HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Phạm Thị Thanh Hiển

NHẬN DẠNG KÝ HIỆU TOÁN HỌC

Chuyên ngành: Khoa học máy tính

Mã số: 60.48.01.01

TÓM TẮT LUẬN VĂN THẠC SĨ

HÀ NỘI - 2013

Luận văn được hoàn thành tại:

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Người hướng dẫn khoa học: TS. Nguyễn Đức Dũng

Phản biện 1: PGS. TS Lương Chi Mai

Phản biện 2: PGS TS. Đỗ Năng Toàn

Luận văn được bảo vệ trước Hội đồng chấm luận văn thạc sĩ tại Học viện Công nghệ Bưu chính Viễn thông

Vào lúc: 14 giờ 15’ ngày 15 tháng 2 năm 2014

Có thể tìm hiểu luận văn tại:

- Thư viện của Học viện Công nghệ Bưu chính Viễn thông

MỞ ĐẦU

Việc số hóa các tài liệu có ký hiệu toán học để lưu trữ, khai thác các thông tin đó trên hệ thống máy tính là một bài toán đang được đặt ra. Trên thực tế, cách duy nhất để sử dụng thông tin toán học này là gõ lại công thức trên bàn phím để có thể thêm nó vào hệ thống máy tính hay sử dụng làm đầu vào trong các ứng dụng toán học. Và với mong muốn tìm hiểu về lĩnh vực nhận dạng kí tự quang học (Optical character recognition – OCR) và đóng góp thêm vào kho ứng dụng về nhận dạng một hệ thống nhận dạng thiết thực, hữu ích.

Vì vậy trong luận văn sẽ tập trung tìm hiểu các kỹ thuật, các công nghệ cần thiết để xây dựng hệ thống "**Nhận dạng ký hiệu toán học**"

Nội dung luận văn gồm phần mở đầu, 3 chương nội dung, phần kết luận, tài liệu tham khảo.

Chương 1: Các phương pháp nhận dạng

Chương 2: Công nghệ nhận dạng ký tự quang học

Chương 3: Ứng dụng mạng neural nhận dạng ký hiệu toán học

Mục đích nghiên cứu:

- Nghiên cứu lý thuyết nhận dạng, xử lý ảnh.
- Hệ thống OCR
- Công nghệ mã nguồn mở Tesseract OCR
- Tạo ra một ứng dụng nhận dạng ký hiệu toán học rời rạc dựa trên mạng noron và mã nguồn mở Tesseract OCR

CHƯƠNG 1: CÁC PHƯƠNG PHÁP NHẬN DẠNG

1.1. Tổng quan về các phương pháp nhận dạng văn bản

Có nhiều phương pháp nhận dạng mẫu khác nhau được áp dụng rộng rãi trong các hệ thống nhận dạng kí tự. Các phương pháp này có thể được tích hợp trong các hướng tiếp cận sau: Đối sánh mẫu, thống kê, cấu trúc, mạng nơ ron và SVM.

1.1.1. Máy vecto hỗ trợ (SVM)

Phương pháp máy véc tơ tựa (SVM - Support Vector Machines) được đánh giá là phương pháp học máy tiên tiến đang được áp dụng rộng rãi trong các lĩnh khai phá dữ liệu và thị giác máy tính… SVM gốc được thiết kế để giải bài toán phân lớp nhị phân, ý tưởng chính của phương pháp này là tìm một siêu phẳng phân cách sao cho khoảng cách lề giữa hai lớp đạt cực đại. Khoảng cách này được xác định bởi các véc tơ tựa (SV - Support Vector), các SV này được lọc ra từ tập mẫu huấn luyện bằng cách giải một bài toán tối ưu lồi.

1.1.2. Phương pháp tiếp cận cấu trúc

Cách tiếp cận của phương pháp này dựa vào việc mô tả đối tượng nhờ một số khái niệm biểu diễn đối tượng cơ sở trong ngôn ngữ tự nhiên. Để mô tả đối tượng người ta dùng một số dạng nguyên thuỷ như đoạn thẳng, cung,… Mỗi đối tượng được mô tả như một sự kết hợp của các dạng nguyên thuỷ.

1.1.3. Phương pháp ngữ pháp (Grammatical Methods)

Các phương pháp ngữ pháp khởi tạo một số luật sinh để hình thành các ký tự từ một tập các công thức ngữ pháp nguyên thủy. Các luật sinh này có thể kết nối bất kỳ kiểu đặc trưng thống kê và đặc trưng hình thái nào dưới một số cú pháp hoặc các luật ngữ nghĩa. Giống như lý thuyết ngôn ngữ, các luật sinh cho phép mô tả các cấu trúc câu có thể chấp nhận được và trích chọn thông tin theo ngữ cảnh về chữ viết bằng cách sử dụng các kiểu ngữ pháp khác nhau .

