

MỤC LỤC

MỞ ĐẦU	7
CHƯƠNG 1 : MẠNG NƠON VÀ QUÁ TRÌNH HỌC CỦA MẠNG NƠON	12
1.1. Giới thiệu về mạng nơon và quá trình học của mạng nơon	12
1.1.1. Mạng nơon và các phương pháp học	12
1.1.2. Đánh giá các nhân tố của quá trình học.....	13
<i>1.1.2.1. Khởi tạo các trọng số</i>	<i>13</i>
<i>1.1.2.2. Bước học α</i>	<i>13</i>
<i>1.1.2.3. Hằng số quán tính</i>	<i>14</i>
1.2. Nhận dạng hệ thống sử dụng mạng nơon	14
1.2.1. Nhận dạng hệ thống	14
1.2.2. Nhận dạng hệ thống sử dụng mạng nơon	16
1.3. Mặt lỗi đặc biệt khi luyện mạng nơon.....	19
1.3.1. Mặt lỗi đặc biệt khi luyện mạng nơon	19
1.3.2. Ví dụ về bài toán dẫn đến mặt lỗi đặc biệt	20
1.4. Mô phỏng quá trình luyện mạng nơon khi sử dụng Toolbox của Matlab	22
1.4.1. Ví dụ với mạng nơon có mặt lỗi bình thường	22
1.4.2. Ví dụ với mạng nơon có mặt lỗi đặc biệt	25
1.5. Tổng quan về tình hình nghiên cứu trong và ngoài nước	26
1.5.1. Điểm qua một số công trình nghiên cứu về mạng nơon và ứng dụng	26
1.5.2. Các công trình trong và ngoài nước nghiên cứu về thuật toán học của mạng nơon ...	31
1.5.3. Bàn luận	37
1.6. Kết luận chương 1	38
CHƯƠNG 2: THUẬT TOÁN VƯỢT KHE TRONG QUÁ TRÌNH LUYỆN MẠNG NƠON	40
2.1. Thuật toán vượt khe	40
2.1.1. Đặt vấn đề	40
2.1.2. Tính hội tụ và điều kiện tối ưu.....	41
2.1.3. Thuật toán vượt khe	46
<i>2.1.3.1. Giới thiệu</i>	<i>47</i>
<i>2.1.3.2. Nguyên lý vượt khe</i>	<i>48</i>

2.1.3.3. Xác định bước vượt khe	51
2.1.3.4. Ví dụ	54
2.2 Ứng dụng thuật toán vượt khe trong quá trình luyện mạng nơron	56
2.3 Minh họa thuật toán	58
2.3.1. Công tác chuẩn bị	58
2.3.1.1. Điều chỉnh trọng số lớp ra	59
2.3.1.2. Điều chỉnh trọng số lớp ẩn	60
2.3.2. Cấu trúc mạng	62
2.3.3. Các thư viện và hàm mạng	64
2.3.3.1. Thư viện	64
2.3.3.2. Hàm khởi tạo trọng số	66
2.3.3.3. Thủ tục tính bước học vượt khe	67
2.3.3.4. Thủ tục huấn luyện mạng, HUANLUYENVUOTKHE()	69
2.3.4. Kết quả chạy chương trình và so sánh	69
2.3.4.1. Chạy chương trình	69
2.3.4.2. So sánh các phương án	73
2.4. Kết luận chương 2	76
CHƯƠNG 3: ĐỀ XUẤT MÔ HÌNH KẾT HỢP THUẬT DI TRUYỀN VÀ THUẬT TOÁN VƯỢT KHE ĐỂ CẢI TIẾN QUÁ TRÌNH HỌC CỦA MẠNG NƠON MLP CÓ MẶT LỖI ĐẶC BIỆT	77
3.1. Đặt vấn đề	77
3.1.1. Khảo sát độ hội tụ của quá trình luyện mạng nơron bằng kỹ thuật lan truyền ngược nguyên thủy với các bộ khởi tạo trọng số ban đầu khác nhau	77
3.1.2. Khảo sát độ hội tụ của quá trình luyện mạng nơron có mặt lỗi đặc biệt bằng kỹ thuật lan truyền ngược kết hợp thuật toán vượt khe với các bộ khởi tạo trọng số ban đầu khác nhau	80
3.2. Đề xuất mô hình kết hợp giải thuật di truyền và thuật toán vượt khe trong quá trình luyện mạng nơron	83
3.2.1. Đặt vấn đề	83
3.2.2. Thuật toán	87
3.3. Áp dụng mô hình kết hợp giải thuật di truyền và thuật toán vượt khe trong quá trình luyện mạng nơron vào bài toán nhận dạng	91
3.4. Kết luận chương 3	94
KẾT LUẬN CHUNG VÀ ĐỀ XUẤT HƯỚNG NGHIÊN CỨU	95

CÁC CÔNG TRÌNH ĐÃ CÔNG BỐ	99
TÀI LIỆU THAM KHẢO	100
PHỤ LỤC 1.....	106

DANH MỤC BẢNG BIỂU, HÌNH VẼ

<i>Bảng 2.1. Các hàm kích hoạt (transfer function) tiêu biểu.....</i>	<i>64</i>
<i>Bảng 2.2: Tập hồ sơ mẫu đầu vào {0 1 2 3 4 5 6 7 8 9}</i>	<i>74</i>
<i>Bảng 2.3: Tập hồ sơ mẫu đầu vào {! @ # \$ % ^ & * ()}</i>	<i>75</i>
<i>Bảng 3.1. Kết quả khi nhận dạng hệ thống phi tuyến tĩnh.....</i>	<i>79</i>
<i>Bảng 3.2: Kết quả khi nhận dạng hệ thống động học phi tuyến</i>	<i>80</i>
<i>Bảng 3.3: Kết quả khi nhận dạng hệ thống có mặt lỗi dạng lòng khe.....</i>	<i>82</i>
<i>Bảng 3.4. So sánh GA và BP với sai số là 0.1</i>	<i>85</i>
<i>Bảng 3.5: So sánh GA và BP với sai số là 0.001</i>	<i>86</i>
<i>Bảng 3.6: So sánh GA và BP với sai số khác nhau</i>	<i>86</i>
<i>Hình 1.1. Điều khiển theo nguyên tắc phản hồi đầu ra</i>	<i>15</i>
<i>Hình 1.2: Mô hình nhận dạng cơ bản</i>	<i>18</i>
<i>Hình 1.3. Mặt sai số dạng lòng khe</i>	<i>19</i>
<i>Hình 1.4: Kỹ nguyên luyện mạng ví dụ 1</i>	<i>24</i>
<i>Hình 1.5: Cấu trúc mạng nơron cho nhận dạng chữ.....</i>	<i>25</i>
<i>Hình 1.6: Kết quả luyện mạng nơron với các phương pháp lan truyền ngược khác nhau (traingd, traingdm, traindx, trainda)</i>	<i>26</i>
<i>Hình 2.1: Quỹ đạo dao động với sai số dạng lòng khe.....</i>	<i>42</i>
<i>Hình 2.2: Hàm khe</i>	<i>48</i>
<i>Hình 2.3: Xác định bước vượt khe α^v</i>	<i>50</i>
<i>Hình 2.4: Lưu đồ thuật toán tính bước vượt khe.....</i>	<i>54</i>
<i>Hình 2.5: Bước lặp $k = 1$</i>	<i>55</i>
<i>Hình 2.6: Các đường đồng mức dạng khe</i>	<i>57</i>
<i>Hình 2.7: Lưu đồ thuật toán huấn luyện mạng nơron với bước học vượt khe.....</i>	<i>58</i>
<i>Hình 3.1: Sơ đồ thuật toán kết hợp giải thuật vượt khe và di truyền cho luyện mạng MLP</i>	<i>90</i>
<i>Hình 3.2: Hoạt động của mạng MLP cải tiến.....</i>	<i>93</i>
<i>Hình a: So sánh hoạt động của mạng MLP thuần túy và MLP cải tiến</i>	<i>97</i>

CÁC TỪ VIẾT TẮT

ADLINE	ADaptive Linear Neural, mạng tuyến tính thích nghi đơn lớp
ANN	Artificial Neural Network, mạng nơron nhân tạo
BP	BackPropagation, lan truyền ngược
BPTT	BackPropagation -Through-Time, lan truyền ngược xuyên tâm
LDDN	Layered Digital Dynamic Network, mạng nơron động
LMS	Least Mean Square, trung bình bình phương nhỏ nhất
NNs	Neural NetworkS, mạng nơron
RTRL	Real-Time Recurrent Learning, thuật học hồi qui thời gian thực
SDBP	Steepest Descent BackPropagation, kỹ thuật lan truyền ngược giảm dốc nhất
OBP	Optical BackPropagation, kỹ thuật lan truyền ngược “tốc độ ánh sáng”
VLBP	Variable Learning rate BackPropagation algorithm, kỹ thuật lan truyền ngược với tốc độ học thay đổi.
MLP	MultiLayer Perceptron, mạng truyền thẳng nhiều lớp
GA	Genetic Algorithms, giải thuật di truyền

LỜI CAM ĐOAN

Tôi xin cam đoan luận án này là công trình nghiên cứu khoa học của tôi và không trùng lặp với bất cứ công trình khoa học nào khác. Các số liệu trình bày trong luận án đã được kiểm tra kỹ và phản ánh hoàn toàn trung thực. Các kết quả nghiên cứu do tác giả đề xuất chưa từng được công bố trên bất kỳ tạp chí nào đến thời điểm này ngoài những công trình của tác giả.

Ngày 14 tháng 10 năm 2013

Tác giả luận án

Nguyễn Thị Thanh Nga

MỞ ĐẦU

Trong rất nhiều lĩnh vực như điều khiển, tự động hóa, công nghệ thông tin..., nhận dạng được đối tượng là vấn đề mấu chốt quyết định sự thành công của bài toán. Phần lớn các đối tượng trong thực tế đều là phi tuyến với độ phi tuyến khác nhau.

Mạng nơron có khả năng xấp xỉ các hàm phi tuyến một cách đầy đủ và chính xác, nó được sử dụng tốt cho các mô hình động học phi tuyến. Điều quan trọng là thuật lan truyền ngược tĩnh và động của mạng nơron được sử dụng để hiệu chỉnh các tham số trong quá trình nhận dạng. Cơ sở toán học của việc khẳng định rằng mạng nơron là công cụ xấp xỉ vạn năng các hàm số liên tục dựa trên các định lý Stone – Weierstrass và Kolmogorov[15].

Việc sử dụng định lý Stone – Weierstrass để chứng minh khả năng xấp xỉ của mạng nơron đã được các tác giả Hornik et al., Funahashi, Cotter, Blum đưa ra từ năm 1989. Các mạng nơron thỏa mãn định lý Stone – Weierstrass có thể kể đến là mạng lượng giác, mạng hai lớp với hàm kích hoạt sigmoid, mạng hai lớp với hàm kích hoạt McCulloch – Pitts(MC - P) và mạng với hàm cơ sở xuyên tâm(RBF)[16], [17], [18], [19].

Việc sử dụng định lý Kolmogorov để biểu diễn chính xác hàm liên tục và đưa ra sơ đồ mạng nơron tương ứng đã được Hecht - Nielsen và Lorentz công bố[20], [21], [22].

Mạng nơron là một trong những công cụ nhận dạng tốt nhất vì các đặc trưng sau: Khả năng học từ kinh nghiệm (khả năng được huấn luyện), khả năng xử lý song song với tốc độ xử lý nhanh, khả năng học thích nghi, khả năng khái quát hoá cho các đầu vào không được huấn luyện, ví dụ dựa vào cách học mạng có thể sẽ tiên đoán đầu ra từ đầu vào không biết trước [23], [24].

Hiện nay, một công cụ phần mềm được ứng dụng rất hiệu quả trong các lĩnh vực về điều khiển, tự động hóa, công nghệ thông tin đó là Matlab. Khi sử dụng bộ công cụ Neural Network Toolbox, chúng ta có thể luyện mạng để nhận dạng được một số đối tượng tuyến tính và phi tuyến. Bộ công cụ cung cấp cho chúng ta một số

phương pháp luyện mạng nơron, trong đó kỹ thuật lan truyền ngược được ứng dụng rộng rãi hơn cả. Ở đó chúng ta có thể lựa chọn các bước học khác nhau phục vụ cho quá trình luyện mạng như: Traingd (Basic gradient descent), Traingdm (Gradient descent with momentum), Traingdx (Adaptive learning rate), Trainbfg (BFGS quasi-Newton)...

Một nhược điểm khi dùng mạng nơron là chưa có phương pháp luận chung khi thiết kế cấu trúc mạng cho các bài toán nhận dạng và điều khiển mà phải cần tới kiến thức của chuyên gia. Mặt khác khi xấp xỉ mạng nơron với một hệ phi tuyến sẽ khó khăn khi luyện mạng vì có thể không tìm được điểm tối ưu toàn cục... Vậy, tồn tại lớn nhất gặp phải là tìm nghiệm tối ưu toàn cục, đặc biệt áp dụng cho các bài toán lớn, các hệ thống điều khiển quá trình.

Giải thuật di truyền (Genetic Algorithms-GA) được biết đến như một giải thuật tìm kiếm dựa trên học thuyết về chọn lọc tự nhiên và nó cho phép ta đạt được tới cực trị toàn cục. Thực ra, GA thuộc lớp các thuật toán xác suất, nhưng lại rất khác những thuật toán ngẫu nhiên vì chúng kết hợp các phần tử tìm kiếm trực tiếp và ngẫu nhiên. Khác biệt quan trọng giữa phương pháp tìm kiếm của GA và các phương pháp tìm kiếm khác là GA duy trì và xử lý một tập các lời giải (quần thể) - tất cả các phương pháp khác chỉ xử lý một điểm trong không gian tìm kiếm. Chính vì thế, GA mạnh hơn các phương pháp tìm kiếm hiện có rất nhiều. [25], [26].

Hiện nay, việc nghiên cứu các thuật toán tìm nghiệm tối ưu toàn cục khi luyện mạng nơron đã được một số tác giả nghiên cứu áp dụng [27], [28], [29]. Tuy nhiên khi sử dụng mạng nơron để xấp xỉ một số đối tượng phi tuyến mà mặt lỗi sinh ra có dạng lòng khe [28], việc huấn luyện mạng gặp rất nhiều khó khăn.

Nội dung đề tài sẽ đi nghiên cứu một thuật toán tìm điểm tối ưu toàn cục trong quá trình luyện mạng nơron bằng thuật toán vượt khe có sự kết hợp với giải thuật di truyền.

Mục tiêu

- Đề xuất mô hình kết hợp thuật toán vượt khe và giải thuật di truyền để huấn luyện mạng nơron.

- Xây dựng bộ công cụ phần mềm để luyện mạng nơron cho một số bài toán có mặt lồi đặc biệt, làm cơ sở bổ sung vào Neural Toolbox Matlab.

Nội dung chính

- Nghiên cứu lý thuyết về mạng nơron và quá trình học của mạng nơron.
- Nghiên cứu lý thuyết về thuật toán vượt khe và xây dựng thuật toán tính bước học vượt khe.
- Xây dựng thuật toán huấn luyện mạng nơron bằng kỹ thuật lan truyền ngược kết hợp với thuật toán vượt khe.
- Đề xuất thuật toán huấn luyện mạng nơron bằng kỹ thuật lan truyền ngược có sử dụng giải thuật di truyền kết hợp với thuật toán vượt khe.
- Viết và cài đặt chương trình huấn luyện mạng nơron trên C++.
- Viết và cài đặt chương trình huấn luyện mạng nơron trên Matlab.

Phương pháp nghiên cứu

Sử dụng cả nghiên cứu lý thuyết, thực nghiệm mô phỏng trên máy tính.

***. Nghiên cứu lý thuyết:**

- Tập trung nghiên cứu vấn đề mạng nơron là gì và ứng dụng của mạng nơron trong nhận dạng. Nghiên cứu những khó khăn tồn tại khi luyện mạng nơron với mặt lồi đặc biệt có dạng lòng khe.
- Nghiên cứu giải bài toán tối ưu tĩnh mà hàm mục tiêu có dạng đặc biệt – dạng lòng khe. Với hàm mục tiêu này bằng các phương pháp thông thường, ví dụ như phương pháp gradient không tìm được cực tiểu, còn thuật toán vượt khe có thể vượt qua được lòng khe để đến điểm tối ưu.
- Nghiên cứu sự ảnh hưởng giá trị ban đầu khi giải bài toán tối ưu tĩnh bằng phương pháp số, đặc biệt khi hàm mục tiêu có dạng lòng khe. Giá trị ban đầu ảnh hưởng lớn tới tính hội tụ và thời gian tính nghiệm tối ưu.
- Nghiên cứu giải thuật di truyền, và ứng dụng của nó trong quá trình tìm nghiệm tối ưu toàn cục.

- Đề xuất mô hình kết hợp thuật toán vượt khe và giải thuật di truyền để luyện mạng nơron có mặt lỗi đặc biệt.

Cơ sở toán học chính gồm lý thuyết về khả năng xấp xỉ vạn năng của mạng nơron với đối tượng phi tuyến có hàm số liên tục là dựa trên các định lý Stone – Weierstrass và Kolmogorov; khả năng tìm ra được vùng chứa cực trị toàn cục của giải thuật di truyền nhờ cơ chế tìm kiếm trải rộng, ngẫu nhiên và mang tính chọn lọc tự nhiên; khả năng tìm đến được cực trị toàn cục của thuật toán tối ưu vượt khe khi hàm phi tuyến có dạng khe.

***. Nghiên cứu thực nghiệm:** Mô phỏng trên máy tính bằng cách sử dụng:

- Bộ công cụ sẵn có trong Toolbox của Matlab.

- Viết chương trình trên C++.

- Viết chương trình trên Matlab

để thực hiện quá trình luyện mạng nơron với mặt lỗi dạng đặc biệt. Đánh giá sự hội tụ để minh chứng cho những kết luận trong phần lý thuyết.

Bố cục của luận án

Luận án chia làm 3 chương

Chương 1 trình bày tổng quan về mạng nơron, quá trình học của mạng nơron, đánh giá các nhân tố của quá trình học. Giới thiệu về mặt lỗi đặc biệt trong quá trình luyện mạng nơron, mặt lỗi có dạng lòng khe, những bài toán dẫn đến mặt lỗi có dạng lòng khe. Sử dụng bộ công cụ Neural Network Toolbox để nhận dạng một số đối tượng có hàm mục tiêu dạng thông thường và dạng đặc biệt. Tóm tắt về tình hình nghiên cứu trong và ngoài nước, từ đó làm xuất phát điểm cho nội dung nghiên cứu của các chương tiếp theo.

Chương 2 trình bày một thuật toán tối ưu áp dụng cho các hàm mục tiêu dạng khe gọi là thuật toán vượt khe. Để giải quyết bài toán nhận dạng đối tượng phi tuyến mà sinh ra hàm mục tiêu dạng khe, tác giả đề xuất việc áp dụng thuật toán vượt khe tính bước học vượt khe trong quá trình học của mạng nơron. Để minh chứng cho hiệu quả của bước học vượt khe, tác giả lấy một ví dụ về nhận dạng chữ

viết tay và chọn hàm kích hoạt là hàm sigmoid do đặc điểm hàm này sinh ra mặt sai số có dạng lòng khe. Ví dụ sẽ được luyện mạng với các phương pháp cập nhật bước học khác nhau. Cuối chương sẽ có đánh giá hiệu quả của các phương pháp này.

Chương 3, tác giả đưa ra các ví dụ về luyện mạng nơron trên những đối tượng có mức độ phi tuyến khác nhau với bộ trọng số khởi tạo khác nhau để thấy sự ảnh hưởng của bộ khởi tạo trọng số đến kết quả luyện mạng, từ đó đề xuất mô hình kết hợp giải thuật di truyền và thuật toán vượt khe trong quá trình luyện mạng nơron. Trong mô hình, giải thuật di truyền làm nhiệm vụ tìm kiếm bộ trọng số khởi tạo tối ưu, khoanh vùng chứa cực trị toàn cục để tiến hành luyện mạng nơron theo kỹ thuật lan truyền ngược có sử dụng bước học vượt khe đã đề xuất từ chương 2.

CHƯƠNG 1

MẠNG NƠRON VÀ QUÁ TRÌNH HỌC CỦA MẠNG NƠRON

Tóm tắt: Trong rất nhiều lĩnh vực như điều khiển, tự động hóa, công nghệ thông tin..., vấn đề nhận dạng được đối tượng là vấn đề mấu chốt quyết định sự thành công của bài toán. Mạng nơron có khả năng xấp xỉ các hàm phi tuyến một cách đầy đủ và chính xác, nó được sử dụng tốt cho các mô hình động học phi tuyến. Tuy nhiên trong quá trình học của mạng nơron, một số nhân tố sẽ có ảnh hưởng mạnh mẽ đến độ hội tụ của bài toán, đặc biệt khi bài toán có dạng lòng khe. Chương 1 sẽ đưa ra mặt lỗi đặc biệt khi luyện mạng nơron và có những đánh giá về sự ảnh hưởng của các nhân tố trong quá trình luyện mạng đến kết quả cuối cùng của bài toán nhận dạng thông qua một số ví dụ đặc trưng. Từ đó làm xuất phát điểm cho hướng đi của luận án.

1.1. Giới thiệu về mạng nơron và quá trình học của mạng nơron

1.1.1. Mạng nơron và các phương pháp học

Mạng nơron nhân tạo, *Artificial Neural Network* (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.

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 sao cho giá trị hàm lỗi là nhỏ nhất.

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 phương pháp học được sử dụng phổ biến nhất và trong đó tiêu biểu nhất là kỹ thuật lan truyền ngược.

Những kiến thức cơ sở về mạng nơron, quá trình học của mạng nơron, kỹ thuật lan truyền ngược sẽ được tác giả giới thiệu trong *phụ lục 1*.

Ở đây, tác giả xin đưa ra một số đánh giá về các nhân tố trong quá trình học của mạng nơron.

1.1.2. Đánh giá các nhân tố của quá trình học

1.1.2.1. Khởi tạo các trọng số

Kỹ thuật lan truyền ngược hội tụ đến một giải pháp mà nó tối thiểu hoá được sai số trung bình bình phương vì cách thức hiệu chỉnh trọng số và hệ số bias của thuật toán là ngược hướng với vector Gradient của hàm sai số trung bình bình phương đối với trọng số. Tuy nhiên, đối với mạng MLP thì hàm sai số trung bình bình phương thường phức tạp và có nhiều cực trị cục bộ, vì thế các phép lặp huấn luyện mạng có thể chỉ đạt được đến cực trị cục bộ của hàm sai số trung bình bình phương mà không đạt đến được cực trị tổng thể. Các giá trị khởi tạo của các trọng số ảnh hưởng rất mạnh đến lời giải cuối cùng. Các trọng số này thường được khởi tạo bằng những số ngẫu nhiên nhỏ. Việc khởi tạo tất cả các trọng số bằng nhau sẽ làm cho mạng học không tốt. Nếu các trọng số được khởi tạo với giá trị lớn thì ngay từ đầu tổng tín hiệu vào đã có giá trị tuyệt đối lớn và làm cho hàm sigmoid chỉ đạt 2 giá trị 0 và 1. Điều này làm cho hệ thống sẽ bị tắc ngay tại một cực tiểu cục bộ hoặc tại một vùng bằng phẳng nào đó gần ngay tại điểm xuất phát. Giá trị khởi tạo ban đầu của các trọng số trên lớp thứ 1 của mạng sẽ được chọn ngẫu nhiên nhỏ trong khoảng $[-1/n, 1/n]$, trong đó n là số trọng số nối tới lớp 1. Do bản chất của giải thuật học lan truyền ngược sai số là phương pháp giảm độ lệch gradient nên việc khởi tạo các giá trị ban đầu của các trọng số các giá trị nhỏ ngẫu nhiên sẽ làm cho mạng hội tụ về các giá trị cực tiểu khác nhau. Nếu gặp may thì mạng sẽ hội tụ được về giá trị cực tiểu tổng thể.

1.1.2.2. Bước học α

Một nhân tố khác ảnh hưởng đến hiệu lực và độ hội tụ của giải thuật lan truyền ngược sai số là bước học α . Không có một giá trị xác định nào cho các bài toán khác nhau. Với mỗi bài toán, bước học thường được lựa chọn bằng thực nghiệm theo phương pháp thử và sai. Giá trị α lớn làm tăng tốc quá trình hội tụ.

Điều này không phải lúc nào cũng có lợi vì nếu ngay từ đầu ta đã cho là mạng nhanh hội tụ thì rất có thể mạng sẽ hội tụ sớm ngay tại một cực tiểu địa phương gần nhất mà không đạt được độ sai số như mong muốn. Tuy nhiên, đặt giá trị bước học quá nhỏ thì mạng sẽ hội tụ rất chậm, thậm chí mạng có thể vượt được qua các cực tiểu cục bộ và vì vậy dẫn đến học mãi mà không hội tụ. Do vậy, việc chọn hằng số học ban đầu là rất quan trọng. Với mỗi bài toán ta lại có phương án chọn hệ số học khác nhau. Như vậy, khi một quá trình huấn luyện theo kỹ thuật lan truyền ngược hội tụ, ta chưa thể khẳng định được nó đã hội tụ đến phương án tối ưu. Ta cần phải thử với một số điều kiện ban đầu để đảm bảo thu được phương án tối ưu.

1.1.2.3. *Hằng số quán tính*

Tốc độ học của giải thuật lan truyền ngược sai số có thể dao động khi hằng số học lớn. Một phương pháp thường dùng cho phép sử dụng hằng số học lớn là thêm thành phần quán tính vào các phương trình hiệu chỉnh các trọng số. Ngoài ra, hằng số quán tính ngăn cản sự thay đổi đột ngột của các trọng số theo hướng khác với hướng mà lời giải đang di chuyển đến. Mặt trái của việc sử dụng thành phần quán tính là chúng ta phải tăng đáng kể bộ nhớ của máy tính gần như gấp đôi để lưu trữ các giá trị hiệu chỉnh ở chu kỳ trước.

1.2. Nhận dạng hệ thống sử dụng mạng nơron

1.2.1. Nhận dạng hệ thống

1.2.1.1. *Tại sao phải nhận dạng*

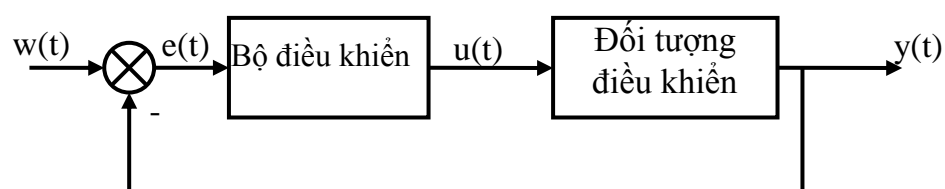
Bài toán nhận dạng là một vấn đề đặt lên hàng đầu trong nhiều các lĩnh vực khác nhau như: điện tử y sinh, điện tử viễn thông, hệ thống điện, tự động hóa và điều khiển... Ví dụ như: nhận dạng vân tay, nhận dạng ký tự, ảnh, tiếng nói, phát hiện và chẩn đoán bệnh... Xét trong lĩnh vực tự động hóa và điều khiển, nhận dạng hệ thống là một trong những công việc đầu tiên phải thực hiện, nó quyết định chất lượng và hiệu quả của công việc điều khiển hệ thống. Tuy ra đời muộn nhưng nhận dạng đã phát triển rất nhanh và đã có những thành tựu vượt bậc. Nguyên nhân của sự phát triển vượt bậc đó một phần từ yêu cầu thực tế, song có lẽ phần chính là nhờ có

những hỗ trợ tích cực của các ngành khoa học có liên quan như tin học, các công cụ tính toán mềm nói chung và mạng nơron nói riêng...

Ví dụ 1: Thiết kế rô bốt giúp người khiếm thị trong học tập và sinh hoạt.

Hiện nay, robot được coi như tâm điểm của cuộc cách mạng lớn sau Internet. Để thiết kế và chế tạo được rô bốt, ta cần có các tri thức của toán học, cơ học, vật lý, điện tử, lý thuyết điều khiển, khoa học tính toán và nhiều tri thức khác. Tại Việt Nam, nghiên cứu phát triển rô bốt đã có những bước tiến đáng kể trong thời gian vừa qua. Nhiều đơn vị trên toàn quốc thực hiện các nghiên cứu cơ bản và nghiên cứu ứng dụng về rô bốt như Trung tâm Tự động hoá, Đại học Bách Khoa Hà Nội; Viện Điện tử, Tin học, Tự động hoá thuộc Bộ Công thương; Đại học Bách khoa TP.HCM; Viện Khoa học và Công nghệ quân sự, Học viện Kỹ thuật Quân sự, Viện Cơ học; Viện Công nghệ thông tin thuộc Viện KHCNVN. Các nghiên cứu hiện nay tập trung nhiều vào vấn đề xử lý ngôn ngữ tự nhiên, nhận dạng và tổng hợp tiếng nói, chữ viết tay đặc biệt là tiếng Việt.

Ví dụ 2: Xét bài toán điều khiển theo nguyên tắc phản hồi như trên *hình 1.1*:



Hình 1.1: Điều khiển theo nguyên tắc phản hồi đầu ra

Muốn thiết kế được bộ điều khiển hệ kín cho đối tượng có được chất lượng như mong muốn thì phải hiểu biết về đối tượng, tức là cần phải có một mô hình toán học mô tả đối tượng. Không thể điều khiển đối tượng khi không hiểu biết hoặc hiểu sai lệch về nó. Kết quả thiết kế bộ điều khiển phụ thuộc rất nhiều vào mô hình mô tả đối tượng. Mô hình càng chính xác, chất lượng của việc điều khiển càng cao.

Như vậy, nhận dạng là cần thiết cho việc ra quyết định tự động và hỗ trợ con người ra quyết định.

Việc xây dựng mô hình cho đối tượng cần nhận dạng được gọi là mô hình hóa. Người ta thường phân chia các phương pháp mô hình hóa ra làm hai loại:

- Phương pháp lý thuyết.
- Phương pháp thực nghiệm.

Phương pháp lý thuyết là phương pháp thiết lập mô hình dựa trên các định luật có sẵn về quan hệ vật lý bên trong và quan hệ giao tiếp với môi trường bên ngoài của đối tượng. Các quan hệ này được mô tả theo quy luật lý – hóa, quy luật cân bằng, ... dưới dạng những phương trình toán học.

Trong các trường hợp mà sự hiểu biết về những quy luật giao tiếp bên trong đối tượng với môi trường bên ngoài không được đầy đủ để có thể xây dựng được một mô hình hoàn chỉnh, nhưng ít nhất từ đó có thể cho biết các thông tin ban đầu về dạng mô hình thì tiếp theo người ta phải áp dụng phương pháp thực nghiệm để hoàn thiện nốt việc xây dựng mô hình đối tượng trên cơ sở quan sát tín hiệu vào $u(t)$ và ra $y(t)$ của đối tượng sao cho mô hình thu được bằng phương pháp thực nghiệm thỏa mãn các yêu cầu của phương pháp lý thuyết đề ra. Phương pháp thực nghiệm đó được gọi là nhận dạng hệ thống.

1.2.2. Nhận dạng hệ thống sử dụng mạng nơron

1.2.2.1. Khả năng sử dụng mạng nơron trong nhận dạng

Xét trường hợp đối tượng phi tuyến có độ phức tạp cao, nếu sử dụng phương pháp giải tích thông thường để nhận dạng sẽ rất khó khăn, thậm chí không thực hiện được do sự hiểu biết nghèo nàn về đối tượng. Vì vậy các nhà khoa học đã đưa ra ý tưởng là sử dụng công cụ tính toán mềm như hệ mờ, mạng nơron, đại số gia tử để xấp xỉ - chính là nhận dạng đối tượng. Các tài liệu [15], [23], [24] chỉ ra rằng, mạng nơron là một trong những công cụ hữu hiệu để nhận dạng mô hình đối tượng. Bằng phương pháp này ta không biết được mô hình toán thực sự của đối tượng nhưng hoàn toàn có thể dùng kết quả xấp xỉ để thay thế đối tượng.

Vì tính phi tuyến của các mạng nơron (hàm kích hoạt phi tuyến), chúng được dùng để mô tả các hệ thống phi tuyến phức tạp. Cybenko đã chứng minh rằng một hàm liên tục có thể xấp xỉ tùy ý bằng một mạng truyền thẳng với chỉ một lớp ẩn.

