# LỜI CAM ĐOAN

# LỜI CÁM ƠN

# TÓM TẮT LUẬN VĂN

# DANH MỤC HÌNH

# DANH MỤC BẢNG

**Danh mục chữ viết tắt**

# GIỚI THIỆU

## ĐẶT VẤN ĐỀ

Ngày nay khi mà hầu hết các tổ chức đều hoạt động trong môi trường không chắc chắn, kế hoạch lập ra hôm nay sẽ ảnh hưởng đến sự sống còn của tổ chức trong ngày mai thì việc dự đoán trước một cách chính xác trở nên rất quan trọng đối với các nhà ra quyết định. Các nhà đầu tư cần phải dự đoán được nhu cầu thị trường, sự biến động của nền kinh tế trong tương lai để có thể đầu tư hiệu quả. Các nhà hoạt định chính sách quốc gia cần dự đoán được về môi trường kinh doanh quốc tế, tỷ lệ lạm phát, tỷ lệ thất nghiệp… trong nhiều năm tới để đưa ra các chính sách phù hợp.

Để đưa ra dự báo chính xác và có cơ sở người ta tiến hành thu nhập dữ liệu về các yếu tố liên quan đến vấn đề mình quan tâm. Một kiểu dữ liệu thu nhập thường thấy là kiểu dữ liệu chuỗi thời gian. Dữ liệu chuỗi thời gian, tức là dữ liệu được thu nhập, lưu trữ và quan sát theo sự tăng dần của thời gian. Ví dụ, số lượng thí sinh dự thi đại học vào Trường Đại Học Bách Khoa thành phố Hồ Chí Minh được lưu trữ theo từng năm, hay số lượng hàng hóa đã bán được của một siêu thị được lưu trữ theo từng quý là các dữ liệu chuỗi thời gian.

Việc dự báo dữ liệu chuỗi thời gian ngày càng chiếm vị trí quan trọng trong hoạt động của các đơn vị tổ chức. Có rất nhiều phương pháp được xây dựng để dự báo chuỗi thời gian, nhiều phương pháp (ví dụ: phương pháp hồi quy) đã được xây dựng từ thế kỷ 19 và nhiều phương pháp (ví dụ phương pháp mạng neuron nhân tạo) được phát triển gần đây. Cơ bản có hai kỹ thuật chủ yếu trong việc dự báo chuỗi thời gian là các phương pháp thống kê: hồi quy, làm trơn, ARIMA… và phương pháp dùng mạng neuron nhân tạo.

## MỤC TIÊU CỦA ĐỀ TÀI

## CẤU TRÚC BÁO CÁO

Bài báo cáo chia làm 6 chương:

**Chương 1:** Giới thiệu về bài toán và nhiệm vụ đề tài.

Chương 2: Chuỗi thời gian và các thành phần của chuỗi thời gian

Chương 3: Mô hình ARIMA

Chương 4: Mô hình ANN

Chương 5: Mô hình kết hợp giữa ARIMA và ANN

Chương 6: Kết quả thực nghiệm

**Chương 7:** Kết luận về kết quả đạt được và hướng phát triển tương lai.

# : Chuỗi thời gian và các thành phần của chuỗi thời gian

## Chuỗi thời gian:

Dữ liệu chuỗi thời gian là dữ liệu được thu nhập, lưu trữ và quan sát theo sự tăng dần của thời gian. Ví dụ, số lượng thí sinh dự thi đại học vào Trường Đại Học Bách Khoa thành phố Hồ Chí Minh được lưu trữ theo từng năm, hay số lượng hàng hóa đã bán được của một siêu thị được lưu trữ theo từng quý là các dữ liệu chuỗi thời gian.

Ta thường ký kiệu chuỗi thời gian là với là các số tự nhiên. là các biến ngẫu nhiên rút ra từ một phân bố xác suất nào đó. Hình 1 là một ví dụ về chuỗi thời gian, số hành khách đặt chổ hàng tháng của hãng hàng không Pan Am.



Hình 1 Số khách hàng đặt chỗ hàng tháng của hãng Pan Am

Để tìm ra được những phương pháp dự báo phù hợp cho chuỗi thời gian, chúng ta phải xác định được các thành phần tạo nên chuỗi thời gian đó. Những thành phần cơ bản của chuỗi thời gian bao gồm: xu hướng, chu kỳ, thành phần mùa, bất qui tắc [1]

* Thành phần xu hướng (trend): chuỗi dữ liệu quan sát tăng hoặc giảm trong suốt thời đoạn quan sát. Đây là thành phần dài hạn (long term). Những chuỗi thời gian có chứa thành phần xu hướng thường gặp như: sự gia tăng dân số, tốc độ lạm phát, tăng trưởng của sản xuất.
* Thành phần chu kì (cyclical): những giá trị quan sát tăng lên, hoặc giảm xuống trong những khoảng thời gian khác nhau.Thành phần chu kì thường có dao động dạng sóng quanh trục xu hướng. Thường xuất hiện bởi sự gia tăng hoặc thu hẹp trong kinh tế.
* Thành phần mùa (seasonal): những thay đổi lặp lại hằng năm được gọi là tính mùa của chuỗi thời gian. Chu kì của những chuỗi thời gian có thể là tháng, quí, hoặc là 6 tháng, … Ví dụ: nhu cầu mua sắm dụng cụ học tập thường cao trong giai đoạn tháng 8, tháng 9 hằng năm. Tính mùa thường phản ánh điều kiện thời tiết, kì nghỉ trong năm.
* Thành phần bất qui tắc: Là thành phần thể hiện sự biến đổi ngẫu nhiên không thể đoán được của chuỗi thời gian.

Trong quá trình nghiên cứu chuỗi thời gian, ta nhận thấy rằng khi một biến được ghi nhận giá trị theo thời gian, các giá trị ở những thời điểm khác nhau có mối quan hệ hoặc tương quan với nhau. Để đo mức độ tương quan này, ta sử dụng hệ số tự tương quan (autocorrelation coefficient).

**Định nghĩa:** Tự tương quan (ACF) là sự tương quan giữa một biến với chính nó theo những độ trễ thời gian khác nhau [1].

Ta tính hê số tự tương quan của biến với độ trễ theo công thức

(2.1.1)

Với

\* là hệ số tự tương quan của ở độ trễ

\* là trung bình của

Nếu khác không thì giữa và có sự tương quan với nhau.

Để biểu diễn sự tự tương quan của một biến theo nhiều độ trễ khác nhau một cách trực quan, ta dùng hàm tự tương quan.

**Định nghĩa:** Hàm tự tương quan (PACF) là một đồ thị biểu diễn các hệ số tự tương quan theo các độ trễ khác nhau [1].

Hình 2 là một ví dụ về hàm tự tương quan.



Hình 2 Hàm tự tương quan

Trong thực tế ta chỉ có thể tính được hệ số tự tương quan lấy mẫu và dùng thống kê để ước lượng các hệ số tự tương quan của đám đông (population).

Công thức tính hệ số tự tương quan lấy mẫu:

(2.1.2)

Với

\* là hệ số tự tương quan lấy mẫu ở độ trễ

\* là trung bình mẫu của

\* là số phần tử của mẫu.

Để kiểm tra xem hệ số tự tương quan ở độ trễ của một chuỗi thời gian có khác không hay không, ta dùng phép thử .

Với :

(2.1.3)

\* là hệ số tự tương quan mẫu ở độ trễ

\* là độ trễ

\* là kích cỡ mẫu

Nếu thì sẽ có phân phối student với bậc tự do. Đối với mẫu có kích thước lớn, với mức ý nghĩa 5%, nếu hệ số tự tương quan mẫu nằm trong khoảng thì ta có thể kết luận hệ số tự tương quan của đám đông bằng không với mức ý nghĩa 5%.

Việc đánh giá các hệ số tự tương quan có ý nghĩa quan trọng trong việc phân tích chuỗi thời gian. Hàm tự tương quan của dữ liệu giúp ta xác định được các thành phần của chuỗi thời gian từ đó có thể lựa chọn mô hình dự báo hợp lý cũng như việc đánh giá tính đầy đủ của mô hình.

## Dự báo chuỗi thời gian

## Khái niệm thống kê liên quan

## Nhận biết các thành phần của chuỗi thời gian bằng phương pháp phân tích hệ số tự tương quan:

Như đã đề cặp ở trên, chuỗi thời gian có thể chứa bốn thành phần: xu hướng (trend), chu kỳ (cyclical), mùa (seasonal), bất quy tắc (irregular).

**Thành phần xu hướng (trend)**

Là thành phần thể hiện sự tăng hay giảm giá trị của chuỗi thời gian trong một giai đoạn dài hạn nào đó [1]. Nếu một chuỗi thời gian có chứa thành phần xu hướng sẽ có hệ số tự tương quan rất lớn ở những độ trễ đầu tiên và giảm dần về 0 khi độ trễ tăng lên. Hình 3 và 4 là một minh họa chuỗi thời gian có thành phần xu hướng. Ở đây dù mức tăng nhiệt độ toàn cầu có biến đổi theo từng năm nhưng nhìn chung mức tăng nhiệt độ trung bình có xung hướng tăng theo thời gian. Hệ số tự tương quan rất lớn ở những độ trễ đầu tiên và giảm dần theo sự tăng của độ trễ.



Hình 3 Độ tăng nhiệt độ trung bình hàng năm từ 1856 đến 2005



Hình 4 Hàm tự tương quan của chuỗi tăng nhiệt độ trung bình hàng năm từ 1856 đến 2005.

Chuỗi thời gian tĩnh (stationary time series) là chuỗi thời gian có trung bình và phương sai độc lập với thời gian [1]. Có nghĩa rằng, một chuỗi thời gian không có sự biến đổi tăng lên hay giảm xuống trong suốt thời gian thì được gọi là chuỗi thời gian tĩnh (hay gọi là chuỗi tĩnh).

Hệ số tự tương quan của chuỗi thời gian tĩnh giảm nhanh về 0, thông thường sau độ trễ thứ 2 hoặc thứ 3.

**Định Nghĩa :** Một chuỗi thời gian có tính chất tĩnh hay gọi là chuỗi thời gian tĩnh nếu nó thỏa mãn hai tính chất sau

(1)

(2) 

Ngược lại, một chuỗi thời gian có chứa thành phần mùa là chuỗi thời gian không tĩnh (nonstationary). Hệ số tự tương quan của chuỗi thời gian không tĩnh thường lớn ở vài độ trễ đầu tiên và sau đó giảm dần về 0 ở những độ trễ tiếp theo.



1. chuỗi thời gian tĩnh



1. chuỗi thời gian không tĩnh

Hình 6 (a) Chuỗi thời gian tĩnh, (b) chuỗi thời gian không tĩnh.

Chuỗi thời gian không tĩnh rất khó để biểu diễn bằng mô hình toán học và thực hiên dự báo [2]. Tuy nhiên, trong thực tế ta gặp rất nhiều dữ liệu thời gian là chuỗi không tĩnh. Do đó ta cần chuyển đổi chuỗi thời gian không tĩnh thành chuỗi thời gian tĩnh, đồng nghĩa với việc loại bỏ thành phần xu hướng ra khỏi chuỗi thời gian không tĩnh. Một trong những phương pháp phổ biến được sử dụng là phương pháp lấy hiệu (differencing)

Đối với một chuỗi thời gian không tĩnh , ta áp dụng toán tử lấy hiệu lên để được một chuỗi thời gian mới với

Nếu là chuỗi tĩnh thì ta xây dựng mô hình mô tả rồi từ đó suy ra . Nếu vẫn là chuỗi không tĩnh, ta tiếp tục áp dụng toán tử cho chuỗi .

**Thành phần chu kỳ (cyclical)**

Là chuỗi biến đổi dạng sóng quanh xu hướng [1]. Trong thực tế thành phần này rất khó xác định bởi vì những dao động dạng sóng quanh xu hướng luôn thay đổi cả về biên độ và khoảng thời gian xuất hiện. Do đó người ta thường xem nó như là một phần của thành phần xu hướng.

**Thành phần bất quy tắc (irregular)**

Là thành phần thể hiện sự biến đổi ngẫu nhiên không thể đoán được của chuỗi thời gian [1].

**Thành phần mùa (Seasonal)**

Theo như định nghĩa bên trên thì đây là thành phần thể hiện sự biến đổi lặp đi lặp lại tại từng thời điểm cố định theo từng năm của chuỗi thời gian [1]. Đối với chuỗi thời gian có thành phần mùa thì giá trị tại những thời điểm cố định theo từng năm sẽ có sự tương quan lớn với nhau. Ví dụ một chuỗi thời gian được ghi nhận theo từng quý có tính chất mùa thì hệ số tự tương quan ở độ trễ là 4 sẽ khác không một cách có ý nghĩa. Hình 5 là đồ thị của một chuỗi thời gian có tính mùa



Hình 5 Chuỗi thời gian có tính mùa.

Việc xác định một chuỗi thời gian có thành phần xu hướng hay thành phần mùa rất quan trong trong bài toán dự đoán chuỗi thời gian. Nó giúp ta lựa chọn được mô hình dự đoán phù hợp hay giúp cải tiến mô hình đã có chính xác hơn.

Bảng tóm tắt một số kĩ thuật dự đoán cho từng thành phần của chuỗi thời gian [1].

|  |  |  |
| --- | --- | --- |
| Thành phần | Kĩ thuật dự báo | Whenever ? |
| Chuỗi tĩnh | Trung bình đơn giản (simple average method)  Trung bình di động (moving average - MA)  Tự hồi quy kết hợp trung bình di động (Autoregressive moving average - ARMA) | Mô hình đơn giản có ít dữ liệu ban đầu như trường hợp mới thành lập của 1 công ty. Hoặc là những chuỗi thời gian được sinh ra trong những điều kiện, môi trường ổn định. |
| Xu hướng | Trung bình di động (MA)  Tự hồi qui tích hợp trung bình di động (autoregressive integrated moving average – ARIMA) | Xuất hiện trong thời đoạn phát triển của dòng đời 1 sản phẩm. Sự gia tăng dân số làm tăng lên nhu cầu về lương thực, thực phẩm, dịch vụ. Tất cả những dữ liệu này đều có tính xu hướng. |
| Mùa | Mô hình làm trơn theo hàm mũ của Winter (Winter’s exponential smoothing)  Tự hồi qui tích hợp trung bình di động (ARIMA) | Những dữ liệu này thường gắn liền với chu kì thời tiết, hoặc là các mốc thời gian đặc biệt của năm (khai giảng, giáng sinh, Tết…) |

Để có thể lựa chọn được kĩ thuật dự báo phù hợp cho chuỗi dữ liệu, ta cần đánh giá được độ tin cậy của kết quả, mức độ hiệu quả mà kĩ thuật đó mang lại.

