HỌC VIỆN KỸ THUẬT QUÂN SỰ

KHOA CÔNG NGHỆ THÔNG TIN

**TRÍ TUỆ NHÂN TẠO**

**NHẬN DẠNG KÝ TỰ SỬ DỤNG MẠNG MLP**

|  |  |
| --- | --- |
| GV Hướng dẫn: | **Ngô Hữu Phúc** |
| Sinh viên: | **Phạm Xuân Lữ** |
| Lớp: | **Tin học 9A** |

HÀ NỘI 06/2014

MỤC LỤC

[**CHƯƠNG I: GIỚI THIỆU VỀ MẠNG NEURAL NHÂN TẠO** 3](#_Toc389456697)

[**I.** **MẠNG NEURAL NHÂN TẠO LÀ GÌ?** 3](#_Toc389456698)

[**II.** **MẠNG NEURAL VỚI TÍNH TOÁN TRUYỀN THỐNG** 3](#_Toc389456699)

[**III.** **NEURON NHÂN TẠO** 4](#_Toc389456700)

[CHƯƠNG II: MẠNG PERCEPTRON NHIỀU LỚP 6](#_Toc389456701)

[**I.** **PERCEPTRON** 6](#_Toc389456702)

[**II.** **MẠNG MULTILAYER PERCEPTRONS** 7](#_Toc389456703)

[CHƯƠNG II: HUẤN LUYỆN MẠNG VÀ THUẬT TOÁN LAN TRUYỀN NGƯỢC 10](#_Toc389456704)

[**I.** **HUẤN LUYỆN MẠNG NEURAL** 10](#_Toc389456705)

[**1.** **Các phương pháp học** 10](#_Toc389456706)

[**2.** **Phương pháp học có giám sát** 11](#_Toc389456707)

[**II.** **THUẬT TOÁN LAN TRUYỀN NGƯỢC** 11](#_Toc389456708)

[**III.** **VARIATIONS OF BACKPROPAGATION** 13](#_Toc389456709)

[**1.** **Momentum** 13](#_Toc389456710)

[**2.** **Variable Learning Rate** 13](#_Toc389456711)

[**IV.** **CÁC VẤN ĐỀ TRONG MẠNG MLP** 14](#_Toc389456712)

[**1.** **Quá khớp (Overfitting)** 14](#_Toc389456713)

[CHƯƠNG III: NHẬN DẠNG KÝ TỰ BẰNG MLP 15](#_Toc389456714)

[**I.** **GIỚI THIỆU BÀI TOÁN NHẬN DẠNG KÝ TỰ** 15](#_Toc389456715)

[**1.** **Phát biểu bài toán** 15](#_Toc389456716)

[**2.** **Mô hình mạng MLP cho bài toán nhận dạng ký tự** 15](#_Toc389456717)

[**II.** **CÀI ĐẶT CHƯƠNG TRÌNH** 15](#_Toc389456718)

[**III.** **KẾT QUẢ THỰC NGHIỆM** 18](#_Toc389456719)

# **CHƯƠNG I: GIỚI THIỆU VỀ MẠNG NEURAL NHÂN TẠO**

1. **MẠNG NEURAL NHÂN TẠO LÀ GÌ?**

**Định nghĩa:** Mạng nơron nhân tạo, ***A****rtificial* ***N****eural* ***N****etwork* (**ANN)** gọi tắt là mạng nơron, *neural network*, là một mô hình xử lý thông tin phỏng theo cách thức xử lý thông tin của các hệ nơron sinh học. Nó được tạo lên từ một số lượng lớn các phần tử (gọi là *phần tử xử lý* hay *nơron*) kết nối với nhau thông qua các liên kết (gọi là *trọng số liên kết*) làm việc như một thể thống nhất để giải quyết một vấn đề cụ thể nào đó.

Một mạng nơron nhân tạo được cấu hình cho một ứng dụng cụ thể (nhận dạng mẫu, phân loại dữ liệu, ...) thông qua một quá trình *học* từ tập các mẫu huấn luyện. Về bản chất học chính là quá trình hiệu chỉnh trọng số liên kết giữa các nơron.

1. **MẠNG NEURAL VỚI TÍNH TOÁN TRUYỀN THỐNG**

Các mạng nơron có cách tiếp cận khác trong giải quyết vấn đề so với máy tính truyền thống. Các máy tính truyền thống sử dụng cách tiếp cận theo hướng giải thuật, tức là máy tính thực hiện một tập các chỉ lệnh để giải quyết một vấn đề. Vấn đề được giải quyết phải được biết và phát biểu dưới dạng một tập chỉ lệnh không nhập nhằng. Những chỉ lệnh này sau đó phải được chuyển sang một chương trình ngôn ngữ bậc cao và chuyển sang mã máy để máy tính có thể hiểu được.