Mạng nơron là một trong những công cụ nhận dạng tốt nhất vì các đặc trưng sau: Khả năng học từ kinh nghiệm (khả năng được huấn luyện), khả năng khái quát

hoá cho các đầu vào không được huấn luyện, ví dụ dựa vào cách học mạng có thể sẽ tiên đoán đầu ra từ đầu vào không biết trước.

Mạng nơron có khả năng xấp xỉ các hàm phi tuyến một cách đầy đủ và chính xác, nó được sử dụng tốt cho các mô hình động học phi tuyến. Điều quan trọng được sử dụng là thuật truyền ngược tĩnh và động của mạng nơron, nó được sử dụng để hiệu chỉnh các tham số trong quá trình nhận dạng.

Nền tảng cho tính xấp xỉ hàm của mạng nơron nhiều lớp là định lý Kolmogorov và định lý Stone – Weierstrass. Các mạng nơron nhân tạo đưa ra những lợi thế qua việc học sử dụng phân loại và xử lý song song, điều này rất phù hợp với việc dùng trong nhận dạng.

1.2.2.2. Mô hình nhận dạng hệ thống sử dụng mạng nơron

Khi xét một bài toán điều khiển, trước tiên ta cần phải có những hiểu biết về đối tượng: số đầu vào, số đầu ra, các đại lượng vật lý vào ra, dải giá trị của chúng, quy luật thay đổi của các đại lượng trong hệ hay mô hình toán học cơ bản của nó,... Tuy nhiên không phải đối tượng nào hay hệ nào cũng cung cấp được đầy đủ các thông tin như trên cũng như xây dựng được mô hình thực từ những thông tin ấy. Việc nhận dạng là việc đầu tiên và quan trọng để việc điều khiển đạt chất lượng mong muốn. Khi thông số của đối tượng là cần thiết để việc điều khiển đạt chất lượng mong muốn. Khi thông số của đối tượng tự thay đổi trong quá trình làm việc (đối tượng phi tuyến) và có tính động học thì việc nhận dạng theo chúng sẽ phức tạp hơn nhiều so với đối tượng có thông số bất biến.

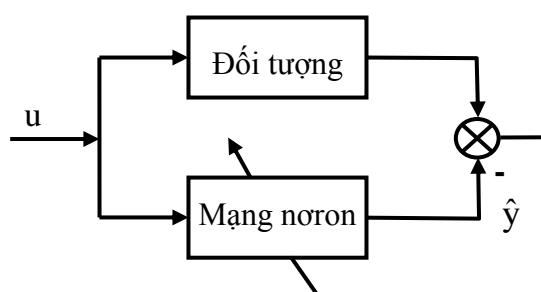
Nhận dạng thường chia ra làm: nhận dạng mô hình và nhận dạng tham số.

Nhận dạng mô hình là quá trình xác định mô hình của đối tượng và thông số trên cơ sở đầu vào và đầu ra của đối tượng.

Mô hình thu được sau khi nhận dạng gọi là tốt nếu nó thể hiện được đúng đối tượng. Như vậy có thể sử dụng mô hình thay cho đối tượng để dự báo, kiểm tra và điều khiển. Mạng nơron được huấn luyện để mô hình hóa quan hệ vào ra của đối tượng. Như vậy quy trình nhận dạng mô hình có bản chất là thuật toán huấn luyện mạng.

Cấu trúc mạng nơron giải bài toán nhận dạng mô hình rất đa dạng, tùy thuộc vào từng bài toán cụ thể.

Nhận dạng tham số chính là huấn luyện mạng. Mô hình cơ bản của mạng nơron được luyện để mô phỏng hành vi của đối tượng giống như mô hình truyền thống được biểu diễn trên *Hình 1.2*



Hình 1.2: Mô hình nhận dạng cơ bản

Tín hiệu sai số $e = y - \hat{y}$ là cơ sở cho quá trình luyện mạng. Mạng nơron ở đây có thể là mạng nhiều lớp hoặc các dạng khác và có thể sử dụng nhiều thuật luyện mạng khác nhau.

1.2.2.3. Nhận dạng hệ thống sử dụng mạng nơron

Như vậy nhận dạng hệ thống cần hai giai đoạn đó là lựa chọn mô hình và tối ưu tham số. Đối với mạng nơron dựa vào nhận dạng lựa chọn số nút ẩn, số lớp ẩn (cấu trúc của mạng) tương đương với mô hình lựa chọn. Mạng có thể được huấn luyện theo kiểu giám sát với kỹ thuật lan truyền ngược, dựa vào luật học sai số hiệu chỉnh. Tín hiệu sai số được lan truyền ngược qua mạng. Kỹ thuật lan truyền ngược sử dụng phương pháp giảm gradient để xác định các trọng của mạng vì vậy tương đương với tối ưu tham số. Mạng nơron được huấn luyện để xấp xỉ mối quan hệ giữa các biến.

Mạng nơron được huấn luyện để tối thiểu hàm sai số. Mạng được huấn luyện để tối thiểu sai số bình phương giữa đầu ra của mạng và đầu vào hệ thống, xác định một hàm truyền ngược. Trong kiểu nhận dạng này đầu ra của mạng hội tụ về đầu vào hệ sau khi huấn luyện, vì vậy mạng đặc trưng cho hàm truyền ngược của hệ. Phương pháp nhận dạng khác cần phải hướng đầu ra hệ thống tới đầu ra của mạng. Trong kiểu này mạng đặc trưng cho hàm truyền thẳng của hệ thống.

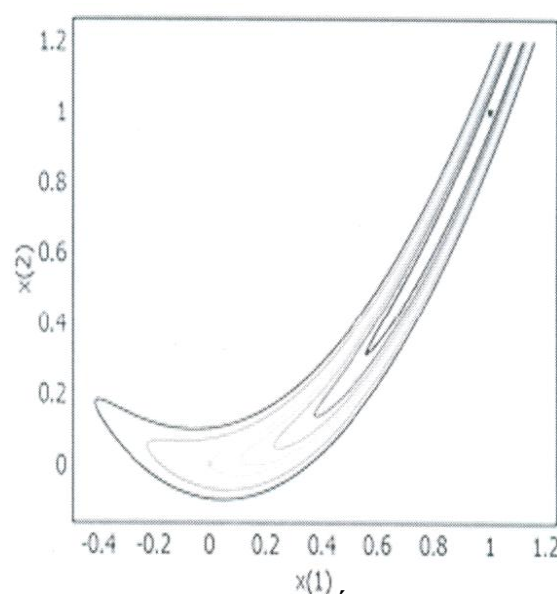
Giả sử các hàm phi tuyến để mô tả hệ thuộc lớp hàm đã biết trong phạm vi quan tâm thì cấu trúc của mô hình nhận dạng phải phù hợp với hệ thống. Với giả thiết các ma trận trọng của mạng nơron trong mô hình nhận dạng tồn tại, cùng các điều kiện ban đầu thì cả hệ thống và mô hình có cùng lượng ra với bất kỳ lượng vào xác định. Do đó quá trình nhận dạng thực chất là điều chỉnh tham số của mạng nơron dựa vào sai lệch giữa các giá trị đầu ra của hệ thống và của mô hình.

1.3. Mặt lỗi đặc biệt khi luyện mạng nơron

1.3.1. Mặt lỗi đặc biệt khi luyện mạng nơron

Trong quá trình nỗ lực thoát ra khỏi các cực tiểu yếu, cực tiểu cục bộ và những mong muốn giảm chi phí thời gian thực hiện của máy tính khi tìm kiếm nghiệm tối ưu thì vấn đề nghiên cứu đặc điểm của các mặt lỗi thường được chọn làm xuất phát điểm cho việc cải tiến hay đề xuất các thuật học mới. Khi nói về mạng nơron thì huấn luyện chất lượng mạng được nhắc đến nhiều hơn cả (loại học có giám sát). Điều này liên quan đến hàm chất lượng của mạng và dẫn đến khái niệm mặt chất lượng mạng. Đôi khi chúng ta còn gọi mặt chất lượng bằng những thuật ngữ khác: mặt sai số, mặt thực thi, mặt lỗi.

Hình 1.3 mô tả một mặt sai số, có một vài điều đặc biệt cần chú ý đối với mặt sai số này đó là độ dốc biến đổi một cách mạnh mẽ trên không gian tham số. Vì lý do đó, nó sẽ khó để mà lựa chọn một tốc độ học phù hợp cho thuật toán giảm dốc nhất. Trong một vài vùng của mặt sai số thì rất phẳng, cho phép tốc độ học lớn, trong khi các vùng khác độ dốc lớn, yêu cầu một tốc độ học nhỏ. Có thể ghi nhận rằng có thể sẽ không đáng ngạc nhiên lắm đối với các



Hình 1.3: Mặt sai số dạng lòng khe

vùng phẳng của mặt sai số bởi một lý do chúng ta dùng hàm truyền sigmoid cho mạng. Hàm sigmoid rất hay được sử dụng trong mạng nơron bởi đặc điểm của nó

(bị chặn, đơn điệu tăng, khả vi) thích nghi với các kỹ thuật tối ưu kinh điển, hàm này có đặc điểm là rất phẳng đối với các đầu vào lớn.

1.3.2. Ví dụ về bài toán dẫn đến mặt lồi đặc biệt

Đặc điểm khe của các bài toán tối ưu hoá trong ngành nhiệt [28]

Do đặc thù của đối tượng nhiệt, các bài toán tối ưu hoá trong ngành nhiệt thường có hàm mục tiêu là phi tuyến, không liên tục, không khả vi, có tính chất khe rõ rệt. Sau đây ta xét cụ thể hơn tính chất khe và độ khe của hàm cực tiểu hoá:

Giả sử $J(x)$ có đạo hàm bậc 2 tại x ta có:

$$\nabla^2 J(x) = \left[\frac{\partial^2 J(x)}{\partial x_i \partial x_j} \right]_{n \times n} = \begin{pmatrix} \frac{\partial^2 J(x)}{\partial x_1 \partial x_1} & \dots & \frac{\partial^2 J(x)}{\partial x_1 \partial x_n} \\ \dots & \dots & \dots \\ \frac{\partial^2 J(x)}{\partial x_n \partial x_1} & \dots & \frac{\partial^2 J(x)}{\partial x_n \partial x_n} \end{pmatrix} \quad (1.1)$$

Ma trận này còn gọi là Hessian: $H(x) \equiv \nabla^2 J(x)$. Giả sử $H(x)$ xác định d-ơng và có các giá trị riêng sắp xếp theo thứ tự giảm dần: $\lambda_1(x) \geq \lambda_2(x) \geq \dots \geq \lambda_n(x) > 0$.

$$\text{Nếu xảy ra:} \quad \lambda_1(x) \gg \lambda_n(x) \quad (1.2)$$

thì $J(x)$ sẽ thay đổi chậm theo 1 hướng nhất định và thay đổi rất nhanh theo hướng vuông góc với nó. Khi đó các mặt mức của hàm số bị kéo dài theo hướng thay đổi chậm và vẽ ra trong mặt cắt hai chiều hình ảnh 1 khe suối hẹp nằm giữa 2 dãy núi song song. Từ đó mà có tên là hàm khe – hàm có tính khe rõ rệt.

Mức độ kéo dài các mặt mức quyết định độ khe của hàm mục tiêu. Để rõ hơn về độ khe, trước hết ta xét hàm bậc 2:

$$J(x) = 0,5 \langle x, Ax \rangle + \langle b, x \rangle + c, \quad x \in E^n. \quad (1.3)$$

Trong đó: A – ma trận xác định d-ơng, b – véc tơ hằng, c – hằng số.

Theo (1.1), Hessian của (1.3) là $H(x) = A$. Giả sử A có các giá trị riêng $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n > 0$. Độ -ớc định của hàm bậc 2 được xác định là: $\mu(A) = \lambda_1 / \lambda_n$.

Trên cơ sở đó ta có định nghĩa độ khe của hàm bậc 2 là:

$$\eta(A) = \frac{\max \lambda(A)}{\min \lambda(A)} - 1 = \frac{\lambda_1}{\lambda_n} - 1 = \mu(A) - 1 \quad (1.4)$$

Trong đó $\lambda(A)$ – giá trị riêng phụ thuộc vào ma trận A . Giá trị độ khe $\eta(A)$ càng lớn (độ - ổn định của bài toán càng xấu) thì các mặt mức của hàm mục tiêu càng bị kéo dài.

Đối với các hàm $J(x)$ phi bậc 2, độ - ổn định mang ý nghĩa cục bộ, đối với

$$x^* \in D_\delta: \quad \mu(x^*) = \lim_{\delta \rightarrow 0} \left(\sup_{x \in D_\delta} \|x - x^*\|^2 / \inf_{x \in D_\delta} \|x - x^*\|^2 \right) \quad (1.5)$$

Trong đó D_δ là miền hữu hạn. Rõ ràng $\mu(x^*) \geq 1$.

Theo (1.5) độ - ổn định đặc tr- ng cho sự kéo dài các mặt mức của hàm $J(x)$ tại lân cận x^* . Nếu $\mu(x^*)$ giảm dần tới 1, thì các mặt mức tiến gần mặt cầu. Nếu $\mu(x^*)$ càng tăng, thì các mặt mức càng bị kéo dài, hàm $J(x)$ càng thể hiện rõ tính khe, quá trình cực tiểu hoá $J(x)$ càng gặp khó khăn, độ - ổn định của bài toán càng xấu. Trong thực tế tính khe của các hàm mục tiêu thể hiện rất đa dạng.

Độ khe đặc tr- ng bởi độ dốc của vách khe, độ rộng và độ dốc của lòng khe. Lòng khe tạo bởi tập các điểm của miền khe, mà tại đó hàm mục tiêu giảm rất chậm theo mọi h- ớng.

Nói chung khe của hàm có thể thẳng hoặc uốn cong và kéo dài. Khe càng dài, càng cong, hoặc tại lòng khe hàm mục tiêu không khả vi liên tục (lòng khe gãy) thì bài toán tối - u hoá càng khó giải. Tính chất khe đó của các bài toán thực tế mang ý nghĩa toàn cục.

Sử dụng mạng nơron để nhận dạng đối tượng

Với các hệ thống có độ phi tuyến cao thì làm thế nào để nhận dạng đối tượng luôn là một câu hỏi đặt ra với chúng ta. Vì tính phi tuyến của các mạng nơron (hàm kích hoạt phi tuyến), chúng được dùng để mô tả các hệ thống phi tuyến phức tạp. Cybenko đã chứng minh rằng một hàm liên tục có thể xấp xỉ tùy ý bằng một mạng truyền thẳng với chỉ một lớp ẩn.

Như đã biết luyện mạng nơron có hai quá trình, quá trình ánh xạ và quá trình học. Học thực chất là quá trình lan truyền ngược.

Thuật học lan truyền ngược là thuật toán hay được sử dụng nhất trong quá trình luyện mạng nơron. Lan truyền ngược giảm dốc nhất (SDBP) cũng như LMS, nó cũng là một thuật toán xấp xỉ giảm dốc nhất cho việc cực tiểu trung bình bình phương sai số. Thật vậy, lan truyền ngược giảm dốc nhất là tương đương thuật toán LMS khi sử dụng trên mạng tuyến tính một lớp. Khi áp dụng với các mạng nhiều lớp thì lan truyền ngược giảm dốc nhất (SDBP) lại hoàn toàn khác, bởi trong các mạng nhiều lớp thì trung bình bình phương sai số liên quan đến mạng tuyến tính một lớp và các mạng phi tuyến nhiều lớp. Vậy, thực hiện kỹ thuật lan truyền ngược chính là giải bài toán tối ưu tĩnh với hàm mục tiêu là mặt sai số.

Hình dạng của mặt sai số phụ thuộc vào số lớp nơron và loại hàm kích hoạt. Trong khi mặt sai số với mạng tuyến tính một lớp có một cực tiểu đơn và độ dốc không đổi, mặt sai số với mạng nhiều lớp có thể có nhiều điểm cực tiểu cục bộ, có thể bị kéo dài, uốn cong tạo thành khe, trục khe và độ dốc có thể thay đổi ở một dải rộng trong các vùng khác nhau của không gian tham số.

Thực tế, việc chọn hàm kích hoạt như thế nào, chọn số lớp mạng nơron bằng bao nhiêu phụ thuộc vào đối tượng cần xấp xỉ. Như vậy, do độ phức tạp của đối tượng cần xấp xỉ khác nhau nên hàm mục tiêu rất khác nhau và dẫn đến quá trình học (giải bài toán tối ưu) có thể rất phức tạp.

Đặc biệt khi đối tượng cần xấp xỉ dẫn đến hàm mục tiêu có dạng lòng khe (ví dụ như đối tượng nhiệt) thì quá trình học rất khó khăn thậm chí không hội tụ nếu ta sử dụng các bộ công cụ có trong Toolbox của Matlab.

1.4. Mô phỏng quá trình luyện mạng nơron khi sử dụng Toolbox của Matlab

1.4.1. Ví dụ với mạng nơron có mặt lỗi bình thường

Xét hệ thống phi tuyến cần nhận dạng có mô hình toán học nh- sau:

$$f(u) = 0.6 \sin(\pi.u) + 0.3 \sin(3.\pi.u) + 0.1 \sin(5.\pi.u)$$

Tín hiệu vào: $u(k) = \sin(2\pi.k/250)$

Mạng nơ-ron đ-ợc dùng là mạng truyền thẳng 3 lớp có một đầu vào và một đầu ra. Các trọng số trong mạng nơ-ron đ-ợc điều chỉnh ở các khoảng $T_i=1$ sử dụng lan truyền ng-ợc tĩnh.

Chương trình

```
% Ch- ơng trình đ- ợc ghi trong file vd1. m – d:\work.

% -----Tạo các biến làm việc-----

% Các thời điểm lấy mẫu

k=0:1:500;

% Tín hiệu vào u(k) có dạng

u=sin(2*pi*k/250);

% Hàm f[u(k)] hay cũng là kết xuất đích của mạng

f=0.6*sin(u*pi)+0.3*sin(3*u*pi)+0.1*sin(5*u*pi);

pause

%-----Thiết kế mạng nơ-ron-----

% NEWFF – tạo một mạng nơ-ron truyền thẳng

% Tạo một mạng nơ-ron truyền thẳng có ba lớp:

% Lớp nhập 8 nơ-ron tansig, lớp ẩn 8 nơ-ron tansig, lớp ra có 1 nơ-ron tansig

% Giới hạn đầu vào nằm trong khoảng [-1 1]

net=newff([-1 1],[8 8 1],{'tansig' 'tansig' 'purelin'});

pause

%-----Luyện mạng nơ-ron-----

% TRAIN huấn luyện mạng nơ-ron

% Kĩ nguyên luyện mạng lớn nhất cho phép

net.trainparam.epochs=1000;

% Tần số hiển thị quá trình

net.trainparam.show=20;
```

% Sai số mục tiêu của mạng

```
net.trainparam.goal=0.00001;
```

% Luyện mạng với kết xuất đầu vào u, kết xuất đích là f

```
net=train(net,u,f);
```

pause

% Khi kết thúc quá trình luyện mạng ta dùng và đ- a mạng về thuộc tính của nó

% ADAPT cho phép mạng nơon tự thích nghi

```
[net,y,e]=adapt(net,u,f);
```

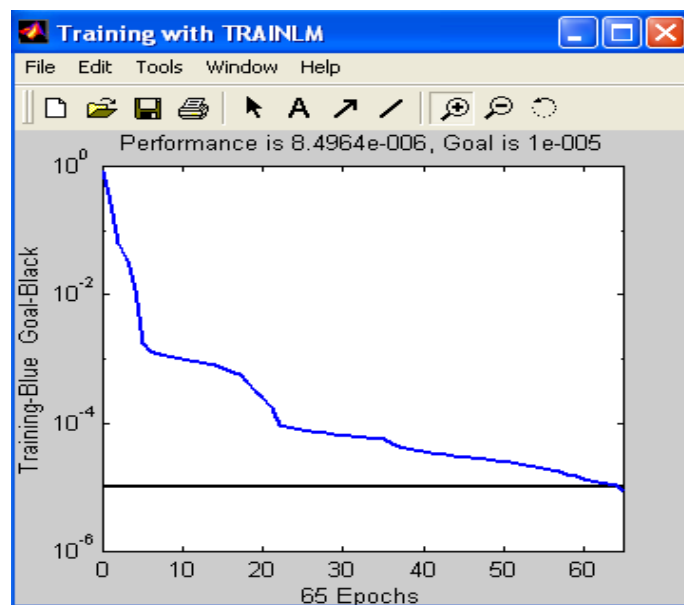
% -----Kết thúc ch- ơng trình -----

Kết quả mô phỏng

Sau khi khởi động phần mềm Matlab, tại dấu nhắc ở cửa sổ Matlab Comand window ta gõ: >> vd1 ↵.

Khi đó ch- ơng trình sẽ mô phỏng và cho kết quả nh- sau:

Hình 1.4 là kỉ nguyên luyện mạng, biểu thị các b- ớc tính trọng số của mạng. Mục tiêu sai số thực thiện đ- ợc sau 65 b- ớc tính.



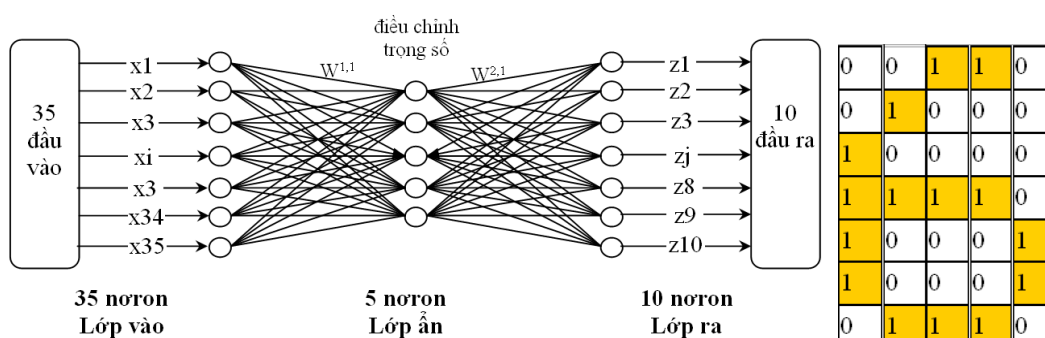
Hình 1.4: Kỉ nguyên luyện mạng ví dụ 1

1.4.2. Ví dụ với mạng nơron có mặt lỗi đặc biệt

Để minh họa, tác giả đề xuất cấu trúc mạng nơron để nhận dạng các chữ số: 0, 1, 2,...,9. Trong đó hàm sigmoid được sử dụng làm hàm kích hoạt. Ví dụ này sẽ theo chúng ta qua các chương của luận án. Cũng xin được nói, nhận dạng chữ số có thể không sinh ra hàm mục tiêu có dạng lòng khe. Nhưng ở đây, để thuận lợi cho quá trình minh họa, tác giả chọn ví dụ này và chọn hàm kích hoạt là hàm sigmoid với mục đích để sinh ra mặt sai số có dạng lòng khe [4].

Để biểu diễn các chữ số, chúng ta sử dụng một ma trận $5 \times 7 = 35$ để mã hóa cho mỗi ký tự. Tương ứng với mỗi vectơ đầu vào x là một vectơ có kích thước 35×1 , với các thành phần nhận các giá trị hoặc 0 hoặc 1. Như vậy, ta có thể lựa chọn lớp nơron đầu vào có 35 nơron. Để phân biệt được mười ký tự, chúng ta cho lớp đầu ra của mạng là 10 nơron. Đối với lớp ẩn ta chọn 5 nơron, ta được cấu trúc mạng như hình 1.5, trong đó:

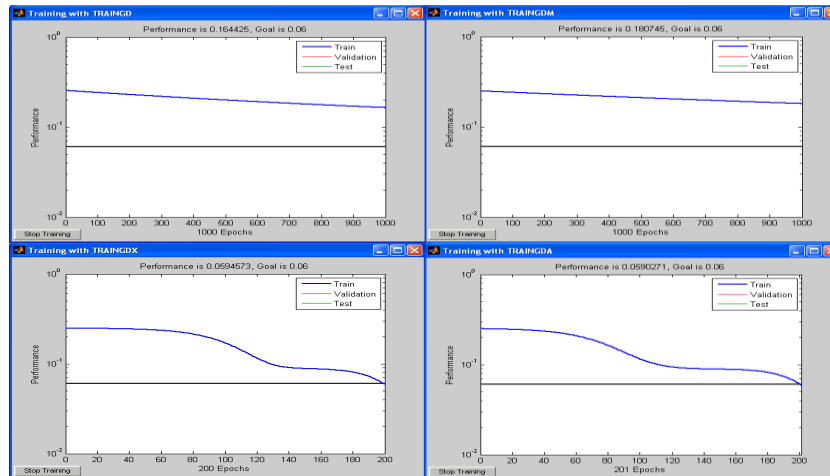
- Vectơ đầu vào x , kích thước 35×1
- Vectơ đầu ra lớp ẩn y , kích thước 5×1
- Vectơ đầu ra lớp ra z , kích thước 10×1
- Ma trận trọng số lớp ẩn: $W^{1,1}$, kích thước 35×5
- Ma trận trọng số lớp ra: $W^{2,1}$, kích thước 5×10



Hình 1.5: Cấu trúc mạng nơron cho nhận dạng chữ

Hàm f được chọn là hàm sigmoid vì thực tế hàm này cũng hay được dùng cho mạng nơron nhiều lớp và hơn nữa do đặc điểm của hàm sigmoid rất dễ sinh ra mặt sai số có dạng lòng khe hẹp. Phương trình của hàm sigmoid là: $f = 1 / (1 + \exp(-x))$

Hàm sai số sử dụng cho luyện mạng: $J = 0.5 * (z - t)^2$ với z là đầu ra của nơron lớp ra và t là giá trị đích mong muốn.



Hình 1.6: Các kết quả luyện mạng nơron với các phương pháp lan truyền ngược khác nhau (*traingd*, *traingdm*, *traindx*, *traingda*)

Hình 1.6 trình bày kết quả của quá trình luyện mạng cho bài toán nhận dạng chữ với các kỹ thuật lan truyền ngược sai số theo phương pháp Batch Gradient Descent (*traingd*), Batch Gradient Descent with Momentum (*traingdm*), Variable Learning Rate (*traingda*, *traindx*). Các phương pháp này đều được tích hợp trên Neural Network Toolbox của Matlab. Nhìn chung các phương pháp đều cho kết quả khá tốt, tuy nhiên để đạt được độ chính xác như mong muốn thì thời gian cần thiết cho luyện mạng là khá lớn. Thậm chí có trường hợp tín hiệu lỗi hầu như thay đổi rất ít qua các chu kỳ luyện mạng.

1.5. Tổng quan về tình hình nghiên cứu trong và ngoài nước

1.5.1. Điềm qua một số công trình nghiên cứu về mạng nơron và ứng dụng

Các nghiên cứu về bộ não con người đã được tiến hành từ hàng nghìn năm nay. Cùng với sự phát triển của khoa học kỹ thuật đặc biệt là những tiến bộ trong ngành điện tử hiện đại, việc con người bắt đầu nghiên cứu các nơron nhân tạo là hoàn toàn tự nhiên. Sự kiện đầu tiên đánh dấu sự ra đời của mạng nơron nhân tạo

diễn ra vào năm 1943 khi nhà thần kinh học Warren McCulloch và nhà toán học Walter Pitts viết bài báo mô tả cách thức các nơron hoạt động. Họ cũng đã tiến hành xây dựng một mạng nơron đơn giản bằng các mạch điện. Các nơron của họ được xem như là các thiết bị nhị phân với ngưỡng cố định. Kết quả của các mô hình này là các hàm logic đơn giản chẳng hạn như “a OR b” hay “a AND b”.

Tiếp bước các nghiên cứu này, năm 1949 Donald Hebb cho xuất bản cuốn sách *Organization of Behavior*. Cuốn sách đã chỉ ra rằng các nơron nhân tạo sẽ trở nên hiệu quả hơn sau mỗi lần chúng được sử dụng.

Những tiến bộ của máy tính đầu những năm 1950 giúp cho việc mô hình hóa các nguyên lý của những lý thuyết liên quan tới cách thức con người suy nghĩ đã trở thành hiện thực. Nathaniel Rochester sau nhiều năm làm việc tại các phòng thí nghiệm nghiên cứu của IBM đã có những nỗ lực đầu tiên để mô phỏng một mạng nơron. Trong thời kì này tính toán truyền thống đã đạt được những thành công rực rỡ trong khi đó những nghiên cứu về nơron còn ở giai đoạn sơ khai. Mặc dù vậy những người ủng hộ triết lý “thinking machines” (các máy biết suy nghĩ) vẫn tiếp tục bảo vệ cho lập trường của mình.

Năm 1956 dự án Dartmouth nghiên cứu về trí tuệ nhân tạo (Artificial Intelligence) đã mở ra thời kỳ phát triển mới cả trong lĩnh vực trí tuệ nhân tạo lẫn mạng nơron. Tác động tích cực của nó là thúc đẩy hơn nữa sự quan tâm của các nhà khoa học về trí tuệ nhân tạo và quá trình xử lý ở mức đơn giản của mạng nơron trong bộ não con người.

Những năm tiếp theo của dự án Dartmouth, John von Neumann đã đề xuất việc mô phỏng các nơron đơn giản bằng cách sử dụng role điện áp hoặc đèn chân không. Nhà sinh học chuyên nghiên cứu về nơron Frank Rosenblatt cũng bắt đầu nghiên cứu về *Perceptron*. Sau thời gian nghiên cứu này Perceptron đã được cài đặt trong phần cứng máy tính và được xem như là mạng nơron lâu đời nhất còn được sử dụng đến ngày nay. Perceptron một tầng rất hữu ích trong việc phân loại một tập các đầu vào có giá trị liên tục vào một trong hai lớp. Perceptron tính tổng có trọng số các đầu vào, rồi trừ tổng này cho một ngưỡng và cho ra một trong hai giá trị mong muốn có thể. Tuy nhiên Perceptron còn rất nhiều hạn chế, những hạn chế này

đã được chỉ ra trong cuốn sách về Perceptron của Marvin Minsky và Seymour Papert viết năm 1969.

Năm 1959, Bernard Widrow và Marcian Hoff thuộc trường đại học Stanford đã xây dựng mô hình ADALINE (ADaptive LINEar Elements) và MADALINE. (Multiple ADaptive LINEar Elements). Các mô hình này sử dụng quy tắc học *Least-Mean-Squares* (LMS: Tối thiểu bình phương trung bình). Thuật học Widrow – Hoff thuộc loại thuật học tối ưu hóa chất lượng mạng, nó cũng được xem như là tiền thân của thuật học lan truyền ngược.

Mạng ADALINE của họ rất giống với Perceptron, trừ hàm truyền là tuyến tính. Cả ADALINE và Perceptron cùng chịu một giới hạn như nhau, đó là các mạng của họ chỉ có thể giải các bài toán mà có thể phân ly tuyến tính.

Thuật toán LMS tìm thấy nhiều ứng dụng thực tế hơn luật học Perceptron. Điều này đặc biệt đúng trong lĩnh vực của xử lý tín hiệu số. Ví dụ, hầu hết các đường điện thoại dài sử dụng các mạng ADALINE cho việc loại nhiễu.

Huấn luyện theo phương pháp Widrow-Hoff là một thuật toán xấp xỉ giảm dốc nhất, trong đó hàm mục tiêu, hay còn gọi là hàm chất lượng, là bình phương trung bình sai số. Thuật toán này quan trọng bởi hai lý do. Thứ nhất, nó được sử dụng rộng rãi trong các ứng dụng xử lý tín hiệu số. Thứ hai, nó giúp ta đến với kỹ thuật lan truyền ngược cho các mạng nhiều lớp nói chung một cách dễ dàng hơn.

Mặc dù thuật toán LMS thành công trong việc xử lý tín hiệu nhưng lại thiếu thành công trong việc thích nghi cho các mạng nhiều lớp. Widrow đã dừng làm việc với các mạng nơron trong những năm 1960 và bắt đầu dành hết thời gian làm việc với xử lý tín hiệu thích nghi, ông trở lại với mạng nơron trong những năm 1980 với việc sử dụng mạng nơron trong điều khiển thích nghi.

Luật học Perceptron của Frank Rosenblatt và thuật toán LMS của Bernard Widrow và Marcian Hoff đã được thiết kế để huấn luyện các mạng giống như Perceptron một lớp. Các mạng đơn lớp chịu một sự bất lợi rằng chúng chỉ có thể giải quyết các lớp bài toán có thể phân ly tuyến tính.

Những năm 60 một Viện sĩ thuộc Viện Hàn Lâm Nga được coi là người tiên phong trong việc nhận dạng hệ thống trên cơ sở sử dụng mạng nơron. Lý thuyết này được công bố trong Я. 3. Цыпкин, Адаптация и обучение в автоматических системах, Главная редакция физико-математической литературы изд-ва «Наука», М., 1968, 400 стр.