1.1.4. Phương pháp đồ thị (Graphical Methods)

Các đơn vị chữ viết được mô tả bởi các cây hoặc các đồ thị. Các dạng nguyên thủy của ký tự (các nét) được lựa chọn bởi một hướng tiếp cận cấu trúc. Đối với mỗi lớp, một đồ thị hoặc cây được thành lập trong giai đoạn huấn luyện để mô tả các nét, các ký tự hoặc các từ. Giai đoạn nhận dạng gán một đồ thị chưa biết vào một trong các lớp bằng cách sử dụng một độ đo để so sánh các đặc điểm giống nhau giữa các đồ thị.

1.1.5. Mô hình Markov ẩn (HMM – Hidden Markov Model)

Mô hình Markov ẩn (HMM) là một trong những mô hình máy học quan trọng nhất trong xử lý ngôn ngữ tự nhiên và nhận dạng. Mô hình này là trường hợp mở rộng của máy hữu hạn trạng thái có hướng, có trọng số. HMM thường được sử dụng để xử lý những sự kiện không quan sát trực tiếp được (sự kiện ẩn). Do vậy, HMM được ứng dụng để giải quyết những bài toán có độ nhiễu lớn, chẳng hạn, dự báo, nhận dạng tiếng nói,…

1.1.6. Đối sánh mẫu

Kỹ thuật nhận dạng chữ đơn giản nhất dựa trên cơ sở đối sánh các nguyên mẫu (prototype) với nhau để nhận dạng ký tự hoặc từ. Nói chung, toán tử đối sánh xác định mức độ giống nhau giữa hai véc tơ (nhóm các điểm, hình dạng, độ cong...) trong một không gian đặc trưng. Các kỹ thuật đối sánh có thể nghiên cứu theo ba hướng sau:

Đối sánh trực tiếp

Các mẫu biến dạng và Đối sánh mềm

Đối sánh giảm nhẹ

Kết luận

Phần này đã giới thiệu một cách tổng quan về lĩnh vực nhận dạng kí tự. Cho đến nay các kết quả nghiên cứu nhận dạng kí tự vẫn còn hạn chế, các ứng dụng chủ yếu chỉ tập trung ở một số lĩnh vực hẹp. Đặc biệt có rất ít kết quả liên quan đến nhận dạng ký hiệu toán học, các kết quả nghiên cứu cũng chỉ tập trung vào các font chữ phổ biến.

Mạng nơ ron được ứng dụng nhiều trong các bài toán phân loại mẫu (điển hình là nhận dạng) bởi ưu điểm nổi trội của nó là dễ cài đặt cùng với khả năng học và tổng quát hoá rất cao. Với thuật toán đơn giản nhưng rất hiệu quả, cùng với thành công của mô hình này trong các ứng dụng thực tiễn, mạng nơ ron hiện đang là một trong các hướng nghiên cứu của lĩnh vực học máy. Trong phần sau sẽ trình bày chi tiết về mạng neural và ứng dụng trong nhận dạng ký hiệu toán học.

1.2. Mạng Neural (neural networks)

1.2.1 Tổng quan về mạng neural nhân tạo

Mạng noron nhân tạo được xây dựng từ những năm 1940 nhằm mô phỏng một số chức năng của bộ não người. Dựa trên quan điểm cho rằng bộ não người là bộ điều khiển. Mạng noron nhân tạo được thiết kế tương tự như neural sinh học sẽ có khả năng giải quyết hàng loạt các bài toán như tính toán tối ưu, điều khiển, công nghệ robot…

Quá trình nghiên cứu và phát triển noron nhân tạo có thể chia thành 4 giai đoạn như sau:

- Giai đoạn 1: 1890 – 1943
- Giai đoạn 2: Vào khoảng gần những năm 1960
- Giai đoạn 3: Vào khoảng đầu thập niên 80
- Giai đoạn 4: Tính từ năm 1987 đến nay

Cho đến nay mạng neural đã tìm và khẳng định được vị trí của mình trong rất nhiều ứng dụng khác nhau.