Trừ khi các bước cụ thể mà máy tính cần tuân theo được chỉ ra rõ ràng, máy tính sẽ không làm được gì cả. Điều đó giới hạn khả năng của các máy tính truyền thống ở phạm vi giải quyết các vấn đề mà chúng ta đã hiểu và biết chính xác cách thực hiện. Các máy tính sẽ trở lên hữu ích hơn nếu chúng có thể thực hiện được những việc mà bản thân con người không biết chính xác là phải làm như thế nào.

Các mạng nơron xử lý thông tin theo cách thức giống như bộ não con người. Mạng được tạo nên từ một số lượng lớn các phần tử xử lý được kết nối với nhau làm việc song song để giải quyết một vấn đề cụ thể. Các mạng nơron học theo mô hình, chúng không thể được lập trình để thực hiện một nhiệm vụ cụ thể. Các mẫu phải được chọn lựa cẩn thận nếu không sẽ rất mất thời gian, thậm chí mạng sẽ hoạt động không đúng. Điều hạn chế này là bởi vì mạng tự tìm ra cách giải quyết vấn đề, thao tác của nó không thể dự đoán được.

Các mạng nơron và các máy tính truyền thống không cạnh tranh nhau mà bổ sung cho nhau. Có những nhiệm vụ thích hợp hơn với máy tính truyền thống, ngược lại có những nhiệm vụ lại thích hợp hơn với các mạng nơron. Thậm chí rất nhiều nhiệm vụ đòi hỏi các hệ thống sử dụng tổ hợp cả hai cách tiếp cận để thực hiện được hiệu quả cao nhất. (thông thường một máy tính truyền thống được sử dụng để giám sát mạng nơron).

1. **NEURON NHÂN TẠO**

Một nơron là một đơn vị xử lý thông tin và là thành phần cơ bản của một mạng nơron. Cấu trúc của một nơron được mô tả trên hình dưới.

****

**Nơron nhân tạo**

Các thành phần cơ bản của một nơron nhân tạo bao gồm:

* **Vectơ đầu vào**: Là các tín hiệu vào (*input signals*) của nơron, các tín hiệu này thường được đưa vào dưới dạng một vector N chiều.
* **Ma trận trọng số liên kết**: Mỗi liên kết được thể hiện bởi một trọng số (gọi là trọng số liên kết – *Synaptic weight*). Trọng số liên kết giữa tín hiệu vào thứ j với nơron k thường được kí hiệu là wkj. Thông thường, các trọng số này được khởi tạo một cách ngẫu nhiên ở thời điểm khởi tạo mạng và được cập nhật liên tục trong quá trình học mạng.
* **Bộ tổng** (*Summing function*): Thường dùng để tính tổng của tích các đầu vào với trọng số liên kết của nó.
* **Bias:**  Bias được đưa vào như một thành phần của hàm truyền.
* **Hàm truyền** *(Transfer function)* : Hàm này được dùng để giới hạn phạm vi đầu ra của mỗi nơron. Nó nhận đầu vào là kết quả của hàm tổng và ngưỡng đã cho. Thông thường, phạm vi đầu ra của mỗi nơron được giới hạn trong đoạn [0,1] hoặc [-1, 1]. Các hàm truyền rất đa dạng, có thể là các hàm tuyến tính hoặc phi tuyến. Việc lựa chọn hàm truyền nào là tuỳ thuộc vào từng bài toán và kinh nghiệm của người thiết kế mạng. Một số hàm truyền thường sử dụng trong các mô hình mạng nơron được đưa ra trong bảng 1 .
* **Đầu ra**: Là tín hiệu đầu ra của một nơron, với mỗi nơron sẽ có tối đa là một đầu ra.

Xét về mặt toán học, cấu trúc của một nơron k, được mô tả bằng cặp biểu thức sau:

a =

trong đó: x1, x2, ..., xp: là các tín hiệu vào; (wk1, wk2, ..., wkp) là các trọng số liên kết của nơron thứ k; uk là hàm tổng; bk là một ngưỡng; f là hàm truyền và yk là tín hiệu đầu ra của nơron.

Như vậy tương tự như nơron sinh học, nơron nhân tạo cũng nhận các tín hiệu đầu vào, xử lý ( nhân các tín hiệu này với trọng số liên kết, tính tổng các tích thu được rồi gửi kết quả tới hàm truyền), và cho một tín hiệu đầu ra ( là kết quả của hàm truyền).