Trong luận văn của Paul Werbos năm 1974, đã trình bày thuật toán trong ngữ cảnh của các mạng nói chung, với mạng nơron như là một trường hợp đặc biệt. Cho đến tận những năm 1980, kỹ thuật lan truyền ngược mới được nghiên cứu lại và mở rộng một cách độc lập bởi David Rumelhart, Geoffrey Hinton và Ronald Williams; David Parker, và Yanm Le Cun. Thuật toán đã được phổ biến hóa bởi cuốn sách Parallel Distributed Processing của nhóm tác giả David Rumelhard và James McClelland. Việc phổ biến của cuốn sách này kích lệ một dòng thác của việc nghiên cứu về mạng nơron Perceptron nhiều lớp, được huấn luyện bởi kỹ thuật lan truyền ngược, mà hiện nay được sử dụng rộng rãi trong mạng nơron.

Năm 1982 trong bài báo gửi tới viện khoa học quốc gia, John Hopfield bằng sự phân tích toán học rõ ràng, mạch lạc, ông đã chỉ ra cách thức các mạng nơron làm việc và những công việc chúng có thể thực hiện được. Công hiến của Hopfield không chỉ ở giá trị của những nghiên cứu khoa học mà còn ở sự thúc đẩy trở lại các nghiên cứu về mạng nơron.

Cũng trong thời gian này, một hội nghị với sự tham gia của Hoa Kỳ và Nhật Bản bàn về việc hợp tác/cạnh tranh trong lĩnh vực mạng nơron đã được tổ chức tại Kyoto, Nhật Bản. Sau hội nghị, Nhật Bản đã công bố những nỗ lực của họ trong việc tạo ra máy tính thế hệ thứ 5. Tiếp nhận điều đó, các tạp chí định kỳ của Hoa Kỳ bày tỏ sự lo lắng rằng nước nhà có thể bị tụt hậu trong lĩnh vực này. Vì thế, ngay sau đó, Hoa Kỳ nhanh chóng huy động quỹ tài trợ cho các nghiên cứu và ứng dụng mạng nơron.

Năm 1985, viện vật lý Hoa Kỳ bắt đầu tổ chức các cuộc họp hàng năm về mạng nơron ứng dụng trong tin học (Neural Networks for Computing).

Năm 1987, hội thảo quốc tế đầu tiên về mạng nơron của Viện các kỹ sư điện và điện tử IEEE (Institute of Electrical and Electronic Engineer) đã thu hút hơn 1800 người tham gia.

Trong những thập niên 1980, 1990 các thuật học phát triển cho mạng nơron động LDDN (Layered Digital Dynamic Network) trên cơ sở lan truyền ngược, đó là lan truyền ngược xuyên thời gian BPTT (BackRropagation-Through-Time) và thuật học hồi qui thời gian thực RTRL (Real_Time Recurrent Learning) dùng cho LDDN.

Ở Việt Nam, từ những năm 90, cũng đã có rất nhiều nhà khoa học quan tâm đến lý thuyết về mạng nơron và những ứng dụng của nó trong nhiều lĩnh vực khác nhau. Tiên phong trong việc đưa kiến thức về mạng nơron phổ biến đến độc giả là quyển sách *“Trí tuệ nhân tạo, Mạng nơron phương pháp và ứng dụng”* của Nguyễn Đình Thúc, NXB Giáo dục năm 2000. Tiếp đó phải kể đến quyển *“Hệ mờ, mạng nơron và ứng dụng”* của Bùi Công Cường, Nguyễn Doãn Phước, NXB Khoa học và Kỹ thuật Hà nội, 2001; quyển *“Hệ mờ & nơron trong kỹ thuật điều khiển”* của Nguyễn Như Hiền, Lại Khắc Lãi, NXB Khoa học tự nhiên và công nghệ. Còn những công trình nghiên cứu về mạng nơron có thể kể đến như:

- ❖ Nguyễn Kỳ Phùng, Nguyễn Khoa Việt Trường, *“Mô hình hoá các quá trình xử lý nước thải bằng mạng nơron nhân tạo”*, Trường Đại học Khoa học Tự nhiên Đại học Quốc gia Thành phố Hồ Chí Minh.
 - Đối tượng là mô hình bể xử lý nước thải, các tác giả đã xây dựng mô hình, tối ưu hoá quá trình luyện mạng và đã kiểm chứng kết quả với sai số nhỏ.
 - Đã xây dựng được chương trình ứng dụng mạng nơron cho dự báo chất lượng đầu ra của hệ thống xử lý nước thải. Cùng với thuật toán tối ưu hoá mạng nơron khi cho số nút ẩn thay đổi để tìm ra cấu trúc mạng tối ưu nhất. Chương trình đã thể hiện rõ ưu việt so với chương trình mạng nơron của Matlab.
 - Thuật toán tối ưu hoá quá trình luyện mạng là một bước cải tiến so với các chương trình ứng dụng mạng nơron thông thường, chẳng hạn như

Matlab. Với quá trình lặp lại nhiều lần và ghi nhận những mạng cho kết quả tốt nhất sau mỗi lần lặp, ta có thể chọn được mạng cho kết quả tốt hơn và sai số ổn định hơn.

- ❖ Đỗ Trung Hải (2008) “*Ứng dụng lý thuyết mờ và mạng nơron để điều khiển hệ chuyển động*”, Luận án tiến sĩ, Trường Đại học Bách khoa Hà Nội.
 - Nghiên cứu và đề xuất cấu trúc hệ mờ - nơron với số lớp và số nơron thích ứng (5 lớp và số nơron lớp 2 tối thiểu là 2 nơron) nhằm đảm bảo độ chính xác và tốc độ tính toán cho hệ điều khiển thời gian thực.
 - Xây dựng thuật toán nhận dạng trực tuyến, cập nhật thích nghi thông số nhằm đảm bảo tối thiểu hoá sai lệch phục vụ cho việc nhận dạng và điều khiển hệ. Việc ứng dụng đại số Lie và điều khiển theo phương pháp tuyến tính hoá chính xác thích nghi có khả năng ứng dụng tổng quát cho một lớp hệ điều khiển chuyển động.
 - Với hệ chuyển động cụ thể và phức tạp là hệ khớp nối mềm công trình đã đưa ra thuật toán mô phỏng hệ. Các kết quả mô phỏng đã chứng tỏ tính đúng đắn của luật nhận dạng và điều khiển, cấu trúc cũng như mô hình điều khiển hệ chuyển động.

1.5.2. Các công trình trong và ngoài nước nghiên cứu về thuật toán học của mạng nơron

Những năm gần đây, những biến thể của thuật học lan truyền ngược vẫn được quan tâm nghiên cứu và được công bố nhằm nâng cao tốc độ hội tụ của quá trình luyện mạng.

Kỹ thuật lan truyền ngược ở đây là lan truyền ngược lỗi (hay sai số) trong mạng, hàm lỗi (hay hàm sai số) thường chọn là hàm mà nó tối thiểu hoá được sai số trung bình bình phương. Chính vì vậy, trong quá trình nỗ lực thoát khỏi các cực tiểu yếu, cực tiểu cục bộ và những mong muốn giảm thời gian thực hiện của máy tính khi tìm kiếm nghiệm tối ưu, thì vấn đề nghiên cứu đặc điểm của các mặt lỗi thường được chọn làm xuất phát điểm cho việc cải tiến hay đề xuất các thuật học mới. Trong các nghiên cứu nhằm cải thiện thuật toán, người ta thường tìm cách thay đổi

bước học để cho phép có thể vượt qua những cực trị địa phương. Không có một giá trị bước học xác định nào cho các bài toán khác nhau. Với mỗi bài toán, bước học thường được lựa chọn bằng thực nghiệm theo phương pháp thử và sai, hoặc sẽ có bước học phù hợp với từng dạng bài toán riêng biệt.

Sau đây là một số các công trình khoa học quan trọng, đề cập đến vấn đề cải tiến kỹ thuật lan truyền ngược nguyên thủy ảnh hưởng đến nội dung của luận án.

- ❖ Các công trình nghiên cứu của Hagan, M.T., và các cộng sự về mạng nơron, tiêu biểu là cuốn “Neural Networks Design”, PWS Publishing Company, Boston, 1996.
- ❖ Công trình của Kandil N., Khorasani K., Patel R.V., Sood V.K., “Optimum learning rate for backpropagation neural networks”, Canadian Conference on Electrical and Computer Engineering, pp: 465-468 vol.1, 1993. Bài báo này đã đưa ra thời gian thay đổi tốc độ học tập tối ưu cho các mạng BP. Kết quả cho thấy thời gian huấn luyện có thể giảm đáng kể trong khi không gây ra bất kỳ dao động trong quá trình huấn luyện đó.
- ❖ Công trình của Dimitri P. Bertsekas, “Nonlinear programming, 2nd Edition, 2004”. Bertsekas nói rất tỉ mỉ về các vấn đề cụ thể ta thường phải đối mặt khi sử dụng một thuật toán tối ưu. Ví dụ, nếu dùng giảm dốc gradient thì cần phải tính đến chuyện điều khiển cập nhật bước nhảy như thế nào,... Trong quyển sách này mô tả khá đầy đủ các phương pháp cổ điển khác như conjugate gradient, golden section,...
- ❖ Mohammed A. Otair, Woalid A. Salamed “Speeding Up BackPropagation Neural Networks”, Jordan University of Science and Technology, Flagstaff, Arizona, USA – June 16-19,2005. Bài báo này trình bày về việc cải tiến thuật học lan truyền ngược, thuật học của họ có tên là OPB, Optical BackPropagation. Nội dung chính của thuật học là việc sử dụng hàm e mũ cho việc tính toán sai số đầu ra của mạng, “sai số đầu ra” = $(1 + \exp(\text{“giá trị mong muốn”} - \text{“giá trị hiện thời”}))$ nếu “giá trị mong muốn”-“giá trị hiện thời” không âm và “sai số đầu ra” = $-(1 + \exp(\text{“giá trị mong muốn”} - \text{“giá trị hiện thời”}))$ nếu “giá trị mong muốn”-“giá trị hiện thời” âm; thay vì việc tính sai số đầu ra của mạng như truyền thống “sai

số đầu ra” = “giá trị mong muốn”-“giá trị hiện thời”. Và họ chứng minh rằng với việc tính toán sai số như vậy, tốc độ hội tụ sẽ nhanh hơn.

- ❖ Công trình của Chi-Chung Cheung, Sin-Chun Ng, “*The multi-phase method in fast learning algorithms*”, International Joint Conference on Neural Networks (IJCNN) 2009, pp: 552-559. Bài báo này đưa ra phương pháp cải tiến kỹ thuật lan truyền ngược BP with two-phase magnified gradient function (2P-MGFPROP). Cụ thể phương pháp 2P-MGFPROP được tăng cường bằng cách phân chia quá trình học của mạng ra nhiều giai đoạn, với mỗi giai đoạn thích nghi khác nhau sẽ chọn một thuật toán học khác nhau. Các kết quả thực nghiệm cho thấy, tốc độ hội tụ nhanh hơn gấp 2 lần so với các thuật toán học nhanh hiện có.
- ❖ Công trình của Islam, M.; Rana, M.R.; Ahmed, S.U.; Enamul Kabir, A.N.M.; Shahjahan, M. “Training neural network with chaotic learning rate” International Conference on Emerging Trends in Electrical and Computer Technology (ICETECT), pp: 781 – 785, 23-24 March 2011. Các tác giả đề cập đến việc thay đổi bước học một cách hỗn loạn “chaotic learning rate” trong quá trình cập nhật trọng số. Đề xuất này đã được kiểm nghiệm qua 6 bài toán trong bộ dữ liệu dùng để phân loại như ung thư vú, tiểu đường, bệnh tim, thẻ tín dụng Úc, ngựa và thủy tinh. Phương pháp mới này nhanh hơn so với BP về khả năng khái quát và cũng như tốc độ hội tụ.
- ❖ Công trình nghiên cứu của PGS.TS. Nguyễn Quang Hoan “*Nhận dạng ký tự viết tay tiếng Việt sử dụng mạng lan truyền ngược*” đăng trên báo cáo của hội nghị Tự động hóa toàn quốc lần thứ 6 (VICA6), 2004. Bài viết nghiên cứu việc kết hợp mạng nơron động với giải thuật di truyền cho nhận dạng âm tiết tiếng Việt. Để thực hiện, tác giả đã sử dụng mạng nơron động với các phần tử trễ và thuật học lan truyền ngược lỗi. Trong đó, giải thuật di truyền đóng vai trò tối ưu các trọng số cho mạng nơron nhằm tăng hiệu quả nhận dạng.
- ❖ Công trình nghiên cứu của Nguyễn Sĩ Dũng, Lê Hoài Quốc, “Một số thuật toán về huấn luyện mạng nơron network trên cơ sở phương pháp conjugate Gradient”, Đại học Công nghiệp TP HCM và Đại học Bách khoa TP HCM.

- Tác giả đã tìm đi hướng đi mới đầy triển vọng là xây dựng thuật toán mới về luyện mạng dựa vào phương pháp Conjugate Gradient, trong đó đặt mục tiêu là cải thiện tốc độ hội tụ của quá trình huấn luyện mạng nơron.
- Trong báo cáo này đã trình bày cơ sở toán học của vấn đề của phương pháp Conjugate Gradient và một thuật toán mới được viết trên Matlab 7.1 để huấn luyện mạng nơron. Xong đối tượng mà tác giả áp dụng là đối tượng phi tuyến tính.
- Phương pháp này có ý nghĩa trong huấn luyện mạng trực tuyến online và ứng dụng nhận dạng và điều khiển trong môi trường động.

Trong quá trình luyện mạng nơron, một nhân tố khác cũng tác động rất lớn đến vấn đề tìm nghiệm tối ưu đó là bộ trọng số khởi tạo ban đầu. Trong kỹ thuật lan truyền ngược nguyên thủy và các thuật toán khác, bộ trọng số ban đầu dùng cho luyện mạng đều được chọn ngẫu nhiên có thể thủ công hay tự động trong một khoảng nào đó. Đã có những công trình chứng minh được rằng, thay vì bằng cách khởi tạo ngẫu nhiên hãy tìm bộ trọng số khởi tạo tối ưu cho quá trình luyện mạng. Trọng số khởi tạo đã được công nhận rộng rãi là một trong những phương pháp tiếp cận hiệu quả trong việc thúc đẩy công tác đào tạo mạng nơron [31-43]. Tiêu biểu có thể kể đến như: Nghiên cứu của Shepanski liên quan đến việc đào tạo một mạng truyền thẳng nhiều lớp [38]. Trọng lượng tối ưu được xác định bằng cách sử dụng phương pháp tính bình phương nhỏ nhất dựa trên một ma trận tiêu chuẩn. Đối với mạng một lớp ẩn, ông đề nghị sử dụng một trong hai phương pháp mô phỏng mềm dẻo hoặc giải thuật di truyền để khởi tạo các trọng số liên kết giữa đầu vào và lớp ẩn, sau đó trọng lượng đầu ra được tính toán sử dụng phân tách giá trị. Đây là một phương pháp tính toán rất phức tạp. Còn Yam và Chow đề xuất hai phương pháp khởi tạo trọng lượng dựa trên phương pháp bình phương nhỏ nhất [40][41]. Trong [40], hệ thống được giả định là tuyến tính. Các thông số đầu vào và đầu ra thu được bằng phương pháp bình phương nhỏ nhất tuyến tính. Từ các thông số này, trọng lượng ban đầu tối ưu giữa các lớp được xác định. Tuy nhiên, thuật toán này không áp dụng cho mạng nơron trong đó số nơron lớp ẩn là nhỏ hơn số nơron trong các lớp trước đó cộng với một. Trong [41], các kết quả đầu ra của lớp ẩn được gán giá

trị trong vùng không bão hòa và trọng lượng ban đầu tối ưu giữa đầu vào và lớp ẩn được đánh giá bằng phương pháp đại số tuyến tính. Tuy nhiên, một số điều tra khác chỉ ra rằng, mạng nơron khởi tạo với [41] đôi khi thu được một tối ưu địa phương xấu, xác suất bị mắc kẹt cho một số ứng dụng có thể lên đến 10%.

Năm 1990, Nguyen và Widrow thúc đẩy quá trình đào tạo của mạng nơron bằng cách thiết lập trọng lượng ban đầu của lớp ẩn [36]; vì vậy, mỗi nút ẩn được gán thêm một loạt các chức năng mong muốn khi bắt đầu luyện mạng. Thông qua xấp hàm kích hoạt với các phân đoạn tuyến tính, trọng lượng được đánh giá. Tiếp theo, các ngưỡng của mạng được lựa chọn bằng cách giả định các biến đầu vào biến thiên từ -1 đến 1. Osowski mở rộng ý tưởng của Nguyen và Widrow. Ông đề nghị một phương pháp để xác định số lượng nơron lớp ẩn và sử dụng thông tin đầu ra mong muốn $y=f(x)$ để xác định trọng lượng ban đầu [37]. Trong ví dụ được Osowski đưa ra, trọng lượng tối ưu thu được sau khi luyện mạng nơron bằng thuật toán BP rất gần với trọng lượng ban đầu được đề xuất bởi thuật toán mới của ông.

Hisashi Shimodaira đề xuất một phương pháp gọi là thiết lập giá trị ban đầu tối ưu (Optimal initial value setting-OIVS) để xác định sự phân bố các giá trị ban đầu của trọng số và chiều dài của vectơ trọng [39]. Còn Drago và Ridella đề xuất phương pháp gọi là SCAWI (Statistically controlled activation weight initialization) để tìm các trọng số ban đầu tối ưu [33]. Họ xác định độ lớn tối đa của trọng lượng thông qua phân tích thống kê. Có thể nói việc ảnh hưởng của bộ trọng số ban đầu đến kết quả luyện mạng nơron là không phải bàn cãi; tuy nhiên, mới mỗi một mạng khác nhau, lại có những phương pháp khác nhau phù hợp. Những công trình nghiên cứu gần đây có ảnh hưởng đến nội dung của luận án:

- ❖ Công trình của Y. F. Yam, T. W. S. Chow, “*Determining initial weights of feedforward neural networks based on least squares method*”, Neural Processing Letters, Vol 2, Issue 2, pp:13-17, 1995. Bài báo đưa ra một thuật toán tối ưu hóa trọng số ban đầu của mạng truyền thẳng dựa trên phương pháp đại số tuyến tính. Với việc sử dụng phương pháp này, lỗi mạng ban đầu là rất nhỏ. Sau đó ta có thể tiếp tục sử dụng kỹ thuật lan truyền ngược để đi đến điểm cực trị.

- ❖ Công trình của Jatinder N.D.Gupta, Randall S. Sexton, “Comparing backpropagation with a genetic algorithm for neural network training”, The International Journal of Management Science, Omega 27, pp: 679-684, 1999. Họ lần đầu tiên đã sử dụng GA để tìm kiếm vectơ trọng số của mạng nơron nhân tạo. Họ so sánh lan truyền ngược với GA và kết quả mỗi giải pháp có nguồn gốc từ GA là vượt trội so với các giải pháp lan truyền ngược tương ứng. GA có thể được sử dụng để tối ưu hóa một số yếu tố của quá trình thiết kế và huấn luyện mạng bao gồm lựa chọn tập hợp tính năng, tối ưu hóa cấu trúc mạng, học tập tối ưu hóa tham số.
- ❖ Công trình nghiên cứu của Gleb Beliakov and Ajith Abraham “*Global Optimisation of Neural Networks Using a Deterministic Hybrid Approach*” đăng trên Hybrid Information Systems, Abraham A. and Koeppen M. (Eds), Physica-Verlag Germany, pp 79-92, 2002. Bài báo đề xuất sử dụng phương pháp “cutting angle” nhằm tối ưu hóa trọng số của mạng. Mạng nơron lần đầu được học theo phương pháp “cutting angle”, sau đó được học theo các kỹ thuật tối ưu khác.
- ❖ Công trình nghiên cứu của P. A. Castillo, M. G. Arenas, J. J. Merelo, G. Romero, F. Rateb, A. Prieto, “Comparing Hybrid Systems to Design and Optimize Artificial Neural Networks”, Genetic Programming Lecture Notes in Computer Science Vol 3003, 2004, pp 240-249. Trong bài báo này các tác giả đã nghiên cứu so sánh giữa các phép lai để tối ưu hóa các perceptron nhiều lớp và đưa ra một mô hình tối ưu hóa cấu trúc và trọng số ban đầu của mạng perceptron nhiều lớp. Kết quả thu được cho thấy mô hình này cần ít chu kỳ huấn luyện hơn nhiều và do đó tăng tốc độ hội tụ.
- ❖ Công trình nghiên cứu của D. Shanthi , G. Sahoo and N. Saravanan, “Evolving Connection Weights of Artificial Neural Networks Using Genetic Algorithm with Application to the Prediction of Stroke Disease”, International Journal of Soft Computing, Vol 4, Issue 2, pp: 95-102, 2009. Bài báo này đề xuất việc kết hợp giải thuật di truyền GA và mạng nơron nhân tạo để tối ưu hóa bộ trọng số ban đầu trong quá trình luyện mạng nơron. Nghiên cứu này được ứng dụng trong việc dự đoán bệnh đột quỵ.

- ❖ Công trình nghiên cứu của Yu-Tzu Chang, Jinn Lin, Jiann-Shing Shieh, Maysam F. Abbod “Optimization the InitialWeights of Artificial Neural Networks via Genetic Algorithm Applied to Hip Bone Fracture Prediction” đăng trên tạp chí Advances in Fuzzy Systems - Special issue on Hybrid Biomedical Intelligent Systems, Vol 2012, January 2012 Article No. 6, New York, NY, United States. Bài báo tìm cách thiết lập trọng số tối ưu ban đầu để nâng cao độ chính xác của mạng nơron bằng giải thuật di truyền trong dự báo xác suất gãy xương hông.

1.5.3. Bàn luận

Như đã nói ở trên, trong quá trình luyện mạng nơron, hàm sai số thường chọn là hàm mà nó tối thiểu hoá được sai số trung bình bình phương. Vậy đây chính là một bài toán tối ưu hóa.

Trong các công trình nghiên cứu về tối ưu hóa, GS.TSKH. Nguyễn Văn Mạnh đã tìm ra được nhiều bài toán tối ưu dẫn đến mặt sai số có dạng lòng khe. Trong quá trình học tập nghiên cứu ở nước ngoài, ông đã đề xuất ra một thuật học mới đó là thuật toán vượt khe. Thuật toán này tỏ ra rất hữu hiệu để có thể vượt qua được “lòng khe”, tìm ra điểm tối ưu, tránh rơi vào cực tiểu yếu, cực tiểu cục bộ. Công trình tiêu biểu, có thể coi như là một đóng góp to lớn cho ngành toán học Việt Nam trước quốc tế đó là:

- ❖ Công trình của N.V. Mạnh nghiên cứu tại MEI, Viện nghiên cứu năng lượng Matxcova, “Optimization of Multiply Connected Control Systems Using Methods of Nonlinear Programming./ Thermal Engineering, ISSN: 0040-6015. 1998, vol. 45. No 10, pp. 829-834”, ông sử dụng thuật toán vượt khe kết hợp với phương pháp chiếu affine để giải quyết bài toán tối ưu nhiệt.
- ❖ Nguyen Van Manh and Bui Minh Tri, “Method of “cleft-overstep” by perpendicular direction for solving the unconstrained nonlinear optimization problem”, Acta Mathematica Vietnamica, vol. 15, N02, 1990.
- ❖ Nguyen Van Manh, On “r-Algorithm” for Minimizing “Ravine” Function /Proceedings of NCST of Vietnam, 1993. No2, pp. 53-60.

- ❖ Manh N.V. “Application of the 'Cleft-Over-Step' method of optimization for identifying the transfer function of controlled plants” Thermal engineering, ISSN: 0040-6015, 1995, vol. 42, No6, pp. 510-518.
- ❖ Manh N.V. “Optimization of the settings of robust controllers by means of the cleft-over-step algorithm of nonlinear minimization” Thermal engineering, ISSN: 0040-6015, Y. 1995, vol. 42, No. 10, pages 845-853.

Trong quá trình luyện mạng nơron, với mặt sai số có dạng lòng khe đã được giới thiệu trong *hình 1.3*, chưa có một công trình nào được công bố trên các tạp chí hội thảo trong và ngoài nước đề cập đến việc làm thế nào hiệu quả nhất để tìm kiếm được nghiệm tối ưu hay tăng tốc độ hội tụ với mặt sai số dạng này.

Dựa trên những nghiên cứu đã có về mạng nơron, về toán tối ưu ở trong và ngoài nước; tác giả sẽ xây dựng một thuật toán luyện mạng nơron cho mặt lỗi đặc biệt có dạng lòng khe mà trong đó sẽ ứng dụng thuật toán vượt khe để cập nhật bước học, ứng dụng giải thuật di truyền để tạo bộ trọng số khởi tạo ban đầu.

1.6. Kết luận chương 1

Trong chương 1, từ việc phân tích các nhân tố trong quá trình học của mạng nơron, tác giả nhận thấy rằng, kết quả luyện mạng nơron phụ thuộc rất lớn vào giá trị ban đầu của vec-tơ trọng số và bước học. Việc mạng sẽ hội tụ đến điểm tối ưu toàn cục hay không nhiều khi còn phụ thuộc vào sự may mắn do việc chọn giá trị khởi tạo là ngẫu nhiên. Thêm nữa, việc lựa chọn bước học sẽ bằng bao nhiêu để có thể hội tụ hay ít nhất là tăng tốc độ hội tụ là một câu hỏi cũng được đặt ra, đặc biệt khi mặt lỗi có dạng đặc biệt. Để minh chứng cho điều đó tác giả đã đưa ra 2 ví dụ: Ở ví dụ 1, khi mặt lỗi dạng bình thường, sử dụng bộ công cụ trong Toolbox của Matlab để luyện mạng, mạng đã luyện thành công chỉ sau 65 bước tính. Nhưng đến ví dụ thứ 2 về nhận dạng chữ viết tay thì thời gian luyện mạng lâu hơn rất nhiều, thậm chí tín hiệu lỗi còn thay đổi rất ít qua các chu kỳ luyện mạng.

Để giải quyết vấn đề này, cần thiết phải tìm ra một thuật toán hiệu chỉnh các bước học nhằm rút ngắn thời gian hội tụ của mạng đồng thời cũng tránh được vấn đề cực trị địa phương.

Trong chương 2, dựa vào những nhận xét đánh giá của chương 1, tác giả sẽ giới thiệu về thuật toán vượt khe và đề xuất việc áp dụng thuật toán vượt khe trong quá trình luyện mạng nơron.

CHƯƠNG 2: THUẬT TOÁN VƯỢT KHE TRONG QUÁ TRÌNH LUYỆN MẠNG NƠRON

Tóm tắt: Kỹ thuật lan truyền ngược là một phát minh chính trong nghiên cứu về mạng nơron. Tuy nhiên, thuật toán nguyên thủy thì quá chậm đối với hầu hết các ứng dụng thực tế; đặc biệt với các bài toán có mặt sai số dạng khe thì ngay cả với các biến thể như gradient liên hợp hay qui tắc mô-men cũng tỏ ra yếu. Trong chương này sẽ trình bày một thuật toán để tính bước học theo nguyên lý vượt khe, thuật toán vượt khe, nhằm cải tiến tốc độ hội tụ của quá trình tìm kiếm nghiệm tối ưu và vấn đề cài đặt thuật toán này kết hợp với kỹ thuật lan truyền ngược trong bài toán luyện mạng nơron.

2.1. Thuật toán vượt khe

2.2. Ứng dụng thuật toán vượt khe trong quá trình luyện mạng nơron

2.3. Minh họa thuật toán

2.4. Kết luận chương 2

2.1. Thuật toán vượt khe

2.1.1. Đặt vấn đề

N.Z.Shor giả thiết dùng toán tử kéo giãn không gian để làm tăng tốc độ hội tụ của thuật toán gradient trong việc giải bài toán tối ưu hóa không điều kiện:[5]

$$\text{Min } J(u) \quad u \in E^n$$

u là vec-tơ trong không gian Euclide n chiều.

Phương pháp “r-Algorithm” tương ứng với công thức lặp cho bởi bước thứ k như sau:

$$u^k = u^{k-1} + \alpha_k s^{k-1}, \quad k = 1, 2, \dots$$

$$s^{k-1} = -B_k B_k^T J'(u^{k-1})$$

$$s^{k-1}, u^{k-1} \in E^n$$

Trong đó u là vec-tơ biến của hàm mục tiêu $J(u)$ tại bước lặp thứ k ; α_k là độ dài bước của hàm theo hướng chuyển động s^{k-1} ; chỉ số trên T ký hiệu chuyển vị của ma trận hoặc vec-tơ, $J'(u^{k-1})$ là gradient của hàm tại điểm u^{k-1} . B^k là ma trận biến đổi không gian, được xác định dựa trên toán tử kéo giãn không gian như sau:

$$B_k = B_{k-1}R_k, \quad B_0 = I \text{ ma trận đơn vị}$$

$$R_k = I + \left(\frac{1}{\rho_k} - 1 \right) \frac{r_k r_k^T}{r_k^T r_k}$$

$$r_k = B_{k-1}^T [J'(u^{k-1}) - J'(u^{k-2})]$$

Trong đó R_k là toán tử kéo giãn không gian theo hướng r_k với hệ số $0 < \rho < 1$.

Hướng chuyển động s^{k-1} là hoàn toàn xác định bởi (2) ÷ (6) tại mỗi bước lặp k . Một quan tâm chính của chúng ta trong phần này là phương pháp xác định độ dài bước α_k , nó tạo ra sự hội tụ hiệu quả và được dùng quy ước.

Một vài công thức xác định giá trị α_k như sau:

$$\alpha_k = \arg \min J(u^{k-1} + \alpha s^{k-1})$$

$$\alpha = \frac{a}{b+k} \quad a, b = \text{const} > 0$$

$$(3.3)$$

$$\alpha_k = \alpha_{k-1} q, \quad 0 < q < 1$$

Các công thức xác định độ dài bước α_k được các tác giả đưa ra khác với mỗi lớp hàm mục tiêu và thường có nhiều tham số đòi hỏi phải chọn giá trị phù hợp cho mỗi lớp hàm cụ thể. Điều này không dễ dàng và là nguyên nhân cản trở trong các ứng dụng. Vì vậy, hiệu quả thực của “r-Algorithm” bị giới hạn, lúc này, sự lồi cuốn của nó bị thu nhỏ lại.

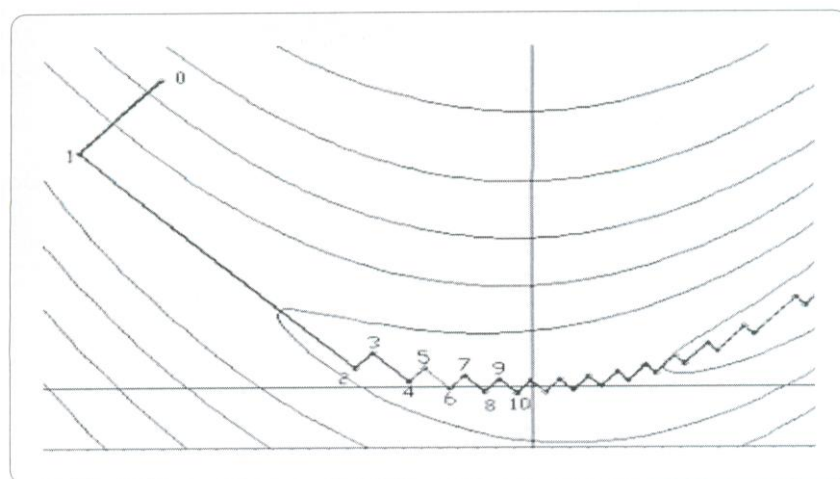
2.1.2. Tính hội tụ và điều kiện tối ưu

a. Tính hội tụ

Chúng ta biết rằng tốc độ học cực đại mà có thể ổn định đối với thuật toán giảm dốc nhất thì được chia hai bởi giá trị riêng cực đại của ma trận Hessian, đối

với hàm mục tiêu dạng toàn phương. Hình dạng của hàm mục tiêu có thể là rất khác nhau trong các vùng khác nhau của không gian tham số. Với các hàm khe thì độ cong sẽ rất khác nhau theo các hướng khác nhau, do đó việc xác định bước học là không thuận lợi.