1.2.1.1 Khái niệm mạng neural. [1]

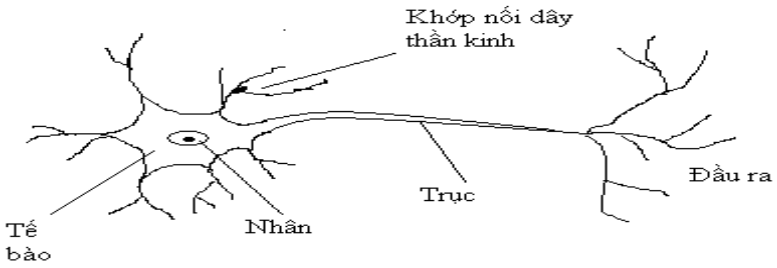
Định nghĩa: Mạng neural nhân tạo, ***A**rtificial **N**eural **N**etwork* **(ANN)** gọi tắt là mạng neural, *neural network*, là một mô hình xử lý thông tin phỏng theo cách thức xử lý thông tin của các hệ neural sinh học. Nó được tạo lên từ một số lượng lớn các phần tử (gọi là *phần tử xử lý* hay *neural*) 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 neural 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 neural.

1.2.1.2 Tìm hiểu về Neural sinh học [1]

Mỗi neural sinh học gồm có 3 thành phần: Thân neural với nhân ở bên trong (soma), một đầu dây thần kinh ra (axon) và một hệ thống phân nhánh hình cây (Dendrite) để nhận các thông tin vào. Hình ảnh đơn giản của một neural thể hiện trong hình 1.5.

Hình 1.5. Mô hình neural sinh học



1.2.1.3. Mạng neural nhân tạo. [1]

a. Giới thiệu

Mạng neural nhân tạo (Artificial Neural Network) là một cấu trúc mạng được hình thành nên bởi một số lượng lớn các neural nhân tạo liên kết với nhau. Mỗi neural có các đặc tính đầu vào, đầu ra và thực hiện một chức năng tính toán cục bộ.

b. Các thành phần của một noron nhân tạo

Phần này mô tả một số thành phần cơ bản của một nơ ron nhân tạo. Những thành phần này là giống nhau cho dù nơ ron đó dùng trong tầng vào, tầng ra hay là ở trong tầng ẩn.

Thành phần 1. Các nhân tố trọng số: Một neural thường nhận nhiều đầu vào cùng lúc. Mỗi đầu vào có trọng số liên quan của riêng nó, trọng số này giúp cho đầu vào có ảnh hưởng cần thiết lên hàm tổng của đơn vị xử lý (thành phần xử lý).

Các trọng số là những hệ số thích nghi bên trong một mạng, chúng xác định cường độ (sức mạnh hay là sức ảnh hưởng) của tín hiệu vào lên neural nhân tạo. Những sức mạnh này có thể được điều chỉnh theo những tập đào tạo đa dạng khác nhau và theo một kiến trúc mạng cụ thể hay là qua các luật học của nó.

Thành phần 2. Hàm tổng: Bước đầu tiên trong hoạt động của một thành phần xử lý là tính toán tổng có trọng số của tất cả các đầu vào. Về mặt toán học, những đầu vào và các trọng số tương ứng là những véc tơ có thể được biểu diễn :

$I = (i_i, i_2, .., i_n)$ và $W = (w_1, w_2, \ldots, w_n)$. Tín hiệu vào tổng là tích vô hướng của mỗi thành phần trong véc tơ I với thành phần tương ứng trong véc tơ W và cộng lại tất cả các tích. $Input_1 = i_1.w_1$, $input_2 = i_2.w_2$… Cuối cùng được cộng lại: $input_1 + input_2 + \ldots + input_n$. Kết quả là một số duy nhất, không phải là một véc tơ.

Thành phần 3. Hàm chuyển đổi: Kết quả của hàm tổng, hầu như luôn là tổng có trọng số, được chuyển đổi thành một đầu ra có ý nghĩa nhờ một quá trình xử lý có thuật toán gọi là hàm chuyển đổi. Trong hàm chuyển đổi tổng có thể được so sánh với một ngưỡng nào đó để quyết định đầu ra của mạng.

Thành phần 4. Hàm ra: Mỗi thành phần xử lý cho phép một tín hiệu đầu ra mà đầu ra này có thể đi tới hàng trăm nơ ron khác.

Thành phần 5. Giá trị truyền ngược và hàm lỗi:

Thành phần 6. Hàm học: Mục đích của hàm học là để thay đổi giá trị của biến trọng số kết nối ở các đầu vào của mỗi thành phần xử lý theo một thuật toán nào đó.

Có hai kiểu học chính là học có giám sát và học không có giám sát.

1.2.2 Các tính chất của mạng neural nhân tạo [8]

1.2.3. Mô hình mạng neural [1]

Một mạng neural là một mô hình tính toán được xác định qua các tham số: kiểu neural (như là các nút nếu ta coi cả mạng neural là một đồ thị), kiến trúc kết nối (sự tổ chức kết nối giữa các neural) và thuật toán học (thuật toán dùng để học cho mạng).

