ĐẠI HỌC CẦN THƠ

MẠNG NƠ-RON NHÂN TẠO

Mã số: CT384, 3 Tín chỉ
(KT Điện tử VT, KT Điều khiển và Cơ – Điện tử)

TS. Nguyễn Chí Ngôn
Bộ môn Tự Động Hóa
Khoa Kỹ thuật Công nghệ
Email: ncngon@ctu.edu.vn

---2008---

Nội Dung

- Chương 1: Tổng quan về mạng nơron nhân tạo (ANN)
- Chương 2: Cấu trúc của ANN
- **Chương 3: Các giải thuật huấn luyện ANN**
- **Chương 4: Một số ứng dụng của ANN (MATLAB)**
- Đồ án môn học
- Chương 5: Mạng nơron mờ (Fuzzy-Neural Networks)
- Chương 6: Một số định hướng nghiên cứu (Case Studies)
- Ôn tập và thảo luận

Tham khảo:

1. *Nguyễn Chí Ngôn, Điều khiển mô hình nội và Neural network: Chương 2 – Mạng nơ-ron nhân tạo, Luận án cao học, ĐHBK Tp. HCM, 2001.*
2. *Nguyễn Đình Thúc, Mạng nơron – Phương pháp và ứng dụng, NXBGD, 2000.*
3. *Simon Haykin, Neural Networks a comprehensive foundation, Prentice Hall, 1999.*
4. *Howard Demuth, Mark Beale and Martin Hagan, Neural Networks toolbox 5 – User's Guide, The Matworks Inc., 2007.*

Tổ chức môn học

- Thời lượng môn học: 3TC
  - 2 TC lý thuyết và bài tập trên lớp
  - 1 TC Đồ án môn học (03 SV thực hiện 1 đề tài)
- Lịch học:
  - Tuần 1: Chương 1
  - Tuần 2 – 3: Chương 2 + Bài tập
  - Tuần 4 – 5: Chương 3 + Bài tập
  - Tuần 6 – 7: Chương 4 + Bài tập
  - Tuần 8 – 11: Đồ án môn học
  - Tuần 12 – 13: Chương 5 + Bài tập
  - Tuần 14: Chương 6
  - Tuần 15: Ôn tập và thảo luận
- Đánh giá
  - Đồ án môn học: 45%
  - Thi hết môn: 55%

Chương 3
Các giải thuật huấn luyện ANN

Giới thiệu

Các phương pháp huấn luyện

Một số giải thuật thông dụng

Hàm mục tiêu

Mặt lỗi và các điểm cực tiểu cục bộ

Qui trình thiết kế một ANN

Các kỹ thuật phụ trợ

Minh họa bằng MATLAB

Bài tập

Giới thiệu

- Gới thiệu về các phương pháp huấn luyện
- Tìm hiểu một số giải thuật thông dụng để huấn luyện ANN. Phần này tập trung vào giải thuật Gradient descent và các giải thuật cải tiến của nó
- Hàm mục tiêu
- Mặt lỗi và các điểm cực tiểu cục bộ
- Một số ví dụ về phương pháp huấn luyện mạng bằng MATLAB
- Qui trình thiết kế một ANN
- Các kỹ thuật phụ trợ
- Hiện tượng quá khớp của ANN

Các phương pháp huấn luyện

- Huấn luyện mạng là quá trình thay đổi các trọng số kết nối và các ngưỡng của nơ-ron, dựa trên các mẫu dữ liệu học, sao cho thỏa mãn một số điều kiện nhất định.
- Có 3 phương pháp học:
  - Học giám có sát (supervised learning)
  - Học không giám sát (unsupervised learning)
  - Học tăng cường (reinforcement learning).

  ☞ **Sinh viên tham khảo tài liệu [1].**
- Giáo trình này chỉ tập trung vào phương pháp học có giám sát. Hai phương pháp còn lại, sinh viên sẽ được học trong chương trình Cao học.

Giải thuật huấn luyện ANN (1)

- Trong phần này chúng ta tìm hiểu về giải thuật truyền ngược (backpropagation) và các giải thuật cải tiến của nó, áp dụng cho phương pháp học có giám sát.
- Giải thuật truyền ngược cập nhật các trọng số theo nguyên tắc:

$$w_{ij}(k+1) = w_{ij}(k) + \eta g(k)$$

trong đó:

wij(k) là trọng số của kết nối từ nơ-ron j đến nơ-ron i, ở thời điểm hiện tại

η là tốc độ học (learning rate, 0< η ≤1)

g(k) là gradient hiện tại

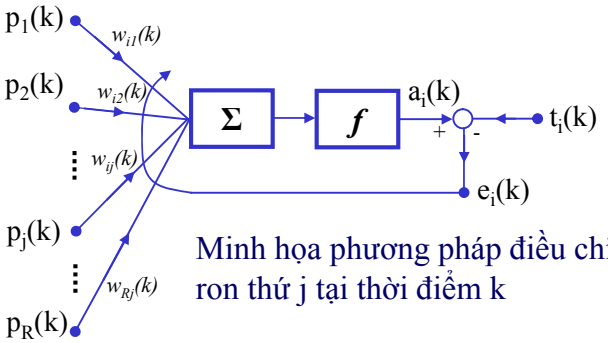
Có nhiều phương pháp xác định gradient *g(k),* dẫn tới có nhiều giải thuật truyền ngược cải tiến.

Giải thuật huấn luyện ANN (2)

- Để cập nhật các trọng số cho mỗi chu kỳ huấn luyện, giải thuật truyền ngược cần 2 thao tác:

Thao tác truyền thuận (forward pass phase): Áp vectơ dữ liệu vào trong tập dữ liệu học cho ANN và tính toán các ngõ ra của nó.

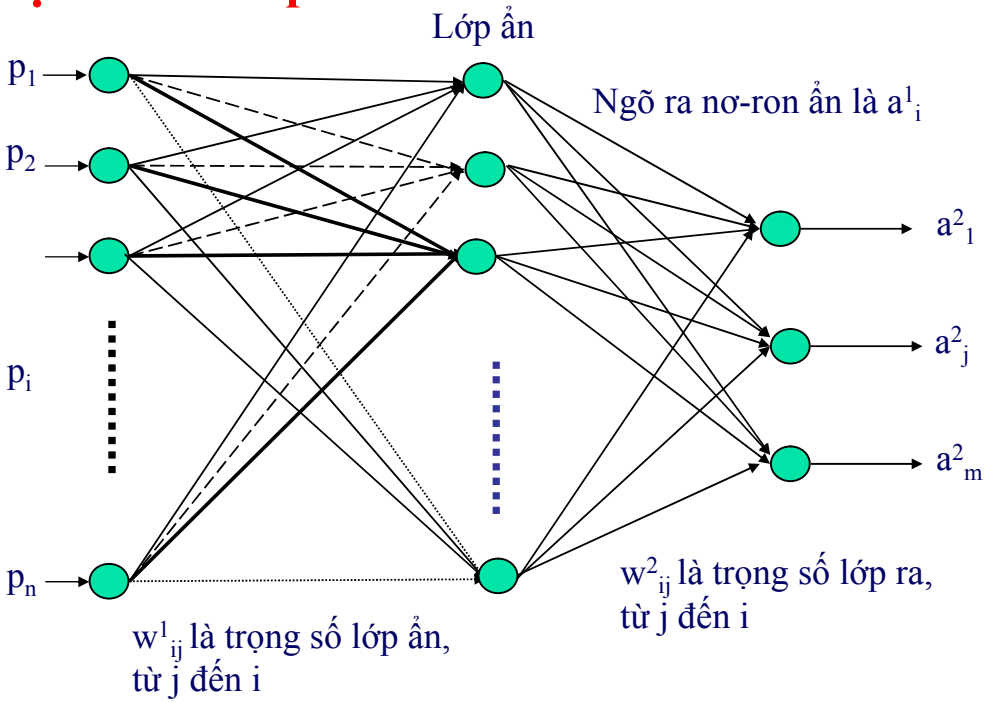
Thao tác truyền ngược (backward pass phase): Xác định sai biệt (lỗi) giữa ngõ ra thực tế của ANN và giá trị ngõ ra mong muốn trong tập dữ liệu học. Sau đó, truyền ngược lỗi này từ ngõ ra về ngõ vào của ANN và tính toán các giá trị mới của các trọng số, dựa trên giá trị lỗi này.



Minh họa phương pháp điều chỉnh trọng số nơ-ron thứ j tại thời điểm k

Giải thuật gradient descent (1)

Xét một MLP 2 lớp:



Giải thuật gradient descent (2)

Thao tác truyền thuận

- Tính ngõ ra lớp ẩn (hidden layer):

  $n^1_i(k) = \Sigma_j w^1_{ij}(k)\ p_j(k)$ tại thời điểm k

  $a^1_i(k) = f^1(n^1_i(k))$

  với f^1 là hàm kích truyền của các nơ-ron trên lớp ẩn.

  Ngõ ra của lớp ẩn là ngõ vào của các nơ-ron trên lớp ra.

- Tính ngõ ra ANN (output layer):

  $n^2_i(k) = \Sigma_j w^2_{ij}(k)\ a^1_j(k)$ tại thời điểm k

  $a^2_i(k) = f^2(n^2_i(k))$

  với f^2 là hàm truyền của các nơ-ron trên lớp ra

Giải thuật gradient descent (3)

Thao tác truyền ngược

- Tính tổng bình phương của lỗi: $E(k) = \frac{1}{2}\sum_i [t_i(k) - a^2{}_i(k)]^2$
  với *t(k)* là ngõ ra mong muốn tại *k*
- Tính sai số các nơ-ron ngõ ra:

$$\Delta_i(k) = -\frac{\partial E(k)}{\partial n^2{}_i(k)} = [t_i(k) - a_i(k)] f^2{}'[n^2{}_i(k)]$$

- Tính sai số các nơ-ron ẩn:

$$\delta_i(k) = -\frac{\partial E(k)}{\partial n^1{}_i(k)} = f^1{}'[n^1{}_i(k)] \sum_j \Delta_j(k) w_{ji}$$

- Cập nhật trọng số

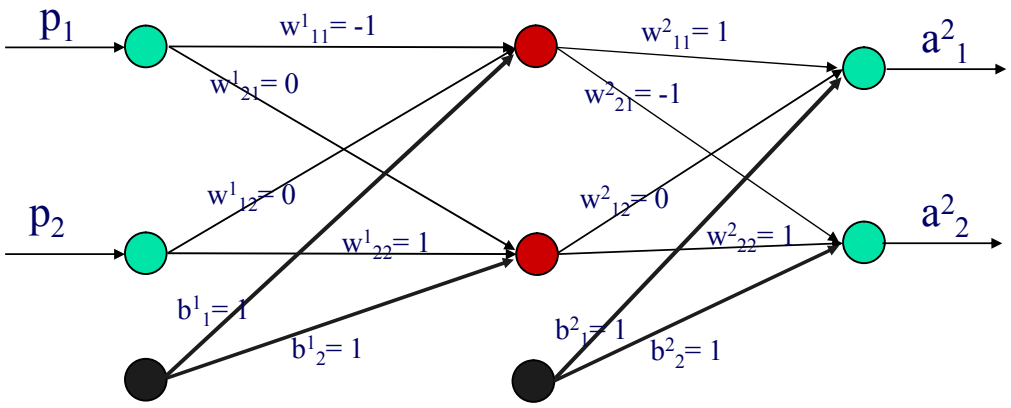
lớp ẩn: $w^1{}_{ij}(k+1) = w_{ij}(k) + \eta \delta_i(k) p_j$

lớp ra: $w^2{}_{ij}(k+1) = w^2{}_{ij}(k) + \eta \Delta_i(k) a^1{}_j(k)$

Sinh viên tham khảo tài liệu [3], trang 161-175.