Chúng ta đã nghiên cứu về hàm mục tiêu, cụ thể là nghiên cứu về hàm mục tiêu có dạng lòng khe ở mục 1.3 chương 1. Nguyên nhân của việc hội tụ chậm là việc thay đổi độ dốc của mặt trên đường đi của quỹ đạo. Hai bên khe rãnh rất dốc tuy nhiên đáy của nó thì lại hầu như bằng phẳng, quỹ đạo sẽ vượt qua mặt lồi rất nhanh cho đến khi nó rơi vào thung lũng có độ nghiêng thoải thoải và mất rất nhiều thời gian bên khe rãnh để rồi tiến chậm chạp đến điểm cực tiểu đôi khi đến mất ổn định khi rơi vào thung lũng, *hình 2.1* mô tả một quỹ đạo dao động với hàm mục tiêu dạng lòng khe.



Hình 2.1: Quỹ đạo dao động với sai số dạng lòng khe

Một phương pháp có thể giúp ta vấn đề này là dùng qui tắc mô-men. Tức là khi quỹ đạo nhảy qua nhảy lại trên khe rãnh, biến thiên trọng số sẽ đổi dấu liên tục và như vậy sẽ cho ta số trung bình cho một thay đổi nhỏ chính xác, như vậy mạng có thể ổn định ở đáy khe rãnh. ở đó nó bắt đầu di chuyển chậm dần theo quán tính. Tuy nhiên, đối với những bài toán mà nghiệm tối ưu nằm ở đáy bề lõm, thì mô-men cũng chẳng giúp được gì mà đôi khi còn gây ra nguy hiểm nếu mô-men lại đẩy quỹ đạo lên cùng một khoảng cách ở phía bên kia của khe rồi cứ xoay vòng tạo thành

dao động. Đó cũng chính là cái lợi và hại của việc dùng mô-men cho dạng bài toán với mặt lỗi lòng khe.

Một phương pháp khác sử dụng tốc độ học thích nghi VLBP (Variable Learning rate Back Propagation algorithm). Chúng ta có thể nâng cao sự hội tụ nếu chúng ta tăng tốc độ học trên vùng phẳng của mặt lỗi và rồi thì giảm tốc độ học khi mà độ cong của mặt lỗi tăng. Nghĩa là, chúng ta có thể nâng cao sự hội tụ bằng việc điều chỉnh tốc độ học trong quá trình huấn luyện, vấn đề của chúng ta sẽ là, xác định khi nào thay đổi tốc độ học và bằng bao nhiêu, hay nói cách khác, chúng ta cần biết chúng ta đang ở đâu trên hàm mục tiêu. Tuy nhiên điều trở ngại chính với các phương pháp VLBP là các điều chỉnh có thể cần 5 hoặc 6 tham số được lựa chọn trước. Thông thường thì vấn đề thực hiện của thuật toán là nhạy cảm với sự thay đổi trong các tham số đó. Sự lựa chọn của các tham số thì cũng phụ thuộc bài toán.

Tính bước học theo nguyên lý vượt khe là một phương pháp tỏ ra rất mạnh để giải quyết bài toán tối ưu, đặc biệt là các bài toán với mặt chất lượng dạng lòng khe, trục khe.

Tuy nhiên, trước khi đến với thuật toán vượt khe thì chúng ta hãy tìm hiểu về vấn đề điều kiện tối ưu, bởi vì nó ảnh hưởng đến những suy nghĩ của chúng ta trong những xuất phát điểm của bất kỳ một thuật toán tối ưu nào.

b. Điều kiện tối ưu

Ta sẽ đi định nghĩa một số khái niệm về điểm tối ưu. Chúng ta giả định rằng điểm tối ưu là điểm cực tiểu của hàm mục tiêu. Ta xét các khái niệm sau trước khi đi đến các điều kiện tối ưu.

Cực tiểu mạnh

Điểm u^* là cực tiểu mạnh của $J(u)$ nếu một số vô hướng $\delta > 0$ tồn tại, để $J(u^*) < J(u + \Delta u)$ với mọi Δu mà $\delta > \|\Delta u\| > 0$. Nói cách khác, nếu chúng ta dịch chuyển theo mọi cách từ điểm cực tiểu mạnh một khoảng nhỏ theo bất kỳ hướng nào hàm sẽ tăng.

Cực tiểu toàn cục

Điểm u^* là một cực tiểu toàn cục duy nhất của $J(u)$ nếu $J(u^*) < J(u + \Delta u)$ với mọi $\Delta u \neq 0$. Đối với cực tiểu mạnh, u^* , hàm có thể nhỏ hơn $J(u^*)$ tại các điểm lân cận nhỏ của u^* . Cho nên đôi khi cực tiểu mạnh còn gọi là cực tiểu cục bộ. Đối với cực tiểu toàn cục thì hàm sẽ là nhỏ nhất so với mọi điểm khác trong không gian tham số.

Cực tiểu yếu

Điểm u^* là một cực tiểu yếu của $J(u)$ nếu nó không phải là cực tiểu mạnh, và một số vô hướng $\delta > 0$ tồn tại, để $J(u^*) \leq J(u + \Delta u)$ với mọi Δu mà $\delta > \|\Delta u\| > 0$.

Chúng ta đã định nghĩa về điểm cực tiểu. Tiếp theo ta đến với một số điều kiện mà cần phải thoả mãn để có điểm tối ưu. Sử dụng khai triển Taylor để tiếp cận các điều kiện đó.

$$J(u) = J(u^* + \Delta u) = J(u^*) + \nabla J(u)^T \Big|_{u=u^*} \Delta u + \frac{1}{2} \Delta u^T \nabla^2 J(u) \Big|_{u=u^*} \Delta u + \dots,$$

trong đó $\Delta u = u - u^*$

Điều kiện thứ nhất

Nếu $\|\Delta u\|$ mà rất nhỏ thì bậc cao nhất trong phương trình

$$J(u) = J(u^* + \Delta u) = J(u^*) + \nabla J(u)^T \Big|_{u=u^*} \Delta u + \frac{1}{2} \Delta u^T \nabla^2 J(u) \Big|_{u=u^*} \Delta u + \dots,$$

sẽ là không đáng kể và ta có thể xấp xỉ hàm là:

$$J(u) = J(u^* + \Delta u) \cong J(u^*) + \nabla J(u)^T \Big|_{u=u^*} \Delta u$$

Điểm u^* là một điểm cực tiểu thí điểm, và điều này nói lên rằng có thể tăng nếu $\Delta u \neq 0$. Để điều này xảy ra thì $\nabla J(u)^T \Big|_{u=u^*} \Delta u \geq 0$. Tuy nhiên, nếu số hạng này mà dương $\nabla J(u)^T \Big|_{u=u^*} \Delta u > 0$ thì điều này nói rằng

$$J(u^* + \Delta u) \equiv J(u^*) - \nabla J(u)^T \Big|_{u=u^*} \Delta u < J(u^*)$$

Nhưng điều này là một sự mâu thuẫn, khi mà u^* là một điểm cực tiểu yếu. Cho nên, một lựa chọn duy nhất phải là $\nabla J(u)^T \Big|_{u=u^*} \Delta u = 0$. Khi điều này phải đúng cho bất kỳ Δu , chúng ta có $\nabla J(u) \Big|_{u=u^*} = 0$

Cho nên gradient phải bằng 0 tại điểm cực tiểu. Đây là điều kiện cần (nhưng không đủ) để u^* là một điểm cực tiểu cục bộ (hay còn gọi là cực tiểu mạnh). Bất kỳ các điểm thỏa mãn phương trình $\nabla J(u) \Big|_{u=u^*} = 0$ được gọi là các điểm tĩnh.

Điều kiện thứ hai

Giả sử rằng ta có điểm tĩnh u^* . Khi gradient của $J(u)$ bằng 0 tại tất cả các điểm tĩnh, khai triển chuỗi Taylor sẽ là:

$$J(u^* + \Delta u) = J(u^*) + \frac{1}{2} \Delta u^T \nabla^2 J(u) \Big|_{u=u^*} \Delta u + \dots,$$

Như trước, chúng ta sẽ chỉ xét các điểm đó trong lân cận nhỏ của u^* , vì $\|\Delta u\|$ là nhỏ và $J(u)$ có thể được xấp xỉ bởi hai số hạng đầu trong phương trình. Cho nên điểm cực tiểu mạnh (cực tiểu cục bộ) sẽ tồn tại, u^* , nếu $\Delta u^T \nabla^2 J(u) \Big|_{u=u^*} \Delta u > 0$

Để cho điều này là đúng với mọi $\Delta u \neq 0$ thì yêu cầu ma trận Hessian phải là ma trận xác định dương. Theo định nghĩa, một ma trận A xác định dương nếu $z^T A z > 0$ đối với bất kỳ vectơ $z \neq 0$. A là bán xác định dương nếu $z^T A z \geq 0$

Đối với vectơ z bất kỳ. Chúng ta có thể kiểm tra các điều kiện đó bởi việc kiểm tra các giá trị riêng của ma trận. Nếu tất cả các giá trị riêng mà dương, thì ma trận là xác định dương. Nếu tất cả các giá trị riêng không âm thì ma trận là xác định bán dương.

Ma trận Hessian xác định dương là một điều kiện thứ hai, điều kiện đủ cho việc tồn tại cực tiểu mạnh (điều kiện đủ để tồn tại cực tiểu cục bộ). Một cực tiểu có thể vẫn là cực tiểu mạnh nếu số hạng thứ hai của chuỗi Taylor là 0, nhưng số hạng thứ ba là dương. Cho nên điều kiện thứ hai là điều kiện cần cho cực tiểu mạnh nếu ma trận Hessian xác định bán dương.

Để tổng kết điều kiện tối ưu ta đi đến các điều sau:

- Các điều kiện cần cho u^* là một cực tiểu, mạnh hoặc yếu, của $J(u)$ là $\nabla J(u)|_{u=u^*} = 0$ và $\nabla^2 J(u)|_{u=u^*}$ xác định bán dương.

- Các điều kiện đủ cho u^* là một điểm cực tiểu mạnh (cực tiểu cục bộ) của $J(u)$ là $\nabla J(u)|_{u=u^*} = 0$ và $\nabla^2 J(u)|_{u=u^*}$ xác định dương.

2.1.3. Thuật toán vượt khe

Chúng ta nhận thấy rằng các điều kiện tối ưu nói chung chỉ là những điều kiện cần mà không đủ đối với cực tiểu toàn cục. Mà các thuật toán đã có thì đều không đảm bảo chắc chắn rằng sẽ tìm được điểm cực tiểu toàn cục mà chỉ có khả năng nâng cao cơ hội tìm đến điểm cực tiểu toàn cục mà thôi.

Như vậy, với những hàm mục tiêu phức tạp thì càng khó khăn hơn trong quá trình tìm nghiệm số tối ưu và vẫn có thể bị tắc tại trục của khe trước khi đạt được điểm cực tiểu, nếu hàm mục tiêu có dạng khe. Có thể có một chiến lược tiếp theo để giải quyết tiếp vấn đề khi mà gặp bài toán khe núi này là sau khi đạt tới gần trục khe bằng phương pháp gradient với bước học được tính bằng cực tiểu hoá theo đường (hoặc với bước học cố định) chúng ta sẽ dịch chuyển dọc theo đáy khe hẹp nhờ sự tiệm cận dần theo dạng hình học của đáy khe hẹp và lúc này thì người ta có thể giả thiết dạng hình học đó là đường thẳng hoặc xấp xỉ cong bậc hai.

Ta đã biết các thuật toán tối ưu hoá lặp được xây dựng trên hai khái niệm cơ bản là hướng thay đổi hàm mục tiêu (hướng tìm kiếm) và độ dài bước. Quá trình đó được mô tả như sau:

$$u^{k+1} = u^k + \alpha_k s^k, k = 0, 1, \dots$$

Trong đó: u^k, u^{k+1} là điểm đầu và điểm cuối của bước lặp thứ k ; s^k là vector chỉ hướng thay đổi các biến số trong không gian n chiều; α_k được gọi là độ dài bước. Các thuật toán tối ưu thông thường thì khác nhau về hướng chuyển động, hoặc (và) quy tắc điều chỉnh độ dài bước.

Sự khác biệt cơ bản của phương pháp vượt khe so với các phương pháp khác là ở quy tắc điều chỉnh bước. Theo quy tắc điều chỉnh bước của phương pháp vượt khe thì độ dài bước của điểm tìm kiếm ở mỗi bước lặp không nhỏ hơn độ dài bước

nhỏ nhất mà tại đó hàm mục tiêu đạt giá trị cực tiểu (địa phương) theo hướng chuyển động tại bước lặp đó. Điều này thể hiện rất rõ trong các nguyên lý cơ bản của phương pháp vượt khe.

2.1.3.1. Giới thiệu

Cho bài toán tối ưu và giải bài toán tối ưu không điều kiện:

$$\text{Min} J(u) \quad u \in E^n \quad (2.1)$$

u là vec-tơ trong không gian Euclide n chiều

Công thức lặp cho bởi bước thứ k như sau:

$$u^{k+1} = u^k + \alpha_k s^k, \quad k = 0, 1, \dots \quad (2.2)$$

trong đó: u là vectơ biến của hàm mục tiêu $J(u)$ tại bước lặp thứ k ;

α_k là độ dài bước của hàm theo hướng chuyển động s^k .

Hướng chuyển động s^k là hoàn toàn xác định tại mỗi bước lặp k (xác định sau). Mỗi quan tâm chính ở đây là phương pháp xác định độ dài bước α_k

Một vài công thức xác định giá trị α_k . Ví dụ: [1], [2]

$$\alpha_k = \arg \min J(u^{k-1} + \alpha.s^{k-1})$$

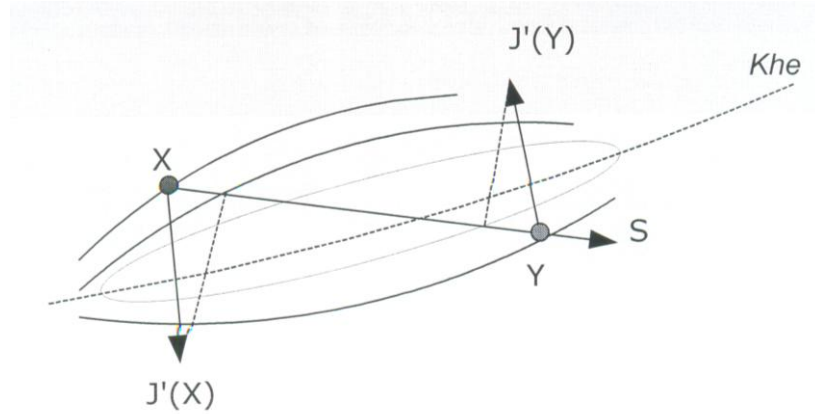
$$\alpha_k = \frac{a}{b+k}, \quad a, b = \text{const} > 0 \quad (2.3)$$

$$\alpha_k = \alpha_{k-1} q, \quad 0 < q < 1$$

Các công thức xác định độ dài bước α_k được các tác giả đưa ra khác với mỗi lớp hàm mục tiêu và thường có nhiều tham số đòi hỏi phải chọn giá trị phù hợp cho mỗi lớp hàm cụ thể.

Đối với các hàm mục tiêu mà có dạng khe thì các bước chuyển động (2.3) kém hiệu quả, trong phần sau chúng ta giả thiết công thức duy nhất định nghĩa độ dài bước α_k . Công thức được gọi là “nguyên lý vượt khe”. Giá trị α_k tìm được bởi nguyên lý được gọi là “bước vượt khe”.

2.1.3.2. Nguyên lý vượt khe



Hình 2.2: Hàm khe

Hàm “khe” là hàm mà mặt đồng mức của nó được kéo dài ra và kết quả là tạo ra một khe dài, hình 2.2. Có nghĩa là độ cong của hàm mục tiêu rất khác nhau đối với các hướng khác nhau. Trên cả hai phía của “khe”, gradient của hàm mục tiêu có hướng ngược lại. Xét điểm X đặt vào một phía của “khe” và Y trên phía khác. Hầu hết trường hợp các điểm X và Y đều thỏa mãn bất đẳng thức sau:

$$J'_s(X)^T J'_s(Y) < 0 \quad (2.4)$$

Trong đó $J'_s(X)$ ký hiệu phép chiếu của $J'(X)$ lên hướng S.

$$\text{Ký hiệu: } h_k(\alpha) = J(u^{k-1} + \alpha \cdot S^{k-1}) \quad (2.5)$$

$$\text{Ta có: } h'_k(\alpha) = \frac{\partial J(u^{k-1} + \alpha \cdot S^{k-1})}{\partial \alpha} = J'(u^{k-1} + \alpha \cdot S^{k-1})^T S^{k-1} \quad (2.6)$$

thế $\alpha = 0$ và $\alpha = \alpha_k$ vào công thức (2.6) ta thu được:

$$h'_k(0) = J'_s(u^{k-1})^T S^{k-1} \quad (2.7)$$

$$h'_k(\alpha_k) = J'_s(u^{k-1} + \alpha_k \cdot S^{k-1})^T = J'_s(u^k)^T S^{k-1} \quad (2.8)$$

Ta được điều kiện sau:

$$h'_k(\alpha_k) \cdot h'_k(0) = J'_s(u^{k-1})^T S^{k-1} J'_s(u^k)^T S^{k-1} < 0 \quad (2.9)$$

Dễ thấy rằng bất phương trình (2.9) là tương đương với (2.4) nếu $S^{k-1} = S^k$, $u^{k-1} = X$, $u^k = Y$.

Điều kiện (2.9) đòi hỏi tại mỗi bước lặp chuyển động của hàm mục tiêu, được gọi là nguyên lý “vượt khe”

Để đảm bảo tính đơn điệu của hàm mục tiêu trong quá trình tối ưu hoá, độ dài bước α_k phải thoả mãn bất phương trình sau:

$$J(u^k + \alpha_k S^k) < J(u^k). \quad (2.10)$$

(2.10) đòi hỏi tại mỗi bước lặp

Xét bài toán cực tiểu hoá không ràng buộc:

$$J(u) \rightarrow \min, u \in E^n$$

trong đó, u là vectơ cực tiểu hoá trong không gian Euclide n chiều, $J(u)$ là hàm mục tiêu bị chặn dưới và thoả mãn điều kiện:

$$\lim_{\|u\| \rightarrow \infty} J(x) = b \quad (2.11)$$

Thuật toán tối ưu hoá bài toán (2.1) có phương trình lặp như dạng sau:

$$u^{k+1} = u^k + \alpha_k S^k, k = 0, 1, \dots$$

trong đó u^k và u^{k+1} là điểm đầu và điểm cuối của bước lặp thứ k , s^k là vectơ chỉ hướng thay đổi các biến số trong không gian n chiều; α_k là độ dài bước

α_k được xác định theo nguyên lý vượt khe thì được gọi là bước vượt khe, còn phương trình (2.2) gọi là thuật toán vượt khe.

Nguyên lý vượt khe đã nêu ở trên phát biểu rằng điểm đầu và điểm cuối của mỗi bước lặp tối ưu hoá luôn nằm về hai phía điểm cực tiểu của hàm mục tiêu xét dọc theo hướng chuyển động tại bước đó. Nói cách khác, nếu tại điểm đầu hàm mục tiêu thay đổi theo chiều giảm, thì đến điểm cuối nó phải có xu hướng tăng.

Quỹ đạo tìm kiếm tối ưu theo nguyên lý vượt khe tạo ra một bức tranh hình học, tựa như điểm tìm kiếm tại mỗi lần lặp đều bước vượt qua lòng khe của hàm

mục tiêu. Để cụ thể hoá nguyên lý vượt khe, ta xét hàm một biến sau đối với mỗi bước lặp:

$$h(\alpha) = J(u^k + \alpha \cdot S^k) \quad (2.12)$$

Giả sử s^k là hướng của hàm mục tiêu tại điểm u^k . Theo điều kiện (2.11), tồn tại một giá trị $\alpha^* > 0$ bé nhất, sao cho $h(\alpha)$ đạt cực tiểu:

$$\alpha^* = \arg \min h(\alpha), \quad \alpha > 0 \quad (2.13)$$

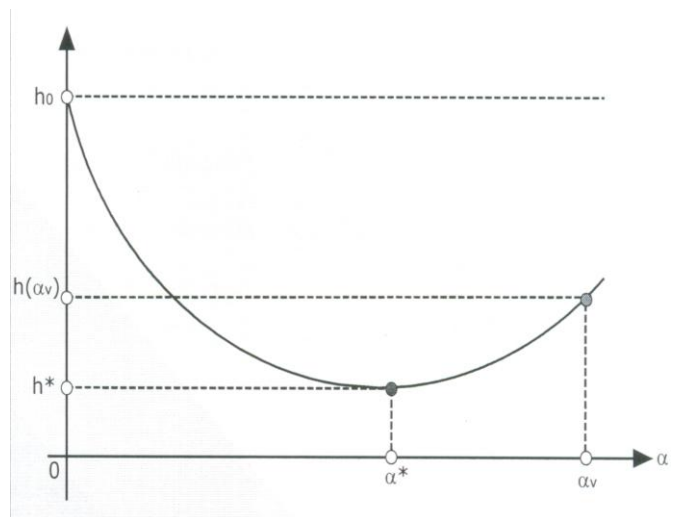
Nếu $J(u^k)$, cũng tức là $h(0)$, khả vi liên tục, ta có định nghĩa bước vượt khe như sau:

$$h'(\alpha) \Big|_{\alpha=\alpha^v} > 0, \quad h(\alpha^v) \leq h(0) \quad (2.14)$$

Trong đó, α^v là bước vượt quá, tức bước vượt khe.

Đồ thị biến thiên của hàm $h(\alpha)$, khi quỹ đạo tối ưu thay đổi từ điểm đầu u^k đến điểm cuối u^{k+1} thể hiện ở hình 2.3, ta thấy rằng, khi giá trị α tăng dần từ 0 vượt qua điểm cực tiểu α^* của $h(\alpha)$ tới giá trị α^v , quỹ đạo tối ưu hoá tương ứng tiến dọc theo hướng s^k theo quan hệ

$u^{k+1} = u^k + \alpha_k S^k$, thực hiện một độ dài bước $\alpha = \alpha^v \geq \alpha^*$. Đồ thị này cũng chỉ ra rằng, xét theo hướng chuyển động, thì hàm mục tiêu thay đổi theo chiều giảm dần từ điểm u^k , còn khi đạt điểm u^{k+1} thì nó đã chuyển sang xu hướng tăng.



Hình 2.3: Xác định bước vượt khe α^v

Nếu ta sử dụng bước chuyển động theo điều kiện (2.13) thì có thể tắc ở trục khe và thuật toán tối ưu hoá tương ứng bị tắc lại ở đó. Còn nếu quá trình tối ưu hoá theo điều kiện (2.14) thì sẽ không cho phép điểm tìm kiếm rơi vào lòng khe trước

khi đạt lời giải tối ưu, đồng thời nó vẽ ra một quỹ đạo luôn vượt lòng khe. Để quá trình lặp có hiệu quả hội tụ cao và ổn định, điều kiện (2.14) được thay đổi bởi:

$$\alpha^v \geq \alpha^* = \arg \min_{\alpha > 0} h(\alpha), \quad h(\alpha^v) - h^* \leq \lambda [h^0 - h^*] \quad (2.15)$$

Trong đó, $0 < \lambda < 1$ được gọi là hệ số vượt

$$h^* = h(\alpha^*)$$

$$h^0 = h(\alpha^0)$$

2.1.3.3. Xác định bước vượt khe

Việc lựa chọn độ dài bước học trong bài toán khe hẹp có ý nghĩa đặc biệt quan trọng. Nếu độ dài bước học mà quá nhỏ thì tốn thời gian thực hiện của máy tính. Nếu độ dài bước học lớn thì có thể gây trở ngại cho việc tìm kiếm vì khó theo dõi được sự uốn lượn của khe hẹp. Do vậy, vấn đề bước học thích nghi với khe hẹp là cần thiết trong quá trình tìm kiếm nghiệm tối ưu. Trong mục này, ta đưa ra một cách rất hiệu quả và đơn giản tìm bước “vượt khe”. Giả sử $J(u)$ liên tục và thoả mãn điều kiện $\lim J(u) = \infty$ khi $|u| \rightarrow \infty$ và tại mỗi bước lặp k , điểm u^{k-1} và véc tơ chuyển động s^{k-1} đã xác định. Cần phải xác định độ dài bước α_k thoả mãn (2.9)

Nếu thay h^* trong (2.15) bởi ước lượng $\bar{h} \approx h^*, \bar{h} > h^*$ thì ta vẫn nhận được bước vượt khe theo định nghĩa. Vì vậy, để đơn giản hoá việc lập trình nên lấy giá trị bé nhất của h được tính một cách đơn giản trong mỗi bước lặp tương ứng, mà không cần xác định chính xác h^* . Điều đó đồng thời làm giảm đáng kể số lần tính giá trị hàm mục tiêu. Ta có α_k xác định theo thuật toán sau:

Đầu vào: điểm khởi tạo, u và hướng tìm kiếm s

Đầu ra: độ dài bước vượt khe α

Các tham số: $a=0.5$, độ chính xác ε và γ

Ta thực hiện các bước sau:

Bước 1:

Giả sử $\alpha = a > 0$ tính $h_k(\alpha)$

Nếu $h_k(\alpha) \geq h_k(0)$ thì $\alpha = 0$, $\beta = a$, sang bước 2

Nếu $h_k(\alpha) < h_k(0)$ thì lặp gán: $\alpha = \beta$; $\beta = 1,5\beta$ cho đến khi $h_k(\beta) \geq h_k(\alpha)$ sang bước 2

Bước 2:

Nếu $|\beta - \alpha| \leq \varepsilon, \varepsilon > 0$ thì sang bước 3.

Tính $\theta = \alpha + \gamma(\beta - \alpha)$ và $h_k(\theta)$, trong đó $\gamma = 0,1$.

Nếu $h_k(\alpha) \geq h_k(\theta)$ thì $\alpha = \theta$ và quay lại bước 2.

Nếu $h_k(\alpha) < h_k(\theta)$ thì $\beta = \theta$ và quay lại bước 2.

Bước 3:

Giữ nguyên giá trị mới $\alpha = \theta$ và chấp nhận bước “vượt khe” $\alpha_k = \beta$. Quá trình xác định bước vượt khe kết thúc.

Có hai tham số (giá trị khởi tạo a và độ chính xác) đòi hỏi chọn giá trị thích hợp, nhưng một giá trị cụ thể ít làm thay đổi hiệu quả của thuật toán và ε phụ thuộc trực tiếp vào giải pháp tối ưu. Vì vậy, có thể đưa ra một hằng số cho mọi trường hợp $a = 0,5$; $\varepsilon = 0,01$.

Các trường hợp chọn độ dài bước mà xét trên “hệ quy chiếu” là hàm mục tiêu dạng lòng khe:

- Trường hợp 1: Giá trị α_k thỏa mãn điều kiện cực tiểu hàm $J(u)$ theo hướng s^k để đạt tới đúng lòng khe là $\alpha_k = \arg \min J(u^{k-1} + \alpha.s^{k-1})$.

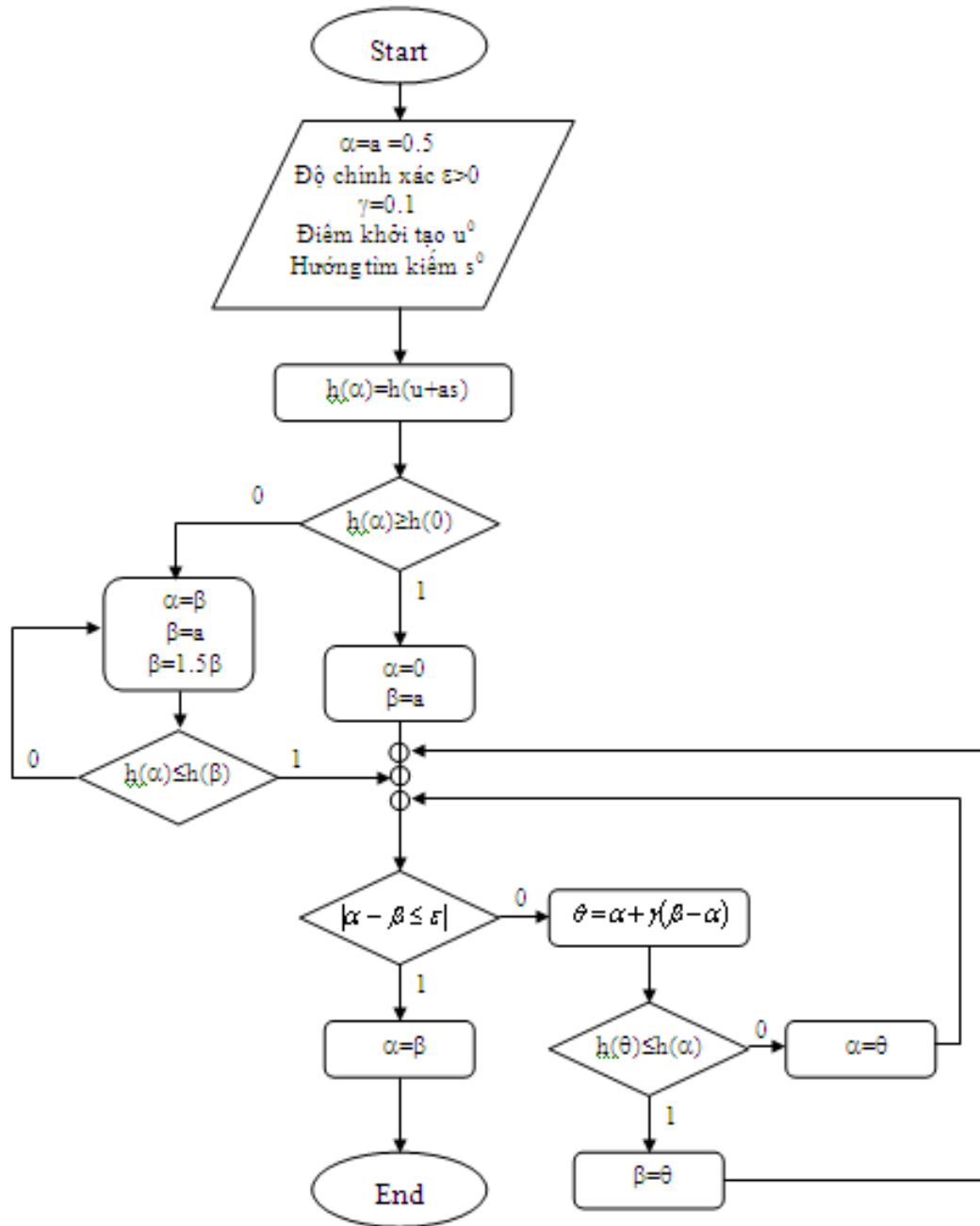
- Trường hợp 2: Giá trị α_k thỏa mãn hệ thức sau để chưa đạt tới lòng khe là $0 < \varepsilon_1 \leq \alpha_k \leq \frac{1}{\varepsilon_2 + L}$, $\varepsilon_2 > 0$. L - hằng số Lipschitz, là giá trị cận trên của độ dốc tối đa của hàm mục tiêu.

- Trường hợp 3: Giá trị α_k vượt qua lòng khe là $\left. \frac{d}{d\alpha} J(u^k + \alpha.s^k) \right|_{\alpha=\alpha^v} > 0$

▪ ***Xác định bước vượt khe***

Việc lựa chọn độ dài bước học trong bài toán khe hẹp có ý nghĩa đặc biệt quan trọng. Nếu độ dài bước học mà quá nhỏ thì tốn thời gian thực hiện của máy tính. Nếu độ dài bước học lớn thì có thể gây trở ngại cho việc tìm kiếm vì khó theo dõi được sự uốn lượn của khe hẹp. Do vậy, vấn đề bước học thích nghi với khe hẹp là cần thiết trong quá trình tìm kiếm nghiệm tối ưu. Trong mục này, ta đưa ra một cách rất hiệu quả và đơn giản tìm bước “vượt khe”. Giả sử $J(u)$ liên tục và thoả mãn điều kiện $\lim_{|u| \rightarrow \infty} J(u) = \infty$ và tại mỗi bước lặp k , điểm u^{k-1} và véc tơ chuyển động s^{k-1} đã xác định. Cần phải xác định độ dài bước α_k thoả mãn điều kiện (2.7).

Đến đây, chúng ta đã có được một thuật toán đầy đủ để tính bước học vượt khe, *hình 2.4*.