1.2.3.1. Phân loại theo kiểu liên kết neural.

Cách thức kết nối các neural trong mạng xác định kiến trúc (*topology*) của mạng. Hai loại kiến trúc mạng chính:

♦ Tự kết hợp (*autoassociative*): là mạng có các neural đầu vào cũng là các neural đầu ra. Mạng Hopfield là một kiểu mạng tự kết hợp.

♦ Kết hợp khác kiểu (*heteroassociative*): là mạng có tập neural đầu vào và đầu ra riêng biệt. Perceptron, các mạng Perceptron nhiều tầng (MLP: MultiLayer Perceptron), mạng Kohonen, … thuộc loại này.

Ngoài ra tùy thuộc vào mạng có các kết nối ngược (*feedback connections*) từ các neural đầu ra tới các neural đầu vào hay không, người ta chia ra làm 2 loại kiến trúc mạng.

♦ Kiến trúc truyền thẳng (*feedforward architechture*): là kiểu kiến trúc mạng không có các kết nối ngược trở lại từ các neural đầu ra về các neural đầu vào; mạng không lưu lại các giá trị output trước và các trạng thái kích hoạt của neural. Các mạng neural truyền thẳng cho phép tín hiệu di chuyển theo một đường duy nhất; từ đầu vào tới đầu ra, đầu ra của một

tầng bất kì sẽ không ảnh hưởng tới tầng đó. Các mạng kiểu Perceptron là mạng truyền thẳng.

♦ Kiến trúc phản hồi (*Feedback architecture*): là kiểu kiến trúc mạng có các kết nối từ neural đầu ra tới neural đầu vào. Mạng lưu lại các trạng thái trước đó, và trạng thái tiếp theo không chỉ phụ thuộc vào các tín hiệu đầu vào mà còn phụ thuộc vào các trạng thái trước đó của mạng. Mạng Hopfield thuộc loại này.

1.2.3.2. Một số loại mạng neural.

a. Perceptron

Perceptron là mạng neural đơn giản nhất, nó chỉ gồm một neural, 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.

b. Mạng nhiều tầng truyền thẳng (MLP)

Mô hình mạng neural được sử dụng rộng rãi nhất là mô hình mạng nhiều tầng truyền thẳng (MLP: Multi Layer Perceptron). Một mạng MLP tổng quát là mạng có n ($n \geq 2$) tầng (thông thường tầng đầu vào không được tính đến): trong đó gồm một tầng đầu ra (tầng thứ n) và (n-1) tầng ẩn.

Kiến trúc của một mạng MLP tổng quát có thể mô tả như sau:

♦ Đầu vào là các vector (x1, x2, ..., xp) trong không gian p chiều, đầu ra là các vector (y1, y2, ..., yq) trong không gian q chiều.

♦ Mỗi neural thuộc tầng sau liên kết với tất cả các neural thuộc tầng liền trước nó.

♦ Đầu ra của neural tầng trước là đầu vào của neural thuộc tầng liền sau nó.

1.2.4. Huấn luyện mạng neural. [1]

1.2.4.1. 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.

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

♦ **Học có giám sát** (*supervised learning)*

♦ **Học không giám sát** (*unsupervised learning*)

♦ **Học tăng cường** (*Reinforcement learning*)

1.2.4.2. Học có giám sát trong các mạng neural

1.2.4.3. Thuật toán lan truyền ngược

Thuật toán lan truyền ngược được mô tả như sau:

Input:

- Mạng feed-forward với n_i đầu vào, n_h nút ẩn và n_o đầu ra.

- Hệ số học η

- Tập dữ liệu huấn luyện D = {là vector đầu vào, là vector đầu ra mong muốn}.

Output: Các vector trọng số

Thuật toán:

Bước 1: Khởi tạo trọng số bởi các giá trị ngẫu nhiên nhỏ.

Bước 2: Lặp lại cho tới khi thỏa mãn điều kiện kết thúc.

Với mỗi mẫu, thực hiện các bước sau:

2.1 Tính đầu ra o_j cho mỗi nút j:

$\mathbf{o_j = f(d - b_j)}$ với $\mathbf{d = \Sigma x_{ji} w_{ji}}$

2.2 Với mỗi nút k thuộc tầng ra, tính δ_k theo công thức:

$$\mathbf{\delta_k = (t_k - o_k)(1 - o_k)o_k}$$

2.3 Với mỗi nút h thuộc tầng ẩn, tính δ_h theo công thức:

$$\mathbf{\delta_h = o_h(1 - o_h)\ \Sigma \delta_k w_{kh}}$$ với $\mathbf{k \in Downstream(j)}$

2.4 Cập nhật: $\mathbf{w_{ji} = w_{ji} + \Delta w_{ji}}$

Trong đó $\mathbf{\Delta w_{ji} = \eta \delta_k x_{ji}}$

1.2.5. Thu thập dữ liệu cho mạng neural.

a. Kích thước mẫu

Hai yếu tố quan trọng ảnh hưởng đến kích thước mẫu:

♦ Dạng hàm đích: khi hàm đích càng phức tạp thì kích thước mẫu cần tăng.