Minh họa giải thuật huấn luyện (1)

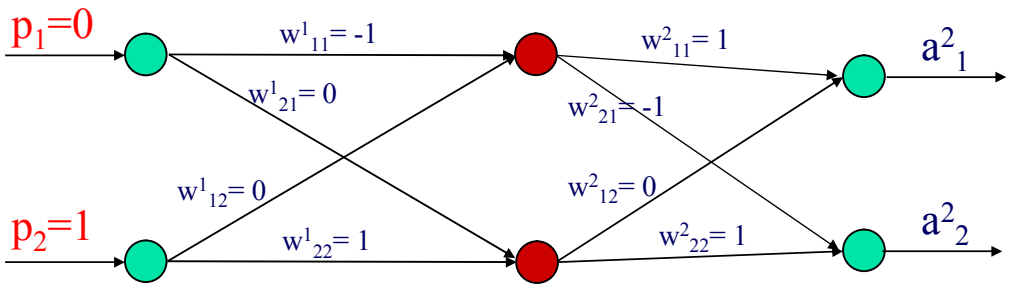
Xét một ANN như hình vẽ, với các nơ-ron tuyến tính.



Minh họa giải thuật truyền ngược như sau

Minh họa giải thuật huấn luyện (2)

Để đơn giản, ta cho cả các ngưỡng bằng 1, và không vẽ ra ở đây

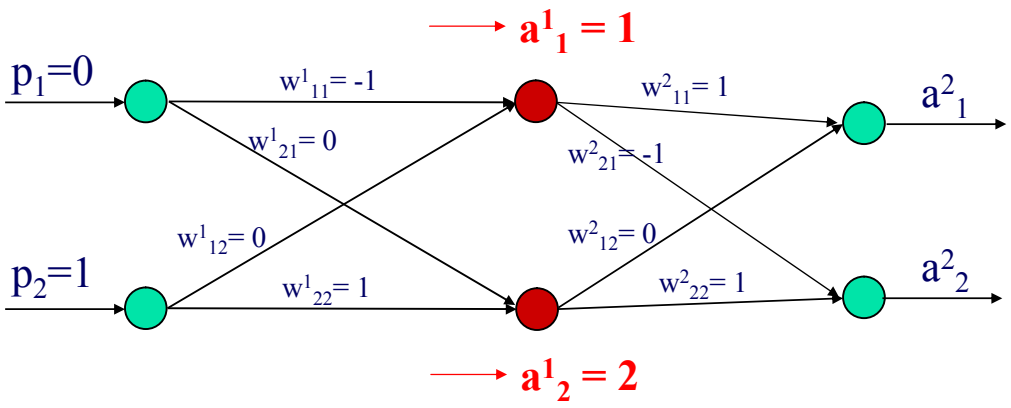


Giả sử ta có ngõ vào **p**=[0 1]' và ngõ ra mong muốn **t**=[1 0]'

Ta sẽ xem xét từng bước quá trình cập nhật trọng số của mạng với tốc độ học $\eta=0.1$

Minh họa giải thuật huấn luyện (3)

Thao tác truyền thuận. Tính ngõ ra lớp ẩn:

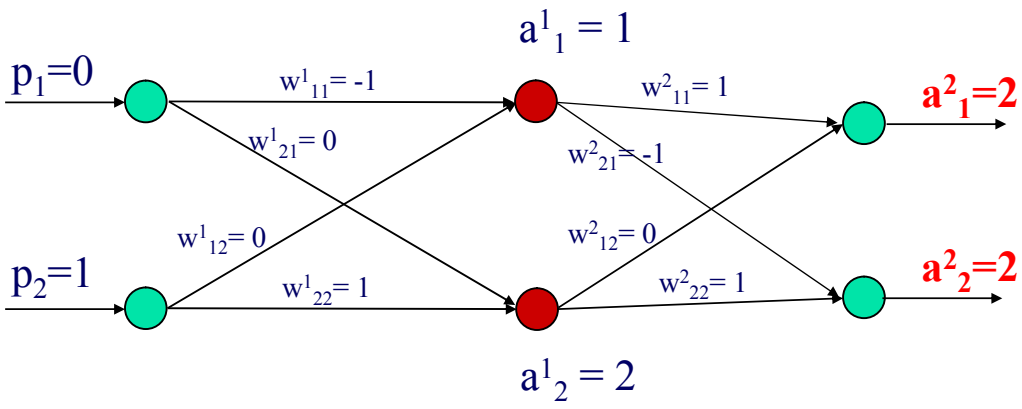


$a^1_1 = f^1(n^1_1) = n^1_1 = (-1*0 + 0*1) + 1 = 1$

$a^1_2 = f^1(n^1_2) = n^1_2 = (0*0 + 1*1) + 1 = 2$

Minh họa giải thuật huấn luyện (4)

Tính ngõ ra của mạng (lớp ra):



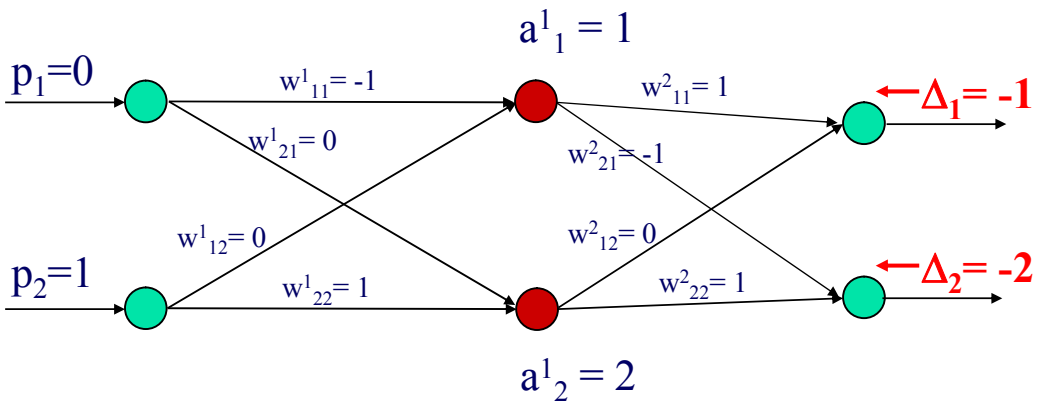
$a^2_1 = f^2(n^2_1) = n^2_1 = (1*1 + 0*2) + 1 = 2$

$a^1_2 = f^2(n^2_2) = n^2_2 = (-1*1 + 1*2) + 1 = 2$

Ngõ ra $\mathbf{a}^2$ khác biệt nhiều với ngõ ra mong muốn $\mathbf{t}=[1\ 0]'$

Minh họa giải thuật huấn luyện (5)

Thao tác truyền ngược $w^2_{ij}(k+1) = w^2_{ij}(k) + \eta\Delta_i(k)a^1_j(k)$



Với ngõ ra mong muốn $\mathbf{t} = [1, 0]'$,

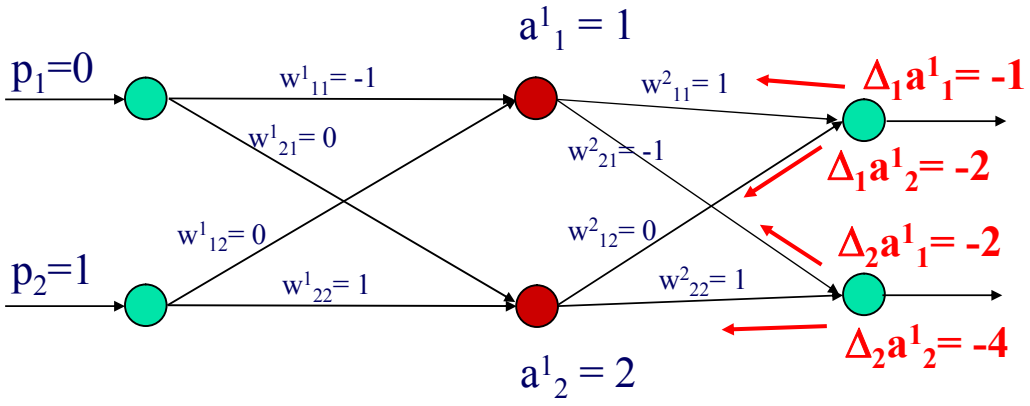
Ta có các error ở ngõ ra:

$\Delta_1 = (t_1 - a^2_1) = 1 - 2 = -1$

$\Delta_2 = (t_2 - a^2_2) = 0 - 2 = -2$

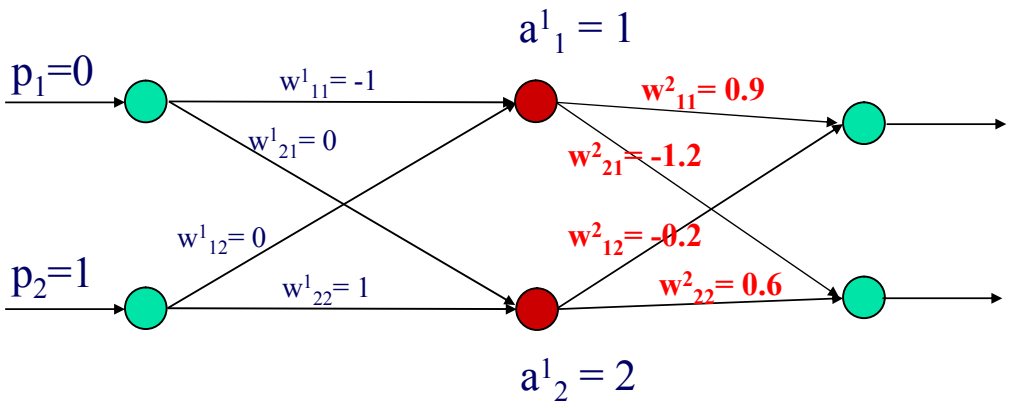
Minh họa giải thuật huấn luyện (6)

Tính các gradient lớp ra



Minh họa giải thuật huấn luyện (7)

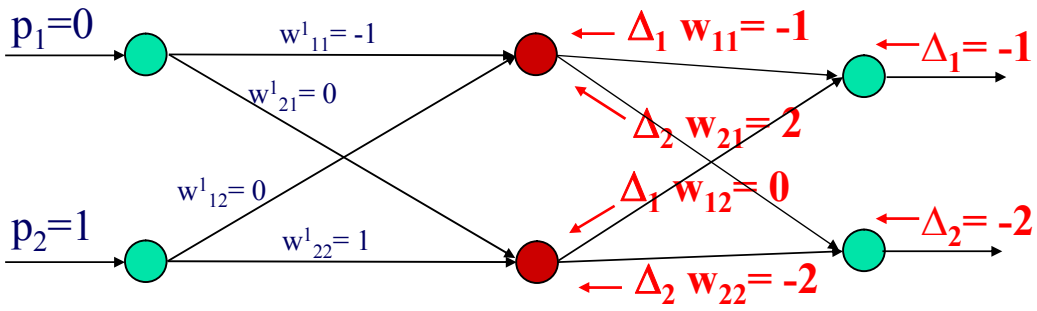
Cập nhật trọng số lớp ra $w^2_{ij}(k+1) = w^2_{ij}(k) + \eta\Delta_i(k)a^1_j(k)$



$w^2_{11}= 1 + 0.1*(-1) = 0.9$
$w^2_{21}= -1 + 0.1*(-2) = -1.2$
$w^2_{12}= 0 + 0.1*(-2) = -0.2$
$w^2_{22}= 1 + 0.1*(-4) = 0.6$

Minh họa giải thuật huấn luyện (8)

