ỨNG DỤNG MẠNG NƠRON trong nhận dạng đối tượng điều khiển

ĐÀO HỮU HÙNG, TA CAO MINH
Bộ môn Tự Động Hóa XNCN - Trường Đại học Bách Khoa Hà Nội

Bài báo chứng minh khả năng nhận dạng điều khiển rất tốt của mạng nơron. Cả nhận dạng mô hình thuận và mô hình ngược đều được thực hiện bởi mạng nơron thẳng và luật học lan truyền ngược. Một ví dụ mô phỏng đơn giản trên MATLAB được đưa ra để chứng minh những nhận định trên.

Mở đầu

Trong thiết kế điều khiển, khi biết được mô hình toán học của đối tượng điều khiển (gọi tắt là đối tượng) thì ta có thể dễ dàng thiết kế được một bộ điều khiển để thu được đáp ứng của hệ thống theo mong muốn. Đồng thời cũng đảm bảo được tính ổn định, bền vững của hệ thống. Tuy nhiên, không phải lúc nào ta cũng biết được mô hình toán học của đối tượng. Với những quá trình vật lý phức tạp, ta hoàn toàn không thể thu được mô hình toán học phản ánh quá trình vật lý đó. Do đó, ta rất khó có thể thiết kế được một bộ điều khiển để đảm bảo các tính năng và chỉ tiêu chất lượng mong muốn cho toàn hệ thống. Trong trường hợp này, để thiết kế được một bộ điều khiển, ít nhất ta cũng phải biết được một mô hình xấp xỉ của đối tượng. Mô hình xấp xỉ đó được gọi là mô hình đồng dạng của đối tượng. Việc ước lượng mô hình xấp xỉ đó được gọi là nhận dạng đối tượng điều khiển.

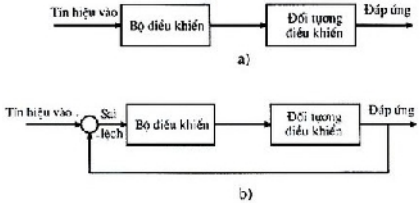
Với khả năng học, mạng nơron tỏ ra rất thích hợp trong việc nhận dạng đối tượng điều khiển. Mạng nơron được chia ra làm hai loại: (1) mạng nơron thẳng và (2) mạng nơron luồn hồi. Trong đó, mạng nơron thẳng có khả năng nhận dạng mẫu rất tốt và mạng nơron luồn hồi lại được sử dụng trong bộ nhớ liên tưởng (associative memories) và giải các bài toán tối ưu. Từ quan điểm lý thuyết hệ thống, mạng nơron thẳng biểu diễn ánh xạ phi tuyến tĩnh. Trong khi đó, mạng nơron luồn hồi được biểu diễn bởi các hệ thống phản hồi, động, phi tuyến. Do đó, trong bài báo này, ta nhấn mạnh vào mạng nơron thẳng kết hợp với luật học lan truyền ngược để nhận dạng đối tượng điều khiển. Phần II trình bày ngắn gọn về thiết kế điều khiển hệ kín và hệ hở. Phần III trình bày việc nhận dạng đối tượng dùng mạng nơron. Phần IV đưa ra một ví dụ nhận dạng đơn giản được mô phỏng trên MATLAB để minh họa.

Thiết kế điều khiển hệ kín và hệ hở

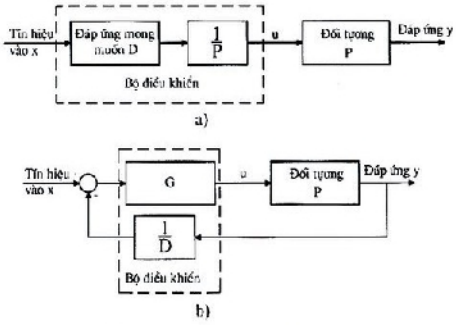
Trong lý thuyết điều khiển, ta có hai loại hệ thống điều khiển quen thuộc là hệ kín và hệ hở được trình bày trên hình 1. Thiết kế bộ điều khiển cho hệ kín và hệ hở là rất khác nhau. Trên hình 2 thể hiện hai cách điển hình để thiết kế bộ điều khiển cho hệ hở và hệ kín.