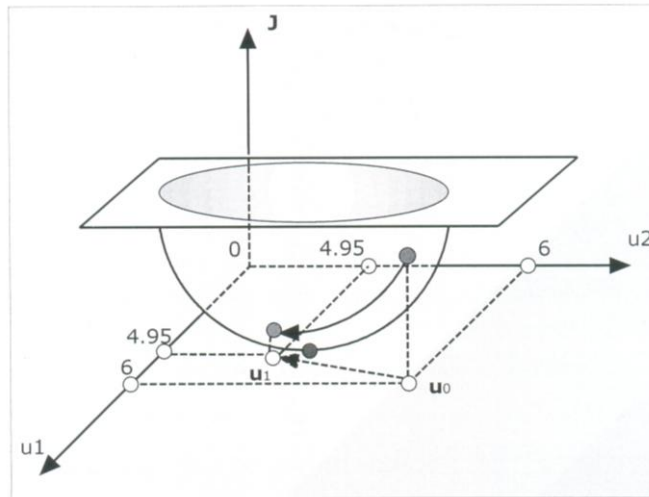


Hình 2.4: Lưu đồ thuật toán tìm bước vượt khe

2.1.3.4. Ví dụ

Để minh họa thuật toán vượt khe, ta làm một bước lặp (thực hiện bằng tay) trong quá trình tìm kiếm lời giải của bài toán tối ưu.

Giả sử ta có hàm $J(u) = (u_1 - 5)^2 + (u_2 - 5)^2$, cần cực tiểu hóa hàm này theo phương pháp hạ nhanh nhất, α_k xác định theo nguyên lý vượt khe.



Hình 2.5: Bước lặp $k = 1$

Điểm bắt đầu $u^0 = (6;6)$ với $J(u^0)=2$, Ta có:

$$\frac{\partial J(u)}{\partial u_1} = 2(u_1 - 5); \quad \frac{\partial J(u)}{\partial u_2} = 2(u_2 - 5).$$

Bước lặp $k = 1$ (xem hình 2.5):

$$u^1 = u^0 + \alpha s^0$$

$$s^0 = -J'(u^0) = \begin{pmatrix} -2 & -2 \end{pmatrix}$$

Tìm độ dài bước α_1

Bước 1(1): Cho $\alpha = a = 0.5$

$$J(u^0 + 0.5s^0) = 0 < J(u^0) \text{ nên } \alpha = a = 0.5; a = 1.5a = 0.75; \text{ quay lại bước 1.}$$

Bước 1(2):

$$J(u^0 + 0.75s^0) = 0.5 > J(u^0 + 0.5s^0) \text{ nên } \beta = a = 0.75; \text{ sang bước 2.}$$

Bước 2: $\theta = 0.5 + 0.1(0.75 - 0.5) = 0.525$;

$$J(u^0 + 0.525s^0) = 0.005 > J(u^0 + 0.5s^0) \text{ và } < J(u^0) \text{ nên sang bước 3.}$$

Bước 3: $\alpha = \theta = 0.525$; Tính u^1 theo $u^0 + \alpha s^0$, $u^1 = (4,95;4,95)$ với $J(u^1) = 0.005$.

Bước lặp $k=2$. Tương tự như bước lặp với $k-1$, quá trình lặp này cứ tiếp diễn cho đến khi $J(u^k) < \varepsilon$ cho trước.

2.2. Ứng dụng thuật toán vượt khe trong quá trình luyện mạng nơron

Nguyên lý cơ bản của tối ưu hóa đã được khám phá trong thế kỷ thứ 17 bởi các nhà toán học như Kepler, Fermat, Newton, và Leibniz. Ngay từ những năm 1950, các nguyên lý tối ưu đã được nghiên cứu và phát triển tiếp để hướng tới một mục đích rằng làm sao thực hiện được các thuật toán đó với tốc độ cao trên nền các máy tính kỹ thuật số. Thành công của sự cố gắng đó đã kích thích việc nghiên cứu các thuật toán mới, và việc nghiên cứu về các thuật toán tối ưu trở thành một ngành quan trọng trong toán học. Các nghiên cứu về mạng nơron ngày nay cũng ứng dụng công cụ toán học đó để giải quyết vấn đề trong quá trình huấn luyện mạng nơron.

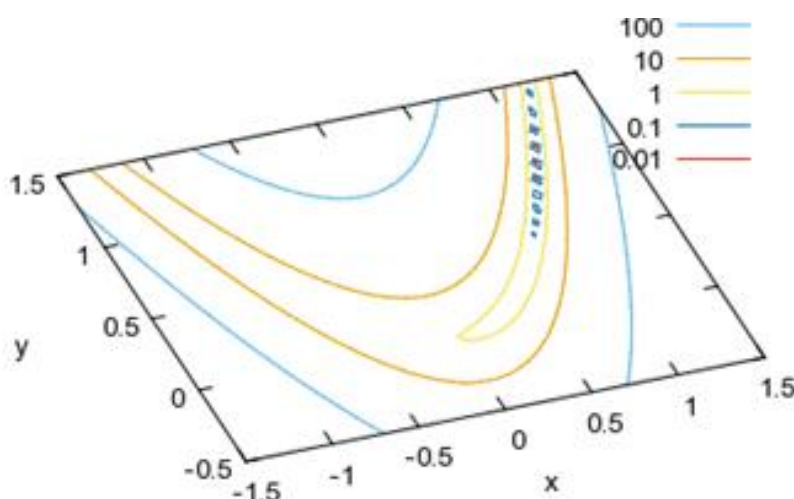
Có thể nhấn mạnh rằng tất cả các thuật toán mà chúng đã biết như thuật toán gradient liên hợp hay thuật toán Levenberg-Marquardt, ... đều sử dụng kỹ thuật lan truyền ngược để huấn luyện mạng nơron, tức là các sai số được xử lý từ lớp cuối cùng đến lớp đầu tiên của mạng.

Vậy, quá trình luyện mạng nơron chính là quá trình tối ưu hóa tham số của mạng, mà ở đây trong phạm vi nghiên cứu của luận án là kỹ thuật lan truyền ngược. Sự khác nhau giữa các thuật toán thể hiện ở cách mà các kết quả của các đạo hàm được sử dụng để cập nhật các trọng số.

Chúng ta đã thấy rằng giảm dốc nhất là một thuật toán đơn giản, và thông thường chậm nhất. Thuật toán gradient liên hợp và phương pháp Newton's nói chung mang đến sự hội tụ nhanh hơn [26]. Khi nghiên cứu về các thuật toán nhanh hơn thì thường rơi vào hai trường phái. Trường phái thứ nhất phát triển về các kỹ thuật tìm kiếm. Các kỹ thuật tìm kiếm bao gồm các ý tưởng như việc thay đổi tốc độ học, sử dụng qui tắc mô-men, bước học thích nghi. Trường phái khác của nghiên cứu nhằm vào các kỹ thuật tối ưu hóa số chuẩn, điển hình là phương pháp gradient liên hợp, hay thuật toán Levenberg-Marquardt (một biến thể của phương pháp Newton). Tối ưu hóa số đã là một chủ đề nghiên cứu quan trọng với 30, 40 năm, nó dường như là nguyên nhân để tìm kiếm các thuật toán huấn luyện nhanh.

Như đã phát biểu trong chương 1 một nhân tố ảnh hưởng đến hiệu lực và độ hội tụ của giải thuật lan truyền ngược sai số là bước học α . Không có một giá trị xác định nào cho các bài toán khác nhau. Với mỗi bài toán, bước học thường được lựa

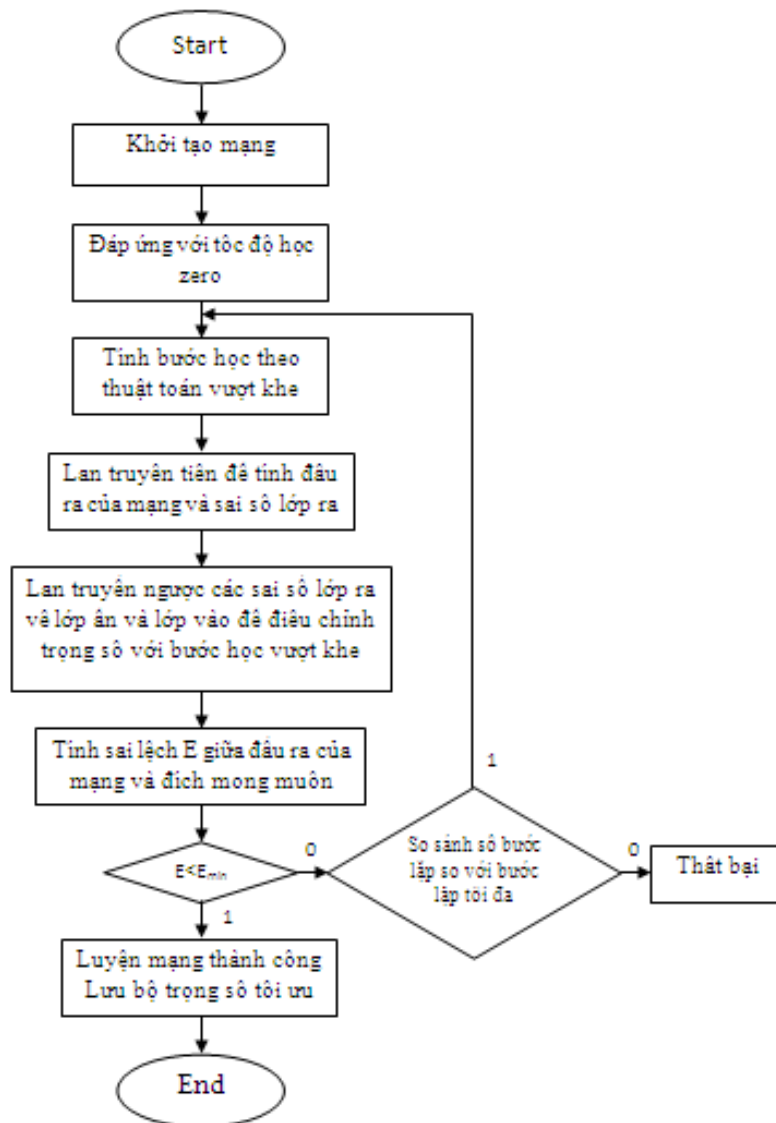
chọn bằng thực nghiệm theo phương pháp thử và sai. Giá trị α lớn làm tăng tốc quá trình hội tụ. Điều này không phải lúc nào cũng có lợi vì nếu ngay từ đầu ta đã cho là mạng nhanh hội tụ thì rất có thể mạng sẽ hội tụ sớm ngay tại một cực tiểu địa phương gần nhất mà không đạt được độ sai số như mong muốn. Tuy nhiên, đặt giá trị bước học quá nhỏ thì mạng sẽ hội tụ rất chậm, thậm chí mạng có thể vượt được qua các cực tiểu cục bộ và vì vậy dẫn đến học mãi mà không hội tụ. Hình 2.6 mô tả một hàm có các đường đồng mức bị kéo dài, cong tạo thành khe. Với các hàm dạng như thế này, chọn bước học bằng bao nhiêu để không bị tắc tại trục khe.



Hình 2.6: Các đường đồng mức dạng khe

Trong phần này tác giả trình bày một kỹ thuật vượt khe để áp dụng cho việc tìm bộ trọng số tối ưu của mạng. Sau đó chúng ta sẽ sử dụng một ví dụ đơn giản để nói rằng kỹ thuật lan truyền ngược giảm dốc nhất (SDBP) có vấn đề với sự hội tụ và tác giả sẽ trình bày sự kết hợp giữa thuật toán vượt khe và kỹ thuật lan truyền ngược nhằm cải tiến tính hội tụ của lời giải.

Hình 2.7 mô tả thuật toán huấn luyện mạng nơron MLP bằng thuật học lan truyền ngược với bước học vượt khe. Thuật toán để tính bước học vượt khe được trình bày trên hình 2.4.



Hình 2.7: Lưu đồ thuật toán huấn luyện mạng nơron MLP với bước học vượt khe

2.3. Minh họa thuật toán

Bài toán ví dụ để minh họa cho thuật toán huấn luyện với bước học vượt khe như sau: Cho một vec-tơ đầu vào tới đầu vào mạng, mạng nơron phải trả lời cho chúng ta biết đầu vào ấy là cái gì.

2.3.1. Công tác chuẩn bị

Mục tiêu: cài đặt thành công phương pháp luyện mạng nơron bằng thủ tục lan truyền ngược kết hợp với bước học tính theo nguyên lý vượt khe để tìm nghiệm tối ưu toàn cục đối với mặt sai số có dạng lòng khe.

Với việc sử dụng phương pháp giảm nhanh nhất để cập nhật trọng số của mạng, ta cần các thông tin liên quan đến đạo hàm riêng của hàm lỗi lấy trên từng trọng số, nghĩa là vấn đề còn lại của chúng ta trong công tác chuẩn bị là trình bày các công thức và thuật toán cập nhật trọng số trong các lớp ẩn và lớp ra.

Với một tập mẫu cho trước, chúng ta sẽ tính đạo hàm hàm sai số bằng cách lấy tổng đạo hàm hàm lỗi trên từng mẫu trong tập đó. Phương pháp phân tích và tính đạo hàm này dựa trên “qui tắc chuỗi”.

Đối với mạng tuyến tính đơn lớp (ADLINE) đạo hàm riêng được tính toán một cách thuận lợi. Đối với mạng nhiều lớp thì sai số không là một hàm tường minh của các trọng số trong các lớp ẩn, chúng ta sẽ sử dụng luật chuỗi để tính toán các đạo hàm. Độ dốc của tiếp tuyến với đường cong sai số trong mặt cắt theo trục w gọi là đạo hàm riêng của hàm lỗi J lấy theo trọng số đó, ký hiệu là $\partial J / \partial w$, dùng quy tắc

chuỗi ta có
$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial w_1} \cdot \frac{\partial w_1}{\partial w_2} \dots \frac{\partial w_n}{\partial w}$$

2.3.1.1. Điều chỉnh trọng số lớp ra

Gọi: b : trọng số lớp ra

z : đầu ra của nơron lớp ra.

t : giá trị đích mong muốn

y_j : đầu ra của nơron trong lớp ẩn

v : tổng trọng hóa $v = \sum_{j=0}^{M-1} b_j y_j$ nên $\partial v / \partial b_j = y_j$ (bỏ qua chỉ số của các

nơron lớp đầu ra)

Ta sử dụng $J = 0.5 * (z - t)^2$, nên $\partial J / \partial z = (z - t)$.

Hàm kích hoạt nơron lớp ra là sigmoid $z = g(v)$, với $\partial z / \partial v = z(1 - z)$.

Ta có:
$$\frac{\partial J}{\partial b} = \frac{\partial J}{\partial z} \cdot \frac{\partial z}{\partial v} \cdot \frac{\partial v}{\partial b} = (z - t) \cdot z(1 - z) \cdot y \quad (2.16)$$

Từ đó ta có công thức cập nhật trọng số lớp ra như sau (bỏ qua các chỉ số):

$$\Delta b = \alpha \cdot (z - t) \cdot z \cdot (1 - z) \cdot y \quad (2.17)$$

Ta sẽ sử dụng công thức (2.17) [21] trong thủ tục DIEUCHINHTRONGSO () để điều chỉnh trọng số lớp ra, trong đó có một điểm nhấn mạnh là tốc độ học α được tính theo nguyên lý vượt khe, đây là mục tiêu của nghiên cứu.

```
void DIEUCHINHTRONGSO(int k)
{
    int i, j;
    float temp;
    SAISODAURA();
    for(i=0; i<SLNRLA; i++)
    {
        for(j=0; j<SLNRLR; j++)
        {
            temp = -TOCDOHOC*y[i]*z[j]*(1-z[j])*SSLR[j];
            MTTSLR[i][j] = MTTSLR[i][j] + temp + QUANTINH*BTMTTSLR[i][j];
            BTMTTSLR[i][j] = temp;
        }
    }
}
```

2.3.1.2. Điều chỉnh trọng số lớp ẩn

Đạo hàm hàm mục tiêu của mạng đối với một trọng số lớp ẩn được tính theo qui tắc chuỗi, $\partial J / \partial a = (\partial J / \partial y) \cdot (\partial y / \partial u) \cdot (\partial u / \partial a)$.

Gọi: a: trọng số lớp ẩn

y: đầu ra của một nơron trong lớp ẩn

x_i : các thành phần của vector vào của lớp vào

u: tổng trọng hóa $u = \sum_{i=0}^{N-1} a_i x_i$ nên $\partial u / \partial a_i = x_i$

k: chỉ số của các nơron trong lớp ra

Lớp ẩn tự chúng không gây ra lỗi nhưng nó góp phần tác động vào sai số của lớp ra. Ta có
$$\frac{\partial J}{\partial y} = \sum_{k=0}^{K-1} \frac{\partial J}{\partial z_k} \frac{\partial z_k}{\partial v_k} \frac{\partial v_k}{\partial y}$$

Ta chỉ xét cho một neuron lớp ẩn nên chỉ số của lớp ẩn ta lược bỏ cho đơn giản. Đến đây ta nhận thấy lượng $\left(\frac{\partial J}{\partial z}\right) \cdot \left(\frac{\partial z}{\partial v}\right)$ được lan truyền ngược về lớp ẩn, chỉ số k nói rằng $\left(\frac{\partial J}{\partial z_k}\right) \left(\frac{\partial z_k}{\partial v_k}\right) = (z_k - t_k) \cdot z_k \cdot (1 - z_k)$ (công thức này được suy ra từ mục 3.1.1) thuộc neuron nào trong lớp ra. Như ta đã biết thì v là tổng trọng hóa, nên $\left(\frac{\partial v}{\partial y}\right) = b$, với chỉ số k ta có $\left(\frac{\partial v_k}{\partial y}\right) = b_k$. Đến đây, số hạng đầu tiên của
$$\left(\frac{\partial J}{\partial a}\right) = \left(\frac{\partial J}{\partial y}\right) \cdot \left(\frac{\partial y}{\partial u}\right) \cdot \left(\frac{\partial u}{\partial a}\right)$$
 đã được hoàn thành.

$$\frac{\partial J}{\partial y} = \sum_{k=0}^{K-1} (z_k - t_k) \cdot z_k \cdot (1 - z_k) \cdot b_k$$
, trong công thức này ta đã lược bỏ chỉ số của neuron lớp ẩn, cái mà được gắn với y (viết đầy đủ phải là $\left(\frac{\partial J}{\partial y_j}\right)$, với j=0...M-1).

Tiếp theo, ta xét số hạng thứ hai $\left(\frac{\partial y}{\partial u}\right)$, số hạng này diễn tả sự biến đổi của đầu ra của một neuron thuộc lớp ẩn theo tổng trọng hóa các thành phần vec-tơ đầu vào của lớp vào. Ta có mối quan hệ giữa y và u theo hàm kích hoạt neuron $y = \frac{1}{1 + e^{-u}}$, nên $\left(\frac{\partial y}{\partial u}\right) = y \cdot (1 - y)$. Còn lại số hạng thứ ba $\left(\frac{\partial u}{\partial a}\right)$ là sự

biến đổi của đầu ra của neuron lớp ẩn theo trọng số lớp ẩn. Vì u là tổng trọng hóa của các thành phần vec-tơ đầu vào $u = \sum_{i=0}^{N-1} a_i x_i$ nên ta có ngay $\left(\frac{\partial u}{\partial a_i}\right) = x_i$. Tóm lại,

ta có đạo hàm hàm mục tiêu theo một trọng số của lớp ẩn:

$$\frac{\partial J}{\partial a_i} = \sum_{k=0}^{K-1} (z_k - t_k) \cdot z_k \cdot (1 - z_k) \cdot b_k \cdot y \cdot (1 - y) \cdot x_i \quad (2.18)$$

Từ đây ta đi đến công thức điều chỉnh trọng số cho lớp ẩn:

$$\Delta a_i = \alpha \cdot \sum_{k=0}^{K-1} (z_k - t_k) \cdot z_k \cdot (1 - z_k) \cdot b_k \cdot y \cdot (1 - y) \cdot x_i \quad (2.19)$$

Trong công thức (2.19) chỉ số i biểu thị neuron thứ i của lớp vào, chỉ số k biểu thị neuron thứ k lớp ra; trong công thức chúng ta đã không biểu thị chỉ số j , chỉ số biểu thị neuron lớp ẩn.

Ta sẽ sử dụng công thức (2.19) [21] trong thủ tục DIEUCHINHTRONGSO() để điều chỉnh trọng số lớp ẩn, tốc độ học α được tính theo nguyên lý vượt khe.

Phần mã thủ tục DIEUCHINHTRONGSO() được trình bày như sau, trong đó biến temp kiểu float chính là giá trị biến thiên trọng số và biến temp được tính theo công thức (2.19). (Một phần trong công thức (2.19) được tính trong thủ tục SAISOLOPAN().)

```
void DIEUCHINHTRONGSO(int k)
{
    int i, j;
    float temp;
    SAISOLOPAN();
    for(i=0; i<SLNRLV; i++)
    {
        for(j=0; j<SLNRLA; j++)
        {
            temp = -TOCDOHOC*x[i]*y[j]*(1-y[j])*SSLA[j];
            MTTSLA[i][j] = MTTSLA[i][j] + temp +
QUANTINH*BTMTTSLA[i][j];
            BTMTTSLA[i][j] = temp;
        }
    }
}
```

2.3.2. Cấu trúc mạng

Từ các thông tin thu được từ các mục 2.3 và mục 2.1, cùng với yêu cầu của bài toán ví dụ đã nêu trong mục 1.4.2. là nhận biết các ký tự là các chữ số 0, 1, ..., 9.

Ngoài ra tác giả còn đưa thêm ví dụ nữa là nhận biết các chữ cái: a, b, c, d, e, g, h, i, k, l.





Hình 1.5 đã mô tả cấu trúc của mạng nơron nhiều lớp với 35 nơron lớp vào, 5 nơron lớp ẩn và 10 nơron lớp ra; hàm f được lựa chọn là sigmoid.

$$f = \frac{1}{1 + \exp(-x)}$$

Chúng ta cũng nói một vài điều về việc sử dụng bias. Có thể các nơron có hoặc không có các bias. Bias cho phép mạng điều chỉnh và tăng cường thêm sức mạnh của mạng. Đôi khi ta muốn tránh đối số của hàm f bằng giá trị không khi mà tất cả các đầu vào của mạng bằng không, hệ số bias sẽ làm cái việc là tránh cho tổng trọng hóa bằng không. Trong cấu trúc mạng hình 1.5 ta bỏ qua sự tham gia của các bias. Một vài điều bàn luận thêm về việc lựa chọn cho bài toán của chúng ta một cấu trúc mạng như thế nào cho tối ưu là chưa giải quyết được, ở đây chúng ta chỉ mới làm tốt việc này dựa trên sự thử sai. Tiếp theo, điều gì sẽ xảy ra nếu chúng ta thiết kế một mạng cần phải có nhiều hơn hai lớp? Số lượng nơron trong lớp ẩn lúc đó sẽ là bao nhiêu? Vấn đề này đôi khi phải dự báo trước hoặc cũng có thể phải qua việc nghiên cứu và thực nghiệm. Một mạng nơron có tốt hay không thì việc đầu tiên là phải chọn được một cấu trúc mạng tốt. Cụ thể là ta sẽ lựa chọn hàm kích hoạt (transfer function) là cái nào trong một loạt các hàm đã có (xem bảng 2.1). Ta cũng có thể tự xây dựng lấy một hàm kích hoạt cho riêng bài toán của chúng ta mà không sử dụng các hàm có sẵn đó. Giả sử, việc lựa chọn sơ bộ một cấu trúc mạng như trên hình 1.5 là hoàn tất. Chúng ta sẽ bắt tay vào công việc huấn luyện mạng, tức là chỉnh định các bộ số (w) của mạng. Nếu sau một nỗ lực huấn luyện mà ta không đạt được các bộ số tối ưu (w*) thì cấu trúc mạng mà ta chọn kia có thể có vấn đề, dĩ nhiên, ta phải làm lại công việc này từ đầu, tức là chọn một cấu trúc mạng khác, và huấn luyện lại... Sau quá trình huấn luyện mạng, chúng ta có thể gặp phải vấn đề là một số nơron trong mạng chẳng tham gia gì vào công việc của bài toán cả, nó thể hiện ở các hệ số trọng w nối với nơron đó gần bằng zero. Tất nhiên, ta sẽ bỏ nơron đó đi cho việc tối ưu bộ nhớ và thời gian hoạt động của mạng.

2.3.3. Các thư viện và hàm mạng

Bảng 2.1. Các hàm kích hoạt (transfer function) tiêu biểu [6]

Tên hàm	Mô tả hàm	Biểu tượng	Hàm Matlab
Hard Limit	$f = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$		Hardlim
Symmetrical Hard Limit	$f = \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$		Hardlims
Linear	$f = x$		Purelin
Log-sigmoid	$f = \frac{1}{1 + \exp(-x)}$		logsig

2.3.3.1. Thư viện

Trong tệp dinhnghia.h, hàm sigmoid được phát biểu cùng với đạo hàm của nó, hàm này có đặc điểm là rất phẳng đối với các đầu vào lớn do đó dễ sinh ra mặt chất lượng có dạng khe núi hẹp. Vậy có một câu hỏi đặt ra là, có bắt buộc tất cả các nơron trong một lớp thì phải có các hàm kích hoạt giống nhau? Câu trả lời là không, ta có thể định nghĩa một lớp của các nơron với các hàm kích hoạt khác nhau. Chúng ta sử dụng hàm kích hoạt mà hay được sử dụng trong kỹ thuật mạng nơron, hàm sigmoid cho toàn bộ các nơron trong mạng.

```
#define SIGMF(x)    1 / (1 + exp(-(double)x))
#define DSIGM(y)    (float) (y) * (1.0-y)
```

Số lượng nơron đầu vào của mạng bằng số thành phần của vec-tơ đầu vào x với kích thước 35×1.

```
#define SLNRLV      35          // SO LUONG NO RON LOP VAO
```

Số lượng nơron lớp ẩn là 5, vec-tơ đáp ứng đầu ra lớp ẩn là y với kích thước 5×1. Có thể nói chưa có công trình nào nói về kích thước của lớp ẩn một cách cụ thể. Tuy nhiên, nếu chọn kích thước lớp ẩn lớn quá thì không nên đối với các bài

toán mà các đầu vào (vec-tơ x) dễ bị nhiễu tác động, bởi rằng với một mạng mà có lớp ẩn kích thước lớn thì mạng nơron sẽ rất “nhạy cảm”, do đó một nhiễu nhỏ tác động lên đầu vào cũng có thể làm ảnh hưởng đến đáp ứng đầu ra của mạng trong quá trình sử dụng mạng.

```
#define SLNRLA      5          // SO LUONG NO RON LOP AN
```

Số lượng nơron lớp ra là 10. Ta cũng có thể chọn là 1 nơron đối với bài toán này, tuy nhiên với cách lựa chọn như vậy thì sẽ rất khó khăn cho chúng ta trong việc giải quyết yêu cầu của bài toán, vec-tơ đầu ra của lớp ra là z với kích thước 10×1 .

```
#define SLNRLR      10         // SO LUONG NO RON LOP RA
```

Trong các thuật toán tối ưu lặp, có một khái niệm là điều kiện dừng thuật toán. Có nhiều phương án để dừng lời giải. Ở đây ta sử dụng một phương pháp rất phổ biến đó là định nghĩa một giá trị ngưỡng sai số để dừng lời giải và coi như đã đạt được nghiệm của bài toán, hay là đã tìm được bộ trọng số tối ưu. Vậy, định nghĩa giá trị ngưỡng là bao nhiêu? Nó có thể ảnh hưởng đến vấn đề mà chúng ta gọi là tổng quát hóa của mạng. Một mạng có khả năng tổng quát hóa tốt hay không thì một trong những yếu tố ảnh hưởng đó là việc dừng luyện mạng sớm hay muộn, tức là liên quan đến việc định nghĩa giá trị ngưỡng sai số. Giá trị ngưỡng sai số mà ta sử dụng để dừng việc huấn luyện mạng nếu giá trị hàm mục tiêu đạt tới là epsilon.

```
#define EPSILON      0.06      // SAI SO TRUNG BINH BINH PHUONG
DE DUNG QUA TRINH LUYEN MANG
```

Ta cũng định nghĩa một giá trị cực đại cho số bước lặp trong quá trình huấn luyện mạng là BLTD, nếu số bước lặp vượt quá giá trị này thì chúng ta coi như quá trình huấn luyện mạng thất bại.

```
#define BLTD          30000     // BUOC LAP TOI DA
```

Giá trị FD được sử dụng trong thuật toán tính bước học vượt khe.

```
#define FD             1e-1
```

Cũng liên quan đến vấn đề khả năng tổng quát hóa của mạng, việc lựa chọn tài liệu học cho mạng ảnh hưởng đến năng lực của mạng sau khi đã được huấn luyện. Ngoài ra, việc chọn tập hồ sơ huấn luyện cũng ảnh hưởng đến chi phí thời

gian để luyện mạng, và mỗi một tập hồ sơ huấn luyện khác nhau đương nhiên cũng sẽ đem lại một mặt chất lượng khác nhau của mạng. Trong tệp `taphuanluyen.h` trình bày bộ tài liệu dùng để huấn luyện mạng. Bộ tài liệu này gồm các vec-tơ mẫu đầu vào và vec-tơ đáp ứng đích. Bảng 2.2 của mục 2.3.4.2. sẽ trình bày về kết quả của việc luyện mạng với tập hồ sơ luyện mạng đã cho để minh họa khả năng của thuật toán vượt khe.

2.3.3.2. Hàm khởi tạo trọng số

Là hàm khởi tạo các giá trị trọng số sẽ được sử dụng cho các mạng nơron. Cấu trúc dữ liệu của giá trị trọng số được khởi tạo và thiết lập một lần bằng việc gọi hàm `KHOITAOMANG()`. Hàm sẽ khởi tạo mảng hai chiều của các trọng số lớp ẩn và lớp ra với các giá trị ngẫu nhiên chuẩn hóa với dải đặc biệt. Với hàm kích hoạt là hàm sigmoid thì ta chọn giá trị này là 0.5. Trong mục 2.3.4. chúng ta sẽ chạy thử chương trình luyện mạng nhiều lần để thấy rằng với việc xuất phát từ một điểm bất kỳ khác nhau trong không gian trọng số của mỗi lần chạy là ảnh hưởng đến quá trình tìm kiếm điểm cực tiểu, xem thông kê ở *bảng 2.2 và 2.3.*

Mô tả hàm `KHOITAOMANG()` trong đoạn mã sau:

```
int KHOITAOMANG()
{
    for(i=0;i<SLNRLV;i++)
        for(j=0;j<SLNRLA;j++)
        {
            MTTSLA[i][j] = -0.5+ (float) rand()/RAND_MAX;
            BTMTSLA[i][j] = 0;
        }
    for(i=0;i<SLNRLA;i++)
        for(j=0;j<SLNRLR;j++)
        {
            MTTSLR[i][j] = -0.5 + (float) rand()/RAND_MAX;
            BTMTSLR[i][j] = 0;
        }
}
```

2.3.3.3. Thủ tục tính bước học vượt khe

Trong mục 2.1 của luận án đã trình bày về nguyên lý vượt khe cũng như thuật toán để tính bước học vượt khe, sau đây là đoạn mã tính bước học. Thủ tục TINHBUOCHOCVUOTKHE() sẽ được gọi trong HUANLUYENVUOTKHE() sau mỗi bước lặp. Khi chúng ta sử dụng bước học được tính theo nguyên lý vượt khe thì đương nhiên máy tính sẽ mất một khoảng thời gian để thực hiện các phép toán của thuật toán. Tuy nhiên, như bảng 2.2 và 2.3 trong mục 2.3.4 thống kê thì thấy rằng số bước lặp của quá trình luyện mạng lại giảm đi thật rõ rệt.

```
void TINHBUOCHOCVUOTKHE(void)
{
    float XL,XU,FL[SLMHL],temp;
    int i,t,j;
    if(NBS==0)
    {
        A=0.5;
        GAMA=0.1;
        TOCDOHOC=A;
    }
    for(t=0;t<SLMHL;t++)
        FL[t]=FX[t];
    XL=0;
BUOC1:
    TOCDOHOC=A;
    BUOCLAP();
    HAMMUCTIEU();
    for(t=0;t<SLMHL;t++)
        F[t] = ECM[t];
    for(t=0;t<SLMHL;t++)
        if(F[t]>FL[t])
        {
            XU=A;
            goto BUOC2;
        }
    //XL=A;
    for(t=0;t<SLMHL;t++)
```

```

        FL[t]=F[t];
    XL=A;
    A=1.5*A;
    XU=A;
    goto BUOC1;
BUOC2:
    if (FD>= (XU-XL) )
        goto BUOC3;
    A=XL+GAMA* (XU-XL) ;
    temp=TOCDOHOC;
    TOCDOHOC=A;
    BUOCLAP() ;
    HAMMUCTIEU() ;
    TOCDOHOC=temp;
    for (t=0;t<SLMHL;t++)
        F[t] = ECM[t];
    for (t=0;t<SLMHL;t++)
        if (FL[t]>F[t])
        {
            XU=A;
            for (j=0;j<SLMHL;j++)
                FL[j]=F[j];
            goto BUOC2;
        }
    for (t=0;t<SLMHL;t++)
        if (F[t]>FL[t])
        {
            for (j=0;j<SLMHL;j++)
                FL[j]=F[j];
            Xl=A;
            goto BUOC3;
        }
BUOC3:
    for (t=0;t<SLMHL;t++)
        FX[t]=F[t];
    NBS=NBS+1;
    TOCDOHOC=A;
}

```

2.3.3.4. Thủ tục huấn luyện mạng, HUANLUYENVUOTKHE ()

Trong thủ tục này có một vòng lặp do...while, mỗi vòng lặp là một bước lặp trong quá trình huấn luyện mạng. Một trong hai yếu tố kết thúc việc huấn luyện mạng là, nếu giá trị hàm mục tiêu đạt tới sai số epsilon=0.06, hoặc số bước lặp vượt quá 3000 lần lặp. Nếu yếu tố thứ nhất thỏa mãn thì mạng được huấn luyện xong và sẵn sàng cho người dùng sử dụng mạng, còn nếu gặp yếu tố thứ hai thì coi như việc huấn luyện mạng bị thất bại.