Thông thường người ta sử dụng mức độ lỗi để đánh giá độ tin cậy và để so sánh giữa các kĩ thuật.

Lỗi (residual, forecast error) là sự sai khác giữa giá trị thực và giá trị dự đoán được.

trong đó: : lỗi

: giá trị thực của biến quan sát

 : giá trị dự đoán được của biến quan sát

: biến giá trị thời gian

Phương thức đánh giá phổ biến là trung bình lỗi tuyệt đối.

# : Mô hình ARIMA

Mô hình tự hồi quy tích hợp với trung bình di động(ARIMA) là một lớp mô hình tuyến tính có khả năng biểu diễn cả chuỗi thời gian tĩnh lẫn không tĩnh. Mô hình ARIMA dựa vào các mẫu tự tương quan trong bản thân của chuỗi thời gian để sinh ra dự đoán. Hệ thống các phương pháp dùng để xác định, kiểm tra và cải tiến mô hình ARIMA có sự đóng góp rất lớn của hai nhà thống kê, G.E.P.Box và G.M.Jenkins. Do đó việc mô hình và dự đoán dựa trên mô hình ARIMA còn được gọi là phương pháp luận Box-Jenkins [1].

## Mô hình trung bình di động bậc q, MA(q)

Giới thiệu mô hình di động

Mô hình MA(q) là mô hình có dạng

Với :

là trung bình của chuỗi thời gian tĩnh .

là sai số, biểu diễn thành phần không thể dự đoán được ở thời điểm i.

là các hệ số ước lượng mức ảnh hưởng của lên .

Biểu diễn (3.1.1) bằng toán tử lấy hiệu *B* ta được

Ở đây và

Vì chuỗi là chuỗi nhiễu trắng (white noise), tức là và

Nên

Từ đó ta có hàm tự tương quan của chuỗi

Dấu hiệu này của hàm tự tương quan giúp ta xác định được bậc của mô hình trung bình di động. Một mô hình trung bình di động có bậc thì hệ số tự tương quan của nó sẽ bằng không với những độ trễ lớn hơn . Trong thực tế do sai biệt khi lấy mẫu, hệ số tương quan mẫu có thể khác không ở những độ trễ lớn hơn . Tuy vậy, nếu hệ số tương quan mẫu nằm trong khoảng , là kích thước mẫu, thì ta có thể kết luận nó bằng không với độ tin cậy 95%.

Ta dùng R để mô phỏng mô hình MA(2) sau:

Ở đây,

*> set.seed(1)*

*> y <- e <- rnorm(1000)*

*> for(t in 3:1000){*

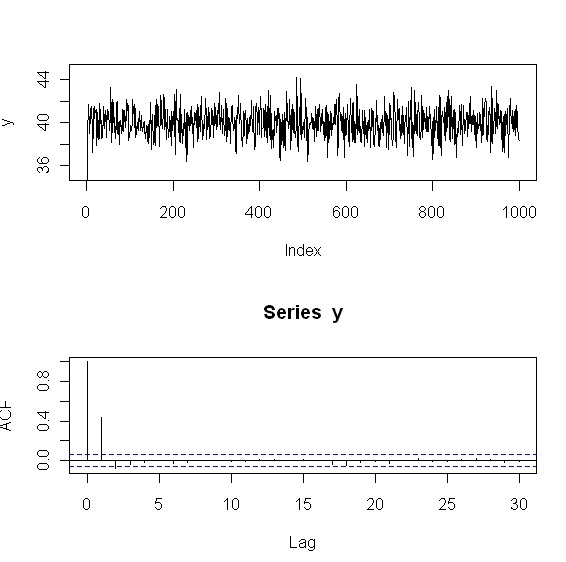
*y[t] <- 40 + y[t] + 0.7\*e[t-1] - 0.28\*e[t-2]*

*}*

*> plot(y,ylim=c(35,45),type ="l")*

*> acf(y)*

Hình 7 là kết quả mô phỏng. Ở đây ta thấy các hệ số tự tương quan ở độ trễ lớn hơn 2 đều bằng không ở mức ý nghĩa 5%.



Hình 7 Hình ảnh của mô hình MA(2)

## Mô hình tự hồi quy bậc p, AR(p)

Giới thiệu mô hình tự hồi qui?

Mô hình tự hồi quy bậc p là mô hình có dạng

(3.2.1)

Với

là hằng số.

là sai số, biểu diễn thành phần không thể dự đoán được từ mô hình ở thời điểm t.

là các trọng số được ước lượng thể hiện sự ảnh hưởng của các giá trị lên .

Từ (3.2.1) ta có:

Suy ra

(3.2.2)

Phương trình (3.2.2) chính là phương trình Yule-Walker cho hàm ACF của mô hình AR(p).

Theo phương trình (3.2.2) ta thấy hàm tương quan của mô hình AR(p) không giảm về 0 sau độ trễ p như mô hình MA(q), do đó ta cần quan sát một dấu hiệu khác cung cấp nhiều thông tin hơn về bậc của mô hình AR, đó là hàm tự tương quan riêng phần.

**Hàm tự tương quan riêng phần.**

Hàm tự tương quan riêng phần ở độ trễ k là sự tương quan giữa và sau khi đã hiệu chỉnh sự ảnh hưởng của các giá trị [1]. Trong mô hình AR(p), hàm tự tương quan riêng phần ở độ trễ sẽ bằng 0.

Xét hệ phương trình Yule-Walker cho hàm tự tương quan của mô hình AR(k) với bậc cố định nào đó

Viết lại hệ phương trình trên với dạng ma trận ta được

Hay

Với

Cho mọi hệ số gọi là hệ số tự tương quan riêng phần của mô hình ở độ trễ k. Đối với mô hình AR(p) thì dễ thấy với . Do đó ta nói rằng PACF của mô hình AR(p) bằng không sau độ trễ p. Đây là một dấu hiệu quan trọng để nhận biết bậc của mô hình tự hồi quy.

Trong thực tế các hệ số tự tương quan riêng phần của mẫu có thể không bằng không sau độ trễ p do sai biệt khi lấy mẫu. Tuy nhiên, ta có thể kết luận hệ số tự tương quan riêng phần bằng không ở mức ý nghĩa 5% nếu nó nằm trong khoảng , với là kích thước mẫu.

Ta dùng R để mô phỏng mô hình AR(2) sau:

*> set.seed(1)*

*> f <- c(0.4,0.5)*

*> y <- e <- rnorm(1000)*

*> for(t in 3:1000){*

*+ y[t] <- 4 + 0.4\*y[t-1] + 0.5\*y[t-2]+ e[t]*

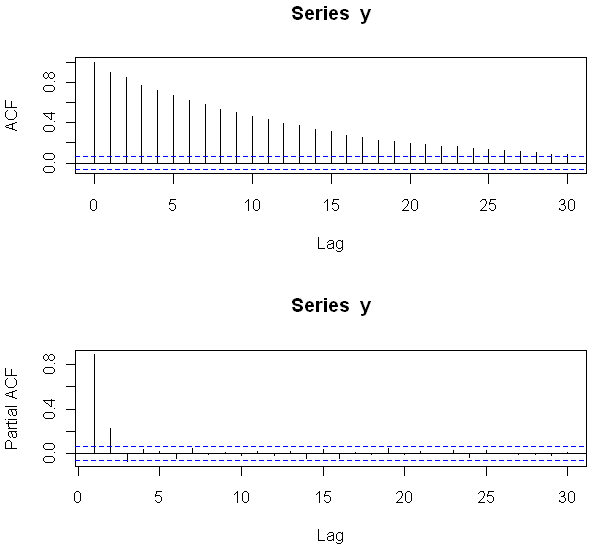
*+ }*

*> layout(1:2)*

*> acf(y)*

*> pacf(y)*

Hình 8 thể hiện hàm tự tương quan và hàm tự tương quan riêng phần của chuỗi thời gian sinh ra bằng mô hình trên. Ở đây ta thấy rằng hàm tự tương quan của mô hình AR(2) giảm dần về không, còn hệ số tự tương quan riêng phần thì bằng không với mọi độ trễ lớn hơn hai.



Hình 8 ACF và PACF của mô hình 

## Mô hình kết hợp tự hồi quy và trung bình di động, ARMA(p,q)

Lí do dẫn đến sự kết hợp này(tìm hiểu thêm)

Mô hình ARMA(p,q) là sự kết hợp hai mô hình AR(p) và MA(q), nó được biểu diễn như sau

Hay

Ý nghĩa của các ký hiệu giống như trong mô hình AR(p) và MA(q).

Mô hình ARMA(p,q) có chứa đựng mô hình AR(p) và MA(q) nên hàm tự tương quan và hàm tự tương quan riêng phần của nó vừa thể hiện hành vi của mô hình AR(p) và MA(q). Tức là hàm tự tương quan riêng phần và hàm tự tương quan giảm dần về không.

Ta dùng R mô phỏng mô hình ARMA(1,1) sau

*> set.seed(1)*

*> y <- e <- rnorm(1000)*

*> for(t in 2:1000){*

*+ y[t] <- 16 + 0.6\*y[t-1]+ e[t] + 0.8\*e[t-1]*

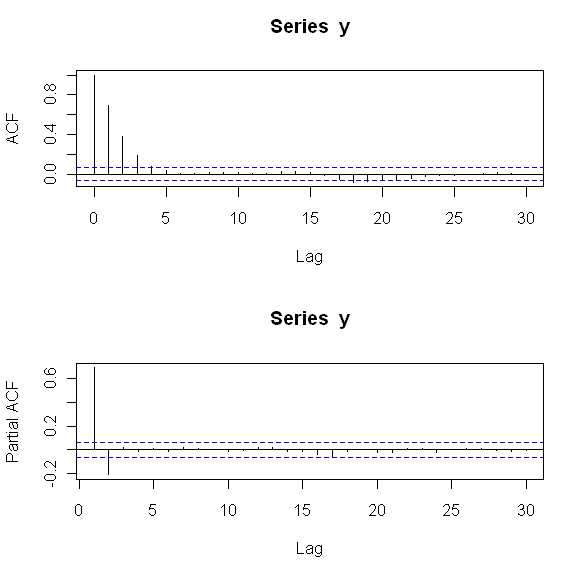
*+ }*

*> layout(1:2)*

*> acf(y)*

*> pacf(y)*

Hình 9 là hình ảnh của hàm tự tương quan và tự tương quan riêng phần của chuỗi thời gian sinh ra bằng mô hình trên. Ta thấy hàm tự tương quan và hàm tự tương quan riêng phần giảm dần về không.



Hình 9 Hàm tự tương quan và tự tương quan riêng phần của mô hình

## Mô hình tự hồi quy kết hợp với trung bình di động, ARIMA(p,d,q)

Mô hình ARMA(p,q) làm việc tốt với các dữ liệu tĩnh, tức là không có thành phần xu hướng (trend). Nhưng trong thực tế dữ liệu thường có một xu hướng đi lên hay đi xuống nào đó, đặc biệt với các chuỗi thời gian trong kinh tế.

Đối với các chuỗi thời gian có xu hướng tuyến tính ta cần chuyển nó thành tuyến tính rồi mới áp dụng mô hình ARMA. Một trong những cách chuyển một chuỗi thời gian có xu hướng về chuỗi thời gian tĩnh là phương pháp lấy hiệu như đã nói ở chương 2. Do đó người ta đưa thêm một hệ số d vào mô hình ARMA(p,q) để tạo thành mô hình ARIMA(p,d,q), với d là số lần lấy hiệu để chuỗi thời gian trở thành chuỗi tĩnh. Khi d bằng không thì mô hình ARIMA(p,d,q) trở thành mô hình ARMA(p,q). Mô hình ARIMA có thể làm việc tốt với dữ liệu tĩnh lẫn dữ liệu có xu hướng tuyến tính.

Dạng toán học của mô hình ARIMA(p,d,q) là

Ý nghĩa của các ký hiệu giống như mô hình AR(p) và MA(q).

## Mô hình ARIMA có tính mùa (SARIMA)

Mô hình ARIMA có thể làm việc tốt với dữ liệu có xu hướng tuyến tính nhưng vẫn chưa xử lý tốt các chuỗi thời gian có tính mùa, tức là chuỗi thời gian có những thành phần thể hiện sự biến đổi lặp đi lặp lại tại từng thời điểm cố định theo từng năm. Do đó người ta đưa vào bốn hệ số P,D,Q,s vào mô hình ARIMA(p,d,q) để tạo thành mô hình ARIMA(p,d,q) x (P,D,Q)s. Ở đây s chính là thời điểm mà thành phần mùa sẽ lặp lại, nếu dữ liệu được quan sát theo tháng thì s = 12, nếu dữ liệu quan sát theo quý thì s = 4. Mô hình ARIMA cho dữ liệu có tính mùa đôi khi còn được gọi mô hình SARIMA.

Dạng toán học của mô hình ARIMA(p,d,q) x (P,D,Q)s là

Với

Việc xác định các thông số P, D, Q tương tự như việc xác định các hệ số p, d, q thông thường chỉ khác ở chỗ các dấu hiệu của hàm tự tương quan và hàm tự tương quan riêng phần khác thể hiện theo các độ trễ là bội của s.