Bộ điều khiển hệ hở gồm hai khâu được nối tầng với nhau được bao bởi phần nét đứt (xem hình 2(a)). Khâu 1/P là mô hình ngược của đối tượng P. Hàm truyền đạt của nó sẽ triệt tiêu với hàm truyền đạt của đối tượng P. Phần đầu của bộ điều khiển là khâu đáp ứng mong muốn D. Vậy đáp ứng của toàn bộ hệ thống bao gồm bộ điều khiển và đối tượng bằng với đáp ứng của khâu D. Đáp ứng của hệ thống:

$$Y(s) = D(s).(1/P(s)).P(s) = D(s) \qquad (1)$$



Hình 1. Sơ đồ khối một hệ thống điều khiển (a) Hệ hở, (b) Hệ kín



Hình 2. Điều khiển đối tượng P (a) Hệ hở, (b) Hệ kín

Trong đó s là toán tử Laplace.

Vậy nếu ta biết được mô hình ngược 1/P của đối tượng P thì ta chỉ cần chọn đáp ứng mong muốn của hệ thống có hàm truyền D(s). Phương pháp thiết kế này đơn giản nếu biết 1/P. Nhưng ước lượng được mô hình ngược 1/P là rất khó nếu mô hình ngược đó không tồn tại duy nhất.

Bộ điều khiển hệ kín được thể hiện bằng phần nét đứt như thấy trên hình 2(b). Hàm truyền của toàn bộ hệ thống là:

$$\frac{Y(s)}{X(s)} = \frac{G(s)P(s)}{1 + G(s)P(s)/D(s)} \qquad (2)$$

Người thiết kế sẽ chọn G(s) và D(s) sao cho

$$\left| \frac{G(s)P(s)}{D(s)} \right|_{S=j\omega} >> 1 \qquad (3)$$

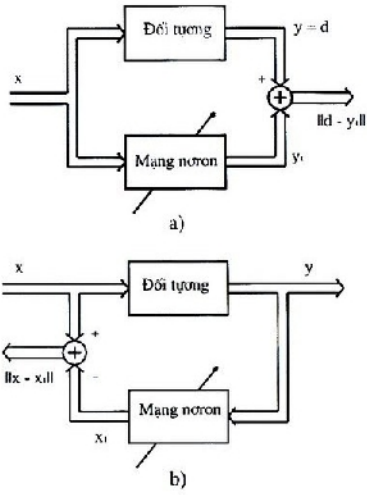
thì hàm truyền của hệ thống sẽ được rút gọn lại như sau:

$$\frac{Y(s)}{X(s)} \cong D(s) \qquad (4)$$

Ta chọn trước đáp ứng mong muốn của hệ thống D(s), sau đó thiết kế G(s) thỏa mãn điều kiện (3).

Tuy nhiên, nếu tham số của đối tượng P thay đổi trong quá trình hoạt động thì bộ điều khiển sẽ không thực hiện tốt công việc của mình. Thậm chí, ta sẽ không thể thiết kế được bộ điều khiển nếu mô hình của P chưa biết hay biết không đầy đủ. Để vượt qua được khó khăn này, ta sẽ sử dụng mạng nơron để nhận dạng mô hình thuận và ngược của P một cách thích nghi nhờ khả năng học của mạng.

Nhận dạng đối tượng điều khiển dùng mạng nơron

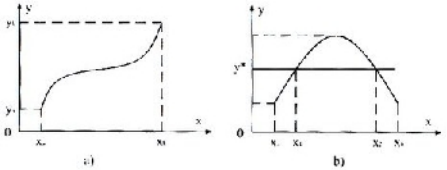


Hình 3. Cấu trúc mạng nơron để nhận dạng đối tượng (a) mô hình thuận, (b) mô hình ngược