HUANLUYENVUOTKHE () sẽ gọi TINHBUOCHOCVUOTKHE () mỗi một lần lặp để có được bước học vượt khe, sau đó BUOCLAP () sẽ được gọi để lan truyền tiến, tính toán đầu ra. Một thủ tục DIEUCHINHTRONGSO () sẽ được gọi tiếp ngay sau đó để điều chỉnh các trọng số lớp ẩn và lớp ra dựa vào thông tin về sai số đầu ra so với tập mẫu đích cùng với bước học vượt khe.

Trong gói phần mềm cũng đưa thêm hai thủ tục, một là HUANLUYENCODINH (), thủ tục này sẽ huấn luyện mạng với bước học chọn cố định là 0.2 và thứ hai là HUANLUYENGIAMDAN () với bước học giảm dần tính theo công thức (2.20) để giúp ta so sánh với cách huấn luyện theo bước học tính theo nguyên lý vượt khe. Mục 2.3.4.2 trong chương này sẽ thống kê 20 lần huấn luyện mạng với ba thủ tục này.

$$toc_do_hoc = \frac{a}{so_buoc_lap*b+c} \quad (2.20)$$

Trong đó a, b, c là các hằng số.

2.3.4. Kết quả chạy chương trình và so sánh

2.3.4.1. Chạy chương trình

Sau khi lập trình bằng Visual C++, chạy chương trình, chúng ta cần lựa chọn một trong ba phương án từ bàn phím: c, g, v tương ứng với việc lựa chọn bước học cố định (viết tắt là c), bước học giảm dần theo công thức (2.20) (viết tắt là g), hay bước học vượt khe tính theo thuật toán vượt khe bằng thủ tục

TINHBUOCHOCVUOTKHE () (viết tắt là v) được sử dụng để luyện mạng. Cách thức nhập từ bàn phím để lựa chọn bước học được mô tả ngay sau đây:

LUA CHON LOAI BUOC HOC

CO DINH: c, GIAM DAN: g, NGUYEN LY VUOT KHE: v

v [enter]

Quá trình luyện mạng bắt đầu, nếu quá trình tìm kiếm bộ trọng số mạng thất bại thì chương trình sẽ thông báo rằng quá trình luyện mạng thất bại, còn nếu việc luyện mạng thành công thì chương trình sẽ cho chúng ta biết số bước lặp của quá trình luyện mạng; kết quả của hai ma trận trọng số lớp ẩn và lớp ra và yêu cầu chúng ta đưa vec-tơ x đầu vào để kiểm tra mạng. Cách thức nhập vec-tơ x từ bàn phím như sau (gồm có 7 hàng, mỗi hàng 5 giá trị; giá trị hoặc 0 hoặc là 1), ví dụ ta nhập mã của số 6

0 0 1 1 0

0 1 0 0 0

1 0 0 0 0

1 1 1 1 0

1 0 0 0 1

1 0 0 0 1

0 1 1 1 0

Và chúng ta chờ câu trả lời của mạng.

* CHUONG TRINH HUAN LUYEN MANG NO-RON *

* BUOC HOC TINH THEO NGUYEN LY VUOT KHE *

DANG HUAN LUYEN MANG THEO BUOC VUOT KHE...

MANG DA DUOC HUAN LUYEN XONG SAU: 34 BUOC LAP!

MA TRAN TRONG SO LOP AN MTTSLA[slnrlv][slnrla]:

-0.513496 +0.764902 +0.833890 -1.213335 +0.821022

-0.714510 -0.330939 +0.718113 -0.010856 +1.041344
 +0.203121 -0.493358 -0.615304 +1.198389 +1.225473
 +0.680735 +0.133827 -1.207137 -0.042108 +1.715010
 +0.013734 -0.783505 +0.020761 +0.770050 -0.108593
 +0.823171 -1.643064 +1.088796 -1.139771 -0.177110
 +0.773920 +0.239387 -1.654652 +0.578060 -0.869230
 +0.727297 -0.028404 +0.788829 -1.379023 -1.399395
 +0.630254 +0.221009 -0.569163 +0.697461 +1.071346
 -0.596292 -0.348468 -0.012247 +0.122078 +1.245557
 -1.321880 -0.141454 -0.235088 +2.864328 +1.306939
 +0.129423 +0.415885 -0.756748 +0.563398 +0.069821
 +0.516451 +0.032283 +0.209667 -0.963300 -0.187824
 +1.728189 -0.967244 -1.690552 -0.385068 -0.347820
 +1.109388 +0.452760 -0.649945 -1.479361 -0.492071
 -0.545680 +0.580958 -0.643666 -0.058043 +0.681030
 -0.139105 +0.502492 -0.103526 -0.416014 +1.761168
 -0.466114 +1.729941 +0.325881 +0.715679 -0.409421
 -0.666974 +1.983714 +0.425334 -0.192603 +1.008505
 -0.766750 +0.952648 -0.091599 -0.618372 +0.769775
 +0.390731 -0.222322 -1.175726 -0.874193 -0.480719
 +0.303599 -0.226470 +0.460789 -0.324308 -0.687494
 -0.466552 -0.199729 +0.305401 -0.112127 -0.616490
 -1.078721 +0.571089 +1.299650 -0.068734 +0.194324
 -1.218586 +1.362693 +0.992297 +1.284863 +0.102053

-0.601627 +0.353629 +1.566376 -0.162777 -1.226421
 +0.335808 +0.359233 -0.639406 +1.286489 -0.565971
 +0.091049 +0.309190 -0.607970 -0.996621 +0.297518
 -0.203598 +0.343273 +0.885806 -1.437262 +0.819597
 -0.382919 +0.682280 +0.220937 +0.767746 -2.170041
 +0.120224 +0.210313 +0.441168 +0.792983 -1.223393
 +0.468991 +0.842258 -1.504078 +0.576556 +0.084106
 -0.352618 -1.862809 +0.389202 +1.284403 +0.617516
 -0.908492 -1.645394 +1.693434 -0.538605 +0.292108
 +0.802787 +1.271673 -0.906446 +1.124133 -0.188477

MA TRAN TRONG SO LOP RA MTSLR[slnrla][slnrlr]:

+2.951620 -4.526521 -3.790868 -2.230710 -1.738504
 -2.769717 +1.312588 -4.664436 -2.827789 +2.371747
 -0.364274 +2.201062 -3.916823 -3.320487 -4.069728
 -1.782830 -4.044702 +3.170280 -4.158247 -3.187445
 -6.282814 +0.281494 -1.669756 +1.434243 +1.132807
 -2.987375 -3.486474 -0.478021 -4.107324 +4.076324
 -1.912957 -2.763546 -3.092701 +1.134861 +2.352585
 -5.310641 +3.295428 +0.162167 -2.746308 -2.727656
 -2.506175 -2.950514 +0.563975 +2.650147 -2.085773
 -2.361584 -0.225960 -4.947299 +3.709565 -3.014404

FINISH.

2.3.4.2. So sánh các phương án

Chúng ta sẽ lần lượt luyện mạng theo ba phương án, phương án thứ nhất là bước học cố định bằng 0.2, phương án thứ hai là bước học giảm dần (bắt đầu từ giá trị 1) sau mỗi bước lặp theo công thức (2.20), phương án thứ ba là bước học tính theo nguyên lý vượt khe. Mỗi phương án, chúng ta thử luyện 20 lần cho một tập hồ sơ luyện mạng. Kết quả cho ta bảng 2.2.

Với bước học cố định, ta thấy rằng số bước lặp cần có để mạng được huấn luyện thành công là rất lớn, trung bình là 10000 chu kỳ, nguyên nhân có thể do bước học chọn là bé (0.2). Tuy nhiên, nếu thử chọn bước học lớn hơn (0.3) thì kết quả là số lần luyện mạng thất bại nhiều hơn. Như trong *bảng 2.2* thống kê thì đã bảy lần thất bại trong tổng số 20 lần luyện mạng với bước học là 0.2.

Với bước học tính theo công thức (2.20) thì ba lần thất bại, số bước lặp để luyện mạng thành công khá ổn định, tuy nhiên chúng ta cũng thấy rằng, theo *bảng 2.2* đã thống kê thì với bước học tính theo nguyên lý vượt khe, tốc độ hội tụ cao hơn với trung bình 37 bước lặp ta đã luyện mạng xong, số lần thất bại khi luyện mạng cũng được giảm đi.

Một nhược điểm của phương án tính bước học vượt khe là chi phí thời gian để máy tính xử lý tính toán bước học trong mỗi bước lặp lớn do ta định nghĩa hằng số $FD=1-e4$ nhỏ, thuật toán sẽ phải lặp nhiều lần để thoát khỏi điều kiện này (bước 2 của thuật toán vượt khe). Tuy nhiên, về tổng chi phí thời gian luyện mạng thì lại có lợi hơn.

Bảng 2.2: Tập hồ sơ mẫu đầu vào {0 1 2 3 4 5 6 7 8 9}

T	Bước học cố định 0.2	Bước học giảm dần từ 1	Bước vượt khe
1	Thất bại	Thất bại	Thất bại
2	7902 (bước lặp)	3634 (bước lặp)	23 (bước lặp)
3	7210	2416	50
4	12370	2908	34
5	Thất bại	2748	31
6	9700	3169	42
7	Thất bại	2315	43
8	10073	2375	33
9	11465	Thất bại	34
10	8410	2820	33
11	10330	2618	32
12	Thất bại	2327	39
13	Thất bại	3238	44
14	9652	2653	Thất bại
15	11980	2652	31
16	12607	Thất bại	53
17	Thất bại	2792	31
18	8165	2322	42
19	10130	2913	42
20	Thất bại	2689	33
Tổng g kết	TB: 10000 bước lặp, 7 thất bại/20	Trung bình: 2740 bước lặp, 3 thất bại/20	TB: 37 bước lặp, 2 thất bại/20

Bảng 2.3: Tập hồ sơ mẫu đầu vào { a b c d e g h i k l }

T	Bước học cố định 0.2	Bước học giảm dần từ 1	Bước vượt khe
1	11212 (bước lặp)	Thất bại	41 (bước lặp)
2	Thất bại	3735 (bước lặp)	28
3	8211	2436	49
4	11365	2868	34
5	8871	2848	35
6	Thất bại	Thất bại	26
7	Thất bại	2341	33
8	11120	2165	36
9	10129	2769	35
10	8860	3220	32
11	9816	2210	32
12	Thất bại	2727	37
13	9712	3018	Thất bại
14	Thất bại	2571	44
15	10010	2541	37
16	11368	3186	33
17	Thất bại	2146	39
18	10923	Thất bại	32
19	Thất bại	2923	45
20	Thất bại	2678	31
Tổng g kết	TB: 10133 bước lặp, 8 thất bại/20	Trung bình: 2728 bước lặp, 3 thất bại/20	TB: 36 bước lặp, 1 thất bại/20

2.4. Kết luận chương 2

Trong chương 2, tác giả đã giới thiệu về một thuật toán mới để tìm bước học, phù hợp cho mặt lồi có dạng khe là thuật toán vượt khe. Để có thể tìm được lời giải tối ưu cho bài toán sử dụng mạng nơron có mặt lồi dạng lòng khe, tác giả đã đưa ra mô hình kết hợp thuật toán vượt khe và lan truyền ngược. Đó là cơ sở để cài đặt thành công thủ tục huấn luyện mạng theo phương pháp vượt khe kết hợp với kỹ thuật lan truyền ngược để tìm bộ trọng số tối ưu. Để chứng minh cho đề xuất này tác giả đã đưa ra một ví dụ về nhận dạng chữ viết tay và có sự so sánh giữa bước học vượt khe với các bước học khác thường hay được sử dụng trong Toolbox của Matlab. Các kết quả mô phỏng cho thấy sự đúng đắn của đề xuất này.

CHƯƠNG 3: ĐỀ XUẤT MÔ HÌNH KẾT HỢP GIẢI THUẬT DI TRUYỀN VÀ THUẬT TOÁN VƯỢT KHE ĐỂ CẢI TIẾN QUÁ TRÌNH HỌC CỦA MẠNG NƠON MLP CÓ MẶT LỖI ĐẶC BIỆT

Tóm tắt: Trong chương này, tác giả sẽ trình bày về việc ứng dụng giải thuật di truyền để tìm bộ trọng số khởi tạo ban đầu và vấn đề cài đặt thuật toán này kết hợp với thuật toán vượt khe nhằm nâng cao khả năng và tốc độ hội tụ trong quá trình luyện mạng nơon.

3.1. Đặt vấn đề

3.2. Luyện mạng nơon kết hợp thuật toán vượt khe và giải thuật di truyền

3.3. Áp dụng mô hình kết hợp giải thuật di truyền và thuật toán vượt khe trong quá trình luyện mạng nơon vào bài toán nhận dạng

3.4. Kết luận chương 3

3.1. Đặt vấn đề

Trong quá trình luyện mạng nơon, hai yếu tố ảnh hưởng mạnh mẽ đến việc tìm được bộ trọng số tối ưu của mạng là bước học và vec-tơ khởi tạo trọng số ban đầu. Trong các nghiên cứu nhằm cải thiện thuật toán, người ta thường tìm cách thay đổi bước học để cho phép có thể vượt qua những cực trị địa phương. Không có một giá trị bước học xác định nào cho các bài toán khác nhau. Với mỗi bài toán, bước học thường được lựa chọn bằng thực nghiệm theo phương pháp thử và sai, hoặc sẽ có bước học phù hợp với từng dạng bài toán riêng biệt. Với bài toán mà mặt lỗi có dạng lòng khe, chương 2 đã đề xuất việc sử dụng thuật toán vượt khe để tìm bước học phù hợp với mặt lỗi dạng này. Còn một nhân tố khác là bộ trọng số khởi tạo ban đầu, nó có ảnh hưởng cụ thể thế nào đến kết quả của luyện mạng nơon, đặc biệt khi mặt lỗi có dạng lòng khe. Để đánh giá nhân tố này, tác giả thử đi luyện mạng nơon trong một số trường hợp sau:

3.1.1. Khảo sát độ hội tụ của quá trình luyện mạng nơron bằng kỹ thuật lan truyền ngược nguyên thủy với các bộ khởi tạo trọng số ban đầu khác nhau.

Kỹ thuật lan truyền ngược ở đây là lan truyền ngược lỗi trong mạng, hàm lỗi thường chọn là hàm mà nó tối thiểu hoá được sai số trung bình bình phương. Cách thức hiệu chỉnh trọng số của thuật toán là ngược hướng với vectơ Gradient của hàm sai số trung bình bình phương. Đối với mạng nơron nhiều lớp thì hàm sai số trung bình bình phương thường phức tạp và có nhiều cực trị cục bộ. Các giá trị khởi tạo của các trọng số ảnh hưởng rất mạnh đến lời giải cuối cùng. Nếu các trọng số được khởi tạo với giá trị lớn thì ngay từ đầu tổng tín hiệu vào đã có giá trị tuyệt đối lớn và làm cho đầu ra của mạng chỉ đạt 2 giá trị 0 và 1. Điều này làm cho hệ thống sẽ bị tắc tại một cực tiểu cục bộ hoặc tại một vùng bằng phẳng nào đó gần ngay điểm xuất phát. Các trọng số này thường được khởi tạo bằng những số ngẫu nhiên nhỏ. Theo nghiên cứu của Wessels và Barnard [42], thì việc khởi tạo các trọng số liên kết w_{ij} nên trong phạm vi $[-3/\sqrt{k_i}, 3/\sqrt{k_i}]$ với k_i là số liên kết của các nơron j tới nơron i .

Hiện nay, bộ công cụ Neural Network Toolbox đã tích hợp sẵn một số thuật toán luyện mạng và bước học khác nhau để chúng ta lựa chọn; còn bộ trọng số ban đầu phục vụ cho quá trình luyện mạng đều lấy ngẫu nhiên trong một khoảng nào đó.

Để thấy rõ được sự ảnh hưởng của vec-tơ khởi tạo trọng số ban đầu đến độ hội tụ của quá trình luyện mạng nơron ta xét hai ví dụ sau:

a). Xét hệ thống phi tuyến tĩnh cần nhận dạng có mô hình toán học như sau:

$$y(u) = 0.6 \sin(\pi.u) + 0.3 \sin(3.\pi.u) + 0.1 \sin(5.\pi.u)$$

Chúng ta phát tín hiệu $u(k) = \sin(2\pi.k/250)$ vào hệ thống trên và đo tín hiệu ra $y(k)$. Sử dụng bộ mẫu $(u(k), y(k))$ này để luyện mạng.

Mạng nơron được dùng là mạng truyền thẳng 3 lớp, có một đầu vào, một đầu ra. Lớp nhập có 8 neural, lớp ẩn có 8 neural, lớp ra có 1 neural, hàm kích hoạt của cả 3 lớp đều là hàm tansig. Sai số cho phép để luyện mạng thành công là 10^{-5} . Ta sử dụng kỹ thuật lan truyền ngược với bước học cố định bằng 0.2.

Gọi $IW^{1,1}$ là ma trận trọng số lớp nhập, ma trận có 1 hàng 8 cột. Gọi $LW^{2,1}$ là ma trận trọng số lớp ẩn, ma trận có 8 hàng, 8 cột. Gọi $LW^{3,2}$ là ma trận trọng số lớp ra, ma trận có 8 hàng, 1 cột. Với mỗi lần luyện mạng khác nhau tức với bộ trọng số ban đầu $[IW^{1,1}, LW^{2,1}, LW^{3,2}]$ lựa chọn ngẫu nhiên khác nhau chúng ta lại thu được một bộ trọng số tối ưu khác nhau, số kỷ nguyên luyện mạng (KNLM) cũng khác nhau. Cụ thể:

Bảng 3.1

TT	KNLM	Sai số (10^{-6})	TT	KNLM	Sai số (10^{-6})
1	66	9.8065	8	24	9.9681
2	11	5.8464	9	45	9.1789
3	28	9.8923	10	62	9.5743
4	22	9.4931	11	55	9.2574
5	46	9.9981	12	37	9.6842
6	29	9.9062	13	29	7.1969
7	207	9.5439	14	60	9.2586

Căn cứ vào *bảng 3.1* ta thấy với một thuật toán không đổi, cấu trúc, tham số của mạng chọn như nhau thì kết quả của quá trình luyện mạng phụ thuộc vào bộ khởi tạo trọng số ban đầu.

b). Xét hệ thống động học phi tuyến cần nhận dạng có mô hình toán học như sau:

$$\dot{y} = 0.00005 - 0.05y - 0.0005u - 0.5uy$$

Chúng ta phát một tín hiệu ngẫu nhiên có giới hạn về biên độ từ 0 đến 2L/sec với thời gian lấy mẫu là 0.1s vào hệ thống trên và đo tín hiệu ra. Lấy tập mẫu vào, ra này để luyện mạng, Tổng thời gian đặt là 100 s, do đó sẽ tạo ra được 1000 bộ mẫu vào ra dưới dạng một mảng dữ liệu.

Cấu trúc mạng nơron được chọn như sau:

Mạng gồm có hai lớp: Lớp vào có 4 nơron, hàm kích hoạt là hàm tansig; lớp ra có 1 nơron, hàm kích hoạt là hàm purelin.

$IW^{1,1}$ là ma trận trọng số lớp nhập, ma trận có 1 hàng 4 cột.

$LW^{2,1}$ là ma trận trọng số lớp ẩn, ma trận có 4 hàng, 1 cột.

$LW^{1,2}$ là ma trận trọng số mạch vòng phản hồi từ đầu ra trở lại đầu vào, ma trận có 1 hàng, 4 cột. Ta sử dụng kỹ thuật lan truyền ngược với bước học cố định bằng 0.2. Sai số cho phép để luyện mạng thành công là 10^{-12} .

Với mỗi lần luyện mạng khác nhau tức với bộ trọng số ban đầu [$IW^{1,1}$, $LW^{2,1}$, $LW^{1,2}$] lựa chọn ngẫu nhiên khác nhau chúng ta lại thu được một bộ trọng số tối ưu khác nhau, số kỷ nguyên luyện mạng (KNLM) cũng khác nhau. Cụ thể:

Bảng 3.2:

TT	KNLM	Sai số (10^{-12})	TT	KNLM	Sai số (10^{-12})
1	210	9.2147	8	301	8.9754
2	151	9.6782	9	229	9.2367
3	234	8.6745	10	234	9.2476
4	193	9.3657	11	167	9.9874
5	271	9.2486	12	205	9.5789
6	146	7.6842	13	212	9.3487
7	231	8.6575	14	203	9.3578

Căn cứ vào *bảng 3.2* ta thấy với một thuật toán không đổi, cấu trúc, tham số của mạng chọn như nhau thì kết quả của quá trình luyện mạng phụ thuộc vào bộ khởi tạo trọng số ban đầu.

3.1.2. Khảo sát độ hội tụ của quá trình luyện mạng nơron có mặt lỗi đặc biệt bằng kỹ thuật lan truyền ngược kết hợp thuật toán vượt khe với các bộ khởi tạo trọng số ban đầu khác nhau.

Khi sử dụng mạng nơron để xấp xỉ một số đối tượng phi tuyến, có thể dẫn đến mặt lỗi khi luyện mạng có dạng lòng khe [27], [28]. Với những đối tượng phức tạp này khi xấp xỉ ta cần chọn mạng nơron có nhiều lớp và đặc biệt cần chọn hàm

kích hoạt là hàm sigmoid để dễ sinh ra mặt lỗi có nhiều cực trị cục bộ và có dạng lòng khe.

Ở chương 1, tác giả đã chứng minh được rằng khi sử dụng bộ công cụ Neural Network Toolbox để luyện mạng nơron có mặt lỗi đặc biệt này thì mạng hội tụ rất chậm thậm chí không hội tụ. Trong chương 2, tác giả đã đề xuất về thuật toán vượt khe và phương pháp tính bước học vượt khe để cập nhật trọng số của mạng nơron. Có thể nhận thấy rằng bước học vượt khe ưu việt hơn hẳn các phương pháp cập nhật bước học khác như bước học cố định, bước học giảm dần qua bảng thống kê kết quả luyện mạng. Cụ thể số lần luyện mạng thất bại và số kỷ nguyên luyện mạng giảm đi đáng kể. Trong phần này, vẫn sử dụng kỹ thuật lan truyền ngược kết hợp với thuật toán vượt khe để luyện mạng nơron có mặt lỗi dạng lòng khe, tác giả sẽ đi đánh giá sự ảnh hưởng của bộ khởi tạo trọng số ban đầu đến vấn đề tìm nghiệm tối ưu toàn cục.

Để minh họa, nhóm tác giả vẫn đề xuất cấu trúc mạng nơron để nhận dạng các chữ số: 0, 1, 2, ..., 9. Trong đó hàm sigmoid được sử dụng với mục đích sinh ra mặt sai số có dạng lòng khe.

Để biểu diễn các chữ số, chúng ta sử dụng một ma trận $5 \times 7 = 35$ để mã hóa cho mỗi ký tự. Tương ứng với mỗi vector đầu vào x là một vector có kích thước 35×1 , với các thành phần nhận các giá trị hoặc 0 hoặc 1. Như vậy, ta có thể lựa chọn lớp nơron đầu vào có 35 nơron. Để phân biệt được mười ký tự, chúng ta cho lớp đầu ra của mạng là 10 nơron. Đối với lớp ẩn ta chọn 5 nơron, ta có cấu trúc mạng như *hình 1.5*.

Hàm f được chọn là hàm sigmoid vì thực tế hàm này cũng hay được dùng cho mạng nơron nhiều lớp và hơn nữa do đặc điểm của hàm sigmoid rất dễ sinh ra mặt sai số có dạng lòng khe hẹp. Phương trình của hàm sigmoid là: $f = 1 / (1 + \exp(-x))$

Hàm sai số sử dụng cho luyện mạng: $J = 0.5 * (z - t)^2$ với z là đầu ra của nơron lớp ra và t là giá trị đích mong muốn.

Bộ trọng số khởi tạo ban đầu với mạng 3 lớp gồm có ma trận trọng số lớp ẩn có kích thước là 35×5 và ma trận trọng số lớp ra có kích thước là 5×10 được lấy là một số ngẫu nhiên xung quanh điểm 0.5 là trung điểm của hàm kích hoạt sigmoid. Sau khi lập trình và cho luyện mạng 14 lần ta có được *bảng 3.3*.

Bảng 3.3

TT	KNLM	TT	KNLM
1	37	8	35
2	Thất bại	9	29
3	42	10	46
4	33	11	38
5	35	12	39
6	28	13	Thất bại
7	44	14	30

Căn cứ vào *bảng 3.3* ta thấy với một thuật toán không đổi, cấu trúc, tham số của mạng chọn như nhau thì kết quả của quá trình luyện mạng phụ thuộc vào bộ khởi tạo trọng số ban đầu, thậm chí còn có 2 lần luyện mạng thất bại trong tổng số 14 lần luyện mạng. Điều đó được giải thích: do bản chất của giải thuật học lan truyền ngược sai số là phương pháp giảm độ lệch gradient nên việc khởi tạo giá trị ban đầu của bộ trọng số các giá trị nhỏ ngẫu nhiên sẽ làm cho mạng hội tụ về các giá trị cực tiểu khác nhau. Nếu gặp may thì mạng sẽ hội tụ được về giá trị cực tiểu tổng thể, còn nếu không mạng có thể rơi vào cực trị địa phương và không thoát ra được dẫn đến luyện mạng thất bại.

Như vậy, tác giả đã đi phân tích sự ảnh hưởng của vec-tơ khởi tạo trọng số ban đầu trong quá trình luyện mạng nơron. Sự ảnh hưởng đó được đánh giá trong 3 ví dụ đặc trưng cho việc xấp xỉ các đối tượng khác nhau: phi tuyến tĩnh, động học phi tuyến và phi tuyến đặc biệt. Thông qua việc nghiên cứu và thực nghiệm trên máy tính cho ta thấy: Với các mặt lỗi thông thường việc khởi tạo bộ trọng số ban đầu ngẫu nhiên trong một khoảng nào đó chỉ ảnh hưởng đến thời gian luyện mạng; còn với mặt lỗi đặc biệt có nhiều cực trị và dạng lòng khe, nó còn có thể làm cho

quá trình luyện mạng thất bại do rơi vào cực trị cục bộ vì xuất phát từ vùng không chứa cực trị toàn cục. Đây là một kết luận quan trọng, làm tiền đề cho việc đề xuất phương pháp tính toán bộ khởi tạo trọng số ban đầu thay cho việc khởi tạo ngẫu nhiên, từ đó tăng độ chính xác và tốc độ hội tụ của quá trình luyện mạng nơron.

3.2. Đề xuất mô hình kết hợp giải thuật di truyền và thuật toán vượt khe trong quá trình luyện mạng nơron

3.2.1. Đặt vấn đề

Quá trình luyện mạng nơron thực chất là giải bài toán tối ưu nhằm cập nhật các trọng số sao cho hàm lỗi đạt cực tiểu, hoặc nhỏ hơn một giá trị cho phép nào đó.

Thuật toán hiện nay thường được sử dụng trong quá trình luyện mạng nơron là thuật toán gradien liên hợp hay thuật toán Levenberg - Marquardt với kỹ thuật lan truyền ngược và còn có thể gọi là kỹ thuật lan truyền ngược.

Kỹ thuật lan truyền ngược hội tụ đến một giải pháp mà nó tối thiểu hoá được sai số trung bình bình phương vì cách thức hiệu chỉnh trọng số và hệ số bias của thuật toán là ngược hướng với vectơ Gradient của hàm sai số trung bình bình phương đối với trọng số. Tuy nhiên, đối với mạng MLP có mặt chất lượng dạng lòng khe thì hàm sai số trung bình bình phương thường phức tạp và có nhiều cực trị cục bộ, vì thế các phép lặp huấn luyện mạng có thể chỉ đạt được đến cực trị cục bộ của hàm sai số trung bình bình phương mà không đạt đến được cực trị tổng thể. Các giá trị khởi tạo của các trọng số ảnh hưởng rất mạnh đến lời giải cuối cùng. Các trọng số này thường được khởi tạo bằng những số ngẫu nhiên nhỏ. Việc khởi tạo tất cả các trọng số bằng nhau sẽ làm cho mạng học không tốt. Nếu các trọng số được khởi tạo với giá trị lớn thì ngay từ đầu tổng tín hiệu vào đã có giá trị tuyệt đối lớn và làm cho hàm sigmoid chỉ đạt 2 giá trị 0 và 1. Điều này làm cho hệ thống sẽ bị tắc ngay tại một cực tiểu cục bộ hoặc tại một vùng bằng phẳng nào đó gần ngay tại điểm xuất phát. Giá trị khởi động ban đầu của các trọng số trên lớp thứ 1 của mạng sẽ được chọn ngẫu nhiên nhỏ trong khoảng $[-1/n, 1/n]$, trong đó n là số trọng số nối tới lớp 1. Do bản chất của giải thuật học lan truyền ngược sai số là phương pháp giảm độ lệch gradient nên việc khởi động các giá trị ban đầu của các trọng số các giá trị nhỏ ngẫu

nhiên sẽ làm cho mạng hội tụ về các giá trị cực tiểu khác nhau. Nếu gặp may thì mạng sẽ hội tụ được về giá trị cực tiểu tổng thể.

Xu thế hiện nay của công nghệ thông tin là kết hợp ưu điểm của các kỹ thuật riêng lẻ. Các kỹ thuật mạng nơron, thuật giải di truyền, logic mờ, ... đang được kết hợp với nhau để hình thành công nghệ tính toán mềm.

Các nghiên cứu về GA kết hợp với ANN bắt đầu bởi Montana and Davis. Năm 1989 các ông đã có báo cáo về việc ứng dụng thành công GA trong mạng ANN. Họ đã chứng minh được rằng GA tìm được bộ trọng số tối ưu tốt hơn BP trong một số trường hợp. Từ đó đến nay các nghiên cứu về sự kết hợp này đã chứng minh được tính ưu việt của nó.

Các nghiên cứu kết hợp GA và ANN (xem thêm trong [14]) gồm:

- Dùng GA để tiền xử lý đầu vào cho ANN:
 - + Chọn dữ liệu (phương pháp biểu diễn dữ liệu, rút gọn dữ liệu) tối ưu khi không có nhiều thông tin về dữ liệu, ...
 - + Khởi tạo bộ trọng cho ANN
- Dùng GA để hậu xử lý đầu ra cho một hoặc nhiều ANN: tìm bộ trọng số tổng hợp kết quả tối ưu từ kết quả của các mô hình ANN thành viên (đã huấn luyện) trong kiến trúc tổng hợp giúp ra quyết định, ...
- GA dùng trong các mô đun độc lập tác động đến kết quả của ANN: thay thế kỹ thuật lan truyền ngược.
- Dùng GA để xác định: kiến trúc, các tham số điều khiển ANN, ...