## Xây dựng mô hình SARIMA

Cấu trúc của mô hình dự đoán SARIMA gồm 4 thành phần chính:

* Thành phần nhận dạng mô hình (Model identification): ước lượng và xấp xỉ cấu trúc cho các mối tương quan tồn tại trong dữ liệu đã thu thập được. Thành phần này nhận dạng được mô hình một cách tự động bằng cách phân tích hệ số tự tương quan và hệ số tự tương quan riêng phần theo các độ trễ khác nhau. Do đó sẽ tránh được việc phải dựa vào các đồ thị dữ liệu để xác định mô hình.
* Thành phần khử mùa và xu hướng (Recursive differencer/integrator): dùng để chuyển đổi các chuỗi dữ liệu không tĩnh hoặc có tính mùa sang chuỗi dữ liệu tĩnh trước khi áp dụng các giải thuật để ước lượng các tham số cho mô hình. Bởi vì mô hình ARIMA chỉ phù hợp với chuỗi dữ liệu tĩnh, các thành phần bị loại bỏ khỏi chuỗi dữ liệu trong quá trình chuyển đổi phải được tích hợp lại vào dữ liệu khi dự đoán.
* Thành phần ước lượng tham số (Parameter estimator): ước lượng tham số của mô hình bằng giải thuật tối thiểu sai số đệ quy. Giải thuật này sử dụng đệ quy để giảm chi phí tính toán và kĩ thuật tối thiểu lỗi để tối thiểu bình phương các ước lượng lỗi. Giải thuật này có hai giả định là: mô hình phải tuyến tính với các tham số và chuỗi lỗi phải có phân phối tự nhiên với giá trị trung bình là 0 và phương sai không đổi. Nếu hai giả định này bị vi phạm thì mô hình ước lượng sẽ không chính xác.
* Thành phần dự đoán (N-step ahead predictor): tạo giá trị dự đoán cho các giai đoạn tiếp theo dựa vào mô hình đã ước lượng được.

MODEL STRUCTURE IDENTIFIER

RECURSIVE DIFFERENCER/ INTEGRATER

PARAMETER ESTIMATOR

N-STEP AHEAD PREDICTOR

Stationary Series

Non-stationary and/or Seasonal series

Forecasts for Original Series

Differenced Series

Estimated Parameters

Estimated Parameters

Forecasts for Original Series

Forecasts for Differenced Series

### Thành phần nhận dạng mô hình

* Nhận dạng mô hình là khâu đầu tiên trong quá trình mô hình hóa SARIMA. Chúng ta cần phải xác định các tham số p, d, q và P, Q, D, S của mô hình SARIMA. Những giá trị trị này được suy ra bằng cách trích rút những hệ số tương quan quan trọng trong chuỗi dữ liệu quan sát được. Kĩ thuật truyền thống dùng để nhận dạng mô hình là quan sát đồ thị và trích rút các hệ số từ đồ thị. Tuy nhiên, phương pháp này có nhược điểm lớn là phụ thuộc vào độ hiểu biết và khả năng phân tích đồ thị của người dự đoán. Do đó cần phải tìm ra một phương pháp tự động, sử dụng các công thức toán học để trích rút ra các hệ số.
* Mô hình SARIMA sử dụng tính chất của ACFs và PACFs để kết hợp một chuỗi dữ liệu với một mô hình thích hợp: mô hình tự hồi quy AR(p), mô hình trung bình di động MA(q), mô hình kết hợp tự hồi quy và trung bình di động ARMA(p,q). Tính chất của các hàm tương quan được chia ra làm 3 loại: giảm gián đoạn (abrupt decay), giảm theo hình sin hoặc hàm mũ, giảm chậm. Tùy theo tính chất của ACFs và PACFs mà chuỗi dữ liệu được kết hợp với 1 mô hình khác nhau
* Mô hình tự hồi quy AR(p): quá trình có ACFs với các độ trễ giảm chậm về không theo hình sin hoặc theo hàm mũ, trong khi đó PACFs gián đoạn về không sau một vài độ trễ.
* Mô hình trung bình di động MA(q): quá trình có tính chất ngược lại với mô hình AR(p), ACFs gián đoạn về không sau một vài độ trễ, trong khi đó PACFs với các độ trễ giảm về không theo hình sin hoặc theo hàm mũ.
* Mô hình kết hợp tự hồi quy và trung bình di động ARMA(p,q): quá trình có ACFs và PACFs giảm về không theo hình sin hoặc theo hàm mũ.

Nếu ACFs giảm chậm thì chuỗi dữ liệu không tĩnh, có tính xu hướng cần phải được chuyển đổi trước khi ước lượng.

|  |  |  |
| --- | --- | --- |
| Process | ACF | PACF |
| AR(p) | Exponential or sinusoidal decay to zero | Spikes cut off to zero after lag p |
| MA(q) | Spikes cut off to zero after lag q | Exponential or sinusoidal decay to zero |
| ARMA(p,q) | Exponential or sinusoidal decay to zero | Exponential or sinusoidal decay to zero |

* Quá trình nhận dạng mô hình SARIMA bao gồm 4 giai đoạn chính:
* Xác định bậc lấy hiệu d để loại bỏ tính xu hướng trong chuỗi dữ liệu. Nếu ACFs của chuỗi dữ liệu mang tính chất giảm chậm thì chuỗi dữ liệu được lấy hiệu cho đến khi ACFs không mang tính chất giảm chậm. Sau khi lấy hiệu d lần, chuỗi dữ liệu đã mất tính xu hướng. Tuy nhiên nếu chuỗi dữ liệu có tính mùa, thì ACFs có giá trị lớn tại các cận của mùa. Khoảng cách giữa các cận chính là chiều dài mùa S.
* Loại bỏ tính mùa: chuỗi dữ liệu được lấy hiệu D lần với khoảng lấy hiệu là S để loại bỏ tính mùa. Với các chuỗi dữ liệu thường gặp chỉ qua 1 lần lấy hiệu là có thể loại trừ được tính mùa trong chuỗi dữ liệu.

Đối với những quá trình có tính mùa, các giá trị quan sát không chỉ phụ thuộc vào những giá trị lân cận mà còn phụ thuộc vào những giá trị cách nó một bội số của chiều dài mùa. Do vậy để xác định chu kì mùa, ta cần:

* Xác định các hệ số tương quan tối ưu: các hệ số này cao hơn các hệ số lân cận và là giá trị cao nhất cục bộ.
* Tính khoảng cách giữa các hệ số tương quan tối ưu: sự sai khác về độ trễ giữa các hệ số liên tiếp.
* Nhận dạng khoảng cách được lặp lại nhiều nhất, nếu tần suất của khoảng cách này lớn thì đó chính là chu kì mùa.
* Xác định mô hình cho thành phần thường: trong chuỗi dữ liệu được biến đổi, ta phân tích ACFs và PACFs trong khoảng thời gian 1 mùa để xác định p, q.

Như đề cập ở trên, các ACFs và PACFs đều có mô hình thuộc một trong ba dạng sau:

* Giảm chậm trong các quá trình không tĩnh
* Giảm theo hình sin hoặc hàm mũ trong quá trình tĩnh
* Giảm gián đoạn trong quá trình tĩnh

Bằng cách tính tỉ lệ thay đổi trung bình và so sánh chúng với các ngưỡng, ta tính được các hệ số của mô hình.

với M là số hệ số tương quan có ý nghĩa, là hệ số tương quan tại độ trễ k.

Phương pháp xác định mô hình trong hệ số tương quan:

if( > 65%) then pattern = abrupt cutoff

else if ( < 10%) then pattern = slow decay

else pattern = exponential decay

Sau khi xác định tính chất của hệ số tương quan ACF và PACF, ta xác định mô hình ARIMA phù hợp theo giải thuật sau:

if (high frequencies exist in ACF or PACF) {

if (ACF pattern = abrupt cut-off at lag q) then

Model = pure MA(q)

else if (PACF pattern = abrupt cut-off at lag p) then

Model = pure AR(p)

else if (both ACF and PACF pattern = exponentila decay) then

Model = ARMA(p,q)

}

else {

p = q =0;

Model = ARMA(0,0)  
}

* Xác định mô hình cho thành phần mùa: tương tự như xây dựng mô hình cho thành phần thường nhưng ta phân tích ACFs và PACFs tại các cận mùa để xác định P, Q.

### Thành phần khử mùa và xu hướng

Trong thực tế, các chuỗi dữ liệu rất phức tạp và thường mang tính xu hướng hoặc tính mùa. Do đó, các chuỗi dữ liệu cần được biến đổi thành các quá trình tĩnh trước khi đưa vào thành phần nhận dạng mô hình. Lấy hiệu là kĩ thuật được sử dụng phổ biến nhất.

Các chuỗi dữ liệu sau khi quá trình lấy hiệu sẽ được chuỗi dữ liệu tĩnh mới để xây dựng mô hình. Khi dùng mô hình này để dự báo, ta cần chuyển đổi lại dữ liệu như ban đầu.

Stationary Series

Non-Stationary Series Seasonal Series

differenced d+D times

integrated d+D times

Với một số chuỗi dữ liệu chỉ với một lần lấy hiệu ta có thể lấy được chuỗi dữ liệu tĩnh. Nhưng đối với một số chuỗi dữ liệu, ta phải lấy liệu nhiều lần và kết hợp cả hai phương pháp mới có được chuỗi dữ liệu tĩnh.

### Thành phần ước lượng tham số

Quá trình ước lượng tham số gồm 3 giai đoạn:

SET INITIAL CONDITIONS

ESTIMATE PARAME

TERS

PREPARE FOR NEXT ESTIMATION

Compute\_AprioriPredictionError Compute\_GainFactor 

Set\_DefaultInitialConditions

Compute\_AposteriorPredictionError Update\_ArpartOfObservationVector Update\_MapartOfObservationVector Update\_RinverseMatrix

* Giai đoạn 1: khởi tạo điều kiện cho quá trình hồi quy. Véc tơ tham số được khởi tạo bằng 0, . Nghịch đảo của ma trận phướng sai với giá trị .
* Giai đoạn 2: giá trị mới của véc tơ tham số được ước lượng dựa vào lỗi trước dự đoán và hệ số khuyếch đại.

Hệ số khuyếch đại được tính hồi quy dựa vào giá trị cũ, véc tơ quan sát và nghịch đảo ma trận phương sai .

Gain factor =

* Giai đoạn 3: cập nhật giá trị cho các biến liên quan để chuẩn bị cho vòng tiếp theo.

Giá trị lỗi sau dự đoán được tính toán dựa vào giá trị quan sát hiện tại và véc tơ tham số mới nhất

Nghịch đảo ma trận phương sai được tính dựa vào giá trị cũ và hệ số khuếch đại

Véc tơ giá trị quan sát và lỗi được cập nhật lại.

Giai đoạn 2 và 3 được lặp lại cho mỗi mẫu dữ liệu quan sát, qua mỗi lần ước lượng như vậy véc tơ tham số sẽ được cập nhật chính xác hơn.

### Thành phần dự đoán

Sau khi xác định được mô hình dự đoán và ước lượng tham số cho mô hình, ta sẽ dùng kết quả đó để dự đoán cho các thời đoạn tiếp theo.

* Dự đoán 1 thời đoạn cho chuỗi dữ liệu tĩnh

Theo phương pháp tối thiểu bình phương lỗi, giá trị dự đoán của mô hình chính là giá trị của mô hình tại thời điểm tiếp theo (t+1)

Bời vì tất cả các tham số trong mô hình ngoại trừ e(t+1), thành phần lỗi Gausian được giả thuyết là phân phối tự nhiên N(0,σ^2), chúng ta có thể ước lượng nó với giá trị kỳ vọng 0.

Do đó giá trị dự đoán là

* Dự đoán nhiều thời đoạn cho chuỗi dữ liệu tĩnh

Dựa vào dự đoán cho 1 giai đoạn, ta có thể mở rộng quá trình dự đoán n giai đoạn dựa vào các giá trị đã tính toán được, các giá trị lỗi tại mỗi thời đoạn được ước lượng với giá trị kỳ vọng 0.

* Dự đoán nhiều thời đoạn cho chuỗi dữ liệu không tĩnh hoặc có tính mùa

Quá trình này gồm hai công đoạn:

* Tính toán n giai đoạn dự đoạn cho chuỗi dữ liệu đã được chuyển đổi
* Khôi phục lại chuỗi dữ liệu gốc: thực hiện ngược lại quá trình lấy hiệu đã thực hiện khi khử đi tính không tĩnh và tính mùa của chuỗi dữ liệu gốc.

## Kết luận về mô hình ARIMA

Mô hình ARIMA là một mô hình tổng quát có thể sử dụng cho nhiều loại chuỗi thời gian trong thực tế, kể cả những chuỗi có thành phần xu hướng và thành phần mùa. Tuy nhiên mô hình ARIMA cũng có một số hạn chế.

1. Để xây dựng mô hình ARIMA cần phải có nhiều dữ liệu. Với những chuỗi không có tính mùa cần có 40 quan sát trở lên, với chuỗi dữ liệu có tính mùa cần dữ liệu được thu thập trong khoảng 6 đến 10 năm [1].
2. Không dễ cập nhập mô hình khi có them dữ liệu mới, thường là mô hình mới sẽ được xây dựng [1].
3. Việc xây dựng một mô hình ARIMA đầy đủ cần tốn nhiều thời gian và tài nguyên hơn các phương pháp truyền thống khác chẳng hạn như phương pháp làm trơn (smoothing) [1].

# Mô hình ANN

## SƠ LƯỢC VỀ MẠNG NEURON NHÂN TẠO

*Mạng neuron nhân tạo* (Artificial Neural Network) là một mô hình toán học định nghĩa một hàm số từ một tập đầu vào đến một tập đầu ra. Mạng neuron nhân tạo được mô phỏng theo mạng neuron sinh học trong bộ não người.

Theo các nhà sinh lý học thì bộ não con người chứa khoảng 1011các phần tử liên kết chặt chẽ với nhau gọi là các neuron. Mỗi neuron được cấu tạo bởi các thành phần: tế bào hình cây, tế bào thân và sợi trục thần kinh. Tế bào hình cây có nhiệm vụ mang tín hiệu điện sinh học tới tế bào thân, tế bào thân sẽ thực hiện tính tổng và *phân ngưỡng* (thresholds) các tín hiệu đến. Sợi trục thần kinh có nhiệm vụ đưa tín hiệu từ tế bào thân ra ngoài. Điểm tiếp xúc giữa sợi trục thần kinh này với tế bào hình cây của neuron kia gọi là *khớp thần kinh* (synapse). Sự sắp xếp của các neuron và độ mạnh yếu của các khớp thần kinh quyết định khả năng của mạng neuron. Cấu trúc của mạng neuron sinh học luôn luôn thay đổi và phát triển theo quá trình học tập và lao động của con người, các liên kết mới được tạo ra và các liên kết cũ bị loại bỏ, hay tang giảm độ mạnh yếu của các liên kết thông qua các khớp thần kinh.

Mạng neuron nhân tạo là một sự mô phỏng đơn giản mạng neuron sinh học. Cấu trúc của mạng bao gồm các đơn vị tính toán đơn giản (tượng trưng cho các neuron) được liên kết với nhau bằng các cạnh có trọng số (tượng trưng cho các khớp thần kinh). Khả năng xấp xỉ hàm số của mạng neuron nhân tạo phụ thuộc vào hình dạng và độ mạnh yếu của các liên kết (giá trị của các trọng số).