Để nhận dạng mô hình thuận của đối tượng P, ta sử dụng cấu trúc như trên hình 3(a). Tín hiệu vào x được đưa vào đồng thời cho cả đối tượng P và mạng nơron. Tín hiệu ra của mạng nơron được so sánh với tín hiệu ra của đối tượng P. Ta có sai lệch $d - y_1$. Trong đó, d = y là tín hiệu ra của đối tượng ứng với tín hiệu vào x – đây cũng chính là tín hiệu ra mong muốn đạt được của mạng nơron và y1 là tín hiệu ra thực của mạng nơron. Chuẩn của vectơ sai lệch $\|d - y_1\|$ sẽ được sử dụng để đào tạo mạng nơron. Nó sẽ được lan truyền ngược trên toàn cấu trúc mạng để thay đổi khối lượng liên kết giữa các nơron để làm cực tiểu sai lệch đó. Kết quả ta sẽ có một mạng nơron sau khi được đào tạo sẽ thực hiện một ánh xạ $x \rightarrow y_1 \sim y$ với bất kỳ một cấp chính xác nào mà ta mong muốn. Mạng nơron sau khi được đào tạo chính là mô hình đồng dạng của đối tượng điều khiển.

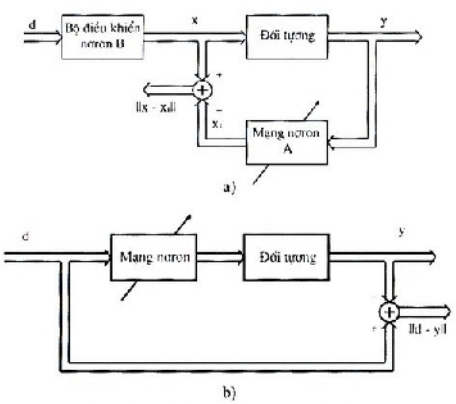
Tương tự như vậy, trên hình 3(b) là cấu trúc nhận dạng mô hình ngược của đối tượng P. Tín hiệu ra của đối tượng P được sử dụng làm tín hiệu vào của mạng nơron. Tín hiệu ra của mạng nơron x_1 được so sánh với tín hiệu vào của đối tượng P. Chuẩn của vectơ sai lệch $\|x - x_1\|$ sẽ được sử dụng để đào tạo mạng nơron. Nó được lan truyền ngược trên toàn cấu trúc mạng để thay đổi khối lượng liên kết giữa các nơron để làm cực tiểu sai lệch này. Mạng nơron sau khi được đào tạo chính là mô hình ngược của đối tượng P. Nó thực hiện một ánh xạ ngược $y \rightarrow x_1 \sim x$ với bất kỳ một cấp chính xác nào mà ta mong muốn. Tuy nhiên, nếu mô hình ngược của đối tượng P không tồn tại duy nhất, nghĩa là ứng với một

giá trị của y có nhiều hơn một giá trị của x, thì ta sẽ không thể thực hiện nhận dạng thành công được, như thấy trên hình 4. Trong trường hợp này mạng nơron chỉ thực hiện được một ánh xạ ngược $y \rightarrow x_1$ = giá trị trung bình của các giá trị x.



Hình 4. Ví dụ nhận dạng mô hình ngược (a) mô hình ngược tồn tại duy nhất, (b) mô hình ngược không tồn tại duy nhất

Khi tham số của đối tượng thay đổi trong quá trình hoạt động, các khối lượng của mạng nơron sẽ được thay đổi thích nghi với sự thay đổi tham số đó của đối tượng nếu mạng được đào tạo on-line. Như vậy, nhận dạng đối tượng điều khiển dùng mạng noron mang tính thích nghi với sự thay đổi tham số của đối tượng trong quá trình hoạt động. Hình 5(a) trình bày một cấu trúc đào tạo on-line cho mạng nơron để nhận dạng một đối tượng điều khiển. Đối tượng được điều khiển trong một hệ hở. Ta thấy trong hình 5 có hai mạng nơron A và B. Bộ điều khiển nơron B là một phiên bản sao chép y nguyên theo mạng nơron A. Mạng nơron A được đào tạo nhận dạng mô hình ngược của đối tượng. Mạng nơron B sẽ bám theo mạng nơron A – chính là mô hình ngược của đối tượng – sau mỗi một bước đào tạo, nên tín hiệu vào bộ điều khiển nơron B (d) sẽ bằng với tín hiệu ra của đối tượng y, nghĩa là d = y. Vậy tín hiệu vào của bộ điều khiển nơron B chính là tín hiệu mong muốn nhận được của hệ thống. Chú ý rằng, sơ đồ này chỉ có



Hình 5. Điều khiển hệ hở với mạng nơron nhận dạng mô hình ngược (a) dùng hai mạng nơron, trong đó (B) là một phiên bản của (A), (b) dùng một mạng nơron.