**Bảng 1: Một số hàm truyền thông dụng**

|  |  |  |
| --- | --- | --- |
| Hàm truyền | Đồ thị | Định nghĩa |
| Symmetrical Hard Limit (hardlims) |  |  |
| Linear (purelin) |  |  |
| Saturating Linear (satlin) |  |  |
| Log-Sigmoid (logsig) |  |  |

CHƯƠNG II: MẠNG PERCEPTRON NHIỀU LỚP

1. **PERCEPTRON**

Perceptron là mạng nơron đơn giản nhất, nó chỉ gồm một nơron, nhận đầu vào là vector có các thành phần là các số thực và đầu ra là một trong hai giá trị +1 hoặc -1.

****

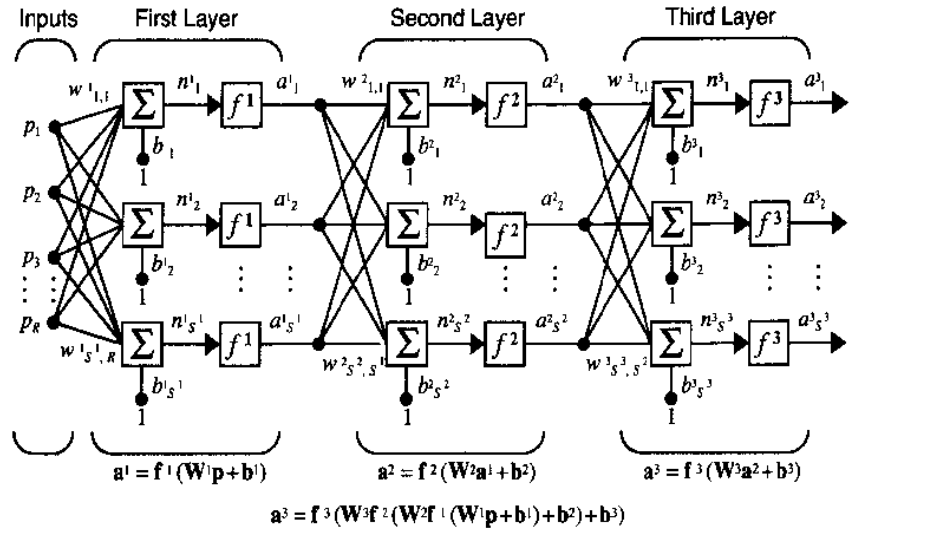
**Mô hình Perceptron**

Đầu ra của mạng được xác định như sau: mạng lấy tổng có trọng số các thành phần của vector đầu vào, kết quả này cùng ngưỡng b được đưa vào hàm truyền (Perceptron dùng hàm Hard-limit làm hàm truyền) và kết quả của hàm truyền sẽ là đầu ra của mạng.

Perceptron cho phép phân loại chính xác trong trường hợp dữ liệu có thể phân chia tuyến tính (các mẫu nằm trên hai mặt đối diện của một siêu phẳng). Nó cũng phân loại đúng đầu ra các hàm AND, OR và các hàm có dạng đúng khi n trong m đầu vào của nó đúng (n ≤ m). Nó không thể phân loại được đầu ra của hàm XOR.

1. **MẠNG MULTILAYER PERCEPTRONS**

Mô hình mạng neural MLP là mạng neuron được sử dụng rộng rãi nhất. Một mạng MLP được cấu thành từ các tầng mà mỗi tầng bao gồm nhiều neuron.



**Kiến trúc mạng MLP**

Kiến trúc mạng MLP như sau:

* Đầu vào là vecto p(p1,p2,…,pr) bao gồm r thành phần mô tả các feature của mẫu được đưa vào mạng trong quá trình huấn luyện mạng cũng như nhận dạng mẫu.
* Đầu ra là vecto a(a1,a2,…,as) đại diện cho lớp mẫu đã được phân loại
* Mỗi nơron thuộc tầng sau liên kết với tất cả các nơron thuộc tầng liền trước nó.
* Đầu ra của nơron tầng trước là đầu vào của nơron thuộc tầng liền sau nó.

Hoạt động của mạng MLP như sau:

* Tại tầng đầu vào các nơron nhận tín hiệu vào xử lý (tính tổng trọng số, gửi tới hàm truyền) rồi cho ra kết quả (là kết quả của hàm truyền); kết quả này sẽ được truyền tới các nơron thuộc tầng ẩn thứ nhất; các nơron tại đây tiếp nhận như là tín hiệu đầu vào, xử lý và gửi kết quả đến tầng ẩn thứ 2;…; quá trình tiếp tục cho đến khi các nơron thuộc tầng ra cho kết quả.