Trong quá trình phát triển của mình mạng neuron nhân tạo đã được ứng dụng thành công trong nhiều bài toán thực tế như nhận dạng chữ viết, nhận dạng tiếng nói, điều khiển tự động, dự báo chuỗi thời gian…

## CẤU TRÚC VỀ MẠNG NEURON NHÂN TẠO

Mạng neuron nhân tạo là một mạng gồm một tập các *đơn vị* (unit) được kết nối với nhau bằng các cạnh có trọng số.

Một đơn vị (Hình 2.1) thực hiện một công việc rất đơn giản: nó nhận tín hiệu vào từ các đơn vị phía trước hay một nguồn bên ngoài và sử dụng chúng để tính tín hiệu ra. Mỗi đơn vị có thể có nhiều tín hiệu đầu vào nhưng chỉ có một tín hiệu đầu ra duy nhất. Đôi khi các đơn vị còn có một giá trị gọi là *độ lệch* (bias) được gộp vào các tính hiệu đầu vào để tính tín hiệu ra. Để đơn giản ký hiệu, độ lệch của một đơn vị được xem như là trọng số nối từ một đơn vị giả có giá trị xuất luôn là 1 đến đơn vị đó.



Hình 2.1: Đơn vị mạng neuron

Trong một mạng neuron có ba kiểu đơn vị:

* Các đơn vị đầu vào, nhận tín hiệu từ bên ngoài.
* Các đơn vị đầu ra, gửi dữ liệu ra bên ngoài.
* Các đơn vị ẩn, tín hiệu vào của nó được truyền từ các đơn vị trước nó và tín hiệu ra được truyền đến các đơn vị sau nó trong mạng.

Khi nhận được các tín hiệu đầu vào, một đơn vị sẽ nhân mỗi tín hiệu với trọng số tương ứng rồi lấy tổng các giá trị vừa nhận được. Kết quả sẽ được đưa vào một hàm số gọi là *hàm kích hoạt* (Activation function) để tính ra tín hiệu đầu ra. Các đơn vị khác nhau có thể có các hàm kích hoạt khác nhau.

Có 4 loại hàm kích hoạt thường dùng:

* Hàm đồng nhất:
* Hàm ngưỡng:
* Hàm sigmoid:
* Hàm sigmoid lưỡng cực

Các đơn vị liên kết với nhau qua các cạnh có trong số tạo thành mạng neuron nhân tạo. Tùy theo số lượng các đơn vị và cách thức liên kết của chúng mà tạo thành các mạng neuron khác nhau có khả năng khác nhau. Có hai loại hình dạng mạng neuron nhân tạo cơ bản là mạng truyền thẳng và mạng hồi quy:

* *Mạng truyền thẳng* (Feed-forward neural network): Một đơn vị ở lớp đứng trước sẽ kết nối với tất cả các đơn vị ở lớp đứng sau. Tín hiệu chỉ được truyền theo một hướng từ lớp đầu vào qua các lớp ẩn (nếu có) và đến lớp đầu ra. Nghĩa là tín hiệu ra của một đơn vị không được phép truyền cho các đơn vị trong cùng lớp hay ở lớp trước. Đây là loại mạng rất phổ biến và được dung nhiều trong việc dự báo dữ liệu chuỗi thời gian. Bài báo cáo này chỉ tập trung vào mô hình mạng này.



Hình 2.2: Mạng neuron truyền thẳng

* *Mạng hồi quy* (Recurrent neural network): Khác với mạng truyền thẳng, mạng hồi quy có chứa các liên kết ngược từ một đơn vị đến các đơn vị ở lớp trước nó.



Hình 2.3: Mạng neuron hồi quy

Chức năng của một mạng nơron được quyết định bởi các nhân tố như: hình dạng mạng (số lớp, số đơn vị trên mỗi lớp, cách mà các lớp được liên kết với nhau) và các trọng số của các liên kết bên trong mạng. Hình dạng của mạng thường là cố định, và các trọng số được quyết định bởi một *thuật toán huấn luyện* (training algorithm). Tiến trình điều chỉnh các trọng số để mạng “nhận biết” được quan hệ giữa đầu vào và đầu ra mong muốn được gọi là *học* (learning) hay *huấn luyện* (training). Rất nhiều thuật toán huấn luyện đã được phát minh để tìm ra tập trọng số tối ưu làm giải pháp cho các bài toán. Các thuật toán đó có thể chia làm hai nhóm chính: *Học có giám sát* (Supervised learning) và *Học không có giám sát* (Unsupervised Learning) [3].

* ***Học có giám sát:*** Mạng được huấn luyện bằng cách cung cấp cho nó các cặp mẫu đầu vào và các *đầu ra mong muốn* (target values). Các cặp này có sẵn trong quá trình thu nhập dữ liệu. Sự khác biệt giữa các đầu ra theo tính toán trên mạng so với các đầu ra mong muốn được thuật toán sử dụng để thích ứng các trọng số trong mạng. Điều này thường được đưa ra như một bài toán xấp xỉ hàm số - cho dữ liệu huấn luyện bao gồm các cặp mẫu đầu vào x, và một đích tương ứng t, mục đích là tìm ra hàm *f(x)* thoả mãn tất cả các mẫu học đầu vào [3]. Đây là mô hình học rất phổ biến trong việc áp dụng mạng neuron vào bài toán dự báo dữ liệu chuỗi thời gian. Hai giả thuật được đề cập trong bài báo cáo này, lan truyền ngược và RPROP là hai giải thuật học thuộc mô hình này.



Hình 2.4: Mô hình học có giám sát

* ***Học không có giám sát:*** với cách học không có giám sát, không có phản hồi từ môi trường để chỉ ra rằng đầu ra của mạng là đúng. Mạng sẽ phải khám phá các đặc trưng, các điều chỉnh, các mối tương quan, hay các lớp trong dữ liệu vào một cách tự động. Trong thực tế, đối với phần lớn các biến thể của học không có giám sát, các đích trùng với đầu vào. Nói một cách khác, học không có giám sát luôn thực hiện một công việc tương tự như một mạng tự liên hợp, cô đọng thông tin từ dữ liệu vào [3]

## NGUYÊN TẮC HOẠT ĐỘNG VÀ CÁC GIẢI THUẬT HUẤN LUYỆN CỦA MẠNG NEURON NHÂN TẠO

### Perceptron

Để hiểu rõ về nguyên tắc hoạt động và cách huấn luyện các mạng neuron nhân tạo trước hết ta khảo sát một mô hình mạng neuron đơn giản được xây dựng trên một đơn vị gọi là perceptron. Một perceptron nhận một vector các giá trị thực, tính tổ hợp tuyến tính của chúng và xuất ra 1 nếu kết quả lớn hơn một ngưỡng nào đó và xuất ra -1 trong các trường hợp còn lại.



Hình 2.5: Đơn vị mạng Neuron

Một cách hình thức, khi nhận một vector đầu vào *n* chiều gồm các giá trị *x1* đến *xn*, giá trị xuất sẽ được tính như sau:

Ở đây các số thực là các trọng số biểu diễn mức độ đóng góp của giá trị nhập vào giá trị xuất của perceptron. Đại lượng ()là ngưỡng mà tổ hợp tuyến các giá trị nhập phải vượt qua để kết quả xuất là 1. Đặt , ta viết lại phương trình trên dưới dạng vector như sau

Ở đây và là các vector có *n* + 1 chiều. Hàm được định nghĩa như sau:

Nếu xem các vector nhập là các điểm trên không gian chiều ( luôn là 1) thì perceptron biểu diễn một *mặt quyết định* (decision surface) xem một điểm có nằm trên một *siêu phẳng* (hyperplane) có phương trình là hay không. Perceptron sẽ xuất ra giá trị 1 cho các điểm nằm trên siêu phẳng này và xuất ra -1 cho các điểm còn lại.

Trong thực tế, ta thường có sẳn một bộ dữ liệu mẫu gồm một tập các điểm được gán nhãn dương và âm. Bài toán huấn luyện perceptron là bài toán xác định vector sau cho siêu phẳng phân chia các điểm trong tập mẫu một cách chính xác theo các nhãn của nó. Thực tế có một số bộ dữ liệu mà không thể tìm thấy bất kỳ siêu phẳng nào có thể phân chia đúng các điểm của nó, các bộ dữ liệu đó được gọi là tập dữ liệu không *khả phân tuyến tính* (linearly separable). Ngược lại nếu một bộ dữ liệu có thể được phân chia đúng bởi một siêu phẳng nào đó thì gọi là khả phân tuyến tính.



Hình 2.6: Mặt quyết định biểu diễn bởi perceptron hai đầu nhập

Hình 2.6 (a) là một tập mẫu khả phân tuyến tính có thể được phân ra bởi một mặt quyết định của perceptron. Hình 2.6 (b) là một tập mẫu không khả phân tuyến tính.

Quá trình huấn luyện một perceptron là một quá trình tìm kiếm một vector trên một không gian thực chiều sau cho nó có khả năng phân xuất ra các giá trị +1, -1 một cách đúng đắn cho một tập dữ liệu nào đó. Có hai giải thuật huấn luyện cơ bản là *luật huấn luyện perceptron* (perceptron training rule) và *luật delta* (delta rule).

1. ***Luật huấn luyện perceptron***: Để tìm một vector thích hợp, trước hết ta áp dụng một perceptron với trọng số ngẫu nhiên qua từng mẫu của tập dữ liệu huấn luyện và hiệu chỉnh các trọng số này khi có sự phân loại sai tập mẫu. Quá trình này được lặp đi lặp lại cho đến khi perceptron đã phân loại đúng tất cả các mẫu của tập huấn luyện. Các trọng số được cập nhập theo luật

Ở đây *o* là giá trị xuất của perceptron, *t* là giá trị đích của mẫu huấn luyện hiện thời, *xi* là giá trị nhập thứ *i,*  là *hệ số học* (learning rate) có vai trò điều tiết mức độ thay đổi của trọng số trong các bước cập nhập. Nó thông thường được gán một giá trị dương nhỏ (ví dụ 0.1) và được điều chỉnh giảm khi số lần cập nhập trọng số tăng lên [2].Giải thuật học này được chứng minh hội tụ sau một số hữu hạn lần cập nhập các trọng số đối với các tập dữ liệu mẫu khả phân tuyến tính và một hệ số học đủ nhỏ nhưng đối với các tập dữ liệu không khả phân tuyến tính thì sự hội tụ là không chắc [2].

1. ***Luật delta:*** Luật perceptron không đảm bảo tính hội tụ đối với các tập mẫu khả phân tuyến tính do đó người ta thiết kế giải thuật luật delta để vượt qua khó khăn này. Luật delta sẽ hội tụ về một xấp xỉ tốt nhất cho các tập không khả phân tuyến tính. Ý tưởng chính của luật delta là áp dụng phương pháp *giảm độ dốc* (gradient descent) để tìm kiếm vector trọng số đáp ứng tốt nhất tập huấn luyện. Xét một perceptron thực hiện việc lấy tổ hợp tuyến tính các giá trị nhập nhưng không phân ngưỡng kết quả. Perceptron này gọi là *perceptron không phân ngưỡng* (unthresholded perceptron) hay còn gọi là *đơn vị tuyến tính* (linear unit). Giá trị xuất của perceptron được tính như sau

Để áp dụng luật delta ta cần định nghĩa một hàm đánh giá, hay còn gọi là *hàm lỗi* (training error function). Có nhiều hàm lỗi được sử dụng nhưng thường dùng nhất là hàm sau

Ở đây *D* là tập dữ liệu huấn luyện, *d* là một mẫu trong tập *D*, *td* là giá trị đích của mẫu *d*, *od* là giá trị xuất của perceptron. Mục đích của luật delta là tìm vector  sau cho  đạt giá trị nhỏ nhất. Hình 2.7 là một biểu diễn hàm lỗi của một đơn vị tuyến tính. Trục thẳng đứng của đồ thị là giá trị hàm lỗi, hai trục ở mặt phẳng ngang là giá trị của các trọng số.

Phương pháp giảm độ dốc bắt đầu tìm với một vector trọng số ngẫu nhiên và duyệt qua các mẫu của tập huấn luyện, mỗi lần duyệt qua các trọng số sẽ được cập nhập theo hướng làm giảm giá trị hàm lỗi. Quá trình này được lặp đi lặp lại cho đến khi đạt được giá trị cực tiểu của hàm lỗi.

Hướng cập nhập các trọng số để làm giảm giá trị hàm lỗi được xác định theo *vector độ dốc* (gradient) của hàm lỗi theo , ký hiệu là

Về mặt toán học vector độ dốc biểu diễn hướng làm tăng giá trị hàm *E* trong không gian trọng số, do đó - sẽ là hướng làm giảm giá trị hàm . Trong hình 2.7 nó được biểu diễn bằng dấu mũi tên. Các trọng số sẽ được cập nhập theo quy luật sau:

Luật huấn luyện này có thể được viết lại cho từng trọng số như sau:



Hình 2.7: Hàm lỗi của một đơn vị tuyến tính

Để thực hiện cập nhập các trọng số, ta thực hiện tính đạo hàm riêng phần của hàm *E* theo từng trọng số:

(2.1)

(2.2)

Thay (2.2) vào (2.1) ta được giá nhập cập nhập trọng số qua từng bước ta được

(2.3)

Ở đây giá trị là giá trị đầu vào thứ *i* của mẫu *d* .