thể thực hiện được khi mô hình ngược của đối tượng tồn tại duy nhất. Tuy nhiên, sơ đồ này là không kinh tế vì phải sử dụng đến hai mạng nơron. Sơ đồ điều khiển trên hình 5(b) có chức năng giống như sơ đồ điều khiển trên hình 5(a), nhưng nó chỉ cần duy nhất một mạng nơron. Tín hiệu vào mạng nơron là giá trị mong muốn nhận được của hệ thống. Chuẩn của vectơ sai lệch ||d – y|| được sử dụng để đào tạo mạng nơron. Mạng nơron được đào tạo chính là mô hình ngược của đối tượng.

Chú ý rằng, các sơ đồ nhận dạng đối tượng điều khiển ở trên chỉ dùng cho các hệ thống tĩnh, nghĩa là tín hiệu ra và vào là bất biến với thời gian. Đối với hệ thống động, mạng nơron sẽ cần đến các tín hiệu ra của hệ thống ở các thời điểm trước đó để nhận dạng đối tượng. Vì thế ta sẽ thêm vào các khâu lấy trễ tín hiệu ra và dùng làm tín hiệu vào cấp cho mạng nơron.

Ví dụ nhận dạng hệ thống

Trong phần này, ta sẽ thực hiện minh hoạ một ví dụ đơn giản thông qua mô phỏng trên MATLAB. Như đã nói ở trên, mục đích của nhận dạng đối tượng là để tìm ra một mô hình phù hợp của đối tượng phi tuyến bằng cách quan sát các tín hiệu vào/ra của đối tượng đó. Đối với một quá trình phi tuyến cho trước, ta cần phải tính ra được mối quan hệ về mặt hàm số giữa tín hiệu vào và tín hiệu ra. Bản thân đối tượng là một hộp đen có cấu trúc chưa biết. Tất cả những gì mà ta có thể quan sát được là cái gì đi vào và đi ra khỏi hộp đen. Đối với quá trình nhận dạng như thế, áp dụng mạng nơron sẽ cho kết quả tốt nhất.

Ta hay xét một ví dụ sau: Một hệ thống phi tuyến được mô tả bởi phương trình:

$$y_p(k+1) = \frac{y_p(k)(y_p(k-1)+2)(y_p(k)+2.5)}{8.5+(y_p(k))^2+(y_p(k-1))^2} + u(k) \quad (5)$$

Đối tượng này hoạt động ổn định đối với u(k) ∈ [-2, 2]. Mục đích trong ví dụ này là đào tạo mạng nơron thắng theo luật học lan truyền ngược, để sau khi đào tạo, tín hiệu ra của mạng và tín hiệu thực của đối tượng là trùng nhau ứng với cùng một đầu vào. Phần dưới đây trình bày các bước thiết kế mạng nơron để nhận dạng đối tượng một cách tuần tự.

Bước 1: Đầu tiên, ta cần tìm cặp dữ liệu vào/ra để đào tạo mạng nơron. Trong ví dụ này, ta mô phỏng hệ phi tuyến trên để tìm dữ liệu đào tạo. Trong các hệ thống thực, dữ liệu đào tạo thu được từ thực nghiệm. Tạo vectơ tín hiệu vào u [-2, 2]. Mục đích là để tạo một vectơ tín hiệu vào ngẫu nhiên để phát ra một vectơ các tín hiệu ra tương ứng. Trong ví dụ này, ta tạo ra một vectơ có 301 tín hiệu vào ngẫu nhiên. Mô phỏng đáp ứng của đối tượng phi tuyến sử dụng vectơ tín hiệu vào ngẫu nhiên đó để tạo ra các cặp dữ liệu đào tạo.

```
yp(1)=0; yp(2)=0; % yp(1)=y(k-1); yp(2)=y(k)
u=rands(1,301)*2; % -2<= u(k) <=2
for k=2:301
```

```
yp(k+1)=yp(k)*(yp(k-1)+2)*(yp(k)+2.5)/
(8.5+yp(k)^2+yp(k-1)^2)+u(k);
out(k-1)=(yp(k+1)-u(k))/20; %output training data
in(k-1)=yp(k)/20;   %input training data
end;
```

Chú ý rằng, dữ liệu đào tạo vào và ra đều được chia cho hệ số tỷ lệ, trong ví dụ này ta lấy bằng 20. Việc chia cho hệ số tỷ lệ là cần thiết khi sử dụng các hàm kích thích dạng sigmoid để nén dữ liệu. Người thiết kế phải có kinh nghiệm trong chọn hệ số tỷ lệ một cách phù hợp để mạng nơron hội tụ nhanh hơn. Hệ số tỷ lệ lớn quá hay nhỏ quá cũng ảnh hưởng đáng kể đến các đặc tính hội tụ của mạng.