Khả năng của mạng MLP:

* Bất kì một hàm Boolean nào cũng có thể biểu diễn được bởi một mạng MLP 2 tầng trong đó các nơron sử dụng hàm truyền sigmoid.
* Tất cả các hàm liên tục đều có thể xấp xỉ bởi một mạng MLP 2 tầng sử dụng hàm truyền sigmoid cho các nơron tầng ẩn và hàm truyền tuyến tính cho các nơron tầng ra với sai số nhỏ tùy ý.
* Mọi hàm bất kỳ đều có thể xấp xỉ bởi một mạng MLP 3 tầng sử dụng hàm truyền sigmoid cho các nơron tầng ẩn và hàm truyền tuyến tính cho các nơron tầng ra.
* Mạng MLP có khả năng nhận dạng hoặc phân cụm các mẫu rất tốt

# CHƯƠNG II: HUẤN LUYỆN MẠNG VÀ THUẬT TOÁN LAN TRUYỀN NGƯỢC

1. **HUẤN LUYỆN MẠNG NEURAL**
2. **Các phương pháp học**

**Khái niệm**: Học là quá trình thay đổi hành vi của các vật theo một cách nào đó làm cho chúng có thể thực hiện tốt hơn trong tương lai.

Một mạng nơron được huyấn luyện sao cho với một tập các vector đầu vào X, mạng có khả năng tạo ra tập các vector đầu ra mong muốn Y của nó. Tập X được sử dụng cho huấn luyện mạng được gọi là tập huấn luyện (*training set*). Các phần tử x thuộc X được gọi là các mẫu huấn luyện (*training example*). Quá trình huấn luyện bản chất là sự thay đổi các trọng số liên kết của mạng. Trong quá trình này, các trọng số của mạng sẽ hội tụ dần tới các giá trị sao cho với mỗi vector đầu vào x từ tập huấn luyện, mạng sẽ cho ra vector đầu ra y như mong muốn

Có ba phương pháp học phổ biến là học có giám sát (*supervised learning*), học không giám sát (*unsupervised learning*) và học tăng cường (*Reinforcement learning*):

* **Học có giám sát**: Là quá trình học có sự tham gia giám sát của một “thầy giáo”. Cũng giống như việc ta dạy một em nhỏ các chữ cái. Ta đưa ra một chữ “a” và bảo với em đó rằng đây là chữ “a”. Việc này được thực hiện trên tất cả các mẫu chữ cái. Sau đó khi kiểm tra ta sẽ đưa ra một chữ cái bất kì (có thể viết hơi khác đi) và hỏi em đó đây là chữ gì?

Với học có giám sát, tập mẫu huấn luyện được cho dưới dạng D = {(x,t) | (x,t) ∈ [IRN x RK]}, trong đó: x = (x1, x2, ..., xN) là vector đặc trưng N chiều của mẫu huấn luyện và t = (t1, t2, ..., tK) là vector mục tiêu K chiều tương ứng, nhiệm vụ của thuật toán là phải thiết lập được một cách tính toán trên mạng như thế nào đó để sao cho với mỗi vector đặc trưng đầu vào thì sai số giữa giá trị đầu ra thực sự của mạng và giá trị mục tiêu tương ứng là nhỏ nhất. Chẳng hạn mạng có thể học để xấp xỉ một hàm t = f(x) biểu diễn mối quan hệ trên tập các mẫu huấn luyện (x, t).

Như vậy với học có giám sát, số lớp cần phân loại đã được biết trước. Nhiệm vụ của thuật toán là phải xác định được một cách thức phân lớp sao cho với mỗi vector đầu vào sẽ được phân loại chính xác vào lớp của nó.

**Học không giám sát**: Là việc học không cần có bất kỳ một sự giám sát nào.

Trong bài toán học không giám sát, tập dữ liệu huấn luyện được cho dưới dạng: D = {(x1, x2, ..., xN)}, với (x1, x2, ..., xN) là vector đặc trưng của mẫu huấn luyện. Nhiệm vụ của thuật toán là phải phân chia tập dữ liệu D thành các nhóm con, mỗi nhóm chứa các vector đầu vào có đặc trưng giống nhau.

Như vậy với học không giám sát, số lớp phân loại chưa được biết trước, và tùy theo tiêu chuẩn đánh giá *độ tương tự* giữa các mẫu mà ta có thể có các lớp phân loại khác nhau.