Phương pháp giảm độ dốc có hai hạn chế chính là tốc độ hội tụ đôi khi khá chậm và nếu có nhiều *cực tiểu cục bộ* (local minimum) trên bề mặt của hàm lỗi thì giải thuật dễ rơi vào cực tiểu cục bộ mà không đạt được *cực tiểu toàn cục* (global minimum). Để giải quyết các khó khăn này người ta đã phát triển phương pháp giảm độ dốc thành phương pháp *giảm độ dốc tăng cường* (incremental gradient descent). Khác với phương pháp giảm độ dốc ở trên phương pháp giảm độ dốc tăng cường thực hiện việc tính toán lỗi và cập nhập các trọng số ngay khi duyệt qua một mẫu của tập dữ liệu. Giá trị cập nhập cho các trọng số của phương pháp giảm độ dốc tăng cường là

(2.3)

Ở đây các giá trị *t, o, xi* lần lượt là giá trị đích, giá trị xuất của mạng và giá trị nhập thứ *i* của mẫu huấn luyện hiện hành. Hàm lỗi của phương pháp giảm độ dốc tăng cường không phải là hàm lỗi toàn cục cho toàn bộ dữ liệu huấn luyện như phương pháp giảm độ dốc thường mà là hàm lỗi cho từng mẫu trong tập dữ liệu

Ở đây giá trị *,* lần lượt là giá trị đích và giá trị xuất của mạng cho mẫu *d* trong tập dữ liệu. Với một hệ số học đủ nhỏ, phương pháp giảm độ dốc tăng cường có thể xấp xỉ tốt tùy ý phương pháp giảm độ dốc thông thường [2]. Theo Tom Mitchell [2] phương pháp giảm độ dốc tăng cường khác với phương pháp giảm độ dốc thông thường ở ba điểm sau. Thứ nhất, giải thuật thực hiện việc tính toán lỗi và cập nhập các trọng số cho mỗi mẫu trong tập huấn luyện chứ không đợi duyệt qua hết các mẩu trong tập huấn luyện. Thứ hai, phương pháp giảm độ dốc thông thường cần nhiều tính toán để cập nhập các trọng số vì nó cần phải tính toán hàm lỗi thực sự cho toàn bộ tập dữ liệu huấn luyện và mỗi lần cập nhập các trọng số được cập nhập một bước lớn hơn phương pháp giảm độ dốc tăng cường. Thứ ba phương pháp giảm độ dốc tăng cường có khả năng không bị rơi vào cự tiểu cục bộ vì nó sử dụng thay cho để tìm kiếm.

Sự khác biệt giữa hai giải thuật huấn luyện luật delta và luật huấn luyện perceptron khác nhau ở tính chất hội tụ của chúng. Luật huấn luyện perceptron hội tụ sau một số lần lặp hữu hạn và tìm ra một mặt phẳng phân loại hoàn hảo một tập dữ liệu huấn luyện khả phân tuyến tính trong khi giải thuật luật delta sẽ hội tụ về một điểm cực tiểu của hàm lỗi với một thời gian khá lâu (có thể là vô hạn) nhưng sự hội tụ của nó không bị ảnh hưởng bởi tính khả phân tuyến tính của tập dữ liệu huấn luyện [2].

### Mạng nhiều lớp và giải thuật lan truyền ngược

Mạng neuron đơn giản như perceptron chỉ biểu diễn được các hàm tuyến tính, nhưng trong thực tế ta cần biểu diễn các hàm phi tuyến như trong các bài toán nhận dạng giọng nói hay dự báo chuỗi thời gian. Để làm được điều này ta sử dụng các mạng neuron nhiều lớp, tức là mạng gồm một lớp đầu vào, một lớp đầu ra và một hay nhiều lớp ẩn.

Các mạng neuron nhiều lớp ít khi sử dụng các đơn vị tuyến tính hay đơn vị phân ngưỡng mà chúng sử dụng các đơn vị có các hàm kích hoạt là các hàm khả vi. Một trong những đơn vị hay dùng nhất là *đơn vị sigmoid* (sigmoid unit). Một đơn vị sigmoid sẽ tính tổ hợp tuyến tính các giá trị đầu vào và đưa kết quả này vào hàm sigmoid để tính giá trị đầu ra.



Hình 2.8: Đơn vị sigmoid

Công thức tính giá trị đầu ra của đơn vị sigmoid là

Với

Một thuận lợi khi sử dụng các đơn vị sigmoid là nhờ đạo hàm của hàm sigmoid rất dễ tính (). Điều này làm cho việc áp dụng phương pháp giảm độ dốc được dễ dàng.

Giải thuật lan truyền ngược tìm tập các trọng số thích hợp cho một mạng neuron truyền thẳng nhiều lớp. Nó áp dụng phương pháp giảm độ dốc để tối thiểu hóa bình phương sai số giữa kết quả xuất của mạng với kết quả xuất mong muốn. Ý tưởng chính của giải thuật là giá trị lỗi sẽ được lan truyền ngược từ tầng xuất về tầng nhập để tính 

Hàm lỗi của giải thuật lan truyền ngược được định nghĩa tổng quát như sau

Ở đây *outputs* là tập các đầu ra của mạng neuron, *tkd* và *okd* lần lượt là giá trị đích và giá trị xuất của đầu ra thứ *k* của mẫu huấn luyện *d*

Giải thuật lan truyền ngược áp dụng phương pháp giảm độ dốc để tìm ra điểm tối ưu của hàm lỗi. Với mỗi mẫu trong tập huấn luyện, mạng neuron được áp dụng để tính đầu ra sau đó giá trị độ dốc của hàm lỗi được tính cho từng đơn của mạng. Cuối cùng giải thuật áp dụng phương pháp giảm độ dốc để cập nhập các giá trị trọng số

Để áp dụng phương pháp giảm độ dốc trước hết ta cần thông tin về đạo hàm riêng phần của hàm lỗi cho từng trọng số

Ta tính đạo hàm riêng phần này như sau:

Với

Ở đây:

* là trọng số của cạnh nối đơn vị *j* đến đơn vị *i*
* là kết quả xuất của đơn vị *j*
* là hàm kích hoạt của các đơn vị
* là các đơn vị đứng trước đơn vị *i* trong mạng

Giá trị được tính theo hai trường hợp tùy theo đơn vị *i* là đơn vị ở tầng xuất hay tầng ẩn:

* Nếu đơn vị *i* là đơn vị ở tầng xuất thì:

Đạo hàm bằng 0 đối với mỗi giá trị *k* khác *i* nên

Thay (2.8) và (2.6) vào (2.5) ta được công thức tính đạo hàm riêng phần của hàm lỗi theo trọng số *wij* của đơn vị xuất *i*

* Nếu đơn vị *i* là đơn vị ở tầng ẩn ở tầng ẩn thì việc tính toán phức tạp hơn bởi vì giá trị xuất của *i* không ảnh hưởng trực tiếp lên giá trị xuất của mạng neuron mà ảnh hưởng gián tiếp thông qua các đơn vị ở sau nó.

Thay (2.10) và (2.6) vào (2.5) ta được công thức tính đạo hàm riêng phần của hàm lỗi theo trọng số *wij* của đơn vị ẩn *i*

Ở đây *succ(i)* là các đơn vị ở lớp ngay sau đơn vị *i*. Các công thức này cho phép ta xây dựng một thủ tục tính đạo hàm riêng của hàm lỗi *E* theo các trọng số *wij* như sau: Bắt đầu tính toán từ các đơn vị ở tầng xuất, sau đó sử dụng kết quả vừa tính được vào việc tính toán ở các đơn vị ở tầng trước. Nói các khác thông tin về độ dốc được lan truyền từ tầng xuất đến tầng nhập. Do đó giả thuật này được gọi là giải thuật lan truyền ngược.

Mỗi khi thông tin về đạo hàm riêng phần đã biết, bước tiếp theo trong giải thuật lan truyền ngược là cập nhập các trọng số *wij*.

Ở đây  là hệ số học có vai trò điều tiết mức độ thay đổi của trọng số trong các bước cập nhập.

Cơ bản có hai phương pháp cập nhập các trọng số phân loại theo thời điểm cập nhập: *học theo mẫu* (learning by pattern) và *học theo epoch* (learning by epoch). Một epoch là một lần học duyệt qua tất cả các mẫu trong tập dữ liệu mẫu dùng để học.

Trong phương pháp học theo mẫu đôi khi còn dược gọi là *học trực tuyến* (online learning) áp dụng phương pháp giảm độ dốc tăng cường, cứ mỗi lần một mẫu trong tập dữ liệu được duyệt qua thì các trọng số sẽ được cập nhập. Phương pháp này cố gắng tối thiểu hàm *lỗi tổng thể* (overall error) bằng cách tối ưu hàm lỗi cho từng mẫu trong tập dữ liệu học. Phương pháp này làm việc tốt cho các tập dữ liệu mẫu có kích cỡ lớn và chứa đựng nhiều thông tin dư thừa [5].

Phương pháp học theo epoch thực hiện lấy tổng tất cả thông tin về *độ dốc* (gradient) cho toàn bộ *tập mẫu* (pattern set) sau đó mới cập nhập các trọng số theo phương pháp giảm độ dốc thông thường, nghĩa là nó thực hiện việc cập nhập trọng số sau khi đã duyệt qua hết các mẫu trong tập dữ liệu. Phương pháp này còn có tên gọi khác là *học theo bó* (batch learning).

Mặc dù giải thuật lan truyền ngược tương đối đơn giản nhưng trong thực tế việc lựa chọn một hệ số học phù hợp là không hề đơn giản. Hệ số học quá nhỏ sẽ dẫn đến thời gian hội tụ của giải thuật quá lâu, ngược lại hệ số học quá lớn sẽ dẫn đến hiện tượng *giao động* (oscillation), ngăn không cho giá trị hàm mục tiêu hội tụ về một diểm nhất định. Hơn nữa, mặc dù điểm tối ưu cục bô có thể được chứng minh là luôn có thể đạt được ở một vài trường hợp cụ thể nhưng không có gì đảm bảo giải thuật sẽ tìm được cực toàn cục của hàm lỗi [5]. Một vấn đề khác nữa là kích cỡ của đạo hàm cũng ảnh hướng đến sự cập nhập các trọng số. Nếu đạo hàm riêng phần quá nhỏ thì  nhỏ, nếu đạo hàm riêng phần lớn thì  lớn. Độ lớn của đạo hàm riêng phần thay đổi không thể biết trước được theo hình dạng của hàm lỗi *E* trong mỗi lần lặp. Do đó quá trình học không ổn định.

Để cho quá trình học ổn định người ta thêm vào một *hệ số quán tính* (momentum term)

Hệ số quán tính có tác dụng điều chỉnh mức độ ảnh hưởng của giá trị ở bước lặp trước lên giá trị . Hệ số này có tác dụng giúp cho giải thuật không bị dừng ở tối ưu cực tiểu và các vùng phẳng của bề mặt lỗi. Nó cũng giúp tăng giá trị cập nhập ở những vùng mà độ dốc không đổi, do đó tăng tốc độ hội tụ [2].

Sau đây là mã giả cho giải thuật lan truyền ngược theo phương pháp học trực tuyến có áp dụng hệ số quán tính:

|  |  |
| --- | --- |
| 1. *Khởi tạo tất cả các trọng số bằng các số nhỏ ngẫu nhiên* 2. ***Loop until*** *điều kiện dừng thỏa*    1. ***For each*** *mỗi mẫu trong tập dữ liệu,* ***do***       1. *Nhập mẫu vào mạng và tính toán giá trị đầu ra.*       2. ***For each*** *mỗi giá trị xuất của đơn vị k*   *Tính*   * + 1. ***For each*** *đơn vị ẩn h, từ lớp ẩn cuối cùng đế lớp ẩn đầu tiên*        1. *Tính*     2. ***For each*** *wij trong mạng*         1. *Tính*        2. *Tính*     3. ***End for***   1. ***End for***  1. ***End loop*** |  |

Hình 2.9: Giải thuật lan truyền ngược

Giải thuật lan truyền ngược cần hai thông số nhập vào đó là hệ số học và hệ số quán tính. Đối với mỗi bài toán khác nhau các thông số này cần có các giá trị khác nhau để đạt được sự hiệu quả trong quá trình học. Việc xác định các thông số này một cách đúng đắn không phải là một việc dễ dàng cần nhiều công sức và kinh nghiệm.

### Giải thuật RPROP

Giải thuật lan truyền ngược gặp một vấn đề ở chỗ giá trị cập nhập trọng số () không những phụ thuộc vào dấu của đạo hàm riêng phần mà còn bị ảnh hưởng bởi độ lớn của nó, điều này làm cho quá trình học không được ổn định. Việc thêm vào hệ số quán tính không giải quyết trọn vẹn vấn đề bởi vì ta không biết giá trị tối ưu cho hệ số này là bao nhiêu.

Nhiều giải thuật đã được phát minh để giải quyết các vấn đề trên, chúng có thể được phân ra làm hai loại: *chiến lược toàn cục* (global strategy) và *chiến lược cục bộ* (local strategy). Chiến lược toàn cục sử dụng kiến thức về trạng thái của toàn bộ mạng để hiệu chỉnh các thông số toàn cục, trong khi chiến lược cục bộ dùng các thông tin riêng của từng trọng số một để thích nghi các thông số đặc biệt cho từng trọng số [4].

RPROP là viết tắt của từ ‘resilient propagation’, nghĩa là lan truyền đàn hồi là một phương pháp thích nghi cục bộ. RPROP thực hiện cập nhập các trọng số *wij* dựa vào thông tin về dấu của các đạo hàm riêng phần điều này giúp nó tránh được sự ảnh hưởng của độ lớn của các đạo hàm riêng phần này. Để thực hiện điều này các trọng số sẽ có một giá trị cập nhập riêng chỉ phụ thuộc vào dấu của . Giá trị này được cập nhập trong quá trình học theo quy luật sau:

Ở đây là các hệ số cố định của quá trình học dùng để hiệu chỉnh các giá trị cập nhập cho từng trọng số tùy theo hình dạng của hàm lỗi.

Mỗi lần đạo hàm riêng phần theo trọng số *wij* của hàm lỗi *E* đổi dấu, nghĩa là giá trị cập nhập vừa thực hiện là quá lớn và giải thuật đã nhảy vượt qua điểm tối ưu cục bộ thì giá trị cập nhập sẽ giảm đi theo một thừa số . Ngược lại nếu đạo hàm riêng phần vẫn giữ nguyên dấu thì giá trị cập nhập sẽ được tăng lên để tăng tốc độ hội tụ. Cứ mỗi lần giá trị cập nhập được biết thì các trọng số được điều chỉnh theo luật sau: nếu đạo hàm riêng phần dương thì trọng số được giảm đi một lượng bằng với giá trị cập nhập (), nếu đạo hàm riêng phần âm thì giá trị cập nhập được cộng thêm vào trọng số.