Để so sánh giải thuật di truyền và lan truyền ngược sai số, ta sử dụng lại bài toán nhận dạng chữ viết đã trình bày trong các chương trước, chọn tham số chung cho cả hai phương pháp:

- Mạng nơron sử dụng là mạng một lớp ẩn
- Số neural trong lớp ẩn: 5
- Ngưỡng sai số dừng lặp: 0.1 hoặc quá 20000 vòng lặp

Tham số của thuật lan truyền ngược sai số:

- Bước học: 0.2

Tham số của giải thuật di truyền:

- Số lượng quần thể: 20

- Xác suất lai: 0.46

- Xác suất đột biến: 0.1

Sau đây là bảng thống kê số bước lặp để mạng hội tụ với mỗi phương án trong 20 lần thử nghiệm khác nhau.

(-) : mạng không hội tụ (số lần lặp lớn hơn 20000)

Bảng 3.4: So sánh GA và BP với sai số là 0.1

TT	GA	BP	TT	GA	BP
1	1356	-	12	865	1890
2	729	3156	13	-	2348
3	1042	2578	14	758	-
4	1783	3640	15	-	2647
5	-	-	16	968	3378
6	879	-	17	1034	-
7	1102	2102	18	779	3018
8	-	2671	19	890	2781
9	891	-	20	904	2585
10	902	2470		TB: 4	TB: 6 thất
11	728	3018		thất bại	bại

Ta thấy rằng giải thuật di truyền có khả năng đạt được yêu cầu về hội tụ (sai số ≤ 0.1) tức tìm vùng chứa cực trị toàn cục dễ dàng hơn so với kỹ thuật lan truyền ngược sai số. Hay nói cách khác kỹ thuật lan truyền ngược sai số dễ rơi vào vùng

chứa cực tiểu cục bộ hơn giải thuật di truyền. Trong 20 lần chạy, GA chỉ có 4 lần không tìm được cực trị toàn cục trong khi đó BP là 6 lần.

Vấn bài toán trên ta thay đổi ngưỡng sai số dừng lặp là 0.001 ta được bảng sau:

Bảng 3.5: So sánh GA và BP với sai số là 0.001

TT	GA	BP	TT	GA	BP
1	-	8019	12	3012	-
2	-	9190	13	-	8601
3	3021	-	14	-	11032
4	-	8701	15	-	9963
5	-	-	16	-	3378
6	2371	10923	17	-	9021
7	-	8971	18	-	
8	-	9801	19	-	-
9	-	-	20	-	10914
10	-	-		TB: 15 thất bại	TB 7 thất bại
11	2038	7781			

Qua kết quả này có thể nhận thấy rằng chỉ rất ít trường hợp GA đạt được giá trị sai số mong muốn. Kết hợp kết quả trong *bảng 3.4* và *3.5* ta có bảng so sánh khả năng hội tụ của mạng nơron khi thay đổi sai số dừng lặp.

Bảng 3.6: So sánh GA và BP với sai số khác nhau

Sai số dừng lặp	Số lần hội tụ trong 20 lần luyện mạng	
	GA	BP
0.1	16	14
0.001	4	13

Nhận xét 1: Nhờ cơ chế tìm kiếm trải rộng, ngẫu nhiên và mang tính chọn lọc tự nhiên nên: GA thường tìm ra được vùng chứa cực trị toàn cục, nhưng khó đạt được cực trị toàn cục. Một mặt ta muốn GA duy trì sự đa dạng quần thể (trải rộng không gian tìm kiếm) để tránh hội tụ sớm đến cực trị cục bộ; mặt khác, khi “*đã khoanh vùng được cực trị toàn cục*”, ta muốn GA thu hẹp vùng tìm kiếm để “*chỉ ra được cực trị toàn cục*”. Mục tiêu thứ nhất thường dễ đạt được bằng cách chọn hàm thích nghi và phương pháp tái tạo quần thể phù hợp. Để đạt được mục tiêu thứ hai đòi hỏi chúng ta phải chia quá trình tiến hóa thành hai giai đoạn, trong giai đoạn hai ta phải chỉnh lại: các toán tử lai, đột biến, tái tạo; phương pháp chọn lọc; đánh giá độ thích nghi; cũng như chỉnh sửa lại các tham số của quá trình tiến hóa để có thể đến cực trị toàn cục. Việc thực thi một mô hình như thế sẽ rất phức tạp. Do đó, cần phải kết hợp GA với các phương pháp tối ưu cục bộ khác.

Nhận xét 2: Các phương pháp học trong ANN thực hiện việc “tìm kiếm cục bộ” trong không gian trọng số (dựa trên thông tin về đạo hàm của lỗi) nên có hai nhược điểm. Thứ nhất bộ trọng số thu được thường không là tối ưu toàn cục. Thứ hai quá trình học có thể không hội tụ hoặc hội tụ rất chậm. Do đó, cần phải kết hợp các phương pháp học “*mang tính cục bộ*” của ANN với các thuật giải “*mang tính toàn cục*” như thuật giải di truyền.

Từ nhận xét 1 và 2, ta thấy rằng có thể kết hợp GA và ANN nhằm nâng cao hiệu quả của ANN. GA sẽ khoanh vùng chứa cực tiểu toàn cục của hàm lỗi, sau đó ANN xuất phát từ bộ trọng số này để tiến đến cực tiểu toàn cục.

Trong phần này sẽ trình bày về giải thuật di truyền (GA) kết hợp với thuật toán “vượt khe” để chế ngự quỹ đạo và rút ngắn thời gian của quá trình tìm kiếm tối ưu với mặt sai số phức tạp dạng lòng khe.

3.2.2. Thuật toán

Có nhiều cách để kết hợp giải thuật di truyền vào mạng nơron nhưng cách đơn giản và khá hiệu quả là ta thực hiện lai ghép hai giải thuật nối tiếp nhau.

Với một cấu trúc mạng cho trước, ta xuất phát bằng giải thuật di truyền, đi tìm tập các trọng số tốt nhất đối với mạng. Một quần thể N chuỗi được khởi tạo

ngẫu nhiên. Mỗi chuỗi là một bản mã hoá của một tập trọng số của mạng. Sau G thế hệ tiến hoá, 5% các cá thể tốt nhất trong G thế hệ sẽ được lưu giữ lại. Các cá thể này sau đó sẽ được giải mã và được đưa vào mạng nơron xây nên các mô hình để học. Sau quá trình học, tập trọng số nào cho kết quả dự báo tốt nhất sẽ được giữ lại làm thông số của mạng nơron cho việc dự báo đó.

1. Khởi tạo ngẫu nhiên tạo một quần thể ban đầu $P^0 = (a_1^0, a_2^0, \dots, a_\lambda^0)$
2. Tính toán giá trị thích nghi $f(a_i')$ của mỗi nhiễm sắc thể a_i' trong quần thể P' hiện tại.
3. Căn cứ vào giá trị thích nghi tạo ra các nhiễm sắc thể mới bằng cách chọn lọc các nhiễm sắc thể cha mẹ, áp dụng các thuật toán lai tạo và đột biến.
4. Loại bỏ nhiễm sắc thể có độ thích nghi kém để tạo chỗ cho quần thể mới.
5. Tính toán các giá trị thích nghi của các nhiễm sắc thể mới $f(a_i'^t)$ chèn vào quần thể.
6. Tăng số lượng các thế hệ nếu chưa đạt đến điều kiện kết thúc và lặp lại từ bước 3. Khi đạt đến điều kiện kết thúc thì dừng lại và đưa ra nhiễm sắc thể tốt nhất.

Các phép toán di truyền sử dụng trong các thực nghiệm được trình bày như sau:

Khởi tạo quần thể:

Quá trình này tạo ra ngẫu nhiên một bộ λ gen (λ là kích thước của quần thể), mã hóa các gen theo số thực ta được độ dài nhiễm sắc thể là L, tập hợp λ nhiễm sắc thể này sẽ tạo thành một quần thể ban đầu.

Hàm thích nghi:

Hàm thích nghi được sử dụng ở đây có dạng như sau:

$$TSSE = f = \sum_{i=1}^s \sum_{j=1}^m (t_{ij} - z_{ij}) \quad (3.1)$$

Trong đó s là tổng số các mẫu học, m là số lượng các nơ ron lớp ra, G là tổng bình phương lỗi của S mẫu và z_{ij} là đầu ra của mạng nơron.

Chọn lọc

Các giá trị thích nghi được tính toán và thực hiện phép chọn lọc bằng phương pháp lựa chọn bánh xe roulette. Kết quả là các cá thể với độ thích nghi cao được chọn vào thế hệ kế tiếp của quần thể.

Lai tạo

Phép lai tạo kết hợp các đặc điểm có trong cá thể cha mẹ hình thành nên cá thể con bằng cách phối ghép các đoạn tương ứng từ các thể cha mẹ. Vị trí lai tạo được lựa chọn tùy theo độ thích nghi trong mỗi thế hệ theo phương trình sau:

$$C_r = \text{ROUND}[F_{fit}(i, j) \times L] \in [0.....L] \quad (3.2)$$

Với $\text{ROUND}(\cdot)$ là hàm xác định số nguyên gần nhất thỏa mãn. Nếu vị trí lai tạo càng lớn các thể con sẽ chứa nhiều đặc điểm trong cá thể mẹ.

Đột biến

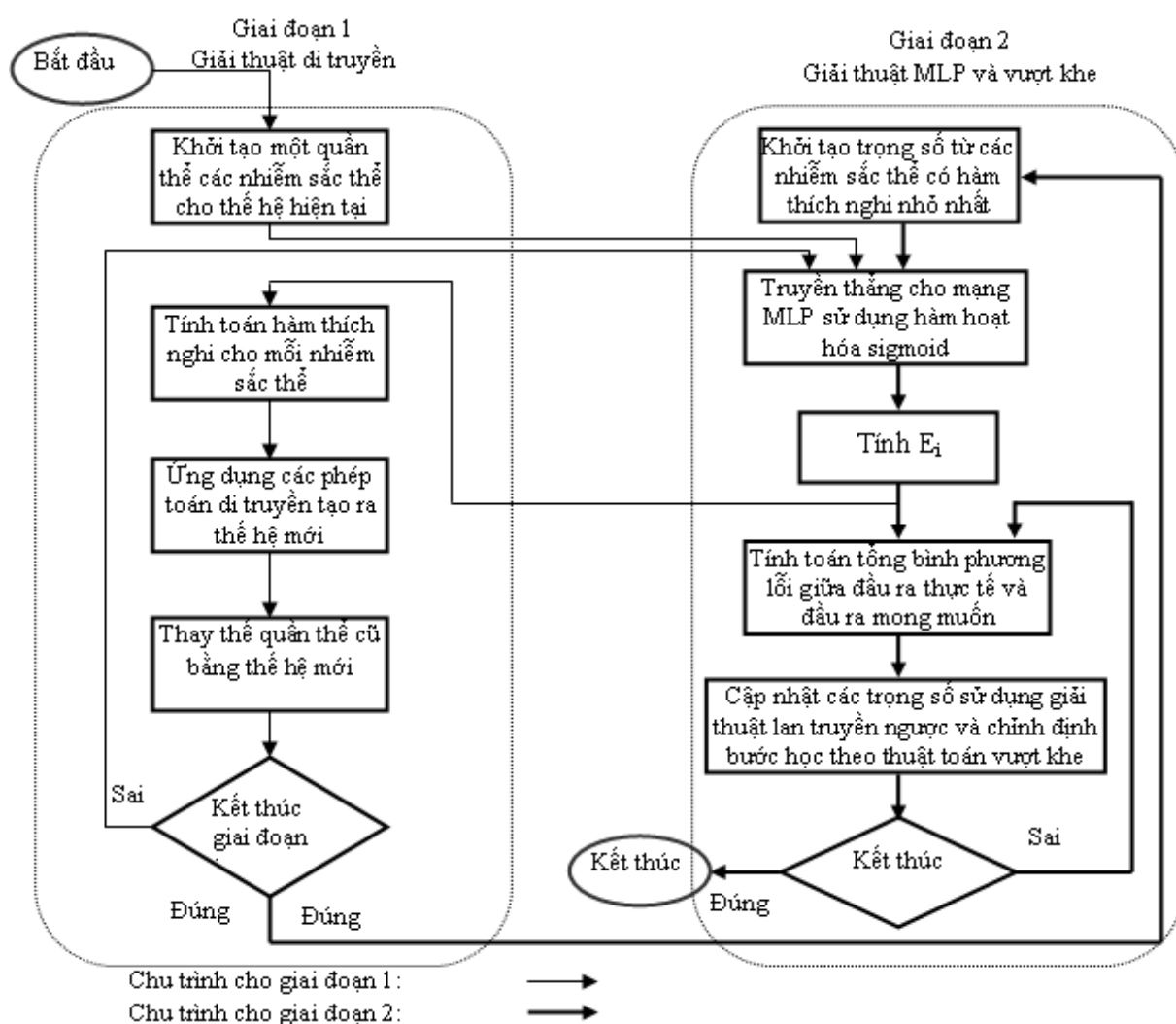
Để tránh rơi vào các điểm tối ưu cục bộ, các cá thể được thay đổi một cách ngẫu nhiên với vị trí đột biến M_r như sau:

$$M_r = \text{ROUND}[(L - C_r) \times M_b / L] \in \{0...M_b\} \quad (3.3)$$

Với M_b là giới hạn trên của vị trí đột biến. Theo thời gian, độ thích nghi sẽ dần tăng và các phép lai tạo đột biến cũng được thực hiện. Quá trình tiến hóa sẽ được thực hiện cho đến khi đạt đến độ thích nghi mong muốn.

Với việc lai ghép này, giải thuật lan truyền ngược sai số lược bỏ đi một số bước sau:

- Không khởi tạo các giá trị trọng số ban đầu vì tập trọng số đã được lấy từ kết quả của giải thuật di truyền.
- Thành phần quán tính trong các phương trình hiệu chỉnh trọng số là không cần thiết vì tập trọng số xuất phát đã khá gần lời giải; tác dụng chống dao động và thay đổi đột ngột các trọng số theo hướng khác với hướng của lời giải trở nên không cần thiết.



Hình 3.1: Sơ đồ thuật toán kết hợp giải thuật vượt khe và di truyền cho luyện mạng MP

Thuật toán kết hợp giải thuật vượt khe và giải thuật di truyền cho mạng MLP được đề xuất trong hình 3.1. Nó bao gồm hai giai đoạn luyện mạng. Giai đoạn đầu tiên sử dụng thuật toán di truyền với bước truyền thẳng nhằm đẩy nhanh toàn bộ quá trình luyện mạng. Thuật toán di truyền thực hiện tìm kiếm toàn cục và tìm kiếm tối ưu gần điểm ban đầu (trọng lượng vec-tơ) cho giai đoạn thứ hai. Trong đó, mỗi nhiễm sắc thể được sử dụng để mã hóa các trọng số của mạng nơron. Hàm thích nghi (hàm mục tiêu) cho các thuật toán di truyền được xác định là tổng bình phương lỗi (TSSE) của mạng nơron tương ứng. Do đó, bài toán sẽ trở thành tối ưu hóa không giới hạn nhằm tìm một tập hợp các biến quyết định giảm thiểu hàm mục tiêu. Trong giai đoạn thứ 2 sẽ sử dụng kỹ thuật lan truyền ngược với các bước học được thay đổi theo thuật toán vượt khe đã được đề xuất ở hình 2.4.

3.3. Áp dụng mô hình kết hợp giải thuật di truyền và thuật toán vượt khe trong quá trình luyện mạng nơron vào bài toán nhận dạng

Trở lại ví dụ về nhận dạng chữ viết tay 0,1,2,... 9.

Việc cài đặt thuật toán trên Matlab được tiến hành như sau:

/* Giai đoạn 1*/

Khởi tạo các nhiễm sắc thể một cách ngẫu nhiên cho thể hệ hiện tại, khởi tạo các tham số làm việc và đặt nhiễm sắc thể đầu tiên là nhiễm sắc thể tốt nhất *best_chromosome*.

a- Lặp từ $i=1$ đến kích thước quần thể, thực hiện công việc sau:

- Khởi tạo *sub_total_fitness* bằng 0 và *sub_best_chromosome* là rỗng

b- Lặp từ $j=1$ đến độ dài của nhiễm sắc thể, thực hiện các công việc sau:

- Thực hiện thủ tục truyền thẳng cho mạng MLP (sử dụng hàm hoạt hóa là sigmoid).

- Tính toán hàm mục tiêu (lỗi hệ thống của mạng nơron)

- Tính toán lỗi tổng cộng *total_fitness* bằng cách tích lũy *sub_total_fitness*

c- Lưu *best_chromosome* vào *sub_best_chromosome*

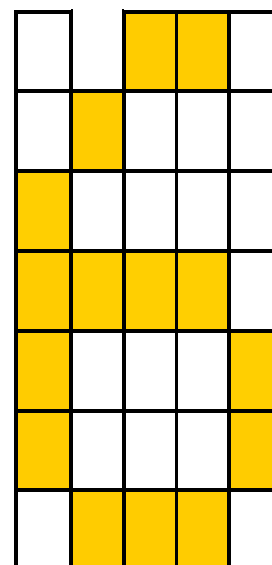
d- So sánh các *sub_best_chromosome* với nhau và đặt *sub_best_chromosome* lớn nhất là *best_chromosome*.

e- Lặp từ $i=0$ đến kích thước quần thể/2, thực hiện các thủ tục sau:

- Khởi tạo *sub_total_fitness* bằng 0 và *sub_best_chromosome* là rỗng

- Lặp từ $j=1$ tới độ dài nhiễm sắc thể, thực hiện các công việc sau:

* Chọn các nhiễm sắc thể cha mẹ sử dụng phương pháp lựa chọn theo bánh xe roulette



* Áp dụng các phép lai tạo và đột biến

- Lặp từ $k=1$ đến độ dài nhiễm sắc thể, thực hiện các công việc sau:

* Thực hiện thủ tục truyền thẳng cho mạng MLP

* Tính toán các giá trị hàm mục tiêu cho các nhiễm sắc thể cha mẹ.

- Tính toán *sub_total_fitness* bằng cách tích lũy giá trị hàm mục tiêu của mỗi nhiễm sắc thể.

- Lưu *best_chromosome* vào *sub_best_chromosome*

g- Thay thế thế hệ cũ bằng thế hệ mới nếu thỏa mãn điều kiện dừng.

/* Giai đoạn 2 */

- Đặt *best_chromosome* là véc tơ trọng số khởi tạo, thiết lập cấu trúc mạng nơron MLP.

- Tính toán đầu ra thực tế của mạng MLP truyền thẳng.

- Tính toán lỗi giữa đầu ra thực tế và đầu ra mong muốn.

- Cập nhật các trọng số bằng kỹ thuật lan truyền ngược, cập nhật các hệ số học bằng thuật toán vượt khe.

END

Các kết quả thực nghiệm khi luyện mạng MLP kết hợp giải thuật vượt khe và di truyền.

Mạng MLP được luyện với bộ các ký tự mẫu chữ với kích thước 7×5 được trình bày ở trên. Các giá trị ban đầu như số đầu vào (35), số lượng lớp ẩn (1), số nơron lớp ẩn (5), các kỹ thuật luyện mạng khác nhau, mã hóa đầu vào và đầu ra nhằm khởi tạo các trọng số đã được đề cập ở trên. Để kiểm tra khả năng của mạng cho quá trình nhận dạng chữ, chúng tôi đề xuất một tham số đánh giá chất lượng của mạng là tỷ lệ lỗi nhận dạng được tính theo công thức:

$$\text{Tỷ lệ lỗi (FR)} = \frac{\text{Số các kí tự đã kiểm tra} - \text{Số ký tự đã nhận dạng}}{\text{Số các kí tự đã kiểm tra}} \times 100\%$$

Các tham số luyện mạng:

Kích thước quần thể = 20 Xác suất lai tạo = 0.46 Mã hóa bằng số thực

Độ dài nhiễm sắc thể = 225 Độ chính xác mong muốn = 90% Số thế hệ: 20

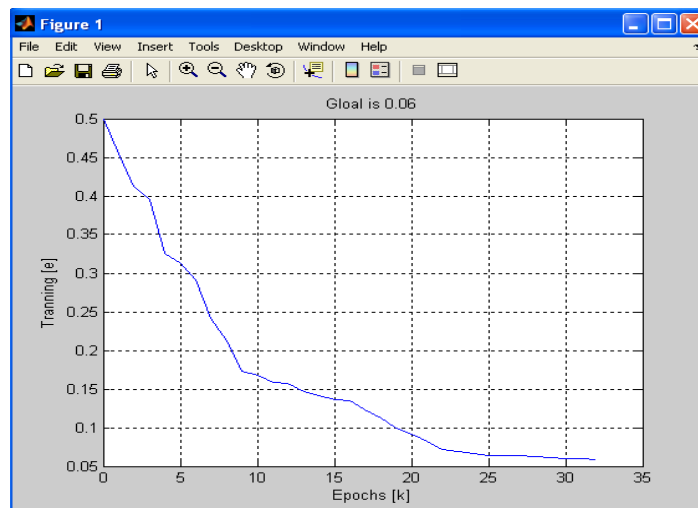
Lỗi hệ thống mong muốn=0.06

Kết quả luyện mạng như sau:

Số thế hệ	1	5	10	15	20
Tổng thích nghi	9.5563	8.1638	6.1383	5.724	5.697

Số chu kỳ luyện	5	10	15	20	33
Tỷ lệ lỗi	93.33%	60.33%	40.67%	37.33%	0%
TSSE	0.4956	0.3274	0.1387	0.0864	0.0589

Như vậy, sau 20 thế hệ đã đạt đến yêu cầu của bài toán. Giá trị thích nghi trung bình đạt được là 5.679. Kết quả của giai đoạn 1 được sử dụng để khởi tạo trọng số cho giai đoạn 2. Với sự thay đổi bước học theo giải thuật vượt khe, sau 33 chu kỳ luyện mạng lỗi hệ thống đã đạt đến mục đích 0.0589 và độ chính xác của quá trình nhận dạng là 100%. Hoạt động của mạng MLP có kết hợp giải thuật vượt khe và di truyền cho nhận dạng chữ được thể hiện trên hình 3.2



Hình 3.2: Hoạt động của mạng MLP cải tiến

3.4. Kết luận chương 3

Trong chương 3 chúng ta đã nghiên cứu các ví dụ cụ thể để thấy sự ảnh hưởng của vec-tơ khởi tạo trọng số ban đầu đến kết quả của quá trình luyện mạng. Đồng thời cũng đi phân tích đánh giá ưu nhược điểm của giải thuật di truyền và thuật học lan truyền ngược sai số trong bài toán nhận dạng. Giải thuật di truyền được biết đến như một giải thuật tìm kiếm dựa trên học thuyết về chọn lọc tự nhiên và nó cho phép ta đạt được tới cực trị toàn cục. Do đó, áp dụng giải thuật di truyền vào bài toán tối ưu hoá trọng số mạng nơron nhân tạo là một cách tiếp cận tiềm năng.

Trong chương này, tác giả đề xuất việc sử dụng giải thuật di truyền kết hợp với thuật toán “vượt khe” để cải tiến quá trình luyện mạng nơron có mặt lỗi đặc biệt và minh họa thông qua ứng dụng nhận dạng chữ. Có thể đánh giá được rằng phương pháp này đã tăng khả năng và tốc độ hội tụ của mạng nơron có mặt lỗi dạng “lòng khe”.

KẾT LUẬN CHUNG VÀ ĐỀ XUẤT HƯỚNG NGHIÊN CỨU

- **So sánh luyện mạng nơron có mặt lỗi đặc biệt với các phương pháp khác nhau**

Để thấy được hiệu quả của việc áp dụng giải thuật di truyền với thuật toán vượt khe trong quá trình luyện mạng nơron sử dụng kỹ thuật lan truyền ngược, tác giả đã đưa ra một ví dụ trong suốt các chương của luận án là bài toán nhận dạng chữ viết tay.

Ở chương 2, bài toán nhận dạng chữ viết được lập trình trên phần mềm C++. Trong ví dụ mạng nơron được luyện với 3 phương pháp có bước học khác nhau lần lượt là bước học cố định, bước học giảm dần và bước học vượt khe.

Với bước học cố định, ta thấy rằng số bước lặp cần có để mạng được huấn luyện thành công là rất lớn, trung bình là 10000 chu kỳ, nguyên nhân có thể do bước học chọn là bé (0.2). Tuy nhiên, nếu thử chọn bước học lớn hơn (0.3) thì kết quả là số lần luyện mạng thất bại nhiều hơn. Như trong *bảng 2.2* thống kê thì đã bảy lần thất bại trong tổng số 20 lần luyện mạng với bước học là 0.2.

Với bước học tính theo công thức (2.20) thì ba lần thất bại, số bước lặp để luyện mạng thành công khá ổn định, tuy nhiên chúng ta cũng thấy rằng, theo *bảng 2.2* đã thống kê thì với bước học tính theo nguyên lý vượt khe, tốc độ hội tụ cao hơn với trung bình 37 bước lặp ta đã luyện mạng xong, số lần thất bại khi luyện mạng cũng được giảm đi.

Ở chương 1 và chương 3 bài toán nhận dạng chữ viết được lập trình trên Matlab. Trong chương 1 bộ công cụ Neural Network Toolbox được sử dụng để luyện mạng.

Các tham số luyện mạng:

Kích thước ký tự = 5 x 7 Số đầu ra = 10 Số đầu vào = 35

Số nơron lớp ẩn = 5 Độ chính xác mong muốn = 90% Tỷ lệ học: 0.6

Lỗi hệ thống mong muốn=0.06

Kết quả luyện mạng như sau:

Số chu kỳ luyện	20	60	100	130	200
Tỷ lệ lỗi	93.33%	60.33%	40.67%	37.33%	0%
TSSE	0.8136	0.6848	0.2834	0.2823	0.06

Trong chương 3, mạng nơron được luyện với sự kết hợp của thuật toán vượt khe và giải thuật di truyền

Các tham số luyện mạng:

Kích thước quần thể = 20 Xác suất lai tạo = 0.46 Mã hóa bằng số thực

Độ dài nhiễm sắc thể = 225 Độ chính xác mong muốn = 90% Số thế hệ: 20

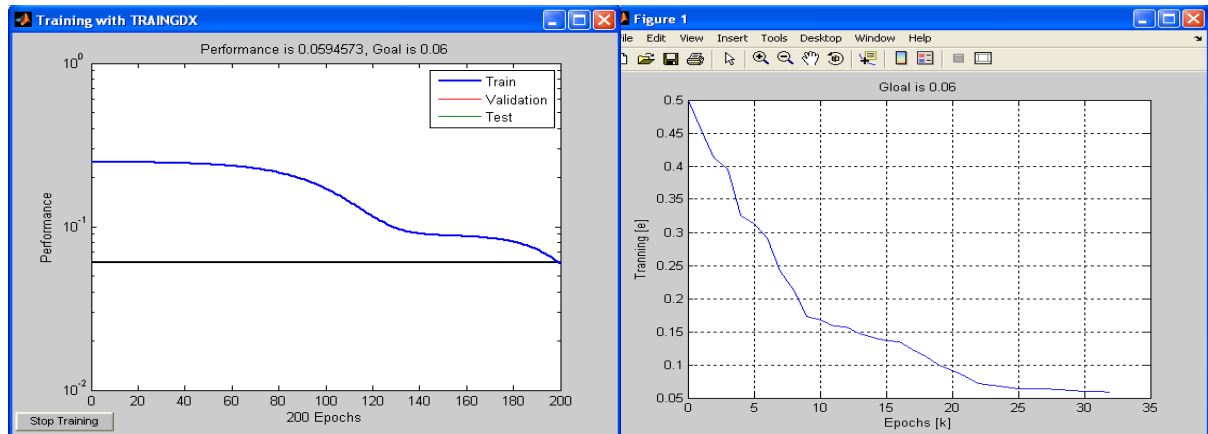
Lỗi hệ thống mong muốn=0.06

Kết quả luyện mạng như sau:

Số thế hệ	1	5	10	15	20
Tổng thích nghi	9.5563	8.1638	6.1383	5.724	5.697

Số chu kỳ luyện	5	10	15	20	33
Tỷ lệ lỗi	93.33%	60.33%	40.67%	37.33%	0%
TSSE	0.4956	0.3274	0.1387	0.0864	0.0589

Như vậy, lỗi hệ thống ở test 1 khi sử dụng luyện mạng MLP bằng giải thuật BG thuần túy là 0.06 sau 200 chu kỳ luyện mạng. Đối với test 2, sau 20 thế hệ đã đạt đến yêu cầu của bài toán. Giá trị thích nghi trung bình đạt được là 5.679. Kết quả của giai đoạn 1 được sử dụng để khởi tạo trọng số cho giai đoạn 2. Với sự thay đổi bước học theo giải thuật vượt khe, sau 33 chu kỳ luyện mạng lỗi hệ thống đã đạt đến mục đích 0.0589 và độ chính xác của quá trình nhận dạng là 100%. Hoạt động của mạng MLP thuần túy và mạng MLP có kết hợp giải thuật vượt khe và di truyền cho nhận dạng chữ được thể hiện trên *hình a*



Hình a: So sánh hoạt động của mạng MLP thuần túy và MLP cải tiến

Qua việc nghiên cứu và thực nghiệm trên máy tính cho ta thấy: với những cấu trúc mạng nơ ron mà mặt lỗi có dạng lòng khe, vẫn sử dụng kỹ thuật lan truyền ngược nhưng việc áp dụng giải thuật di truyền kết hợp với thuật toán “vượt khe” để luyện mạng sẽ cho ta độ chính xác và tốc độ hội tụ nhanh hơn nhiều so với phương pháp gradient.

Kết quả nghiên cứu này được giải thích như sau:

- Kết quả luyện mạng nơron phụ thuộc rất lớn vào giá trị ban đầu của vec-tơ trọng số. Việc sử dụng giải thuật di truyền thực hiện quá trình tìm kiếm toàn cục cho phép có được vec-tơ trọng số ban đầu tốt cho giai đoạn sau của quá trình luyện mạng.
- Khi mặt lỗi đặc biệt có dạng lòng khe, nếu luyện mạng bằng thuật toán gradien liên hợp hay thuật toán Levenberg – Marquardt sẽ chậm hội tụ và gặp phải vấn đề cực trị địa phương. Thuật toán “vượt khe” nhằm tìm kiếm các bước học tối ưu trong giai đoạn 2 của quá trình luyện mạng nên đã khắc phục các nhược điểm này và do đó làm tăng tốc độ hội tụ cũng như độ chính xác của quá trình luyện mạng.

Việc sử dụng giải thuật di truyền kết hợp với thuật toán “vượt khe” có thể ứng dụng để luyện một số cấu trúc mạng nơ ron mà có mặt lỗi đặc biệt khác. Vì vậy, kết quả nghiên cứu này có thể ứng dụng cho nhiều bài toán khác trong lĩnh vực viễn thông, điều khiển, và công nghệ thông tin.

- **Những đóng góp chính của luận án**

1. Đề xuất một dạng thuật toán vượt khe để giải bài toán tối ưu với hàm mục tiêu có dạng đặc biệt, dạng lòng khe.
2. Phân tích, đánh giá độ hội tụ của quá trình luyện mạng nơron phụ thuộc vào bộ trọng số khởi tạo ban đầu và bước học.
3. Đề xuất mô hình kết hợp giải thuật di truyền và thuật toán vượt khe trong quá trình luyện mạng nơron có mặt lỗi đặc biệt. Trong đó giải thuật di truyền có vai trò thực hiện quá trình tìm kiếm toàn cục để có được vec-tơ trọng số ban đầu tốt cho giai đoạn sau của quá trình luyện mạng. Còn thuật toán vượt khe là để tìm bước học tối ưu, làm tăng tốc độ hội tụ cũng như độ chính xác của quá trình luyện mạng.
4. Để kiểm chứng kết quả nghiên cứu, một ví dụ về nhận dạng chữ viết tay đã được đưa ra để luyện mạng với những phương pháp khác nhau.

- **Đề xuất hướng nghiên cứu**

- Bổ sung vào Toolbox Matlab một lựa chọn tính bước học mới: bước học vượt khe.
- Ứng dụng thuật toán này cho một số bài toán trong lĩnh vực điều khiển, tự động hóa và kỹ thuật điện tử.
- Phát triển mô hình kết hợp cho các bài toán tối ưu có hàm mục tiêu phức tạp khác.