**Học tăng cường**: đôi khi còn được gọi là học thưởng-phạt (*reward-penalty learning*), là sự tổ hợp của cả hai mô hình trên. Phương pháp này cụ thể như sau: với vector đầu vào, quan sát vector đầu ra do mạng tính được. Nếu kết quả được xem là “tốt” thì mạng sẽ được thưởng theo nghĩa tăng các trọng số kết nối lên; ngược lại mạng sẽ bị phạt, các trọng số kết nối không thích hợp sẽ được giảm xuống. Do đó học tăng cường là học theo nhà phê bình (*critic*), ngược với học có giám sát là học theo thầy giáo (*teacher*).

1. **Phương pháp học có giám sát**

Học có giám sát có thể được xem như việc xấp xỉ một ánh xạ: X→ Y, trong đó X là tập các vấn đề và Y là tập các lời giải tương ứng cho vấn đề đó. Các mẫu (x, y) với x = (x1, x2, . . ., xn) ∈ X, y = (yl, y2, . . ., ym) ∈ Y được cho trước. Học có giám sát trong các mạng nơron thường được thực hiện theo các bước sau:

* **Bước 1**: Xây dựng cấu trúc thích hợp cho mạng nơron và khởi tạo các trọng số liên kết của mạng.
* **Bước 2**: Đưa một vector x trong tập mẫu huấn luyện X vào mạng
* **Bước 3**: Tính vector đầu ra o của mạng
* **Bước 4**: So sánh vector đầu ra mong muốn y (là kết quả được cho trong tập huấn luyện) với vector đầu ra o do mạng tạo ra; nếu có thể thì đánh giá lỗi.
* **Bước 5**: Hiệu chỉnh các trọng số liên kết theo một cách nào đó sao cho ở lần tiếp theo khi đưa vector x vào mạng, vector đầu ra o sẽ giống với y hơn.
* **Bước 6**: Nếu cần, lặp lại các bước từ 2 đến 5 cho tới khi mạng đạt tới trạng thái hội tụ. Việc đánh giá lỗi có thể thực hiện theo nhiều cách, cách dùng nhiều nhất là sử dụng lỗi tức thời: Err = (**d** – **a**), lỗi trung bình bình phương (MSE: mean-square error): Err = (**d**- **a**)2/2;

Thuật toán tổng quát ở trên cho học có giám sát trong các mạng nơron có nhiều cài đặt khác nhau, sự khác nhau chủ yếu là cách các trọng số liên kết được thay đổi trong suốt thời gian học. Trong đó tiêu biểu nhất là thuật toán lan truyền ngược.

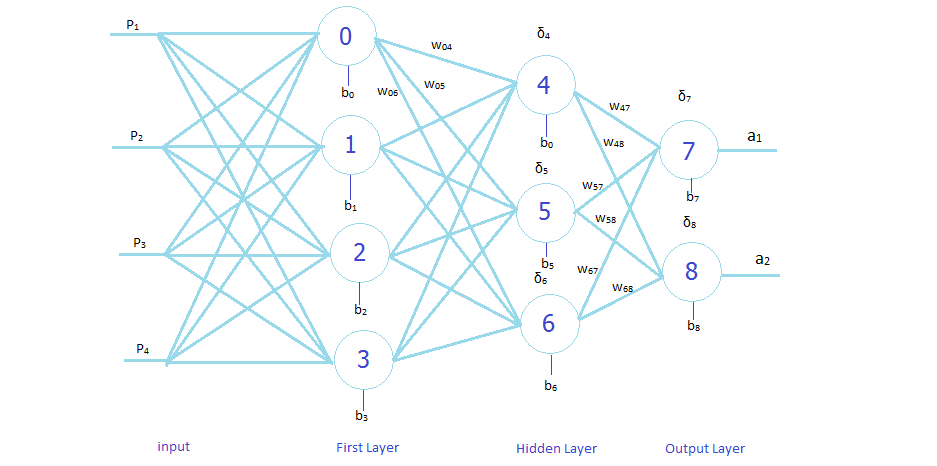
1. **THUẬT TOÁN LAN TRUYỀN NGƯỢC**

Với mạng Perceptron 1 lớp ta có thể tính toán trực tiếp được **E = (d-a)** với d là đầu ra mong muốn của neuron và a là đầu ra thực tế, từ đó có thể cập nhật lại trọng số liên kết theo mong muốn.

Tuy nhiên với mạng Perceptrons nhiều lớp chỉ có lớp output là có đầu ra mong muốn và có thể tính được E theo công thức trên, còn đối với các lớp ẩn thì không có đầu ra mong muốn mà chỉ có đầu ra thực tế. Chính vì vậy, thuật toán lan truyền ngược sẽ tính toán và lan truyền ngược sai số từ lớp output về các lớp ẩn còn lại từ đó có thể thay đổi được trọng số các liên kết.