♦ Nhiễu: khi dữ liệu bị nhiễu (thông tin sai hoặc thiếu thông tin) kích thước mẫu cần tăng.

b. Mẫu con

Trong xây dựng mô hình cần chia tập mẫu thành 2 tập con: một để xây dựng mô hình gọi là tập huấn luyện (*training set*), và một để kiểm nghiệm mô hình gọi là tập kiểm tra (*test set*). Thông thường dùng 2/3 mẫu cho huấn luyện và 1/3 cho kiểm tra. Điều này là để tránh tình trạng quá khớp (*overfitting*).

c. Sự phân tầng mẫu

d. Chọn biến

Khi tạo mẫu cần chọn các biến sử dụng trong mô hình. Có 2 vấn đề cần quan tâm:

♦ Cần tìm hiểu cách biến đổi thông tin sao cho có lợi cho mạng hơn: thông tin trước khi đưa vào mạng cần được biến đổi ở dạng thích hợp nhất, để mạng đạt được hiệu xuất cao nhất.

♦ Chọn trong số các biến đã được biến đổi biến nào sẽ được đưa vào mô hình: không phải bất kì thông tin nào về mẫu cũng có lợi cho mạng..

1.2.6. Xác định các tham số cho mạng cho mạng neural.

a. Chọn hàm truyền

Một số quy tắc khi chọn hàm truyền như sau:

♦ Không dùng hàm truyền tuyến tính ở tầng ẩn.

♦ Chọn các hàm truyền sao cho kiến trúc mạng neural là đối xứng

b. Xác định số neural tầng ẩn

Lựa chọn số lượng noron trong tầng ẩn của một mạng MLP phụ thuộc vào bài toán cụ thể và kinh nghiệm của nhà thiết kế mạng. Nếu tập dữ liệu huấn luyện được chia thành các nhóm với các đặc tính tương tự nhau thì số lượng các nhóm này có thể được sử dụng để chọn số lượng neural ẩn. Trong trường hợp dữ liệu huấn luyện nằm rải rác và không chứa các đặc tính chung, số lượng kết nối có thể gần bằng với số lượng các mẫu huấn luyện để mạng có thể hội tụ.

c. Khởi tạo trọng

Trọng thường được khởi tạo bằng phương pháp thử sai, nó mang tính chất kinh nghiệm và phụ thuộc vào từng bài toán. Một số quy tắc khi khởi tạo trọng:

♦ Khởi tạo trọng sao cho mạng neural thu được là cân bằng

♦ Tạo trọng sao cho giá trị kết xuất của các nút có giá trị trung gian.

1.2.7. Một số vấn đề của mạng neural

1.2.7.1. Vấn đề lãng quên (catastrophic forgetting)

1.2.7.1. Vấn đề quá khớp

1.2.9. Ứng dụng của mạng neural

Ngày nay, mạng nơ ron ngày càng được ứng dụng nhiều trong thực tế ví dụ: các bài toán nhận dạng mẫu, xử lý, lọc dữ liệu, và điều khiển.

1.2.9. Ưu nhược điểm của mạng neural

1.3 Kết luận

Mỗi phương pháp nhận dạng đều có những ưu điểm, nhược điểm riêng. Trong chương này đã giới thiệu một cách tổng quan về các phương pháp nhận dạng và lý thuyết về phương pháp mạng nơron để áp dụng trong giải quyết bài toán nhận dạng ký hiệu toán học.

CHƯƠNG 2: CÔNG NGHỆ NHẬN DẠNG KÍ TỰ QUANG HỌC

2.1 Giới thiệu chung

2.1.1 Sơ lược về nhận dạng ký tự quang học – OCR

Nhận dạng ký tự quang học (tên tiếng anh là Optical Character Recognition – OCR) là kỹ thuật được sử dụng để chuyển đổi ảnh văn bản sang dạng văn bản có thể chỉnh sửa trong máy tính. Đầu vào của quá trình này là tập tin hình ảnh và đầu ra sẽ là các tập tin văn bản chứa nội dung là các chữ viết, ký hiệu có trong hình ảnh đó.