Tiếp tục truyền ngược

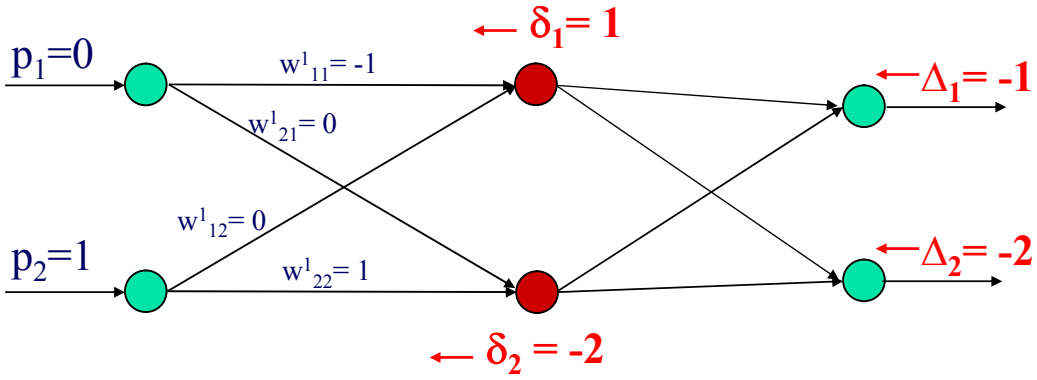


Sử dụng lại các trọng số trước khi cập nhật cho lớp ra, để tính gradient lớp ẩn

$$\delta_i(k) = f^{1\prime}\left[n^1{}_i(k)\right]\sum_j \Delta_j(k) w_{ji}$$

Minh họa giải thuật huấn luyện (9)

Tính các error trên lớp ẩn

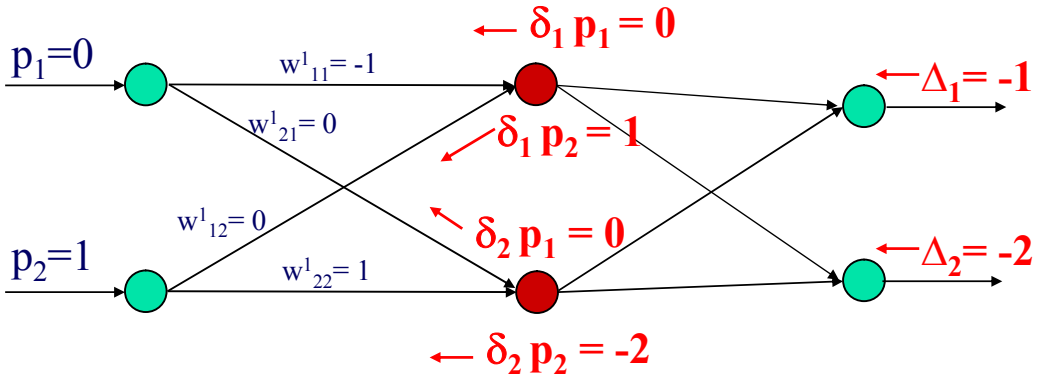


$\delta_1 = \Delta_1 w_{11} + \Delta_2 w_{21} = -1 + 2 = 1$

$\delta_2 = \Delta_1 w_{12} + \Delta_2 w_{22} = 0 - 2 = -2$

Minh họa giải thuật huấn luyện (10)

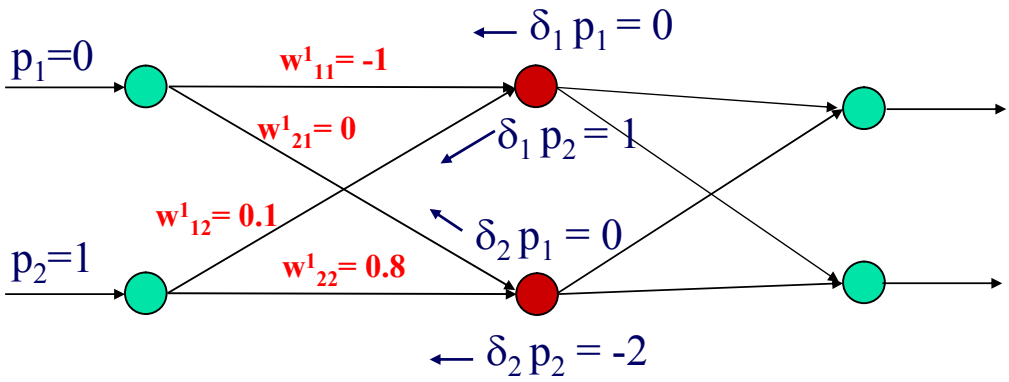
Tính gradient lớp ẩn



Minh họa giải thuật huấn luyện (11)

Cập nhật trọng số lớp ẩn

$$w^1_{ij}(k+1) = w_{ij}(t) + \eta\delta_i(k)p_j(k)$$



$w^1_{11} = -1 + 0.1*0 = -1$

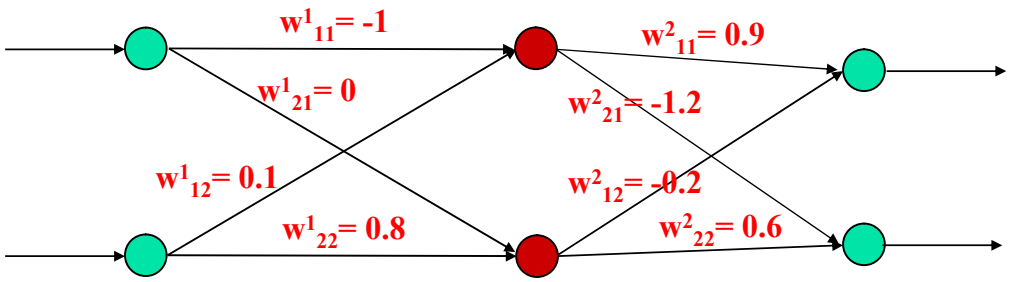
$w^1_{21} = 0 + 0.1*0 = 0$

$w^1_{12} = 0 + 0.1*1 = 0.1$

$w^1_{22} = 1 + 0.1*(-2) = 0.8$

Minh họa giải thuật huấn luyện (12)

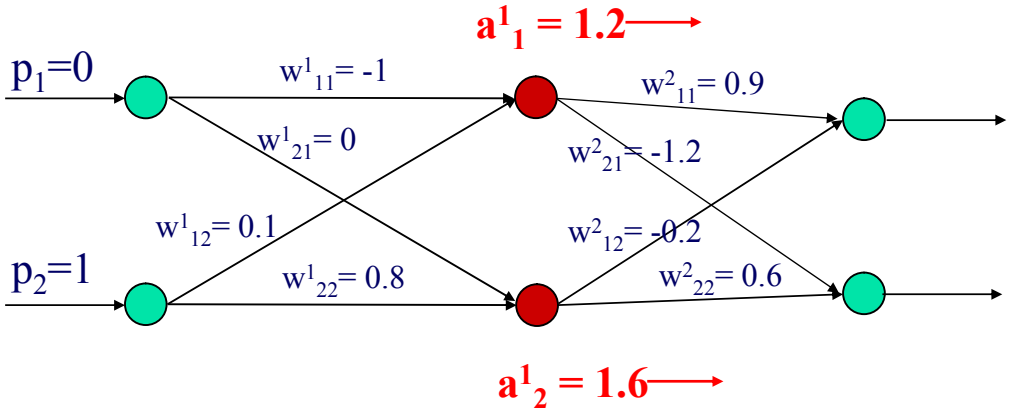
Giá trị trọng số mới:



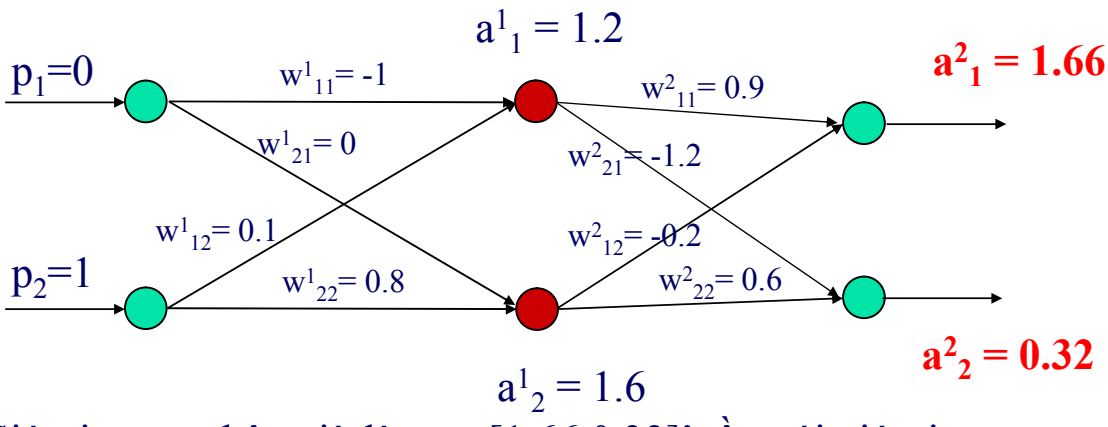
Quá trình cập nhật các giá trị ngưỡng hoàn toàn tương tự.

Minh họa giải thuật huấn luyện (13)

Truyền thuận 1 lần nữa để xác định ngõ ra của mạng với giá trị trọng số mới



Minh họa giải thuật huấn luyện (14)



Giá trị ngõ ra bây giờ là $\mathbf{a}_2$ = [1.66 0.32]'gần với giá trị mong muốn **t**=[1 0]' hơn.

Bài tập: Từ kết quả này, sinh viên hãy thực hiện thao tác truyền ngược và cập nhật trọng số ANN 1 lần nữa.

Huấn luyện đến khi nào?

- Đủ số thời kỳ (epochs) ấn định trước
- Hàm mục tiêu đạt giá trị mong muốn
- Hàm mục tiêu phân kỳ

Hàm mục tiêu MSE (1)

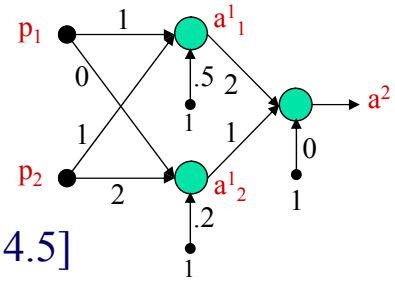
- Mean Square Error – MSE là lỗi bình phương trung bình, được xác định trong quá trình huấn luyện mạng. MSE được xem như là một trong những tiêu chuẩn đánh giá sự thành công của quá trình huấn luyện. MSE càng nhỏ, độ chính xác của ANN càng cao.
- Định nghĩa MSE:

  Giả sử ta có tập mẫu học: $\{p_1,t_1\}$, $\{p_2,t_2\}$, ..., $\{p_N,t_N\}$, với $\boldsymbol{p}=[p_1, p_2, \ldots p_N]$ là vectơ dữ liệu ngõ vào, $\boldsymbol{t}= [t_1, t_2, \ldots, t_N]$ là vectơ dữ liệu ngõ ra mong muốn. Gọi $\boldsymbol{a}=[a_1, a_2, \ldots, a_N]$ là vectơ dữ liệu ra thực tế thu được khi đưa vectơ dữ liệu vào $\boldsymbol{p}$ qua mạng. MSE:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(t_i - a_i)^2$$ ***được gọi là hàm mục tiêu***

Hàm mục tiêu MSE (2)

- **Ví dụ 1:** Cho ANN 2 lớp tuyến tính như hình vẽ.
  p=[1 2 3; 0 1 1] là các vectơ ngõ vào.
  t=[2 1 2] là vectơ ngõ ra mong muốn.
  Tính MSE.