Về cơ bản, thuật toán lan truyền ngược được chia làm 2 pha tính toán, Pha truyền thằng (ForwardPass) đưa đầu vào và tính toán truyền thẳng qua mạng bình thường không thay đổi trọng số. Pha truyền ngược (BackwardPass) từ đầu ra thực tế ở pha truyền thẳng và tính toán sai số, lan truyền ngược sai số lại các lớp ẩn và thay đổi trọng số tại pha này.

Dưới đây là thuật toán lan truyền ngược với hàm truyền LogSig



***Mô hình mạng MLP***

Thuật toán lan truyền ngược:

* Bước 1: Đưa mẫu input vào mạng và tính toán đầu ra, đầu ra khởi đầu có thể là bất cứ gì vì trọng số ban đầu chúng ta khởi tạo ngẫu nhiên.
* Bước 2: Tính lỗi cho các neuron lớp output.

E = a(1-a)(d-a)

Hay nói cách khác E= y’(d-a) với y’ là đạo hàm của hàm truyền

* Bước 3: Thay đổi trọng số của kết nối.

Với là trọng số mới của liên kết giữa neuron 4 và neuron 7

là trọng số cũ

là lỗi của neuron 7

là đầu ra thực tế của neuron 4

* Bước 4: Tính lỗi cho các neuron lớp ẩn. Giả sử với neuron 4 trên hình ta có thể tính lỗi theo công thức sau.
* Bước 5: Thay đổi trọng số lớp ẩn

1. **VARIATIONS OF BACKPROPAGATION**
2. **Momentum**

Momen quán tính làm tăng tốc độ hội tụ của mạng bằng cách sử dụng cả thông tin giảm trọng số của bước trước nữa, bằng cách đó có thể tăng mức thay đổi nếu sự thay đổi đang là giảm dốc hoặc giảm mức thay đổi nếu sự thay đổi đang là tăng dốc

Với momen quán tính thì trọng số thay đổi theo công thức:

1. **Variable Learning Rate**

Trong quá trình huấn luyện ta có thể thay đổi tốc độ học của mạng để tăng tốc độ hội tụ.

Thuật toán thay đổi tốc độ học như sau:

* Thay đổi trọng số
* Tính tổng bình phương sai số trên toàn bộ tập dữ liệu huấn luyện
* Nếu lỗi tăng lớn hơn một tỉ lệ teta % cho trước thì
  + Hủy bỏ việc thay đổi trọng số vừa thực hiện
  + Giảm tốc độ học đi alpha % (thông thường alpha=5%)
  + Đặt lại Momentum=0
* Nếu lỗi tăng nhỏ hơn tỉ lệ teta cho trước
  + Việc thay đổi trọng số được chấp nhận
  + Tốc độ học tăng lên beta %
  + Nếu Momentum đã đặt về 0 thì reset về giá trị ban đầu
* Nếu lỗi giảm
  + Thay đổi trọng số được chấp nhận
  + Tốc độ học không đổi
  + Momentum không đổi

1. **CÁC VẤN ĐỀ TRONG MẠNG MLP**
2. **Quá khớp (Overfitting)**

Khi mạng huấn luyện đến một mức nào đó mà độ chính xác trên tập dữ liệu huấn luyện là rất cao nhưng với những dữ liệu mạng chưa từng “nhìn thấy” thì độ chính xác rất thấp, lúc này mạng rơi vào tình trạng quá khớp các mẫu đã học (Overfitting).

Cách để chống lại hiện tượng Overfitting là sử dụng phương pháp huấn luyện Early Stopping.

Phương pháp Early Stopping được phát biểu như sau:

Chia bộ dữ liệu học thành 3 phần theo tỉ lệ thích hợp.

* Training set chỉ dùng để đưa vào mạng huấn luyện và thay đổi trọng số.
* Validation set: tính toán sai số trên tập này sau mỗi bước học
* Test set: Kiểm tra độ tổng quát hóa của mạng sau khi huấn luyện

Với early stopping, mạng sẽ dừng trước khi đạt hội tụ hoặc đạt đến ngưỡng lỗi nếu sai số trên tập Validation bắt đầu tăng.