Tuy nhiên có một trường hợp đặc biệt đó là khi đạo hàm riêng phần đổi dấu, nghĩa là bước cập nhập trước quá lớn làm cho điểm tối ưu bị nhảy vượt qua. Giá trị trọng số phải được trả về giá trị trước khi thay đổi, giá trị cập nhập  sẽ được giảm xuống và ở bước kế sau ta sẽ không cập nhập giá trị này. Thực tế ta có thể làm việc này bằng cách gán

Giải thuật RPROP thực hiện việc thích nghi các giá trị cập nhập của các trọng số tùy theo độ dốc của hàm lỗi E, mà thông tin về tổng độ dốc của hàm lỗi cho toàn bộ tập dữ liệu mẫu đáng tin hơn thông tin về độ dốc chỉ cho một mẫu trong tập mẫu nên giải thuật RPROP thực hiện theo mô hình học theo bó (học theo epoch). Các thông tin về đạo hàm riêng phần sẽ được cộng dồn qua từng mẫu trong tập huấn luyện và các trọng số sẽ được cập nhập sau khi đã duyệt qua hết các mẫu.

Giải thuật RPROP ban đầu cũng thực hiện các bước giống như giải thuật lan truyền ngược, các thông tin về đạo hàm riêng phần của hàm lỗi theo các trọng số sẽ được lan truyền ngược từ các lớp sau đến các lớp trước. Khi các thông tin về các đạo hàm riêng phần này có đủ thì giải thuật sẽ thực hiện việc cập nhập các trọng số theo các quy tắc nêu ở trên.

Mã giả cho phần cập nhập trọng số của giải thuật RPROP như sau:

|  |
| --- |
| ***For*** *mọi trọng số và độ lệch {*  ***if*** *()* ***then*** *{*  *= minimum ()*  *= - sign ()\**  *= +*  *}* ***else if*** *()* ***then*** *{*  *= maximum ()*  *= -*    *}*  ***else if*** *()* ***then*** *{*  *= - sign ()\**  *= +*  *}*  *}* |

Hình 2.10: Giải thuật RPROP

Ở đây hàm số *minimum* và *maximum* lần lược là hai hàm trả về giá trị nhỏ nhất và lớn nhất của hai số. Hàm số *sign(x)* trả về 1 nếu x dương, trả về -1 nếu x âm và trả về 0 trong các trường hợp còn lại.

Ban đầu các giá trị cập nhập  sẽ được khởi tạo một giá trị dương ban đầu . Lựa chọn tốt cho  là 0.1 nhưng theo các nghiên cứu thì việc lựa chọn tham số này không ảnh hưởng nhiều đến tốc độ hội tụ của giải thuật [4]. Các thông số về  và để tránh vấn đề tràn số của các biến thực. Giá trị  được thiết lập thường là 1.0e-6, còn giá trị  là 50.0. Thông thường độ hội tụ của giải thuật không bị ảnh hưởng bởi các thông số này nhưng đôi khi thông số  được chọn là một giá trị nhỏ (ví dụ 1.0) để ngăn giải thuật không rơi quá nhanh vào một cực tiểu cục bộ [4]. Hai thông số  và được cố định ở hai giá trị lần lược là 1.2 và 0.5, để việc lựa chọn các tham số cho giải thuật được đơn giản. Trong thực tế, hai thông số cần được lưa chọn cho giải thuật RPROP là  và .

Một trong các thuận tiện của RPROP là trong nhiều bào toán không cần phải lựa chọn các tham số một các cẩn thận cũng đạt được tốc độ hội tụ tối ưu hay gần tối ưu [4].

Trong giải thuật lan truyền ngược, giá trị cập nhập trọng số phụ thuộc vào độ lớn của đạo hàm riêng phần của hàm lỗi theo trọng số, mà giá trị này lại giảm theo khoảng cách của các trọng số đối với lớp xuất. Do đó các trọng số ở xa lớp xuất sẽ ít được hiệu chỉnh hơn và việc huấn luyện các trọng số này sẽ chậm hơn các trọng số gần lớp xuất. Tuy nhiên khi dùng RPROP thì giá trị cập nhập trọng số chỉ phụ thuộc vào dấu nên sự huấn luyện sẽ trải đều trên toàn bộ mạng: những trọng số gần lớp nhập cũng có cơ hội được cập nhập và phát triển ngang với các trọng số gần lớp xuất [4].

### Hiện tượng quá khớp

Trong việc huấn luyện mạng neuron, đôi khi ta gặp phải hiện tượng mạng xấp xỉ rất tốt tập dữ liệu huấn luyện nhưng cho ra kết quả dự đoán thiếu chính xác, giảm khả năng tổng quát của mạng. Đây được gọi là hiện tượng *quá khớp* (overfitting).

Để đảm bảo tính tổng quát hóa của mạng và tránh hiện tượng quá khớp, ta cần chuẩn bị một tập dữ liệu kiểm tra. Tập dữ liệu này được sử dụng trong giai đoạn huấn luyện, sau khi huấn luyện xong một cấu hình mạng, ta cần tiến hành kiểm tra trên tập dữ liệu này để xem mạng có xấp xỉ tốt tập dữ liệu kiểm tra này hay không. Nếu sai số kiểm tra nhỏ thì mô hình mạng vừa được huấn luyện có khả năng tổng quát hóa tốt và có thể được sử dụng để dự báo. Ngược lại, nếu sai số kiểm tra lớn, ta buộc phải thực hiện việc huấn luyện mạng lại.

## Áp dụng mạng Neuron vào dự báo dữ liệu chuỗi thời gian

Việc sử dụng mạng neuron nhân tạo vào việc dự báo dữ liệu chuỗi thời gian dựa chủ yếu vào dữ liệu mà ta thu nhập. Mạng neuron nhân tạo truyền thẳng với ít nhất một lớp ẩn và đủ số đơn vị cho lớp ẩn có thể xấp xỉ bất kỳ *hàm khả đánh giá* (measurable function) tuyến tính hay phi tuyến nào[6].



Hình 3.5: Mô hình học với chuỗi thời gian

Như đã đề cập ở trên, dữ liệu chuỗi thời gian là dữ liệu được thu nhập, lưu trữ và quan sát theo sự tăng dần của thời gian X1, X2, …, Xn.

Mạng neuron học cấu hình mạng từ dữ liệu chuỗi thời gian bằng cách ánh xạ từ một vectơ dữ liệu đầu vào sang dữ liệu đầu ra. Một số lượng dữ liệu liên tiếp của dữ liệu chuỗi thời gian (cửa sổ đầu vào Xt-s, Xt-s+1, …, Xt) được ánh xạ sang khoảng thích hợp (ví dụ [0,1] hoặc [-1,1]) và được sử dụng như dữ liệu đầu vào của tầng nhập. Giá trị s của “cửa sổ đầu vào” tương ứng với số đơn vị ở tầng nhập. Trong giai đoạn truyền tiến, những giá trị đó được truyền qua tầng ẩn rồi đến các đơn vị đầu ra. Khi truyền tới đơn vị đầu ra, giá trị lỗi được tính toán dựa vào sự khác biệt giữa giá trị đầu ra với giá trị của dữ liệu chuỗi thời gian tại thời điểm t+1. Sau đó, giá trị lỗi này được truyền ngược lại tới các kết nối giữa tầng ẩn và tầng đầu ra, kết nối giữa tầng đầu vào và tầng ẩn để cập nhập lại trọng số của các kết nối này.

Các cửa sổ đầu vào có thể được chọn một cách ngẫu nhiên hoặc liên tiếp nhau từ dữ liệu chuỗi thời gian. Chọn cửa sổ đầu vào một cách ngẫu nhiên sẽ phức tạp hơn, tuy nhiên sẽ đảm bảo cấu hình mạng tốt hơn và tránh được lỗi tối ưu cục bộ [6].

## Các bước xây dựng một mô hình mạng neuron để dự báo dữ liệu chuỗi thời gian

Theo Kaastra và các cộng sự [7], quá trình xây dựng một mô hình mạng neuron cho bài toán dự báo thường gồm 8 bước:

* Lựa chọn các biến
* Thu thập dữ liệu
* Tiền xử lý dữ liệu
* Phân chia tập dữ liệu
* Xây dựng cấu trúc mạng
* Xác định tiêu chuẩn đánh giá
* Huấn luyện mạng
* Dự đoán và cải tiến

Quá trình này thường không phải là một quá trình liêm tiếp các bước, một số bước có thể được lặp lại đặc biệt là: lựa chọn các biến và huấn luyện mạng

### Lựa chọn các biến

* Thành công trong việc xây dựng một mạng neuron phụ thuộc vào việc hiểu rõ ràng vấn đề cần giải quyết. Biết được những biến nào cần được xem xét là điểm mấu chốt.
* Trong bài toán dự báo các dữ liệu thương mại thì các học thuyết kinh tế có thể giúp chọn lựa các biến là các chỉ số kinh tế quan trọng. Đối với một bài toán cụ thể cần thực hiện xem xét các vấn đề lý thuyết mà từ đó sẽ xác định được các nhân tố ảnh hưởng đến bài toán. Tại bước này trong quá trình thiết kế, điều cần quan tâm đó là các dữ liệu thô từ đó có thể phát triển thành các chỉ số quan trọng. Các chỉ số này sẽ tạo ra các đầu vào cho mạng.
* Khi lựa chọn các biến, ta có thể chọn biến kĩ thuật hoặc biến cơ bản. Biến kĩ thuật bao gồm các giá trị cũ, trong quá khứ của biến đó hoặc các chỉ số được tính toán từ các giá trị cũ đó. Biến cơ bản bao gồm dữ liệu của các biến khác mà ảnh hưởng đến biến đang xem xét. Mô hình neuron đơn giản nhất sử dụng các dữ liệu của biến kĩ thuật hoặc *lấy hiệu* (differencing) của nó như dữ liệu đầu vào của mạng. Hiệu của một chuỗi thời gian *{Xt}* cũng là một chuỗi thời gian *{Yt},* với các giá trị *Yt = Xt+1 - Xt*. Việc lấy hiệu có thể loại bỏ tính xu hướng hay tính mùa của một chuỗi thời gian và làm cho việc xấp xỉ nó đơn giản hơn. Một mô khác cũng được áp dụng phổ biến là sử dụng dữ liệu của các biến cơ bản trong quá khứ để dự đoán.
* Tần suất của dữ liệu được ghi nhận phụ thuộc vào mục đích của nhà dự báo. Nếu dùng để dự đoán tình hình giao dịch chứng khoán thì dữ liệu được ghi nhận hằng ngày. Đối với các vấn đề đầu tư dài hạn thì các dữ liệu hàng tuần, hàng tháng được dùng làm đầu vào cho mạng neuron.

### Thu thập dữ liệu

* Ta cần phải xem xét chi phí và khả năng có thể thu thập được dữ liệu của các biến đã chọn ra ở bước trước. Các dữ liệu kỹ thuật có thể thu thập được dễ dàng và chi phí ít tốn kém hơn là các dữ liệu cơ bản. Để đảm bảo tính chính xác của mạng neuron, ta phải đảm dữ liệu có chất lượng cao. Sau khi được thu thập, các dữ liệu phải được kiểm tra để đảm tính hợp lệ, tính nhất quán và tránh các dữ liệu bị thiếu sót.
* Các dữ liệu bị thiếu sót thường xuyên xuất hiện và có thể được xử lý bằng nhiều cách khác nhau. Các dữ liệu bị thiếu sót có thể được bỏ qua hoặc chúng có thể xem như không thay đổi so với dữ liệu trước nó, và được tính toán bằng phương pháp nội suy hoặc trung bình các giá trị lân cận.

### Tiền xử lý dữ liệu

* Tiền xử lý dữ liệu liên quan đến việc phân tích và chuyển đổi giá trị các tham số đầu vào, đầu ra mạng để tối thiểu hóa nhiễu, nhấn mạnh các đặc trưng quan trọng, phát hiện các xu hướng và cân bằng phân bố của dữ liệu. Bởi vì, mạng neuron dùng để học mẫu từ tập dữ liệu, sự biểu diễn dữ liệu có vai trò quyết định trong việc học các mẫu thích hợp. Các dữ liệu dùng cho đầu vào, đầu ra của mạng neuron hiếm khi được đưa trực tiếp vào mạng dưới dạng dữ liệu thô. Chúng thường được chuẩn hóa vào khoảng giữa cận trên và cận dưới của hàm chuyển (thường là giữa đoạn [0;1] hoặc [-1;1]).
* Hai phương pháp chuyển đổi dữ liệu thường dùng nhất là lấy hiệu và lấy logarit tự nhiên của biến số. Lấy hiệu sử dụng sự thay đổi trong giá trị của biến số, nó có thể được sử dụng để loại bỏ khuynh hướng tuyến tính của dữ liệu. Việc lấy logarit tự nhiên của biến số là hữu ích trong trường hợp biến số lấy các giá trị rất khác nhau, sự thay đổi trong giá trị rất lớn. Việc lấy logarit tự nhiên đồng thời cũng có thể chuyển từ mối liên hệ tỷ lệ sang mối liên hệ cộng giữa các biến dự báo.
* Ngoài phường pháp lấy hiệu và lấy logarit tự nhiên của biến số, ta có thể sử dụng tỉ số của biến đầu vào, trung bình di động. Ta có thể kết hợp các phương pháp để hạn chế dư thừa dữ liệu và cung cấp mạng với tính chính xác cao.

### Phân chia tập dữ liệu

* Trong thực tế, khi huấn luyện, người ta thường chia tập dữ liệu thành các tập: huấn luyện, kiểm tra và kiểm định (ngoài các mẫu). Tập huấn luyện thường là tập lớn nhất được sử dụng để huấn luyện cho mạng. Tập kiểm tra thường chứa khoảng 10% đến 30% tập dữ liệu huấn luyện, được sử dụng để kiểm tra mức độ tổng quát hóa của mạng sau khi huấn luyện. Kích thước của tập kiểm định cần được cân bằng giữa việc cần có đủ số mẫu để có thể kiểm tra mạng đã được huấn luyện và việc cần có đủ các mẫu còn lại cho cả pha huấn luyện và kiểm tra. Tập kiểm định nên bao gồm các giá trị liên tục mới nhất.
* Có hai cách thực hiện xác định tập kiểm tra. Một là lấy ngẫu nhiên các mẫu từ tập huấn luyện ban đầu. Lợi điểm của cách này là có thể tránh được nguy hiểm khi mà đoạn dữ liệu được chọn có thể chỉ điển hình cho một tính chất của dữ liệu (đang tăng hoặc đang giảm). Hai là chỉ lấy các dữ liệu ở phần sau của tập huấn luyện, trong trường hợp các dữ liệu gần với hiện tại là quan trọng hơn các dữ liệu quá khứ.
* Tập dữ liệu kiểm tra ngẫu nhiên không nên lặp lại trong tập huấn luyện, bởi vì điều này có thể làm mất khả năng tổng quát hóa của mạng neuron, đặc biệc trong trường hợp kích thước của tập kiểm tra tương đối lớn so với tập huấn luyện ( khoảng 30 %). Phương pháp tất định, như sử dụng mỗi dữ liệu thứ n làm dữ liệu kiểm tra, cũng không nên được sử dụng bởi vì nó chịu ảnh hưởng bởi tính chu kỳ của dữ liệu.
* Một phương pháp chặt chẽ dùng để đánh giá mạng neuron là walk-forward. Phương pháp walk-forward chia tập dữ liệu thành một chuỗi các tập dữ liệu nhỏ hơn huấn luyện-kiểm tra-kiểm định gối chồng lên nhau.