Bước 2: Thiết lập các vectơ dữ liệu vào/ra cho đào tạo mạng nơron. Ở đây, dữ liệu vào "plantin" được thiết lập thành một cặp dưới dạng:

$$\begin{bmatrix} y_p(k) \\ y_p(k+1) \end{bmatrix} = \begin{bmatrix} in(1) \\ in(2) \end{bmatrix}, \begin{bmatrix} in(2) \\ in(3) \end{bmatrix}, \begin{bmatrix} in(3) \\ in(4) \end{bmatrix}, \ldots, \begin{bmatrix} in(299) \\ in(300) \end{bmatrix} \quad (6)$$

và dữ liệu ra "plantout" cho mỗi cặp dữ liệu vào tương ứng được thiết lập dưới dạng:

$$[out(1), out(2), out(3), \ldots, out(299)] \quad (7)$$

```
plantin=[in(1:299);in(2:300)];
plantout=out(1:299);
```

Bước 3: Chọn sơ đồ mạng nơron, chọn phương pháp đào tạo và thiết lập các tham số chỉ tiêu trong quá trình đào tạo. Trong ví dụ này ta chọn sơ đồ mạng nơron thẳng 3 lớp, trong đó có 1 lớp ẩn có 10 nơron. Lớp nơron vào và ra đều có 1 nơron. Các nơron đều sử dụng hàm kích thích dạng tansig. Đào tạo mạng theo thuật toán lan truyền ngược kết hợp với thuật toán tìm tối ưu Levenberg-Marquardt. Trước tiên, ta khởi tạo các khối lượng, sau đó thiết lập số epochs đào tạo và sai số nhỏ nhất có thể chấp nhận được. Ta có thể phải tăng số epochs để quá trình đào tạo mạng hội tụ. Mặt khác, ta cũng có thể phải tăng giá trị sai số nhỏ nhất có thể chấp nhận được để quá trình đào tạo mạng hội tụ. Trong ví dụ này ta chọn epochs = 500 và sai số nhỏ nhất là: 0,0005.

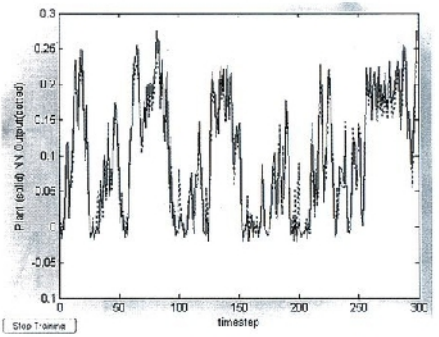
```
net=newff(minmax(plantin),[1 10 1],{'tansig' 'tansig'
'tansig'}, 'trainlm', 'learngdm','mse')
net = init(net);
net.iw{1,1}
net.b{1}
net.trainParam.epochs = 500;
net.trainParam.goal = 0.0005;
```

Bước 4: Đào tạo mạng nơron với dữ liệu đào tạo.

```
net=train(net,plantin,plantout);
```

Bước 5: Tìm đáp ứng của mạng nơron với các tín hiệu vào ngẫu nhiên đã sử dụng để đào tạo. Đồng thời so sánh đáp ứng của mạng nơron với đáp ứng thực của đối tượng.

```
trainedout=sim(net,plantin);
plot(plantout,'b');
hold on;
plot(trainedout,':k');
axis([0, 300,-0.1, 0.3]);
xlabel('timestep');
ylabel('Plant (solid) NN Output(dotted)');
pause;
```



Hình 6. Đáp ứng của mạng nơron và đáp ứng thực của đối tượng ứng với đầu vào là tín hiệu ngẫu nhiên.