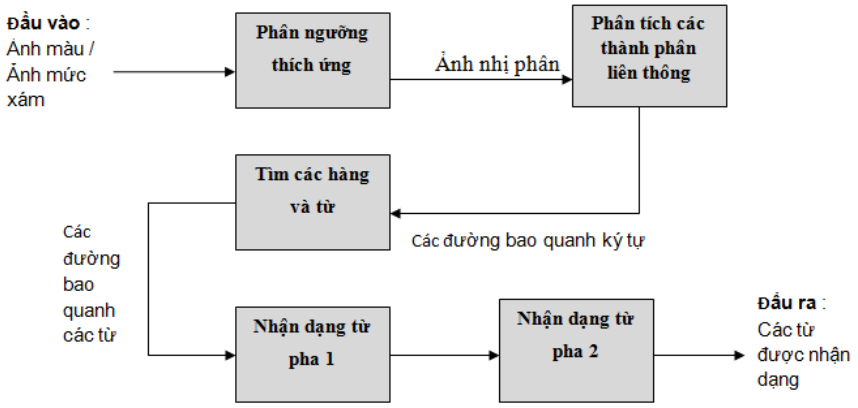
2.1.2 So sánh các thư viện/ công cụ nhận dạng ký tự quang học

- Tesseract OCR, GOCR, FreeOCR, MS Office Document Imaging (MODI), JavaOCR, TopOCR, SimpleOCR

2.2 Giới thiệu về bộ nhận dạng ký tự quang học Tesseract

2.2.1 Lịch sử

2.2.2 Kiến trúc hoạt động



Hình 2.2 Kiến trúc tổng thể của Tesseract

2.2.3 Cài đặt và sử dụng thư viện tesseract

2.2.4 Huấn luyện dữ liệu trên tesseract

1. Sinh hình ảnh huấn luyện
2. Tạo các tập tin *.box
3. Bắt đầu chạy huấn luyện Tesseract
4. Clustering (tập hợp lại)
5. Thêm dữ liệu từ điển (tùy biến)

6. Tổ hợp kết quả lại với nhau:

Trong việc ứng dụng Tesseract enginer cho nhận dạng ký hiệu toán học, thực chất các vấn đề khó khăn nhất nằm ở khâu huấn luyện hơn là các vấn đề về lập trình và tích hợp phần mềm.

2.3 Kết luận

Trong các thư viện nhận dạng ký tự quang học trên thì bộ nhận dạng Tesseract OCR nổi trội nhất nhiều ưu điểm sau:

- Có lịch sử phát triển lâu dài và mang độ chính xác cao ngay từ khi mới ra mắt.
- Khả năng mở rộng và tùy biến cao đồng thời được Google tài trợ và đông đảo các nhà phát triển tham gia đóng góp cho Tesseract.
- Phiên bản được cập nhật thường xuyên, hỗ trợ ngày càng nhiều ngôn ngữ, có khả năng huấn luyện trên các ngôn ngữ mới và nhiều loại font chữ khác nhau.

CHƯƠNG 3. ỨNG DỤNG MẠNG NEURAL NHẬN DẠNG KÝ HIỆU TOÁN HỌC

3.1. Xác định bài toán

Input : Vì đề tài chỉ tập trung vào nghiên cứu và cài đặt quá trình nhận dạng ký hiệu toán học, nên các dữ liệu đầu vào phải qua xử lý và đạt chuẩn yêu cầu có độ phân giải 300dpi trước khi đưa vào nhận dạng.

Output : là một file text chứa các ký hiệu toán học tương ứng mà chương trình mong muốn huấn luyện mạng học thuộc.

Do hạn chế về thời gian và các ký hiệu phân tán ở nhiều tài liệu khác nhau nên chương trình thử nghiệm huấn luyện và nhận dạng với 80 chữ số từ 0 – 9 và các ký hiệu toán học cơ bản.

