (2,0 đ)	
Feedforward Khi trọng số và bias được khởi tạo, nghĩa là mạng đã sẵn sàng cho việc huấn luyện. Mạng có thể được huấn luyện để xấp xỉ hàm số, liên kết mẫu và phân loại mẫu. Trong quá trình huấn luyện, trọng số và bias được điều chỉnh để cực tiểu hóa hàm hiệu suất của mạng *net.performFcn*. Hàm hiệu suất mặc định là sai số bình phương trung bình *mse*	
Lan truyền ngược Nguyên tắc cơ bản của lan truyền ngược là cập nhật trọng số theo vòng lặp: Trong đó x_k là vector trọng số và bias hiện tại, gk là gradient hiện tại và α_k là tốc độ học. Có hai cách để thực hiện thuật toán suy giảm độ dốc: chế độ tích lũy và chế độ khối *Huấn luyện khối:* Hàm huấn luyện suy giảm độ dốc khối là *traingd*. Nếu muốn sử dụng thuật toán này thì ta thiết lập *trainFcn* thành *traingd* Có 7 thông số liên quan tới traingd: *epochs, show, goal, time, min_grad, max_fail, lr* *Suy giảm độ dốc với xung lượng:* Suy giảm độ dốc với xung lượng được thực hiện bởi hàm *traindm*, cho phép mạng không chỉ đáp ứng với gradient cục bộ mà còn với hướng vừa qua của mặt phẳng sai số. Suy giảm độ dốc với xung lượng phụ thuộc vào 2 yếu tố: tốc độ học *lr*, và hằng số xung lượng *mc*	