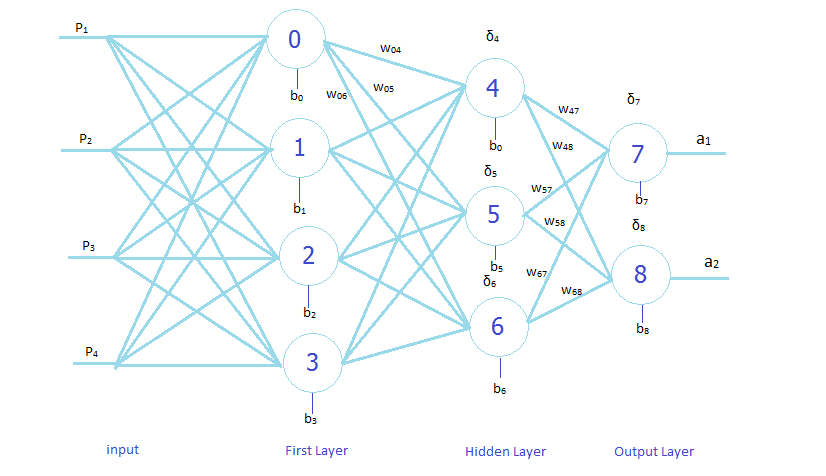
CHƯƠNG III: NHẬN DẠNG KÝ TỰ BẰNG MLP

1. **GIỚI THIỆU BÀI TOÁN NHẬN DẠNG KÝ TỰ**
2. **Phát biểu bài toán**

Thực nghiệm này xây dựng chương trình nhận dạng ký tự với các font khác nhau.

Với đầu vào là ảnh chứa ký tự và đầu ra là ký tự mà máy tính hiểu được.

1. **Mô hình mạng MLP cho bài toán nhận dạng ký tự**



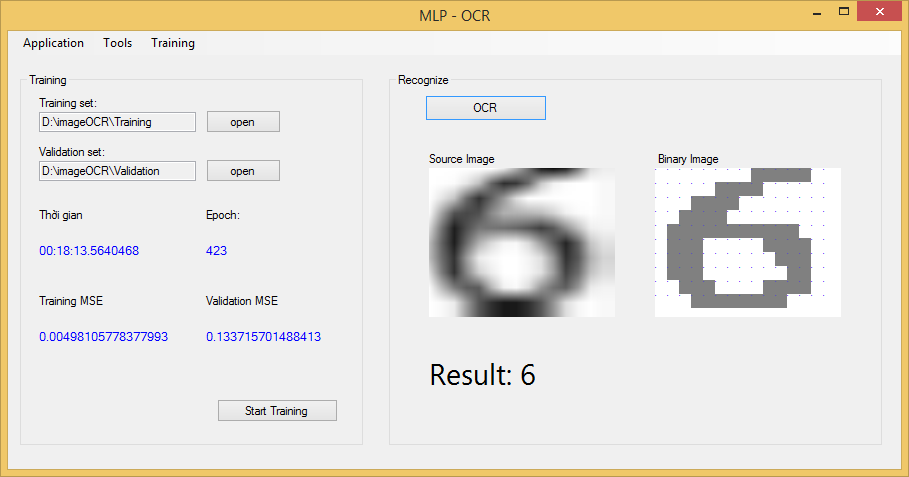
* Mạng MLP bao gồm 3 lớp mạng neuron và một lớp đầu vào input.
* Lớp thứ nhất có 36 neurons
* Lớp ẩn có 28 neurons
* Lớp đầu ra có 16 neurons (là 16 bit đại diện cho ký tự)
* Hàm truyền: Mạng sử dụng hàm truyền LogSig:

có đạo hàm là

* Các giá trị trọng số và bias khởi đầu ngẫu nhiên trong khoảng [-1,1]
* Tốc độ học Eta = 0.01
* Momentum = 0.06
* Ngưỡng lỗi chấp nhận được cho mạng là 0.005