3.2. Quá trình thực hiện

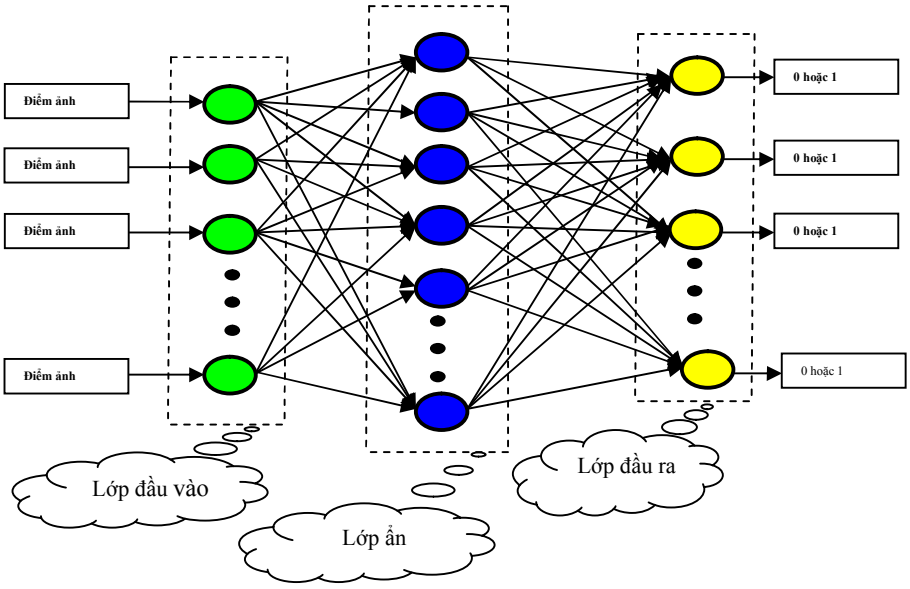
3.2.1. Xây dựng mạng neural

Mạng noron được xây dựng theo phương pháp học có giám sát. Bài toán lựa chọn mạng Feed-forward 3 lớp với cấu trúc như sau

-Số noron lớp đầu vào : 150 noron.

-Số noron lớp ẩn : 500 noron.

-Số noron đầu ra : 16 noron tương ứng với 16 bit nhị phân của mã Unicode.



Hình 3.2 Mô hình mạng neural

Thuật toán huấn luyện mạng.

Mạng feed- forward sử dụng giải thuật lan truyền ngược sai số Back Propagation.

✓ **Các tham số sử dụng trong chương trình**

Tốc độ học $\eta = 0.15$.

Hệ số góc α hàm Sigmoid= 0.014.

Giá trị ngưỡng hay độ lệch : 30

Số lần dạy 300-600 tùy độ phức tạp của từng loại ký hiệu.

Ngưỡng sai số ε= 0.0002.

3.2.2. Xử lý dữ liệu

Quá trình phân tích ảnh dựa trên việc nhận dạng từ ảnh đầu vào bằng phương pháp tính giá trị màu. Giới hạn của giá trị là điểm đen RGB(0,0,0) hoặc điểm trắng RGB(255, 255, 255).

Những ảnh đầu vào được định dạng *.bmp, *.jpg, *.png, *.gif, *.tiff.

3.2.3. Huấn luyện mạng neural

Một mạng neural đã được xây dựng sẽ phải được huấn luyện trên một không gian đầu vào đã được chuẩn bị trước. Khi hoạt động mạng neural sẽ đọc giá trị đã được huấn luyện.

Thuật toán:

1. Xây dựng mạng tương ứng với mô hình tham số
2. Khởi tạo giá trị trọng số với giá trị ngẫu nhiên. Nạp file huấn luyện. Tìm biên ảnh
3. Đọc giá trị đầu ra mong muốn từ file và lưu trữ các ký hiệu học được
4. Với mỗi ký hiệu:
    a. Tính toán giá trị đầu ra của mạng
    b. So sánh giá trị đầu ra mong muốn tương ứng với từng kí tự và tính toán lỗi.
    c. Truyền ngược giá trị từ đầu vào và với mỗi liên kết điều chỉnh trọng số liên kết.
5. Chuyển sang ký hiệu tiếp theo và lặp lại bước 6 cho đến khi hết các ký hiệu.
6. Tính toán lỗi trung bình cho tất cả các ký hiệu.
7. Lặp lại từ bước 6 đến bước 8 cho tới khi đạt số đưa vào của số lần lặp tối đa.
    a. Với phạm vi lỗi đạt đến ngưỡng. Nếu như vậy thì bỏ lặp lại.
    b. Ngược lại, tiếp tục quá trình huấn luyện

3.3. Cài đặt ứng dụng nhận dạng ký hiệu toán học

3.3.1. Cài đặt chương trình

Chương trình được viết bằng ngôn ngữ C# trong bộ Visual Studio 2010 của Microsoft

3.3.2 Mô tả chương trình

Các module dùng trong chương trình bao gồm:

Module huấn luyện

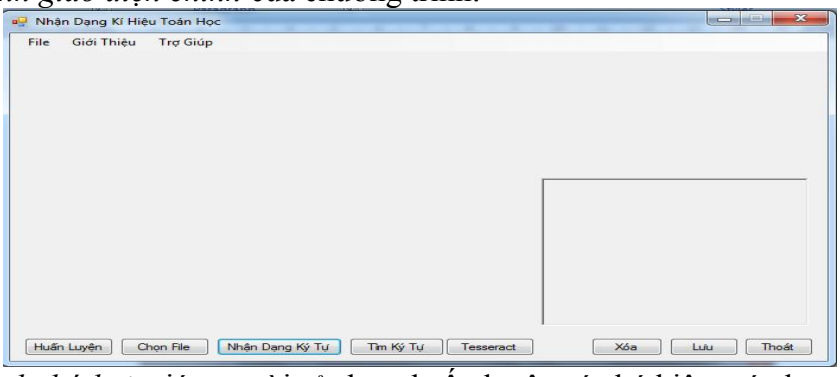
Module nhận dạng ký hiệu

Module nhận dạng dựa trên Tesseract.