- **Giải:**
  - Ngõ ra lớp ẩn: $a^1_1=f(n^1_1)=n^1_1=[1.5\ \ 3.5\ \ 4.5]$

    $a^1_2=f(n^1_2)=n^1_2=[0.2\ \ 2.2\ \ 2.2]$
  - Lớp ra: $a^2=f(n^2)=n^2=[3.2\ \ 9.2\ \ 11.2]$
  - $MSE = \frac{1}{3}\sum_{i=1}^{3}(t_i - a_i)^2 = \frac{1}{3}\left[(2-3.2)^2 + (1-9.2)^2 + (2-11.2)^2\right]$

    MSE = 51.1067

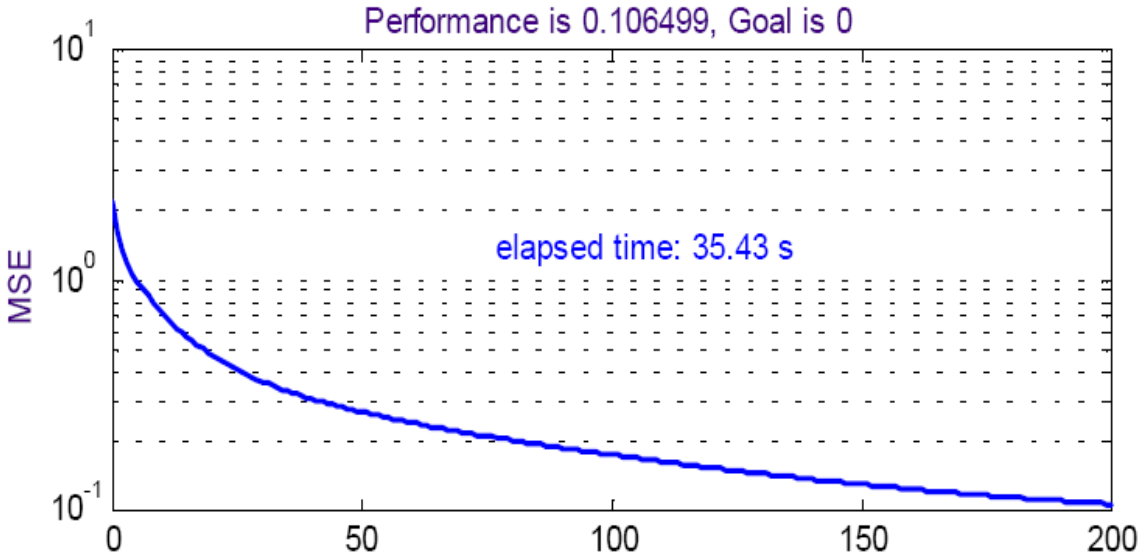
Hàm mục tiêu MSE (3)

- **Mô phỏng:**

```
» net=newff([-5 5; -5 5], [2 1], {'purelin', 'purelin'});
» net.IW{1,1}=[1 1; 0 2];        % gán input weights
» net.LW{2,1}=[2 1];             % gán layer weights
» net.b{1}=[.5; .2];             % gán ngưỡng nơ-ron lớp ẩn
» net.b{2}=0;                    % gán ngưỡng nơ-ron lớp ra
» p=[1 2 3; 0 1 1];              % vectơ dữ liệu vào
» t=[2 1 2];                     % ngõ ra mong muốn
» a=sim(net,p)                   % ngõ ra thực tế của ANN
» mse(t-a)                       % tính mse
   ans =
     51.1067
```

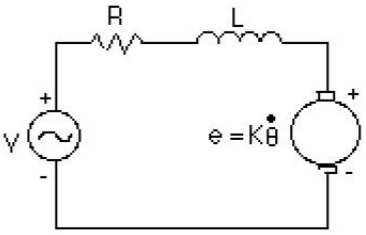

Hàm mục tiêu MSE (4)

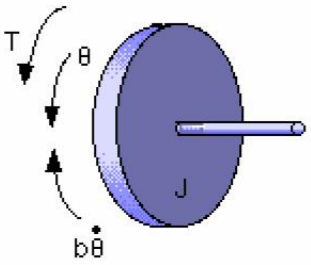
MSE được xác định sau mỗi chu kỳ huấn luyện mạng (epoch) và được xem như 1 mục tiêu cần đạt đến. Quá trình huấn luyện kết thúc (đạt kết quả tốt) khi MSE đủ nhỏ.



Ví dụ về GT Gradient descent (1)

- **Bài toán:** Xây dựng một ANN để nhận dạng mô hình vào ra của hệ thống điều khiển tốc độ motor DC sau:





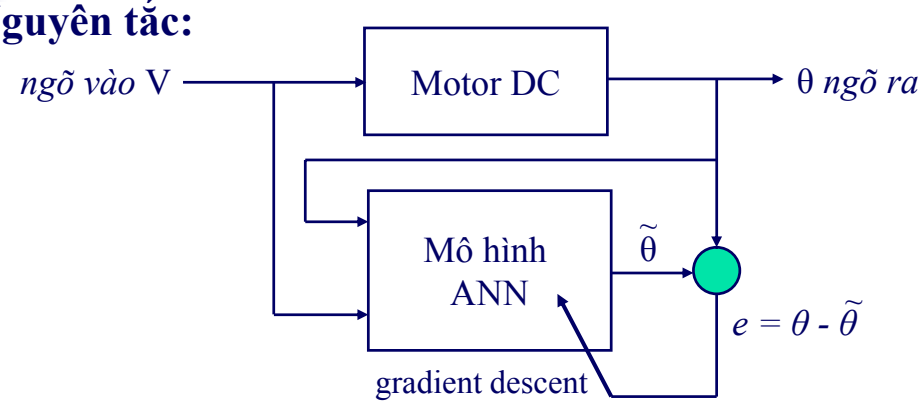
$$J\frac{d^2\theta}{dt^2} + b\frac{d\theta}{dt} = Ki$$

$$L\frac{di}{dt} + Ri = V - K\frac{d\theta}{dt}$$

$J = 0.01\ kgm^2/s^2$	là moment quán tính của rotor
$b = 0.1\ Nms$	hệ số ma sát
$K=K_e=K_t=0.01\ Nm/Amp$	các hằng số sức điện động
$R = 1\ ohm$	điện trở
$L = 0.5\ H$	điện cảm
I:	dòng điện chạy trong cuộn dây của motor
V:	điện áp trên hai đầu cuộn dây motor - ngõ vào
θ:	vị trí trục - ngõ ra

Ví dụ về GT Gradient descent (2)

- **Nguyên tắc:**

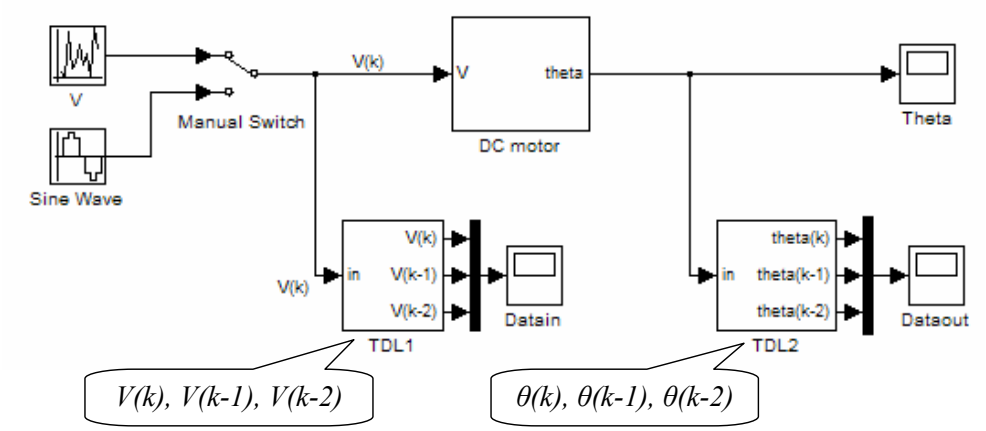


$$\theta(k)=f_{ANN}[V(k), V(K-1), V(K-2), \theta(k-1), \theta(k-2)]$$

- **Các bước cần thiết:**
  - Thu thập và xử lý dữ liệu vào ra của đối tượng
  - Chọn lựa cấu trúc và xây dựng ANN
  - Huấn luyện ANN bằng giải thuật gradient descent
  - Kiểm tra độ chính xác của mô hình bằng các tín hiệu khác

Ví dụ về GT Gradient descent (3)

- **Thực hiện:**
  - *Mô hình Simulink để thu thập dữ liệu: data_Dcmotor.mdl*



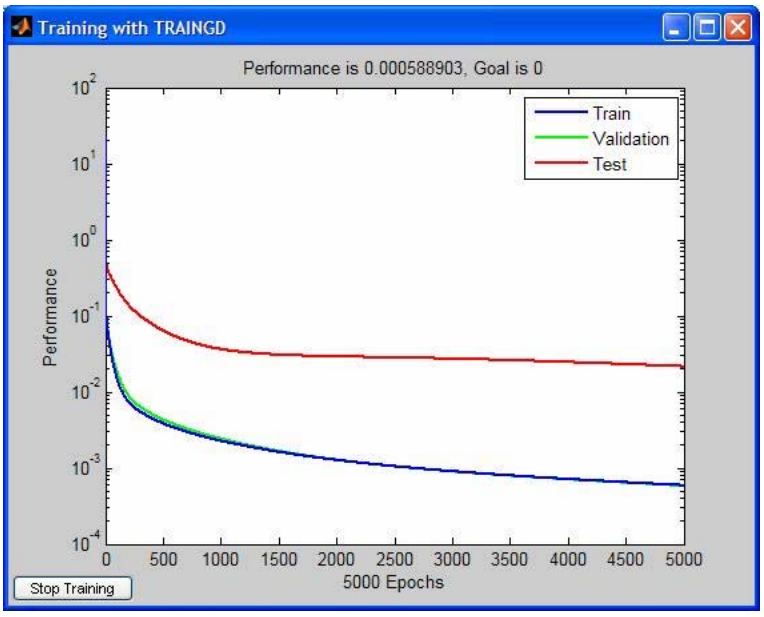
  - *Chuẩn bị dữ liệu huấn luyện: ANN_Dcmotor.m*

```
load data_DCmotor;                % nap tap du lieu hoc
P=[datain'; dataout(:, 2:3)'];    % [V(k), V(k-1), V(k-2), θ(k-1), θ(k-2)]
T=dataout(:,1)';                  % theta(k)
```


Ví dụ về GT Gradient descent (4)

- **Thực hiện:**
  - *Tạo ANN và huấn luyện: ANN_Dcmotor.m*

```
>> net=train(net, Ptrain,Ttrain, [], [], VV,TV);
```



Nhận xét:

Tốc độ hội tụ của giải thuật Gradient descent quá chậm.

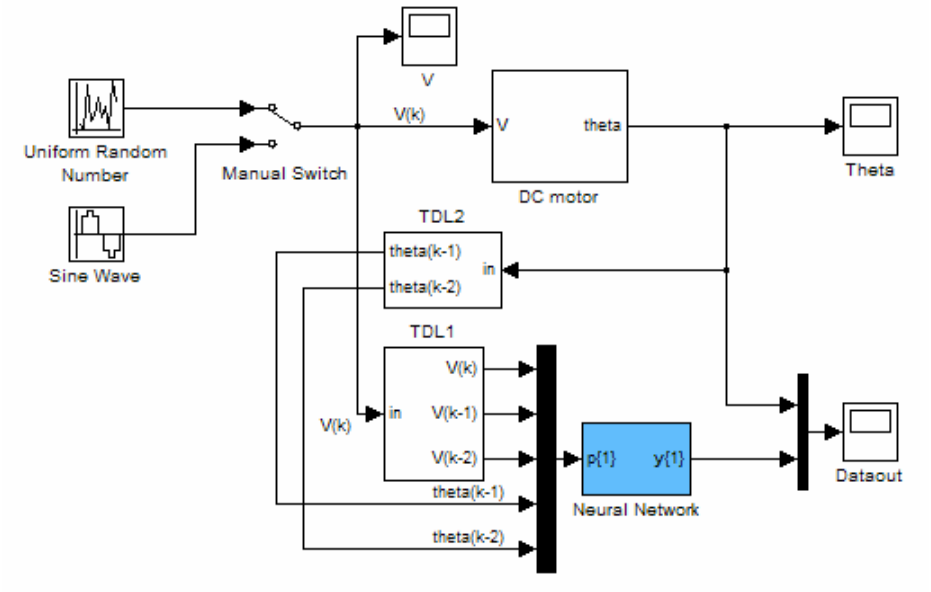
Sau 5000 Epochs, MSE chỉ đạt 6.10^{-4}.