Hình 3.6: Thủ tục sử dụng phương pháp walk-forward chia tập dữ liệu

### Xây dựng cấu trúc mạng

* Phương pháp thực hiện xây dựng cấu trúc mạng neuron bao gồm việc xác định sự liên kết giữa các neuron, đồng thời xác định cấu trúc của mạng bao gồm số lớp ẩn, số neuron trong từng lớp. Ta có thể thực hiện lựa chọn số neuron trong các lớp ẩn bằng cách bắt đầu bằng một số nào đó dựa trên các luật. Sau khi thực hiện huấn luyện, kiểm tra lỗi tổng quát hóa của từng cấu trúc, có thể tăng hoặc giảm số các neuron.
* Việc thiết kế cấu hình một mạng neuron có ý nghĩa quyết định quan trong việc dự đoán dự đoán dữ liệu chuỗi thời gian. Nếu xây dựng mạng có quá nhiều tầng ẩn, hoặc số lượng đơn vị ở mỗi tầng quá nhiều sẽ dẫn đến vấn đề quá khớp. Tức là khi đó, cấu hình mạng neuron giải thích tập dữ liệu huấn luyện rất tốt, nhưng lại không có khả năng tổng quát hóa, vì thế không thể dùng cấu hình này để dự đoán. Tuy nhiên số tầng hoặc số đơn vị trên mỗi tầng quá ít thì mạng neuron không có khả năng giải thích và dự đoán tốt các chuỗi thời gian phức tạp.
* Thực tế đã chứng minh: một mạng neuron với một tầng đầu vào, một tầng ẩn, một tầng đầu ra cùng với sự thay đổi số đơn vị tại mỗi tầng là đủ để xấp xỉ bất kì một hàm liên tục nào [7]. Thông thường các mạng neuron được khởi tạo với một hoặc nhiều nhất là hai lớp ẩn. Nếu kết quả huấn luyện từ mạng trên mà vẫn không thỏa mãn sau khi đã thử với nhiều giá trị khởi tạo ngẫu nhiên của trọng số thì ta nên xem xét hiệu chỉnh lại số đơn vị trên các lớp ẩn hay kiểm tra dữ liệu đầu vào (ví dụ dữ liệu dùng để huấn luyện mạng có phải đã lỗi thời không?) chứ không nên tăng thêm số tầng ẩn. Cả lý thuyết và các kết quả thực nghiệm gần đây đều kết luận rằng các mạng với hơn hai tầng ẩn sẽ không cải thiện được kết quả dự đoán [7].
* Số lượng đơn vị trong mỗi lớp cũng là một vấn đề cần phải xem xét vì nó cũng ảnh hưởng nhiều đến chất lượng của công tác dự báo. Số lượng các đơn vị ở tầng xuất luôn là 1 cho bài toán dự báo chuỗi thời gian. Tuy nhiên việc chọn số đơn vị cho tầng ẩn và tầng nhập là việc không dễ. Số đơn vị ở tầng nhập bằng số giá trị trong cửa sổ nhập, việc lựa chọn này dựa trên giả định của nhà dự báo về giá trị tại thời điểm hiện tại của chuỗi thời gian sẽ bị chi phối chủ yếu bởi giá trị của bao nhiêu thời điểm trước nó. Việc lựa chọn thông số này phụ thuộc vào kinh nghiệm và sự hiểu biết của nhà dự báo vào chuỗi thời gian đang xét. Số lượng đơn vị ở tầng ẩn cũng là một thông số cần phải lựa chọn cẩn thận và cũng không có một thủ tục hình thức nào giúp ta xác định được một cách tối ưu thông số này. Việc lựa chọn sao cho phù hợp phải dựa vào thực nghiệm. Thông thường có hai cách chủ yếu để tìm giá trị tối ưu cho số đơn vị ở lớp ẩn. Cách thứ nhất ta chuẩn bị một nhóm các mạng neuron chỉ khác nhau số đơn vị ở lớp ẩn (số lượng đơn vị có thể tăng dần theo một, hai hoặc ba), sau đó ta thực hiện huấn luyện và kiểm tra các mạng này trên tập dữ liệu đã chuẩn bị. Mạng neuron có sai số nhỏ nhất là là mạng có cấu hình tốt nhất. Phương pháp này khá tốn thời gian nhưng khá hiệu quả. Cách thứ hai là thay đổi số đơn vị trong lớp ẩn ngay trong quá trình huấn luyện. Cách này không cần phải tạo ra nhiều mạng neuron riêng biệt nhưng lại rất phức tạp. Rất ít các hệ thống thương mại cho phép việc thay đổi số đơn vị trong quá trình huấn luyện[7].
* Nhiều mô hình mạng neuron tầng vào-tầng ẩn-tầng ra đã được sử dụng hiệu quả trong bài toán dự báo chuỗi thời gian như: 8-8-1, 6-6-1, 5-5-1 [6].

### Xác định tiêu chuẩn đánh giá

Để đánh giá khả năng xấp xỉ một chuỗi thời gian của mạng neuron người ta thường dùng hàm *tổng bình phương lỗi* (sum of squared errors) sau:



Ở đây *n* là số điểm trong tập dữ liệu dùng để kiểm tra mạng, *tk* và *ok* lần lượt là giá trị mong muốn trong bộ dữ liệu và giá trị xuất của mạng neuron. Mạng neuron có tổng bình phương lỗi càng nhỏ càng tốt.

Ngoài ra người ta còn dùng các hàm khác là hàm *độ lệch tuyệt đối nhỏ nhất* (least absolute deviation), *hiệu phần trăm* (percentage differences).

### Huấn luyện mạng

* Huấn luyên mạng để học các mẫu từ dữ liệu bằng cách lần lượt đưa các mẫu vào cùng với những giá trị mong muốn. Mục tiêu của việc huấn luyện mạng đó là tìm ra tập các trọng số cho ta giá trị nhỏ nhất toàn cục của chỉ số hiệu năng hay hàm lỗi.
* Một vấn đề quan trọng trong quá trình huấn luyên mạng neuron là xác định điều kiện dừng của quá trình huấn luyện. Có ba cách thường dùng để dừng một quá trình huấn luyện. Cách thứ nhất nhấn mạnh vào việc tránh bị rơi vào điểm tối ưu cục bộ, nhà dự báo chỉ dừng quá trình học khi không có một sự cải thiện đáng kể nào của hàm lỗi. Điểm mà mạng neuron không còn cải thiện được nữa gọi là điểm hội tụ. Cách thứ hai là sử dụng một thông số cố định là số lần lặp tối đa, quá trình huấn luyện sẽ dừng nếu số số lần lặp (epoches) vượt quá thông số này. Mạng neuron sẽ được kiểm tra, nếu kết quả không tốt thì quá trình học sẽ được tiếp tục lại. Cách thứ ba là ta sử dụng một tập dữ liệu ngoài dữ liệu huấn luyện gọi là tập *dữ liệu xác thực* (validation set). Trong quá trình huấn luyện, cứ mỗi lần vector trọng số của mạng neuron thay đổi, tập dữ liệu xác thực này sẽ được đưa vào mạng và tính ra sai số. Giải thuật huấn luyện sẽ dừng khi sai số này nhỏ hơn một ngưỡng mà nhà dự đoán mong muốn. Phương pháp này có khả năng tránh được quá khớp.

### Dự đoán và cải tiến

Sau khi đã thực hiện các bước trên, ta có được một mô hình mạng neuron dùng để dự đoán. Các giá trị dự đoán của mạng được lưu lại và so sánh với các giá trị thực tế khi chúng xuất hiện. Sau một thời gian, có thể mô hình mạng không còn đúng nữa thể hiện qua việc kết quả dự đoán ngày càng xa các giá trị thật, ta cần phải tiến hành cải tiến mạng hoặc học lại và xây dựng mạng mới theo các bước trước.

## Kết luận

Chất lượng dự báo của mạng neuron nhân tạo phụ thuộc nhiều vào cấu hình của mạng (số lớp, số đơn vị mỗi lớp) và các tham số của giải thuật huấn luyện

Về mặt lý thuyết mạng neuron nhân tạo có khả năng xấp xỉ bất cứ hàm liên tục nào, điều này đã làm cho mạng neuron trở thành một công cụ mạnh trong công tác dự báo chuỗi thời gian. Tuy nhiên để tìm một cấu hình tối ưu cho một mạng neuron nhân tạo trong công tác dự báo một chuỗi thời gian nào đó là một việc khó khăn. Ta phải tiến hành việc lưa chọn bằng việc xây dựng nhiều cấu hình khác nhau, qua một quá trình lặp các công đoạn huấn luyện và kiểm tra lựa chọn một cấu hình tốt nhất. Trong quá trình áp dụng mô hình mạng để dự báo, khi các giá trị mới được thu nhập sai số nhiều so với kết quả dự báo, ta cần phải tiến hành huấn luyện lại mạng với các dữ liệu mới.

Giải thuật RPROP tốc độ hội tụ và độ chính xác nhìn chung tốt hơn giải thuật lan truyền ngược và cũng ít bị chi phối bởi việc lựa chọn các tham số huấn luyện. Tuy nhiên điều này không luôn đúng cho mọi tập dữ liệu, vì vậy để thực hiện một công tác dự báo chuỗi thời gian bằng mạng neuron nhân tạo ta nên xem xét lựa chọn và thử nghiệm trên nhiều cấu hình và giải thuật khác nhau để tìm ra một mô hình tốt nhất.

Đối với những chuỗi thời gian có xu hướng và tính mùa chương trình dự đoán với độ chính xác chưa cao.

# Mô hình kết hợp giữa ARIMA và ANN

Cả 2 mô hình ARIMA và ANN đều cho kết quả dự đoán với độ tin cậy cao trong môi trường tuyến tính và phi tuyến tính. Tuy nhiên cả 2 mô hình đều không dự báo tốt cho những chuỗi thời gian tổng quát có chứa cả 2 thành phần trên.

Để giải quyết vấn đề trên, người ta xấp xỉ chuỗi thời gian về dạng tuyến tính, sau đó dùng mô hình ARIMA để dự đoán chuỗi thời gian sau khi xấp xỉ. Tuy nhiên kết quả đạt được có độ tin cậy không cao vì việc xấp xỉ 1 chuỗi thời gian tổng quát về chuỗi thuần tuyến tính sẽ làm mất đi tính phi tuyến tính của chuỗi thời gian và việc xấp xỉ rất phức tạp và tốn chi phí cao.

Áp dụng mô hình ANN để dự báo chuỗi thời gian tổng quát thì ta nhận được kết quả khả quan hơn mô hình ARIMA. Tuy nhiên ANN vẫn không dự báo được hoàn toàn thành phần tuyến tính của chuỗi thời gian.

1 mô hình mới được đề xuất là kết hợp 2 mô hình để dự báo 1 chuỗi thời gian tổng quát.

## Nguyên nhân

Mô hình ARIMA sử dụng những đặc tính thống kê để xây dựng lên mô hình. Mặc dù ARIMA khá linh hoạt, nó có thể dùng để biểu diễn cho nhiều loại chuỗi thời gian khác nhau: AR(p), MA(q), ARMA(p,q), ARIMA(p,d,q), SARIMA(p,q,d)(P,Q,D)S, nhưng hạn chế chính của nó là giả định về mối liên hệ tuyến tính trong chuỗi thời gian. Bởi vì giả định về sự tương quan tuyến tính giữa các giá trị trong chuỗi dữ liệu thời gian nên mô hình ARIMA không thể nhận biết được các thành phần không tuyến tính. Do đó dùng mô hình tuyến tính để dự đoán cho các bài toán thực tế có thể dẫn đến kết quả không thỏa đáng.

Mô hình nơron nhân tạo ANNs được xây dựng dựa trên mô phỏng bộ óc con người. Ưu điểm chính của mô hình này là khả năng mô hình hóa các chuỗi dữ liệu mà các giá trị của nó có mối liên hệ phi tuyến. Với ANNs ta không cần xây dựng một mô hình cụ thể, mà mô hình được thích nghi dần qua quá trình huấn luyện. Phương pháp này thích hợp đối với chuỗi dữ liệu mà ta chưa xác định được các lý thuyết để tìm mối liên hệ trong dữ liệu.

Mô hình lai sử dụng cả hai phương pháp:

* Trong thực tế, thường rất khó để xác định mối liên hệ trong một chuỗi thời gian và tuyến tính hay phi tuyến và phương phương pháp nào hữu hiệu hơn trong việc dự đoán. Do đó rất khó để chọn một mô hình thích hợp, nên ta thường thử nhiều phương pháp và chọn cái ra kết quả tốt nhất. Tuy nhiên, mô hình được chọn không nhất nhiết là mô hình tốt nhất cho dự báo tương lai do sự tồn tại các yếu tố tiềm ẩn: sai số lấy mẫu, mô hình chưa chắc chắn, cấu trúc thay đổi.
* Các chuỗi dữ liệu thực hiếm khi thuần tuyến tính hay thuần phi tuyến, mà chúng thường bao gồm cả hai thành phần. Trong trường hợp đó, một mình mô hình ARIMA cũng như ANNs không đủ tin cậy để mô hình và dự đoán bởi vì mô hình ARIMA không thể giải quyết được mối liên hệ phi tuyến, một mình mô hình ANNs cũng không thể giải quyết được cả hai mối liên hệ phi tuyến và tuyến tính tốt được. Do đó kết hợp cả hai mô hình sẽ ra được kết quả tốt.
* Hầu hết các tài liệu dự báo đều cho rằng không một phương pháp đơn lẻ nào là tốt trong mọi trường hợp. Điều đó là vì các bài toán thực tế rất phức tạp và mang nhiều đặc tính khác nhau, không một phương pháp đơn lẽ nào có thể mô phỏng các bài toán tốt như nhau.