1. **CÀI ĐẶT CHƯƠNG TRÌNH**

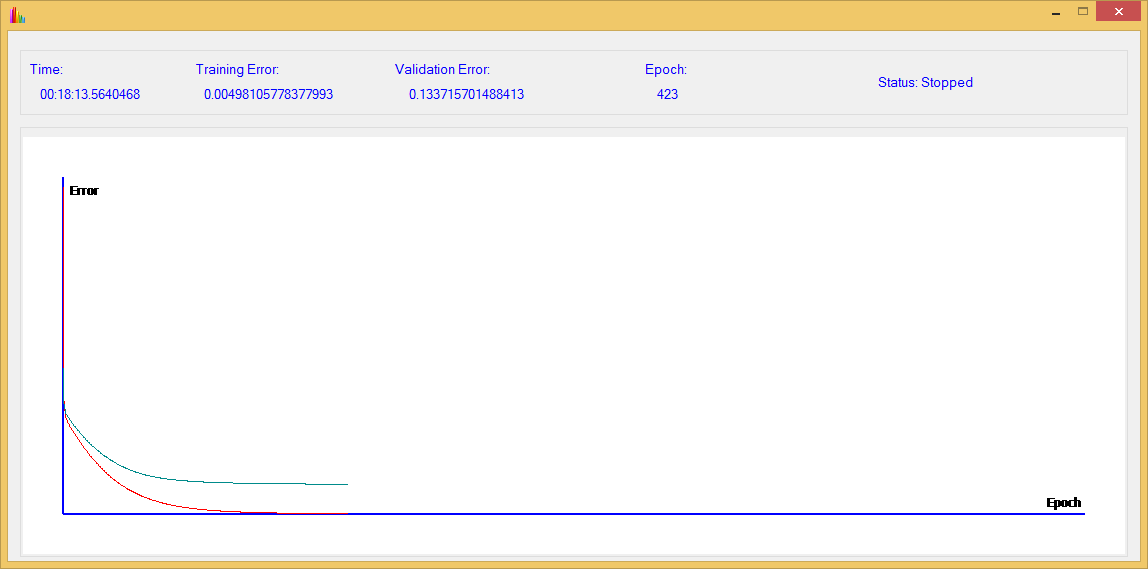
Chương trình được lập trình bằng C# cho phép chúng ta huấn luyện hoặc nhận dạng một ảnh kích thước cố định.



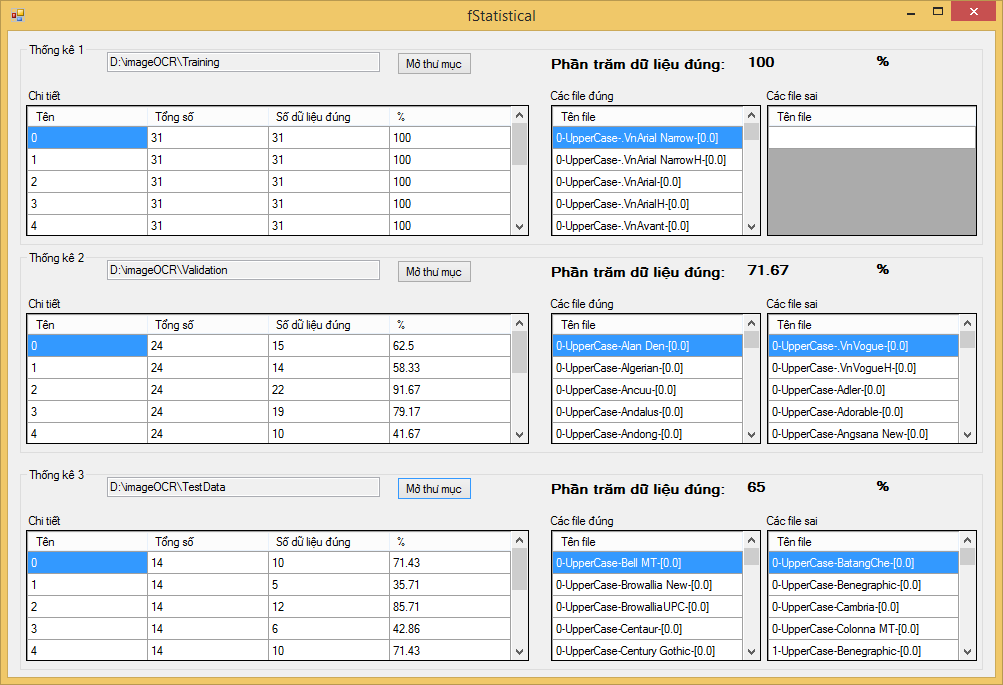
Tập dữ liệu được chia làm 3 phần:

* Training set: chiếm khoảng 50% là tập dữ liệu đưa vào mạng học và tính toán thay đổi trọng số
* Validation set: chiếm khoảng 30% là tập dữ liệu kiểm tra quá trình học có rơi vào trạng thái overfitting hay không
* Test set: chiếm khoảng 20% để kiểm tra độ tổng quát hóa của mạng sau khi huấn luyện

Chương trình có các công cụ giúp trực quan hóa quá trình huấn luyện



*Đồ thị lỗi trên tập Training và tập Validation*



*Công cụ thống kê độ chính xác trên các tập dữ liệu*

Ngoài ra chương trình có thể lưu lại bộ trọng số đã huấn luyện.

1. **KẾT QUẢ THỰC NGHIỆM**