Kết quả kiểm tra cho thấy lỗi lớn.

Cần giải thuật cải tiến.

Ví dụ về GT Gradient descent (5)

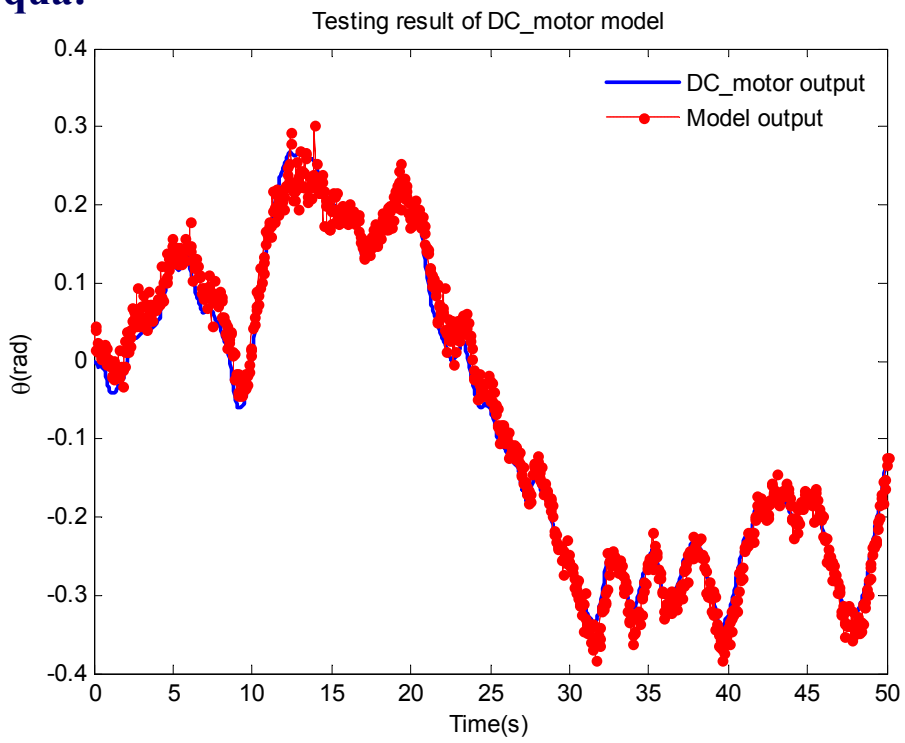
- **Thực hiện:**
  - *Mô hình kiểm tra độ chính xác ANN: Test_Dcmotor.mdl*



$$\theta(k)=f_{ANN}[V(k), V(K-1), V(K-2), \theta(k-1), \theta(k-2)]$$

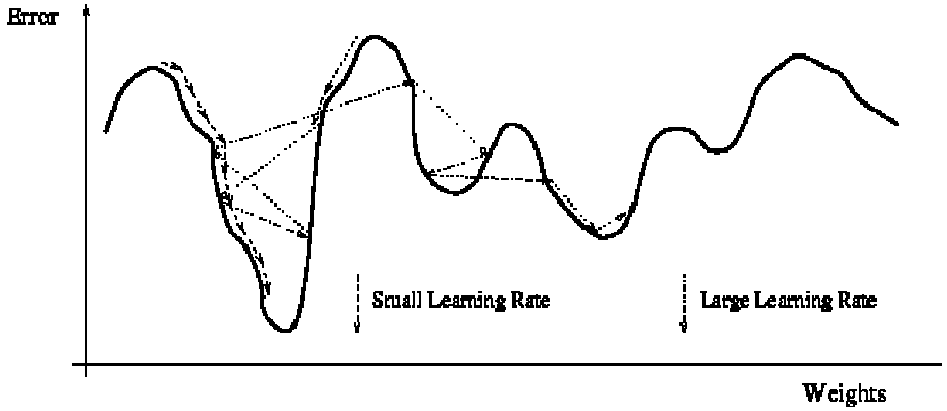
Ví dụ về GT Gradient descent (6)

- **Kết quả:**

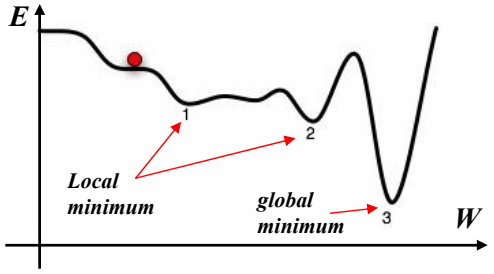


Cực tiểu cục bộ

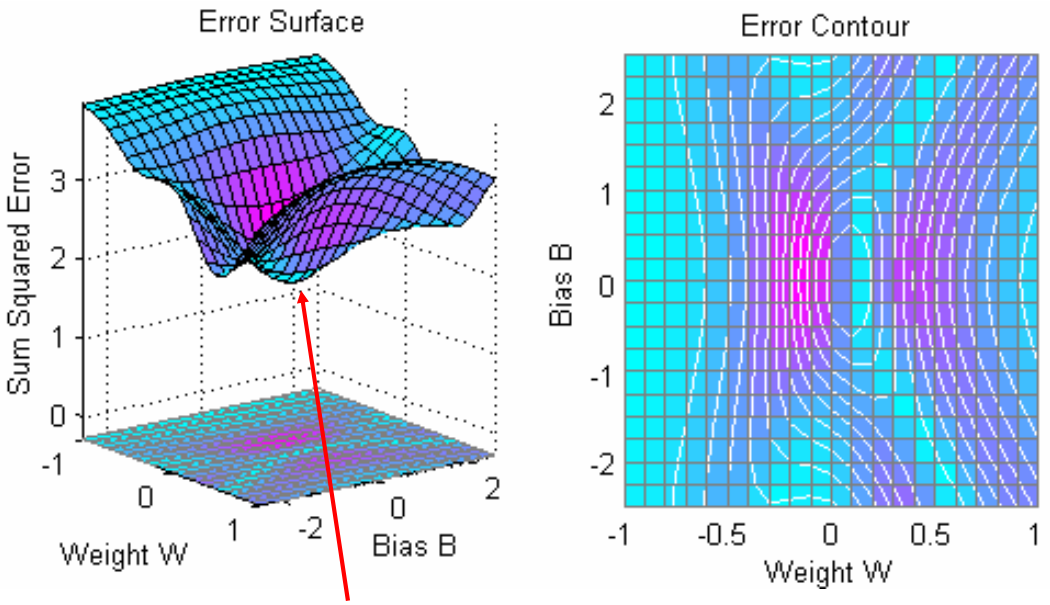
- Ảnh hưởng của tốc độ học



- Quá trình huấn luyện mạng, giải thuật cần vượt qua các điểm cực tiểu cục bộ (ví dụ: có thể thay đổi hệ số momentum), để đạt được điểm global minimum.



Mặt lỗi



Cực tiểu mong muốn

Gradient des. with momentum (1)

- Nhằm cải tiến tốc độ hội tụ của giải thuật gradient descent, người ta đưa ra 1 nguyên tắc cập nhật trọng số của ANN:

$$w_{ij}(k+1) = w_{ij}(k) + \nabla w_{ij}(k)$$

$$\nabla w_{ij}(k) = \eta g(k) + \mu \nabla w_{ij}(k-1)$$

với *g(k): gradient;* *η: tốc độ học*

$\nabla_{ij}(k-1)$ *là giá trị trước đó của* $\nabla_{ij}(k)$

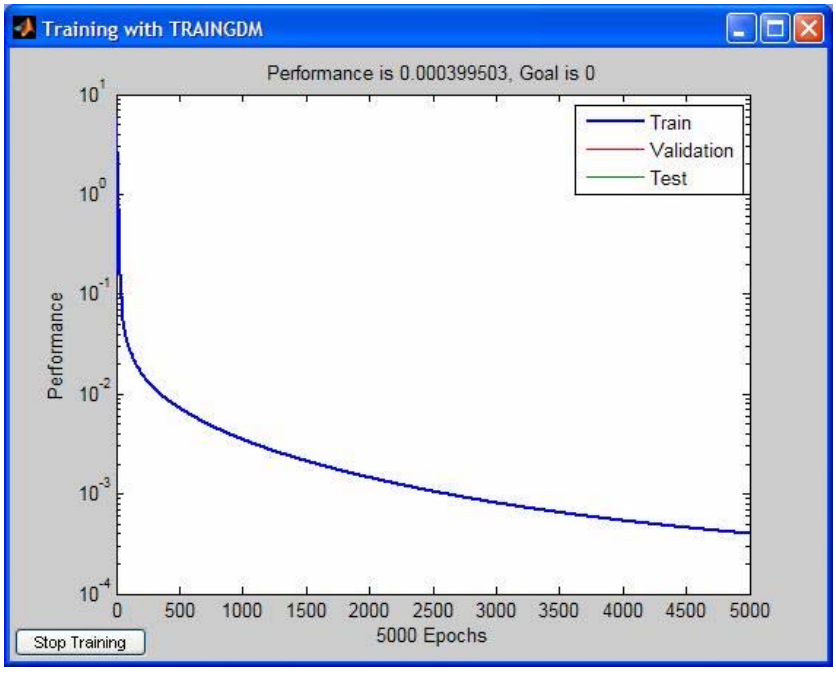
μ: *momentum*

- Để đạt hiệu quả huấn luyện cao, nhiều tác giả đề nghi giá tổng giá trị của moment và tốc độ học nên gần bằng 1:

$$\mu \in [0.8 \quad 1]; \quad \eta \in [0 \quad 0.2]$$

Gradient des. with momentum (2)

- Áp dụng cho bài toán nhận dạng mô hình của motor DC:



Nhận xét:
Sau 5000 Epochs, MSE đạt 4.10^{-4}, nhanh hơn giải thuật gradient descent.