Đáp ứng của mạng nơron được đào tạo trong bước 5 và đáp ứng thực của đối tượng tương đối trùng nhau như được chỉ ra trong hình trên.

Bước 6: Ta muốn rằng mạng nơron vừa được đào tạo ở trên có thể đưa ra các đáp ứng trùng với đáp ứng thực của đối tượng ứng với cùng một tín hiệu vào khác với các tín hiệu vào đã được sử dụng để đào tạo mạng. Nếu hai đáp ứng trùng nhau thì mạng nơron đó có thể sử dụng để nhận dạng đối tượng rất tốt. Còn nếu hai đáp ứng chưa trùng nhau, thì ta cần phải quay lại bước 1 để thu thêm dữ liệu đào tạo và đào tạo thêm cho mạng.

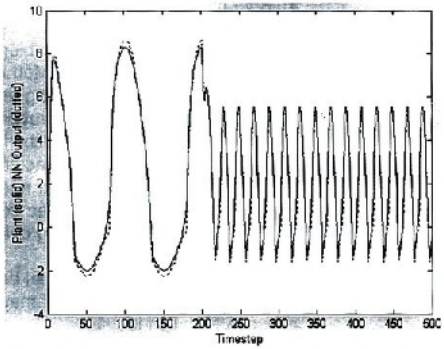
Trong ví dụ này, ta tìm đáp ứng của mạng nơron và đáp ứng thực của đối tượng ứng với tín hiệu vào theo hàm cos và sin như được minh họa trong đoạn chương trình và trong hình dưới đây.

```
yp(1)=0;yp(2)=0;out1(1)=0;out1(2)=0;
for k=2:500
   if(k<=200) u(k)=2.0*cos(2*pi*k*0.01);
   else
      u(k)=1.2*sin(2*pi*k*0.05);
   end;
     yp(k+1)=yp(k)*(yp(k-1)+2)*(yp(k)+2.5)/(8.5+
     yp(k)^2+yp(k-1)^2)+u(k);
     out1(k)=yp(k)/20;
     out1(k-1)=yp(k-1)/20;
    nnout(k+1)=20*sim(net,[out1(k);out1(k-1)])+u(k);
end;
plot(yp,'b');
hold on;
plot(nnout,':k');
axis([0, 500, -4.0, 10.0]);
xlabel('Timestep');
ylabel('Plant (solid) NN Output (dotted)');
```

(Xem tiếp trang 27)

ỨNG DỤNG MẠNG NƠRON ...

(Tiếp theo trang 25)



Hình 7. Đáp ứng của mạng nơron và đáp ứng thực của đối tượng ứng với một đầu vào khác.

Kết quả mô phỏng cho thấy đáp ứng thực và đáp ứng của mạng nơron là tương đối trùng nhau. Do vậy, ta không cần phải đào tạo thêm cho mạng.

Kết luận

Nhờ khả năng học, mạng có thể thích nghi với sự thay đổi tham số của đối tượng điều khiển trong quá trình hoạt động. Vì thế, nhận dạng đối tượng điều khiển sử dụng mạng nơron là rất hiệu quả. Kết quả mô phỏng trên MATLAB cũng cho thấy đáp ứng của mạng nơron bám theo đáp ứng thực của đối tượng điều khiển là rất tốt. Ngoài mục đích phân tích lý thuyết nhận dạng đối tượng điều khiển sử dụng mạng nơron, bài báo còn đề cập chi tiết từng bước thiết kế một mạng nơron. Điều này rất có ích cho những ai mới bắt đầu tiếp xúc với lĩnh vực mạng nơron - một lĩnh vực không những đang thu hút được sự chú ý của các nhà khoa học, các nhà nghiên cứu, các kỹ sư mà còn đang thu hút được sự chú ý ngày càng tăng của các bạn sinh viên.❑

Tài liệu tham khảo

1. Jacek M. Zurada, "Introduction to Artificial Neural Systems," West Publishing Company, St. Paul, 1992.

2. Hung T. Nguyen, Nadipuram R. Prasad, Carol L. Walker, and Elbert A. Walker, "A first course in Fuzzy and Neural Control," Chapman & Hall / CRC Press, Boca Raton, FL, 2002.

3. Math Works Inc., "Neural Network Toolbox User's guide," 1998.