## Mô hình kết hợp ARIMA và Nơron :

1. Hiện thực mô hình

Ta giả sử rằng mỗi chuỗi thời gian gồm cả thành phần tuyến tính và thành phần phi tuyến:

trong đó:

L: biểu diễn thành phần tuyến tính của chuỗi thời gian

N: biểu diễn thành phần phi tuyến của chuỗi thời gian

Hai thành phần đó sẽ được ước lượng từ dữ liệu. Đầu tiên, ta dùng mô hình ARIMA để dự đoán thành phần tuyến tính của chuỗi thời gian, lúc đó phần lỗi của mô hình ARIMA sẽ chỉ chứa thành phần phi tuyến tính. Với là lỗi tại thời điểm t của mô hình tuyến tính ARIMA, ta có: trong đó là giá trị dự đoán từ mô hình ARIMA tại thời điểm t.

Sau khi mô hình hóa dữ liệu bằng ARIMA thì lỗi có thể vẫn còn mang tính phi tuyến. Mô hình hóa lỗi này bằng ANNs, ta có thể phát hiện ra được mối liên hệ phi tuyến. Với n giá trị đầu vào, mô hình ANNs cho phần lỗi là:

Trong đó là hàm phi tuyến được xác định bởi mô hình nơron và lỗi ngẫu nhiên.

Sau khi mô hình đã được xây dựng, giá trị dự đoán từ hai mô hình thành phần là:

Trong đó là giá trị dự đoán từ mô hình ARIMA, là giá trị dự đoán từ mô hình ANNs.

Time series data

Artificial Neural Network

Seasonal ARIMA model

Forecast

ARIMA forecast

Tóm lại, mô hình lai gồm hai bước chính để xây dựng. Ở bước đầu tiên, mô hình ARIMA dùng để phân tích thành phần tuyến tính của chuỗi dữ liệu. Ở bước thứ hai, mô hình ANNs dùng để phân tích thành phần lỗi từ mô hình ARIMA. Bởi vì ARIMA không thể nhận biết được cấu trúc phi tuyến của dữ liệu nên lỗi từ mô hình sẽ chứa thành phần phi tuyến. Kết quả từ mô hình mạng nơron được xem là giá trị dự đoán cho lỗi từ mô hình ARIMA. Mỗi thành phần mô hình sẽ giải quyết được những đặc tính khác nhau của dữ liệu, bằng cách kết hợp cả hai phương pháp, ta sẽ nâng cao hiệu suất mô phỏng và dự đoán.

# KẾT QUẢ THỰC NGHIỆM

Trong đề tài này chúng tôi thực hiện chạy các giải thuật trên các bộ dữ liệu khác nhau để so sánh trên các tiêu chí: khả năng tổng quát hóa (thể hiện qua sai số trong quá trình kiểm tra), khả năng phù hợp với các loại dữ liệu khác nhau, khả năng dự đoán.

Các dữ liệu được thực nghiệm là: dữ liệu về (1) Nhu cầu năng lượng ở Italia, (2) Tỉ giá giữa đồng euro và đồng đô la, (3) Chỉ số tiêu dùng xăng dầu của người của người dân thành thị ở Mỹ, (4) Tổng số người sinh theo tháng ở New York. Các bộ dữ liệu trên được lấy từ nguồn internet.

Các kết quả này được các thành viên trong nhóm chạy thực nghiệm với chương trình trên hệ điều hành Window 7/Window 8, cài đặt sẵn .NET framework với cấu hình máy: Ram 2GB, CPU Core 2 Duo 2.0 GHz.

## Dữ liệu thực nghiệm

Để đánh giá kết quả, dữ liệu mà chúng tôi sử dụng trong đề tài này được lấy từ Time-Series Data Library (nguồn: <http://robjhyndman.com/TSDL/>).

Các dữ liệu này trải rộng từ dữ liệu thị trường tài chính cho đến các quá trình tự nhiên và được phân thành bốn loại *Seasonal*, *Trended*, *Seasonal and Trended* and *Nonlinear*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Series** | **Type** |  | **Domain** | **Description** |
| Passengers  Paper | Seasonal  & Trended |  | Tourism  Sales | Monthly international airline passengers  Monthly sales of French paper |
| Deaths  Maxtemp | Seasonal |  | Traffic  Meteorology | Monthly deaths & injuries in UK roads  Maximum temperature in Melbourne |
| Chemical  Prices | Trended |  | Chemical  Economy | Chemical concentration readings  Daily *IBM* common stock closing prices |
| Sunspots  Kobe | Nonlinear |  | Physics  Geology | Annual Wolf’s Sunspot Numbers  Seismograph of the Kobe earthquake |

Các hình biểu diễn đồ thị của các chuỗi dữ liệu thời gian:



Đồ thị chuỗi dữ liệu Passengers



Đồ thị chuỗi dữ liệu Paper



Đồ thị chuỗi dữ liệu Deaths



Đồ thị chuỗi dữ liệu Maxtemp



Đồ thị chuỗ dữ liệu Chemical



Đồ thị chuỗi dữ liệu Prices



Đồ thị chuỗi dữ liệu Sunspots



Đồ thị chuỗi dữ liệu Kobe

## Kết quả thực nghiệm và đánh giá

### Bộ dữ liệu có tính mùa và xu hướng

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Model | MAE | MSE | MAPE |
| ANNs | Neuron(6,6,1), Algorithm=RPROP, epoches=153, Residual=1E-6, default\_update=0.1, max\_update=10 | 24.46 | 974.08 | 8.92 |
| Neuron(6,6,1),Algorithm=Back Propagation, epoches=616, Residual=1E-6, learning\_rate=0.4, momemtum = 0,2 | 26.81 | 1228.73 | 9.74 |
| SARIMA | ARIMA(1, 1, 0)(1, 1, 0)12 | 8.34 | 129.07 | 2.97 |
| Hybrid | ARIMA(1, 1, 0)(1, 1, 0)12 Neuron(6,6,1),Algorithm=RPROP, epoches=88, Residual=1E-6, default\_update=0.1, max\_update=10 | 7.9 | 120.22 | 2.75 |
| ARIMA(1, 1, 0)(1, 1, 0)12 Neuron(6,6,1),Algorithm=Back Propagation, epoches=114, Residual=1E-6, learning\_rate=0.4, momemtum = 0,2 | 8.2 | 132.78 | 2.82 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pape | Model | MAE | MSE | MAPE |
| ANNs | Neuron(6,6,1), Algorithm=RPROP, epoches=749, Residual=1E-6, default\_update=0.1, max\_update=10 | 60.15 | 6951.32 | 10.46 |
| Neuron(6,6,1),Algorithm=Back Propagation, epoches=627, Residual=1E-7, learning\_rate=0.4, momemtum = 0,2 | 65.73 | 7404.44 | 11.52 |
| SARIMA | ARIMA(0, 0, 0)(0, 1, 0)12 | 36.04 | 2302.3 | 5.27 |
| Hybrid | ARIMA(0, 0, 0)(0, 1, 0)12 Neuron(6,6,1),Algorithm=RPROP, epoches=276, Residual=1E-6, default\_update=0.1, max\_update=10 | 30 | 1704.85 | 4.32 |
| ARIMA(0, 0, 0)(0, 1, 0)12 Neuron(6,6,1),Algorithm=Back Propagation, epoches=156, Residual=1E-6, learning\_rate=0.4, momemtum = 0,2 | 39.17 | 2718.07 | 5.97 |

### Bộ dữ liệu có tính mùa

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Deat | Model | MAE | MSE | MAPE |
| ANNs | Neuron(5,5,1), Algorithm=RPROP, epoches=90, Residual=1E-6, default\_update=0.1, max\_update=10 | 143.69 | 34592.54 | 8.57 |
| Neuron(5,5,1),Algorithm=Back Propagation, epoches=1000, Residual=1E-6, learning\_rate=0.4, momemtum = 0,2 | 164.99 | 48948.66 | 9.23 |
| SARIMA | ARIMA(2, 0, 0)(0, 1, 1)12 | 132.75 | 31667.15 | 8.17 |
| Hybrid | ARIMA(2, 0, 0)(0, 1, 1)12 Neuron(5,5,1),Algorithm=RPROP, epoches=251, Residual=1E-6, default\_update=0.1, max\_update=10 | 117 | 25154.94 | 7.23 |
| ARIMA(2, 0, 0)(0, 1, 1)12 Neuron(5,5,1),Algorithm=Back Propagation, epoches=857, Residual=1E-8, learning\_rate=0.4, momemtum = 0,2 | 109.96 | 21805.63 | 6.77 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Maxt | Model | MAE | MSE | MAPE |
| ANNs | Neuron(5,5,1), Algorithm=RPROP, epoches=78, Residual=1E-6, default\_update=0.1, max\_update=10 | 1.21 | 2.35 | 6.16 |
| Neuron(5,5,1),Algorithm=Back Propagation, epoches=23, Residual=1E-6, learning\_rate=0.4, momemtum = 0,2 | 1.26 | 2.48 | 6.5 |
| SARIMA | ARIMA(0, 0, 0)(0, 1, 0)12 | 1.21 | 2.54 | 6.02 |
| Hybrid | ARIMA(0, 0, 0)(0, 1, 0)12 Neuron(5,5,1),Algorithm=RPROP, epoches=71, Residual=1E-6, default\_update=0.1, max\_update=10 | 1.19 | 2.5 | 5.92 |
| ARIMA(0, 0, 0)(0, 1, 0)12 Neuron(5,5,1),Algorithm=Back Propagation, epoches=7, Residual=1E-6, learning\_rate=0.4, momemtum = 0,2 | 1.19 | 2.5 | 5.92 |

### Bộ dữ liệu có tính xu hướng

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Chem | Model | MAE | MSE | MAPE |
| ANNs | Neuron(6,6,1), Algorithm=RPROP, epoches=144, Residual=1E-6, default\_update=0.1, max\_update=10 | 0.23 | 0.09 | 1.34 |
| Neuron(7,7,1),Algorithm=Back Propagation, epoches=98, Residual=1E-6, learning\_rate=0.4, momemtum = 0,2 | 0.22 | 0.09 | 1.32 |
| SARIMA | ARIMA(2, 0, 2)(0, 0, 0)0 | 0.24 | 0.1 | 1.42 |
| Hybrid | ARIMA(2, 0, 2)(0, 0, 0)0 Neuron(6,6,1),Algorithm=RPROP, epoches=130, Residual=1E-6, default\_update=0.1, max\_update=10 | 0.22 | 0.087 | 1.32 |
| ARIMA(2, 0, 2)(0, 0, 0)0 Neuron(6,6,1),Algorithm=Back Propagation, epoches=145, Residual=1E-6, learning\_rate=0.4, momemtum = 0,2 | 0.28 | 0.13 | 1.7 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Price | Model | MAE | MSE | MAPE |
| ANNs | Neuron(6,6,1), Algorithm=RPROP, epoches=91, Residual=1E-6, default\_update=0.1, max\_update=10 | 9.98 | 174.11 | 2.2 |
| Neuron(6,6,1),Algorithm=Back Propagation, epoches=1000, Residual=1E-6, learning\_rate=0.4, momemtum = 0,2 | 15.53 | 356 | 3.08 |
| SARIMA | ARIMA(2, 1, 2)(0, 0, 0)0 | 5.18 | 51.99 | 1.19 |
| Hybrid | ARIMA(2, 1, 2)(0, 0, 0)0 Neuron(6,6,1),Algorithm=RPROP, epoches=35, Residual=1E-6, default\_update=0.1, max\_update=10 | 5.14 | 51.74 | 1.18 |
| ARIMA(2, 0, 2)(0, 0, 0)0 Neuron(6,6,1),Algorithm=Back Propagation, epoches=136, Residual=1E-6, learning\_rate=0.4, momemtum = 0,2 | 5.18 | 54.85 | 1.19 |

### Bộ dữ liệu có tính phi tuyến

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sun | Model | MAE | MSE | MAPE |
| ANNs | Neuron(8,8,1), Algorithm=RPROP, epoches=200, Residual=1E-6, default\_update=0.1, max\_update=10 | 10.36 | 191.81 | 33.25 |
| Neuron(8,8,1),Algorithm=Back Propagation, epoches=1000, Residual=1E-6, learning\_rate=0.4, momemtum = 0,2 | 12.72 | 243.03 | 40.77 |
| SARIMA | ARIMA(2, 0, 0)(0, 0, 0)0 | 12.38 | 272.48 | 37.72 |
| Hybrid | ARIMA(2, 0, 0)(0, 0, 0)0 Neuron(8,8,1),Algorithm=RPROP, epoches=189, Residual=1E-7, default\_update=0.1, max\_update=10 | 11.43 | 273.63 | 35.69 |
| ARIMA(2, 0, 0)(0, 0, 0)0 Neuron(8,8,1),Algorithm=Back Propagation, epoches=1000, Residual=1E-6, learning\_rate=0.4, momemtum = 0,2 | 15.08 | 457.49 | 39.9 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kobe | Model | MAE | MSE | MAPE |
| ANNs | Neuron(6,6,1), Algorithm=RPROP, epoches=688, Residual=1E-7, default\_update=0.1, max\_update=10 | 787.53 | 1238980 | 30.33 |
| Neuron(6,6,1),Algorithm=Back Propagation, epoches=1000, Residual=1E-7, learning\_rate=0.4, momemtum = 0,2 | 812.89 | 1134819 | 32.58 |
| SARIMA | ARIMA(2, 0, 2)(1, 1, 1)10 | 1090 | 2497855 | 37.37 |
| Hybrid | ARIMA(2, 0, 2)(1, 1, 1)10 Neuron(6,6,1),Algorithm=RPROP, epoches=170, Residual=1E-6, default\_update=0.1, max\_update=10 | 698.36 | 955424 | 28.71 |
| ARIMA(2, 0, 2)(1, 1, 1)10 Neuron(6,6,1),Algorithm=Back Propagation, epoches=1000, Residual=1E-6, learning\_rate=0.4, momemtum = 0,2 | 467.98 | 421860 | 22.25 |

# KẾT LUẬN

## Đánh giá kết quả

### Những công việc làm được

### Những đúc kết về mặt lý luận

### Mặt hạn chế

## Hướng phát triển đề tài

# TÀI LIỆU THAM KHẢO

[1] Bussiness forecasting book

[2] http://www.investopedia.com/articles/trading/07/stationary.asp#axzz2Bu5LkrNQ