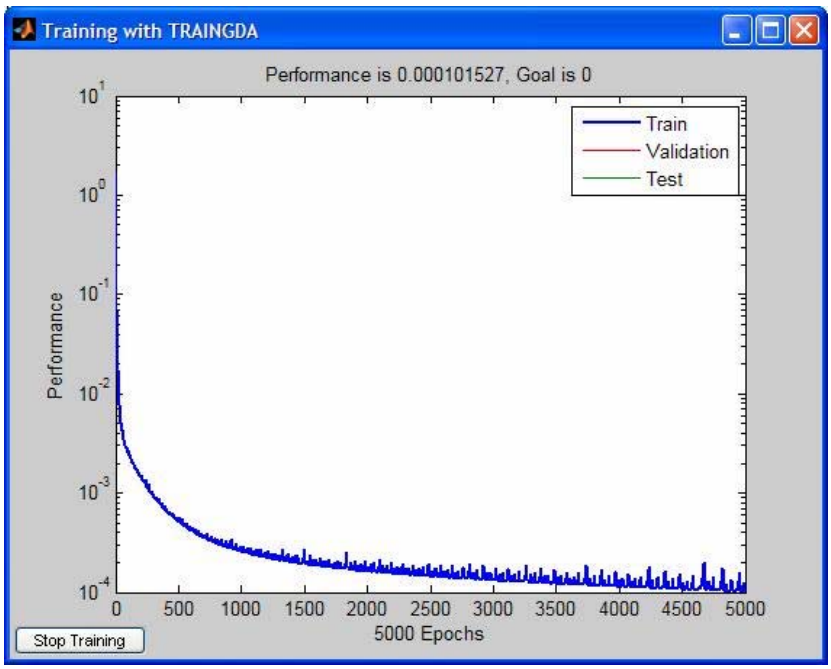
GD với tốc độ học thích nghi (1)

- Tốc độ hội tụ của giải thuật Gradient descent phụ thuộc vào tốc độ học η. Nếu η lớn giải thuật hội tụ nhanh nhưng bất ổn. Nếu η nhỏ thời gian hội tụ sẽ lớn.
- Việc giữ tốc độ học η là một hằng số suốt quá trình huấn luyện, tỏ ra kém hiệu quả. Một giải thuật cải tiến nhằm thay đổi thích nghi tốc độ học theo nguyên tắc:

if (*new_error* - *old_error*) > *max_perf_inc* (*thường, max_perf_inc=1.04*)
    Loại trọng số và ngưỡng mới
    *Tốc độ học = tốc độ học * lr_dec* (*giảm lr, thường thì lr_dec=0.7*)
elseif new_error < old_error
    Giữ lại trọng số và ngưỡng mới
    *Tốc độ học = tốc độ học * lr_inc* (*tăng lr, thường thì lr_inc=1.05*)
end

GD với tốc độ học thích nghi (1)

- Áp dụng cho bài toán nhận dạng mô hình của motor DC:



Nhận xét:
Sau 5000 Epochs, MSE đạt 10^{-4}, nhanh hơn giải thuật gradient descent with momentum

GD với η và μ thích nghi (1)

- Là giải thuật gradient descent cải tiến, mà cả tốc độ học η và hệ số momentum μ được thay đổi thích nghi trong quá trình huấn luyện.
- Việc thay đổi thích nghi μ được thực hiện tương thự như việc thích nghi tốc độ học μ.
- Thủ tục cập nhật trọng số giống như giải thuật *GD with momentum:*

$$w_{ij}(k+1) = w_{ij}(k) + \nabla w_{ij}(k)$$

$$\nabla w_{ij}(k) = \eta g(k) + \mu \nabla w_{ij}(k-1)$$

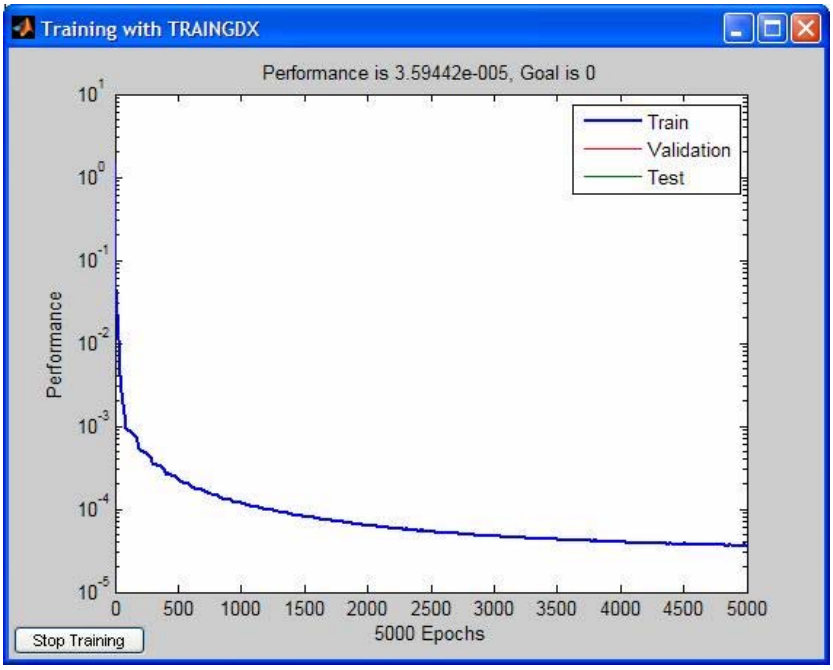
với *g(k): gradient;* *η: tốc độ học thích nghi*

$\nabla_{ij}(k\text{-}1)$ *là giá trị trước đó của* $\nabla_{ij}(k)$

μ: momentum thích nghi

GD với η và μ thích nghi (2)

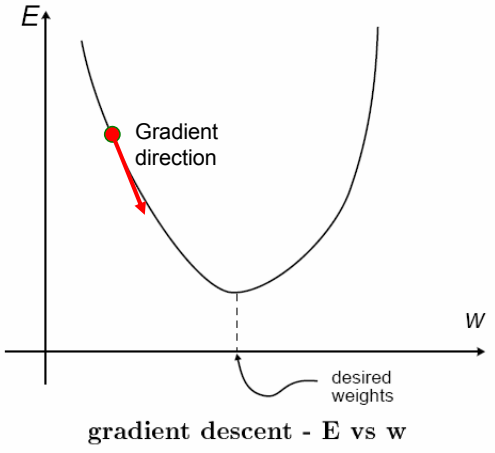
- Áp dụng cho bài toán nhận dạng mô hình của motor DC:



Nhận xét:
Sau 5000 Epochs, MSE đạt 3.10^{-5}, nhanh hơn giải thuật gradient descent with η thích nghi

GT. truyền ngược Resilient (1)

- Các hàm truyền Sigmoid "nén" các ngõ vào vô hạn thành các ngõ ra hữu hạn làm phát sinh một điểm bất lợi là các gradient sẽ có giá trị nhỏ, làm cho các trọng số chỉ được điều chỉnh một giá trị nhỏ, mặc dù nó còn xa giá trị tối ưu.
- Giải thuật Resilient được phát triển nhằm loại bỏ điểm bất lợi này bằng cách sử dụng đạo hàm của hàm lỗi để quyết định hướng tăng/giảm của gradient.



gradient descent - E vs w

- Nếu $\left(\frac{\partial E_k}{\partial w_{ij}}\right) \& \left(\frac{\partial E_{k+1}}{\partial w_{ij}}\right)$ cùng dấu: $w_{ij}(k+1)$ được tăng thêm 1 lượng Δ_{inc}
- Nếu $\left(\frac{\partial E_k}{\partial w_{ij}}\right) \& \left(\frac{\partial E_{k+1}}{\partial w_{ij}}\right)$ khác dấu: $w_{ij}(k+1)$ được giảm đi 1 lượng Δ_{dec}

GT. truyền ngược Resilient (2)

1. *Khởi tạo giá trị nhỏ của step size* $\Delta w_{ji}(0)$

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}$$ *(n là thời kỳ huấn luyện hiện tại)*

2. *Thay đổi step size và cập nhật trọng số*

if $\frac{\partial E}{\partial w_{ij}}(k+1) * \frac{\partial E}{\partial w_{ij}}(k) > 0$ *(2 thời kỳ liên tiếp, đạo hàm cùng dấu)*

$$\Delta w_{ij}(k+1) = \Delta w_{ij}(k) * \Delta_{inc}$$ *(thường thì* $\Delta_{inc}=1.2$*)*

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}(k+1)$$ *(tăng trọng số)*

elseif $\frac{\partial E}{\partial w_{ij}}(k+1) * \frac{\partial E}{\partial w_{ij}}(k) < 0$ *(2 thời kỳ liên tiếp, đạo hàm khác dấu)*

$$\Delta w_{ij}(k+1) = \Delta w_{ij}(k) * \Delta_{dec}$$ *(thường thì* $\Delta_{dec}=0.5$*)*

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}(k+1)$$ *(giảm trọng số)*

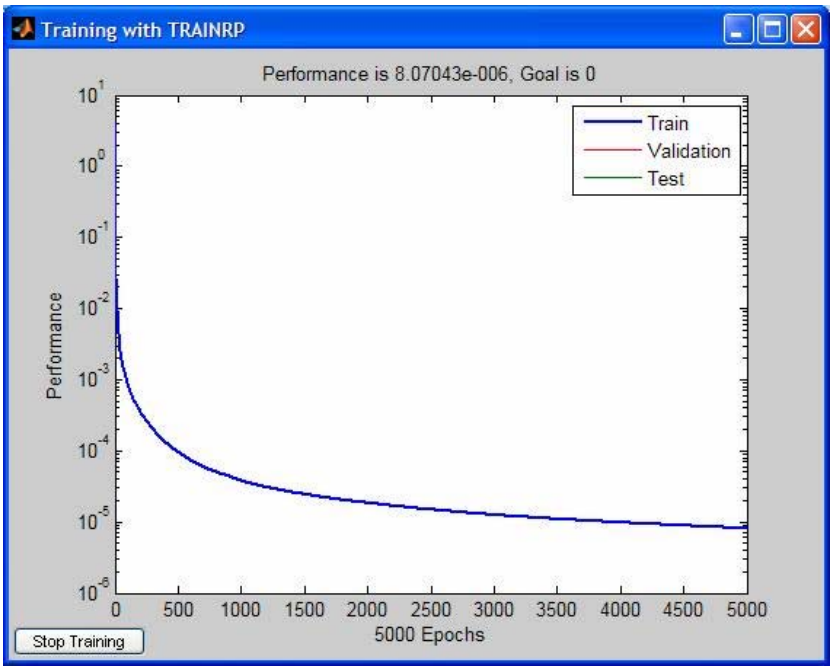
else *(đạo hàm bằng không)*

$$w_{ij}(k+1) = w_{ij}(k)$$ *(giữ nguyên trọng số cũ)*

end

GT. truyền ngược Resilient (3)

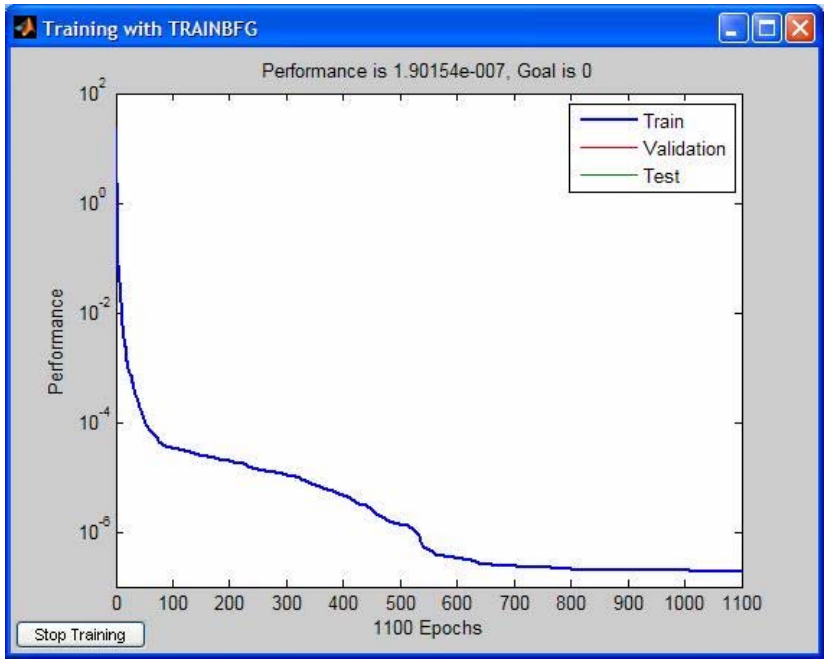
- Áp dụng cho bài toán nhận dạng mô hình của motor DC:



Nhận xét:
Sau 5000 Epochs, MSE đạt 8.10^{-6}, nhanh hơn giải thuật gradient descent with η & μ thích nghi

Giải thuật BFGS Quasi-Newton

- Sinh viên tự đọc tài liệu [1] và [3]
- Áp dụng cho bài toán nhận dạng mô hình motor DC:



Nhận xét:
Sau 1100 Epochs, MSE đạt 2.10^{-7}, nhanh hơn giải thuật Resilient

Giải thuật Levenberg-Marquardt (1)

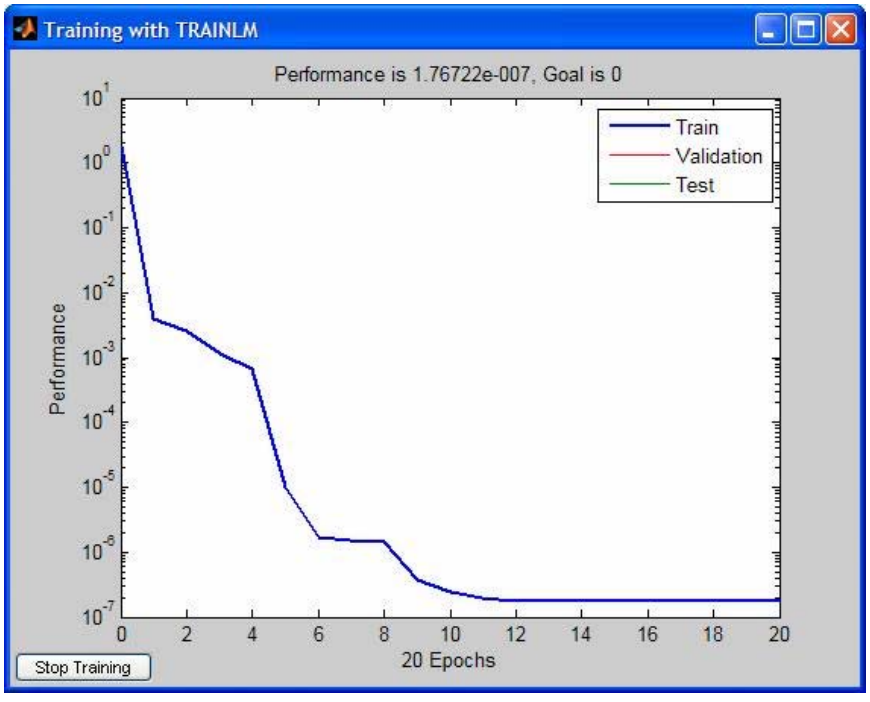
- Giải thuật Levenberg-Marquardt được xây dựng để đạt tốc độ hội tụ bậc 2 mà không cần tính đến ma trận Hessian như giải thuật BFGS Quasi-Newton.
- Ma trận Hessian được tính xấp xỉ: $\boldsymbol{H}=\boldsymbol{J}^T\boldsymbol{J}$ và giá trị gradient được xác định: $\boldsymbol{g}=\boldsymbol{J}^T\boldsymbol{e}$
  trong đó, $\boldsymbol{J}$ là ma trận Jacobian, chứa đạo hàm bậc nhất của hàm lỗi $(\delta \boldsymbol{e}/\delta wij)$, với $\boldsymbol{e}$ là vectơ lỗi của mạng.
- Nguyên tắc cập nhật trọng số:

$$w_{ij}(k+1)=w_{ij}(k) - [\boldsymbol{J}^T\boldsymbol{J} + m\boldsymbol{I}]^{-1}\,\boldsymbol{J}^T\boldsymbol{e}$$

  - Nếu $m=0$, thì đây là giải thuật BFGS Quasi-Newton.
  - Nếu m có giá trị lớn nó là giải thuật gradient descent.
  - Giải thuật Levenberg-Marquardt luôn sử dụng giá trị m nhỏ, do giải thuật BFGS Quasi-Newton tốt hơn giải thuật gradient descent.

Giải thuật Levenberg-Marquardt (1)

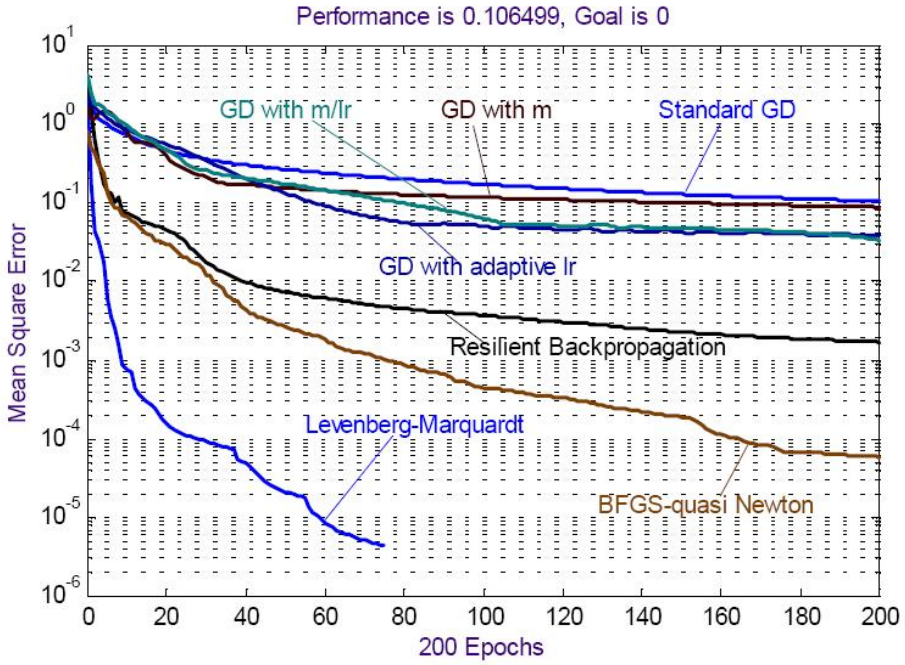
- Áp dụng cho bài toán nhận dạng mô hình của motor DC:



Nhận xét:
Sau 20 Epochs, MSE đạt $1{,}7.10^{-7}$, nhanh hơn giải thuật Newton

So sánh các giải thuật

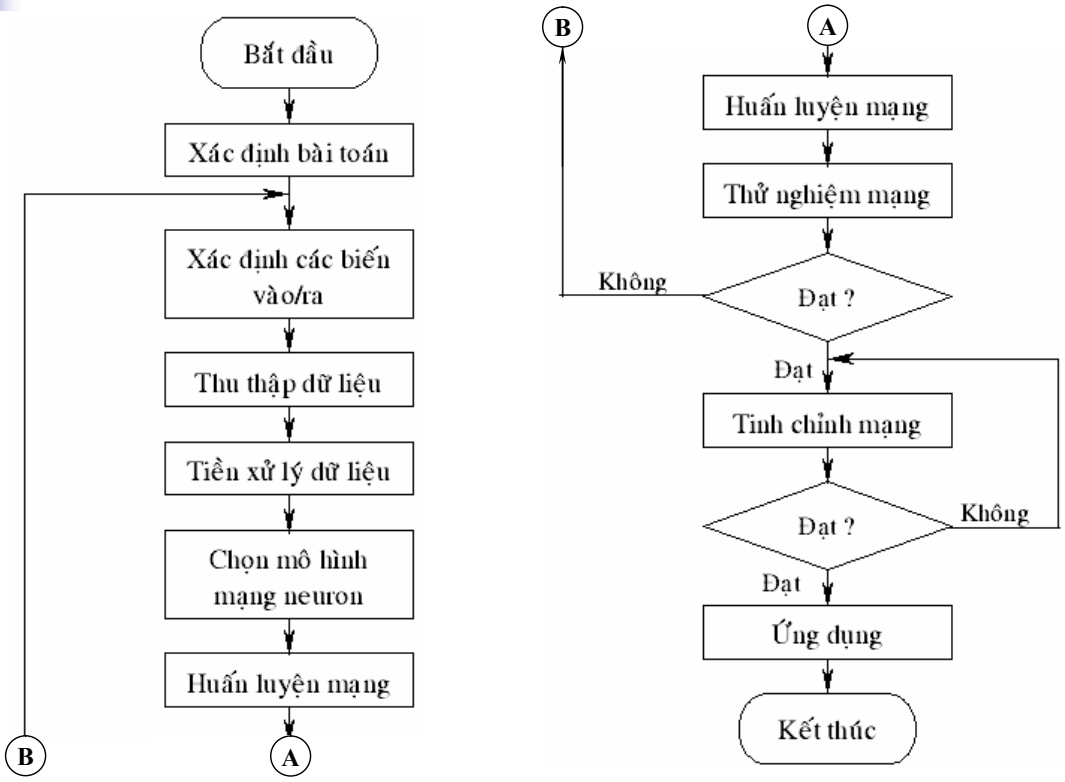
- So sánh trên bài toán nhận dạng mô hình một đối tượng phi tuyến, được trình bày trong tài liệu [1]:



Các giải thuật của NN. toolbox

Trainb	Batch training with weight & bias learning rules.
Trainbfg	BFGS quasi-Newton backpropagation.
Trainbr	Bayesian regularization.
Trainc	Cyclical order incremental training w/learning functions.
Traincgb	Powell-Beale conjugate gradient backpropagation.
Traincgf	Fletcher-Powell conjugate gradient backpropagation.
Traincgp	Polak-Ribiere conjugate gradient backpropagation.
Traingd	Gradient descent backpropagation.
Traingdm	Gradient descent with momentum backpropagation.
Traingda	Gradient descent with adaptive lr backpropagation.
Traingdx	Gradient descent w/momentum & adaptive lr backpropagation.
Trainlm	Levenberg-Marquardt backpropagation.
Trainoss	One step secant backpropagation.
Trainr	Random order incremental training w/learning functions.
Trainrp	Resilient backpropagation (Rprop).
Trains	Sequential order incremental training w/learning functions.
Trainscg	Scaled conjugate gradient backpropagation.

Qui trình thiết kế một ANN

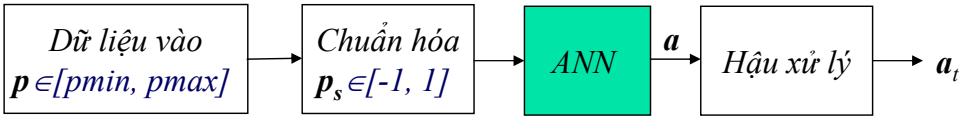


Một số kỹ thuật phụ trợ

Tiền xử lý dữ liệu (1)

- **Phương pháp chuẩn hóa dữ liệu:**

Chuẩn hóa để tập dữ liệu nằm trong khoảng [-1 1].