3.3.3 Giới thiệu chương trình

Chương trình được thiết kế với 4 cửa sổ màn hình, mỗi cửa sổ thực hiện một chức năng riêng biệt, người dùng sẽ thao tác với các cửa sổ thông qua các nút lệnh trên màn hình giao diện chính.

Màn hình giao diện chính của chương trình:



Màn hình thứ hai, giúp người sử dụng huấn luyện các ký hiệu toán học mới. Các ký hiệu có thể được huấn luyện thông qua file ảnh hoặc người sử dụng tự vẽ lại ký hiệu vào 4 khung text. Các ký hiệu huấn luyện có thể lựa chọn trong hộp thoại ký tự học và nhập tên ký hiệu trong label Tên ký tự

Màn hình thứ ba, giúp người dùng lựa chọn các file ảnh ký hiệu toán học cần nhận dạng:

Màn hình thứ tư, nhận dạng các ký hiệu toán học thông qua mã nguồn Tesseract.

Kết quả nhận dạng lưu lại dạng file *.txt:

3.4 Kết quả

Thử nghiệm việc huấn luyện mạng với tập mẫu các chữ số và ký hiệu toán học. Chương trình thu được các kết quả như sau:

3.4.1 Thử nghiệm tìm số lần lặp lại

Số ký hiệu = 80, hệ số học η =0.15, hệ số góc α =0.014

Epoch	**300**	**600**	**900**
Số ký hiệu sai	7	3	1
% Error	8.75	3.75	1.25

3.4.2 Thử nghiệm tìm tham số hệ số học

Số ký hiệu = 80, số Epoch = 600, hệ số góc α =0.014

Hệ số học η	**0.05**	**0.1**	**0.12**
Số ký hiệu sai	62	15	5
% Error	77.5	18.75	6.25

3.4.3 Thử nghiệm huấn luyện mạng

Các tham số được lựa chọn như sau:

Hệ số học $\eta = 0.15$.

Hệ số góc α hàm Sigmoid= 0.014.

Giá trị ngưỡng hay độ lệch : 30

Số lần dạy 300-600 tùy độ phức tạp của từng loại ký hiệu.

Ngưỡng sai số ε= 0.0002.

Kết luận: Với các kết quả thu được trong quá trình thử nghiệm ta nhận thấy mạng hoạt động khá tốt và ổn định với các thông số như trên.

Để chương trình hoạt động tốt và thực sự hữu ích, chúng ta cần phải giải quyết một số vấn đề sau đây:

- Dữ liệu huấn luyện phải được thu thập một cách chính xác
- Ảnh đưa vào nhận dạng phải đạt chuẩn và có độ phân giải từ 300dpi
- Tối ưu hơn nữa cấu trúc và các thông số mạng, điều này đòi hỏi thời gian thử nghiệm và huấn luyện mạng lâu hơn.

KẾT LUẬN

Luận văn nghiên cứu về lý thuyết mạng neural, mã nguồn mở Tesseract và ứng dụng của nó trong lĩnh vực nhận dạng ký hiệu toán học. Những kết quả mà luận văn đã đạt được:

Lý thuyết:

- Đã nghiên cứu các phương pháp nhận dạng văn bản
- Đã nghiên cứu một cách có hệ thống các khái niệm cơ bản của lý thuyết mạng neural
- Nghiên cứu các ứng dụng của nhận dạng kí tự quang học trong nhận dạng văn bản
- Nghiên cứu phương pháp, cách thu thập dữ liệu áp dụng xây dựng chương trình nhận dạng ký hiệu toán học dựa trên mạng neural và mã nguồn Tesseract.

Ứng dụng:

Trên cơ sở nghiên cứu lý thuyết, đã xây dựng một chương trình nhận dạng các ký hiệu toán học

Hướng nghiên cứu tiếp theo:

- Nâng cao hiệu quả để chương trình chạy nhanh hơn, thời gian huấn luyện nhanh hơn
- Nghiên cứu chương trình ứng dụng phát triển nhận dạng các công thức toán học
- So sánh kết quả với các phương pháp khác để tìm ra một phương pháp tối ưu.