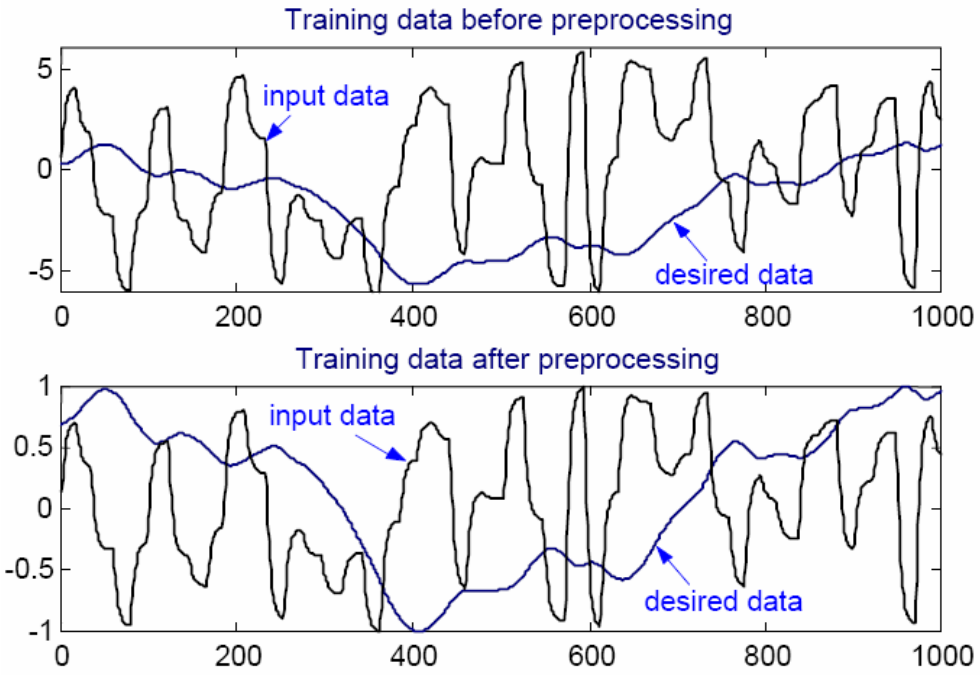


- Gọi $\boldsymbol{p} \in [p_{min}, p_{max}]$ là vectơ dữ liệu vào
- $\boldsymbol{p}_s$ là vectơ dữ liệu sau khi chuẩn hóa, thì: $p_s = 2\dfrac{p - p_{min}}{p_{max} - p_{min}} - 1$
- Nếu ta đưa tập dữ liệu đã được xử lý vào huấn luyện mạng, thì các trọng số được điều chỉnh theo dữ liệu này. Nên giá trị ngõ ra của mạng cần có thao tác hậu xử lý.
- Gọi $\boldsymbol{a}$ là dữ liệu ra của mạng, $\boldsymbol{a}_t$ giá trị hậu xử lý, thì:

$$a_t = \frac{1}{2}(a+1)(p_{max} - p_{min}) + p_{min}$$

Tiền xử lý dữ liệu (2)

- **Phương pháp chuẩn hóa dữ liệu (ví dụ)**



Tiền xử lý dữ liệu (3)

- **Phương pháp trị trung bình và độ lệch chuẩn:**
  Tiền xử lý để tập dữ liệu có trị trung bình bằng 0 *(mean=0)* và độ lệch chuẩn bằng 1 *(standard deviation=1)*.
  - Gọi ***p*** là vectơ dữ liệu vào, có trị trung bình là $mean_p$ và độ lệch chuẩn là std_p, thì vectơ dữ liệu được xử lý là:

$$p_s = \frac{p - mean_p}{std_p}$$

  - Gọi a là vectơ dữ liệu ra của ANN, thì vectơ dữ liệu ngõ ra sau khi thực hiện thao tác hậu xử lý:

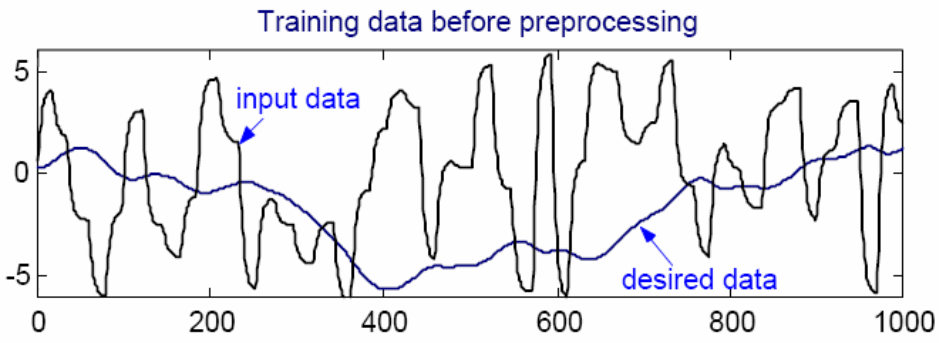
$$a_t = a * std_p + mean_p$$

Tiền xử lý dữ liệu (4)

- **Phương pháp trị trung bình và độ lệch chuẩn (ví dụ)**

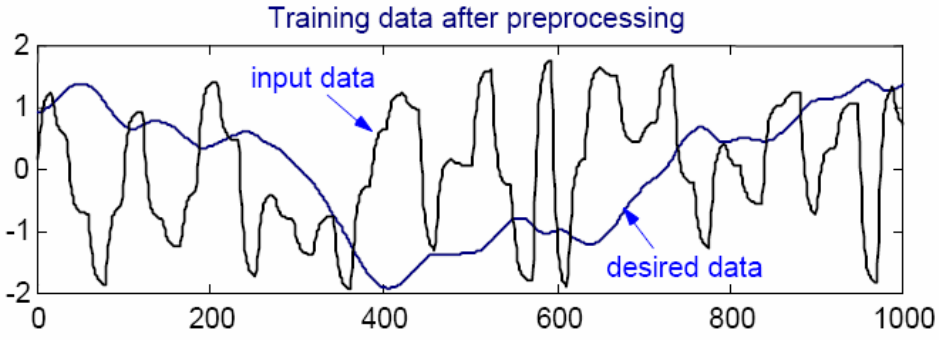
Mean ≠ 0

Std ≠ 1



Mean = 0

Std = 1



Nâng cao khả năng tổng quát hóa (1)

- Một vấn đề xuất hiện trong quá trình huấn luyện mạng, đó là hiện tượng quá khớp (overfitting).
- Khi kiểm tra mạng bằng tập dữ liệu đã huấn luyện, nó cho kết quả tốt (lỗi thấp). Nhưng khi kiểm tra bằng dữ liệu mới, kết quả rất tồi (lỗi lớn). Do mạng không có khả năng tổng quát hóa các tình huống mới ("học vẹt").
- Có 2 phương pháp khắc phục: Phương pháp định nghĩa lại hàm mục tiêu và phương pháp ngừng sớm.

Nâng cao khả năng tổng quát hóa (2)

- **Phương pháp định nghĩa lại hàm mục tiêu:**
  - Thông thường hàm mục tiêu được định nghĩa là:

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N} e_i^{\,2} = \frac{1}{N}\sum_{i=1}^{N} (t_i - a_i)^2$$

  - Hàm mục tiêu được định nghĩa lại bằng cách thêm vào đại lượng tổng bình phương trung bình của các trọng số và ngưỡng, *MSW*, khi đó:

$$MSEreg = \gamma MSE + (1-\gamma) MSW$$

Với γ là một hằng số tỉ lệ và $\mathrm{MSW} = \frac{1}{n}\sum_{j=1}^{n} w_j^{\,2}$

n tổng số trọng số
và ngưỡng của mạng

Nâng cao khả năng tổng quát hóa (3)

- **Phương pháp ngừng sớm:**
  - Phương pháp này đòi hỏi chia tập dữ liệu học thành 3 phần, gồm dữ liệu huấn luyện, dữ liệu kiểm tra và dữ liệu giám sát.

$$P = [P_{train}, P_{test}, P_{validation}]$$

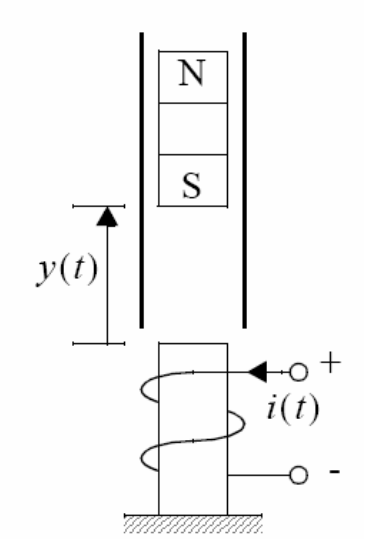
  - Sau lỗi thời kỳ huấn luyện, tập dữ liệu dám sát $P_{validation}$ được đưa vào mạng để kiểm tra lỗi. Nếu lỗi thu được giảm, quá trình huấn luyện được tiếp tục. Nếu lỗi thu được bắt đầu tăng (hiện tượng quá khớp bắt đầu xảy ra), quá trình huấn luyện được dừng lại – gọi là ngừng sớm.
  - Sinh viên đọc thêm ở tài liệu [1].

Minh họa bằng MATLAB

Bài toán:

Nhận dạng mô hình hệ thống đệm từ, có phương trình vi phân mô tả hệ:

$$\frac{d^2y(t)}{dt^2} = -g + \frac{\alpha}{M}\frac{i^2(t)\,\mathrm{sgn}[i(t)]}{y(t)} - \frac{\beta}{M}\frac{dy(t)}{dt}$$



Với i(t) ∈ [0, 4A] dòng điện ngõ vào

y(t) là khoảng cách từ nam châm vĩnh cửu đến nam châm điện.

các tham số: β=12, α=15, g=9.8 và M=3.

Bài tập

1. *Sinh viên thực hiện lại bài toán nhận dạng mô hình motor DC và huấn luyện mạng bằng tất cả các giải thuật của NN toolbox của MATLAB. So sánh tốc độ hội tụ của các giải thuật.*

Chương 4
Một số ứng dụng của ANN

Giới thiệu

Nhận dạng ký tự (OCR)

Nhận dạng tiếng nói

Thiết kế các bộ điều khiển

Kết luận

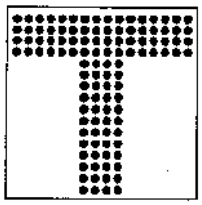
Bài tập

Giới thiệu

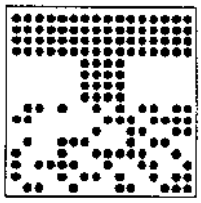
- Giới thiệu một số hướng ứng dụng ANN
- Phát triển thành Luận văn tốt nghiệp hay đề tài NCKH sinh viên

Nhận dạng ký tự

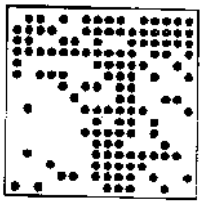
- Ma trận hóa bitmap của ký tự
- Giả lập các hình thức nhiễu
- Tập hợp dữ liệu huấn luyện, mỗi ký tự là 1 vectơ dữ liệu ngõ vào
- Qui ước ngõ ra, giả sử là mã ASCII tương ứng của ký tự.
- Xây dựng cấu và huấn luyện ANN



Original 'T'

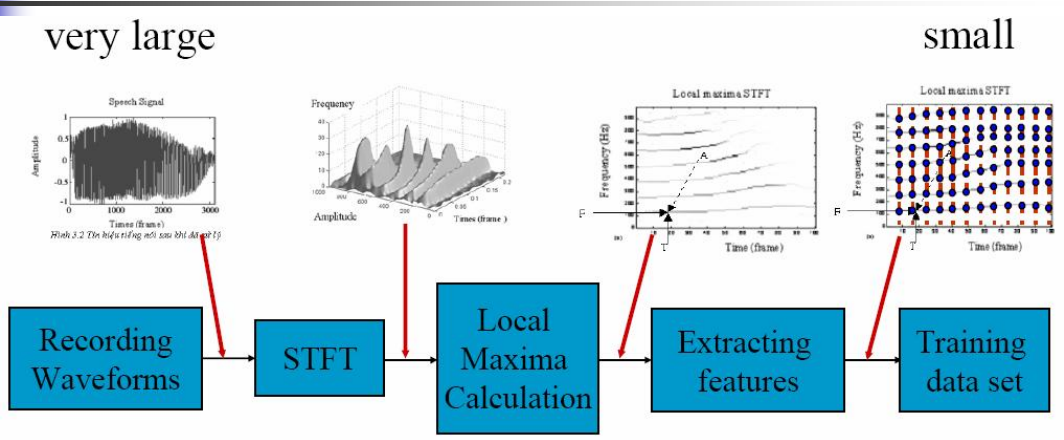


half of image corrupted by noise



20% corrupted by noise (whole image)

Nhận dạng tiếng nói



- Trích đặc trưng tín hiệu tiếng nói
- Tập hợp dữ liệu vào, qui ước dữ liệu ra
- Huấn luyện và thử nghiệm
- Phương pháp LPC & AMDF để xác định đặc trưng tiếng nói