CÁC CÔNG TRÌNH ĐÃ CÔNG BỐ

1. **“Research and Development of an adaptive control system for extremal systems”**; Cong Nguyen Huu, Dung Nguyen Tien, Nga Nguyen Thi Thanh, The 2009 international forum on strategic technologies (IFOST 2009), October 21-23, 2009, Ho Chi Minh city, Vietnam; page 235-238.
2. **“Nghiên cứu ứng dụng mạng hồi quy thời gian liên tục trong nhận dạng và điều khiển hệ thống xử lý nước thải”**; Nguyễn Hữu Công, Nguyễn Thị Thanh Nga, Phạm Văn Hưng; Tạp chí khoa học công nghệ Đại học Thái Nguyên số 12 tập 74 năm 2010; trang 4-8.
3. **Research on the application of genetic algorithm combined with the “cleft-overstep” algorithm for improving learning process of MLP neural network with special error surface.**; Cong Nguyen Huu, Nga Nguyen Thi Thanh, Huy Nguyen Phuong; The 7th International Conference on Natural Computation (ICNC'11) and the 8th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'11), 26-28 July, Shanghai, China, 2011; page 222-227.
4. Đề tài nghiên cứu khoa học cấp bộ **“Nghiên cứu thuật toán tìm nghiệm tối ưu toàn cục trong quá trình luyện mạng nơron - ứng dụng để nhận dạng, điều khiển đối tượng động học phi tuyến”**. Chủ nhiệm đề tài: Nguyễn Thị Thanh Nga, Nghiệm thu chính thức năm 2011.
5. **“Research to improve a learning algorithm of neural networks”**; Cong Nguyen Huu, Nga Nguyen Thi Thanh, Ngọc Văn Đông; Tạp chí Khoa học Công nghệ - Đại học Thái Nguyên, tháng 5 năm 2012; page 53-58.
6. **“The Influence of Initial Weights During Neural Network Training”**; Cong Nguyen Huu, Nga Nguyen Thi Thanh, Huy Vũ Ngọc, Anh Bùi Tuấn; Tạp chí Khoa học Công nghệ các Trường Đại học Kỹ thuật, No.95 (2013); page 18-25.

Trong tổng số 06 công trình tác giả đã công bố, tiêu biểu có: bài báo số 03 được nằm trong danh sách ISI, và đề tài NCKH cấp bộ số 04 mà tác giả làm chủ nhiệm đề tài.

TÀI LIỆU THAM KHẢO

- [1] Bùi Công Cường, Nguyễn Doãn Phước. *Hệ mờ, mạng nơron và ứng dụng*. Nhà xuất bản Khoa học và kỹ thuật. Hà Nội 2001.
- [2] Nguyễn Hữu Công, *ứng dụng mạng nơron trong nhận dạng hệ thống phi tuyến*, Tạp chí Khoa học và Công nghệ Thái Nguyên, số 3 – 2007.
- [3] Nguyễn Quang Hoan. *Một số mô hình và luật điều khiển mạng nơron dùng trong điều khiển*. Hội nghị toàn quốc lần thứ nhất về Tự động hóa. Hà Nội, 20-22/4/1994.
- [4] Nguyễn Thế Vinh, *Tìm nghiệm bài toán tối ưu tĩnh theo thuật toán vượt khe bằng mạng nơron*, Luận văn thạc sĩ, 2007
- [5] Bùi Minh Trí, *Tối ưu hóa*, tập II, Nhà xuất bản Khoa học và Kỹ thuật Hà nội, 2005.
- [6] Lê Minh Trung. *Giáo trình mạng nơron nhân tạo*. Nhà xuất bản thống kê. 1999.
- [7] Nguyễn Mạnh Tùng. *Nghiên cứu ứng dụng mạng nơron nhân tạo cho các bài toán đo lường*. Luận án tiến sĩ kỹ thuật, trường đại học Bách Khoa Hà Nội, 2003.
- [8] Nguyễn Quang Hoan (1996), “Một số thuật học của mạng nơron”, *Tuyển tập các báo cáo khoa học Hội nghị toàn quốc lần thứ II về Tự động hóa*, tr.220–229.
- [9] Chu Văn Hỷ (1998), “Điều khiển thích nghi phi tuyến trên cơ sở mạng nơron RBF”, *Tuyển tập các báo cáo khoa học Hội nghị toàn quốc lần thứ III về Tự động hóa*, tr.238–241.
- [10] Nguyễn Doãn Phước, Phan Xuân Minh (2000), *Điều khiển tối ưu và bền vững*, Nhà xuất bản Khoa học và Kỹ thuật, Hà Nội.
- [11] Nguyễn Doãn Phước, *Lý thuyết điều khiển nâng cao*, Nhà xuất bản Khoa học và Kỹ thuật Hà nội, 2005.
- [12] Nguyễn Đình Thúc và Hoàng Đức Hải, *Trí tuệ nhân tạo – Mạng nơ ron, phương pháp và ứng dụng*, Nhà xuất bản Giáo dục, Hà nội.
- [13] Đỗ Trung Hải (2008), “*Ứng dụng lý thuyết mờ và mạng nơron để điều khiển hệ chuyển động*”, luận án tiến sĩ kỹ thuật, Đại học Bách Khoa Hà Nội.

-
- [14] Lê Hoàng Thái (2004), “*Xây dựng phát triển ứng dụng một số mô hình kết hợp giữa mạng nơron (NN), logic mờ (FL) và thuật giải di truyền (GA)*”, Luận án tiến sĩ tin học, Đại học Khoa học Tự nhiên Thành phố Hồ Chí Minh, Việt Nam.
 - [15] Kolmogorov A. N. (1957). “*On the representation of continuous functions of many variables by superposition of continuous function of one variable and addition*”. Dokl, Akad, Nauk SSSR, 114, 953-956, Trans. Am. Math-Soc. 2(28), 55-59.
 - [16] Blum, E. K. and L. K. Li. “*Approximation Theory and feedforward networks*”, Neural Networks, 1991, Vol. 4, pp. 511-515.
 - [17] Cotter, N. E. “*The Stone-Weierstrass theorem and its application to neural networks*”, IEEE Transactions on Neural Networks, 1990, Vol. 1, pp. 290-295.
 - [18] Funahashi, K. “*On the approximate realization of continuous mappings by neural networks*”, Neural Networks, 1989, Vol. 2, pp. 183-192.
 - [19] Hornik, K. “*Approximation capabilities of multilayer feedforward networks*”, Neural Networks, 1991, Vol. 4, pp. 251-257.
 - [20] Hecht-Nielsen, R. 1987. “*Kolmogorov’s mapping neural network existence theorem*”. Proceeding of the IEEE International Conference on Neural Networks (pp. 11-13), New York, IEEE Press.
 - [21] Hecht-Nielsen, R. 1989. “*Theory of backpropagation neural network*”. In Proceedings of the International Joint Conference on Neural Networks, pp. I-593-P605, Washington DC., June 1989. IEEE TAB Neural Network Committee.
 - [22] Lorentz, G. G. “*Approximation of Functions*”, Holt, Rinehart and Winston, N.Y., 1966. Meyer, Y. Ondulettes et Operateurs, Hermann, Paris, 1990.
 - [23] Jeffrey T. Spooner, Mangredi Maggiore, Raúl Ordóñez, Kelvin M. Passino (2002), *Stable Adaptive Control and Estimation for Nonlinear Systems: Neural and Fuzzy Approximator Techniques*, Wiley Interscience, USA.
-

-
- [24] Jyh-Shing Roger Jang, Chuen-Tsai Sun, Eiji Mizutani (1996), *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Prentice Hall, USA.
 - [25] L. Davis, “*Hand book of Genetic Algorithms*”, Van Nostrand Reinhold, New York, 1991.
 - [26] D.E. Goldberg, “*Genetic Algorithms in Search, Optimization, and Machine Learning*”, Addison-Wesley Pub. Comp. Inc., Reading, MA, 1989
 - [27] Cong Huu Nguyen, Thanh Nga Thi Nguyen, Phuong Huy Nguyen, “*Research on the application of genetic algorithm combined with the “cleft-overstep” algorithm for improving learning process of MLP neural network with special error surface*”, ICNC 2011, July, Shanghai China, 223-227.
 - [28] Nguyen Van Manh and Bui Minh Tri, “*Method of “cleft-overstep” by perpendicular direction for solving the unconstrained nonlinear optimization problem*”, Acta Mathematica Vietnamica, vol. 15, N02, 1990.
 - [29] Hagan, M.T., H.B. Demuth and M.H Beal, *Neural Networks Design*, PWS Publishing Company, Boston, 1996.
 - [30] Simon Haykin, *Neural Networks – A Comprehensive Foundation*, Macmillan, 1994.
 - [31] L.-Y. Bottou, *Reconnaissance de la parole par reseaux multi-couches*, Proceedings of the International Workshop Neural Networks Application, Neuro-Nimes'88, EC2 and Chambre de Commerce et d'Industrie de Nimes, 1988, pp. 197-217.
 - [32] T. Denoeux, R. Lengellé, *Initializing back propagation networks with prototypes*, Neural Networks 6 (1993), pp 351-363.
 - [33] G.P. Drago, S. Ridella, *Statistically controlled activation weight initialization (SCAWI)*, IEEE Trans. Neural Networks 3 (1992) 627-631.
 - [34] J.-P. Martens, *A stochastically motivated random initialization of pattern classifying MLPs*, Neural Process. Lett. 3 (1996) 23-29.
 - [35] T. Masters, *Practical Neural Network Recipes in C++*, Academic Press, Boston, 1993.
-

-
- [36] D. Nguyen, B. Widrow, *Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights*, International Joint Conference on Neural Networks, Vol. 3, San Diego, CA (1990) 21-26.
 - [37] S. Osowski, *New approach to selection of initial values of weights in neural function approximation*, Electron. Lett. 29 (1993) 313-315.
 - [38] J.F. Shepanski, *Fast learning in artificial neural systems: multilayer perceptron training using optimal estimation*, IEEE International Conference on Neural Networks 1, IEEE Press, New York, 1988, pp. 465-472.
 - [39] H. Shimodaira, *A weight value initialization method for improving learning performance of the back propagation algorithm in neural networks*, Proceedings of the International Conference on Tools with Artificial Intelligence, New Orleans, LA, 1994, pp. 672-675.
 - [40] Y.F. Yam, T.W.S. Chow, *Determining initial weights of feedforward neural networks based on least squares method*, Neural Process. Lett. 2 (1995) 13-17.
 - [41] Y.F. Yam, T.W.S. Chow, *A new method in determining the initial weights of feedforward neural networks*, Neurocomputing 16 (1997) 23-32.
 - [42] L.F.A. Wessels, E. Barnard, *Avoiding false local minima by proper initialization of connections*, IEEE Trans. Neural Networks 3 (1992) 899-905.
 - [43] N. Weymaere, J.P. Martens, *On the initialization and optimization of multilayer perceptrons*, IEEE Trans. Neural Networks 5 (1994) 738-751.
 - [44] D. Anthony, E. Hines, *"The use of genetic algorithms to learn the most appropriate inputs to neural network"*, Application of the International Association of Science and Technology for Development-IASTED, June, 1990, 223-226.
 - [45] L. Fauselt, *"Fundamentals of Neural Networks"*, Prentice-Hall, International Inc., Englewood Cliffs, NJ, 1994.
 - [46] R.K. Al Seyab, Y. Cao (2007), *"Nonlinear system identification for predictive control using continuous time recurrent neural networks and automatic differentiation"*, School of Engineering Cranfield University, College Road, Cranfield, Bedford MK43 0AL, UK, Science Direct
 - [47] Cong Nguyen Huu; Nam Nguyen Hoai, *"Optimal control for a distributed parameter and delayed – time system based on the numerical method"*, Teth
-

- international conference on Control, Automotion, Robotics and vision (ICARCV'2008).
- [48] M.Norgaard, O.Ravn, N.K. Poulsen and L.K. Hansen. *Neural Network for Modelling and Control of Dynamic System*. Springer 2000.
 - [49] Kumpati S. Narendra fellow, IEEE, and Kannan parthasarathy. *Identification and control of Dynamical Systems Using Neural Networks*.
 - [50] Jaroslava Žilková, Jaroslav Timko, Peter Girovský, “*Nonlinear System Control Using Neural Networks*”, Department of Electrical Drives and Mechatronic, Technical University of Kosice, Hungary.
 - [51] Maciej Lawrynczuk (2010), “*Training or neural models for predictive control*”, Insitute of control and computation Engineering, Faculty of Electronics and Information Technology, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warsaw, Poland, Neurocomputing 73.
 - [52] A. Griewank, J. David, U. Jean, ADOL-C: *a package for the automatic differentiation of algorithms written in C/C++*, ACM Transactions on Mathematical Software 22 (2) (1996) 131–167.
 - [53] S.P. Moustakidis, G.A. Rovithakis, and J.B. Theocharis (2006), “*An Adaptive Neuro-Fuzzy Control Approach for Nonlinear Systems via Lyapunov Function Derivative Estimation*”, The 2006 IEEE International Symposium on Intelligent Control, Munich, Germany, pp.1602–1607.
 - [54] S. Seshagiri and H. K. Khalil (2000), “*Output feedback control of nonlinear systems using RBF neural networks*”, IEEE Transactions on Neural Networks, Vol. 11, No. 1, pp. 69–79.
 - [55] S.S. Ge, C. C. Hang, Tao Zhang (1999), “*Adaptive Neural Network Control of Nonlinear Systems by State and Output Feedback*”, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, Vol. 29, No. 6.
 - [56] Tianping Zhang and Shuzhi Sam Ge (2006), “*Robust Adaptive Neural Control of SISO Nonlinear Systems with Unknown Nonlinear Dead-Zone and Completely Unknown Control Gain*”, The 2006 IEEE International Symposium on Intelligent Control, Munich, Germany, pp.88–93.
 - [57] Tomohisa Hayakawa (2005), “*A New Characterization on the Approximation of Nonlinear Functions via Neural Networks: An Adaptive*

- Control Perspective*”, The 44th IEEE Conference on Decision and Control, and the European Control Conference, Spain, pp.4117–4122.
- [58] Tomohisa Hayakawa, Wassim M. Haddad, James M. Bailey, Naira Hovakimyan (2005), “*Passivity-Based Neural Network Adaptive Output Feedback Control for Nonlinear Nonnegative Dynamical Systems*”, IEEE Transactions on Neural Networks, Vol. 16, No. 2, pp.387–398.
- [59] Tomohisa Hayakawa, Wassim M. Haddad, Naira Hovakimyan, VijaySekhar Chellaboina (2005), “*Neural Network Adaptive Control for Nonlinear Nonnegative Dynamical Systems*”, IEEE Transactions on Neural Networks, Vol. 16, No. 2, pp.399–413.
- [60] Tomohisa Hayakawa, Wassim M. Haddad, and Naira Hovakimyan (2008), “*Neural Network Adaptive Control for a Class of Nonlinear Uncertain Dynamical Systems with Asymptotic Stability Guarantees*”, IEEE Transactions on Neural Networks, Vol. 19, No. 1, pp.80–89.
- [61] T. Zhang, S. S. Ge, C. C. Hang (2000), “*Stable Adaptive Control for a Class of Nonlinear Systems using a Modified Lyapunov Function*”, IEEE Transactions on Automatic Control, Vol. 45, No. 1, pp.129–132.

PHỤ LỤC 1: MỘT SỐ KIẾN THỨC CƠ SỞ LIÊN QUAN ĐẾN ĐỀ TÀI

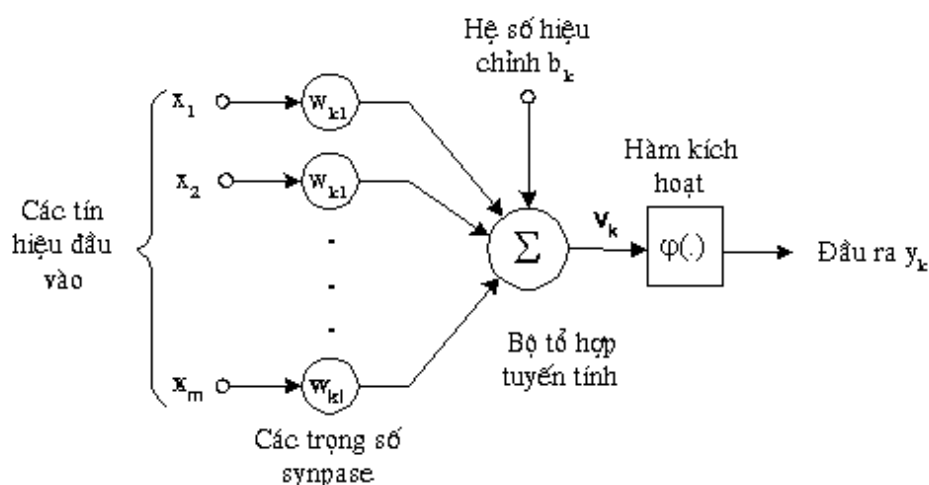
Giới thiệu về mạng nơron

I.1. Định nghĩa

Mạng nơron nhân tạo, *Artificial Neural Network (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.

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.



Hình 1: Nơron nhân tạo

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

♦ **Tập các đầu vào:** Là các tín hiệu vào (*input signals*) của nơron, các tín hiệu này thường được đưa vào dưới dạng một vec-tơ m chiều.

♦ **Tập các 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 w_{kj} . 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.

♦ **Ngưỡng** (còn gọi là một độ lệch - *bias*): Ngưỡng này thường 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 tùy thuộc vào từng bài toán và kinh nghiệm của người thiết kế mạng.

♦ **Đầu ra**: Là tín hiệu đầu ra của một nơron, với mỗi nơron sẽ có tối đa là một đầu ra.

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).

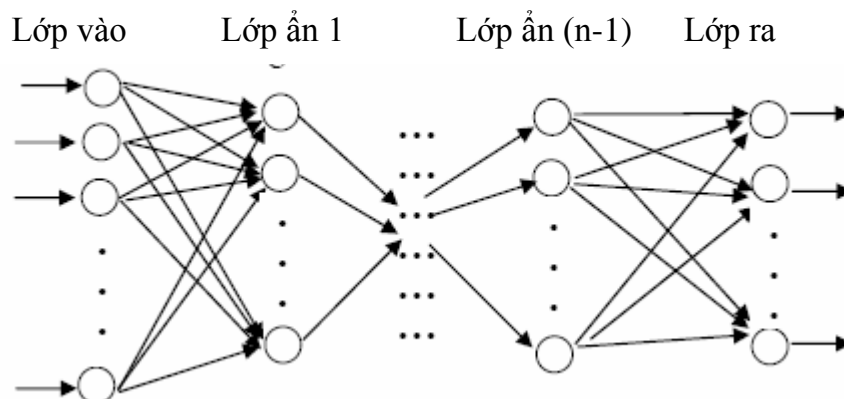
Mô hình mạng nơron

Mặc dù mỗi nơron đơn lẻ có thể thực hiện những chức năng xử lý thông tin nhất định, sức mạnh của tính toán nơron chủ yếu có được nhờ sự kết hợp các nơron trong một kiến trúc thống nhất. Một mạng nơron là một mô hình tính toán được xác định qua các tham số: kiểu nơron (như là các nút nếu ta coi cả mạng nơron là một đồ thị), kiến trúc kết nối (sự tổ chức kết nối giữa các nơron) và thuật toán học (thuật toán dùng để học cho mạng).

Về bản chất một mạng nơron có chức năng như là một hàm ánh xạ $F: X \rightarrow Y$, trong đó X là không gian trạng thái đầu vào (*input state space*) và Y là không gian trạng thái đầu ra (*output state space*) của mạng. Các mạng chỉ đơn giản là làm nhiệm vụ ánh xạ các vec-tơ đầu vào $x \in X$ sang các vec-tơ đầu ra $y \in Y$ thông qua “bộ lọc” (*filter*) các trọng số. Tức là $y = F(x) = s(W, x)$, trong đó W là ma trận trọng số liên kết. Hoạt động của mạng thường là các tính toán số thực trên các ma trận.

Mô hình mạng nơron 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 ($n \geq 2$) lớp (thông thường lớp đầu vào không được tính đến): trong đó gồm một lớp đầu ra (lớp thứ n) và $(n-1)$ lớp ẩn.



Hình 2: Mạng MLP tổng quát

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

♦ Đầu vào là các vec-tơ (x_1, x_2, \dots, x_p) trong không gian p chiều, đầu ra là các vec-tơ (y_1, y_2, \dots, y_q) trong không gian q chiều. Đối với các bài toán phân loại, p chính là kích thước của mẫu đầu vào, q chính là số lớp cần phân loại. Xét ví dụ trong bài toán nhận dạng chữ số: với mỗi mẫu ta lưu tọa độ (x, y) của 8 điểm trên chữ số đó, và nhiệm vụ của mạng là phân loại các mẫu này vào một trong 10 lớp tương ứng với 10 chữ số 0, 1, ..., 9. Khi đó p là kích thước mẫu và bằng $8 \times 2 = 16$; q là số lớp và bằng 10.

♦ Mỗi nơron thuộc lớp sau liên kết với tất cả các nơron thuộc lớp liền trước nó.

♦ Đầu ra của nơron lớp trước là đầu vào của nơron thuộc lớp liền sau nó.

Hoạt động của mạng MLP như sau: tại lớp 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 lớp ẩn thứ nhất; các nơron tại đây tiếp nhận như là tín hiệu đầu vào, xử lý và gửi kết quả đến lớp ẩn thứ 2;...; quá trình tiếp tục cho đến khi các nơron thuộc lớp ra cho kết quả.

Một số kết quả đã được chứng minh:

♦ Bất kì một hàm Boolean nào cũng có thể biểu diễn được bởi một mạng MLP 2 lớp 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 lớp sử dụng hàm truyền sigmoid cho các nơon lớp ẩn và hàm truyền tuyến tính cho các nơon lớp ra với sai số nhỏ tùy ý.

♦ Mọi hàm bất kỳ đều có thể xấp xỉ bởi một mạng MLP 3 lớp sử dụng hàm truyền sigmoid cho các nơon lớp ẩn và hàm truyền tuyến tính cho các nơon lớp ra.

Quá trình học của mạng nơ-ron

Các phương pháp học

Khái niệm: Học là quá trình cập nhật trọng số sao cho giá trị hàm lỗi là nhỏ nhất.

Một mạng nơon được huấn luyện sao cho với một tập các vec-tơ đầu vào X , mạng có khả năng tạo ra tập các vec-tơ 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 vec-tơ đầu vào x từ tập huấn luyện, mạng sẽ cho ra vec-tơ 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 trong các mạng nơon

Học có giám sát có thể được xem như việc xấp xỉ một ánh xạ: $X \rightarrow Y$, 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 $x = (x_1, x_2, \dots, x_n) \in X$, $y = (y_1, y_2, \dots, y_m) \in Y$ được cho trước. Học có giám sát trong các mạng nơon thường được thực hiện theo các bước sau:

♦ **B1:** Xây dựng cấu trúc thích hợp cho mạng nơon, chẳng hạn có $(n + 1)$ nơon vào (n nơon cho biến vào và 1 nơon cho ngưỡng x_0), m nơon đầu ra, và khởi tạo các trọng số liên kết của mạng.

♦ **B2:** Đưa một vec-tơ x trong tập mẫu huấn luyện X vào mạng

♦ **B3:** Tính vec-tơ đầu ra z của mạng

♦ **B4:** So sánh vec-tơ đầu ra mong muốn t (là kết quả được cho trong tập huấn luyện) với vec-tơ đầu ra z do mạng tạo ra; nếu có thể thì đánh giá lỗi.

♦ **B5:** Hiệu chỉnh các trọng số liên kết theo một cách nào đó sao cho ở lần tiếp theo khi đưa vec-tơ x vào mạng, vec-tơ đầu ra z sẽ giống với t hơn.

♦ **B6:** 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: $Err = (z - t)$, hoặc $Err = |z - t|$; lỗi trung bình bình phương (MSE: mean-square error): $Err = (z - t)^2/2$;

Có hai loại lỗi trong đánh giá một mạng nơron. Thứ nhất, gọi là lỗi rõ ràng (*apparent error*), đánh giá khả năng xấp xỉ các mẫu huấn luyện của một mạng đã được huấn luyện. Thứ hai, gọi là lỗi kiểm tra (*test error*), đánh giá khả năng tổng quát hóa của một mạng đã được huấn luyện, tức khả năng phản ứng với các vec-tơ đầu vào mới. Để đánh giá lỗi kiểm tra chúng ta phải biết đầu ra mong muốn cho các mẫu kiểm tra.

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.

Thuật toán lan truyền ngược

II.3.1. Một vài điều về thuật toán lan truyền ngược

Có một vài lớp khác nhau của các luật huấn luyện mạng bao gồm: huấn luyện kết hợp, huấn luyện cạnh tranh,... Huấn luyện chất lượng mạng, performance learning, là một lớp quan trọng khác của luật huấn luyện, trong phương pháp này thì các thông số mạng được điều chỉnh để tối ưu hóa chất lượng của mạng. Thuật toán lan truyền ngược là một phát minh chính trong nghiên cứu về mạng nơron, thuộc loại thuật học chất lượng mạng (học có giám sát). Ngược dòng thời gian, chúng ta thấy rằng sau khoảng mười năm kể từ khi lan truyền ngược bắt đầu được thai nghén, năm 1974, thì thuật học lan truyền ngược được chính thức nghiên cứu lại và mở rộng ra một cách độc lập bởi David Rumelhart, Geoffrey Hinton và Ronald Williams; David Parker và Yann Le Cun. Thuật toán đã được phổ biến hóa bởi cuốn sách *Parallel Distributed Processing* của nhóm tác giả David Rumelhart và James McClelland. Tuy nhiên, thuật toán nguyên thủy thì quá chậm chạp đối với hầu hết các ứng dụng thực tế [1], có nhiều lý do cho việc hội tụ chậm trong đó có sự ảnh hưởng của bước học.

Nhắc lại rằng lan truyền ngược, tiền thân của nó là thuật học Widow-Hoff (thuật toán LMS, Least Mean Square), là một thuật toán xấp xỉ giảm dốc nhất. Giống với luật học LMS, hàm mục tiêu là trung bình bình phương sai số. Điểm khác giữa thuật toán LMS và lan truyền ngược chỉ là cách mà các đạo hàm được tính. Đối với mạng tuyến tính một lớp đơn giản, sai số là hàm tuyến tính tường minh của các trọng số, và các đạo hàm của nó liên quan tới các trọng số có thể được tính toán một cách dễ dàng. Trong các mạng nhiều lớp với các hàm phi tuyến, mối quan hệ giữa các trọng số mạng và sai số là cực kỳ phức tạp. Để tính các đạo hàm, chúng ta cần sử dụng luật chuỗi.

Chúng ta đã thấy rằng giảm dốc nhất là một thuật toán đơn giản, và thông thường chậm nhất. Thuật toán gradient liên hợp và phương pháp Newton's nói chung mang đến sự hội tụ nhanh hơn [6]. Khi nghiên cứu về các thuật toán nhanh hơn thì thường rơi vào hai trường phái. Trường phái thứ nhất phát triển về các kỹ thuật tìm kiếm. Các kỹ thuật tìm kiếm bao gồm các ý tưởng như việc thay đổi tốc độ học, sử dụng qui tắc mô-men, bước học thích nghi. Trường phái khác của nghiên cứu nhằm vào các kỹ thuật tối ưu hóa số chuẩn, điển hình là phương pháp gradient liên hợp, hay thuật toán Levenberg-Marquardt (một biến thể của phương pháp Newton). Tối ưu hóa số đã là một chủ đề nghiên cứu quan trọng với 30, 40 năm, nó dường như là nguyên nhân để tìm kiếm các thuật toán huấn luyện nhanh.

Ta biết rằng, thuật toán LMS được đảm bảo để hội tụ tới một lời giải cực tiểu hóa trung bình bình phương sai số, miễn là tốc độ học không quá lớn. Điều này là đúng bởi vì trung bình bình phương sai số cho một mạng tuyến tính một lớp là một hàm toàn phương. Hàm toàn phương chỉ có một điểm tĩnh. Hơn nữa, ma trận Hessian của hàm toàn phương là hằng số, cho nên độ dốc của hàm theo hướng là không thay đổi, và các hàm đồng mức có dạng hình e-lip.

Lan truyền ngược giảm dốc nhất (SDBP) cũng như LMS, nó cũng là một thuật toán xấp xỉ giảm dốc nhất cho việc cực tiểu trung bình bình phương sai số. Thật vậy, lan truyền ngược giảm dốc nhất là tương đương thuật toán LMS khi sử dụng trên mạng tuyến tính một lớp.

Bây giờ chúng ta sẽ tập trung nghiên cứu một kỹ thuật rất phổ biến của mạng neural nhiều tầng. Chúng ta sẽ xem xét cách mà một mạng học một ánh xạ từ một tập dữ liệu cho trước.

Chúng ta đã biết việc học dựa trên định nghĩa của hàm lỗi, hàm lỗi này sau đó sẽ được tối thiểu hoá dựa vào các trọng số và các trọng ngưỡng trong mạng.

Trước tiên ta sẽ xem xét trường hợp mạng sử dụng hàm ngưỡng. Vấn đề cần bàn ở đây chính là cách để khởi tạo các trọng số cho mạng như thế nào. Công việc này thường được gọi là ‘credit assignment problem’. nếu một nút đầu ra tạo ra một đáp số sai lệch thì chúng ta phải quyết định xem liệu nút ẩn nào phải chịu trách nhiệm cho sự sai lệch đó, cũng chính là việc quyết định trọng số nào cần phải điều chỉnh và điều chỉnh là bao nhiêu.

Để giải quyết vấn đề gán trọng số này, chúng ta hãy xem xét một mạng với các hàm truyền phân biệt, do đó giá trị tổng trọng của các nút xuất sẽ trở thành một hàm phân biệt của các biến nhập và của trọng số và trọng ngưỡng. Nếu ta coi hàm lỗi, ví dụ có dạng sai số trung bình bình phương, là một hàm riêng biệt cho các giá trị xuất của mạng thì bản thân nó cũng chính là một hàm phân biệt của các trọng số.

Do đó chúng ta có thể tính toán được đạo hàm hàm lỗi theo các trọng số, và giá trị đạo hàm này lại có thể dùng để làm cực tiểu hoá hàm lỗi bằng cách sử dụng phương pháp giảm gradient (gradient descent) hoặc các phương pháp tối ưu hoá khác.

Giải thuật ước lượng đạo hàm hàm lỗi được biết đến với tên gọi *lan truyền ngược*, nó tương đương với việc lan truyền ngược lỗi trong mạng. Kỹ thuật về lan truyền ngược được biết đến rất rộng rãi và chi tiết qua các bài báo cũng như các cuốn sách của Rumelhart, Hinton và Williams (1986). Tuy nhiên gần đây một số ý tưởng tương tự cũng được một số nhà nghiên cứu phát triển bao gồm Werbos (1974) và Parker (1985).

Cần nói thêm rằng giải thuật lan truyền ngược được sử dụng trong mạng neural có ý nghĩa rất lớn. Ví dụ như, kiến trúc của mạng perceptron nhiều tầng cũng thường được gọi là mạng lan truyền ngược. Khái niệm *lan truyền ngược cũng thường được sử dụng để mô tả quá trình huấn luyện của mạng perceptron nhiều tầng sử dụng phương pháp gradient descent áp dụng trên hàm lỗi dạng sai số trung bình bình phương*. Để làm rõ hơn về thuật ngữ này chúng ta cần xem xét quá trình luyện mạng một cách kỹ càng. Phần lớn các giải thuật luyện mạng đều liên quan đến một thủ tục được lặp đi lặp lại nhằm làm tối thiểu hàm lỗi, bằng cách điều chỉnh trọng số trong một chuỗi các bước.

Tại mỗi bước như vậy, chúng ta có thể chia thành hai bước phân biệt.

Tại bước thứ nhất, cần phải tính đạo hàm hàm lỗi theo các trọng số. Chúng ta đã biết rằng một đóng góp rất quan trọng của kỹ thuật lan truyền ngược đó là việc cung cấp một phương pháp hết sức hiệu quả về mặt tính toán trong việc đánh giá

các đạo hàm. Vì tại bước này lỗi sẽ được lan truyền ngược trở lại mạng nên chúng ta sẽ sử dụng khái niệm lan truyền ngược để đặc trưng riêng cho việc đánh giá đạo hàm này.

Tại bước thứ hai, các đạo hàm sẽ được sử dụng trong việc tính toán sự điều chỉnh đối với trọng số. Và kỹ thuật đơn giản nhất được sử dụng ở đây là kỹ thuật gradient descent, kỹ thuật này được Rumelhart et al. (1986) đưa ra lần đầu tiên.

Một điều hết sức quan trọng là phải nhận thức được rằng hai bước này là phân biệt với nhau. Do đó, quá trình xử lý đầu tiên, được biết đến là quá trình lan truyền ngược các lỗi vào trong mạng để đánh giá đạo hàm, có thể được áp dụng đối với rất nhiều loại mạng khác nhau chứ không chỉ đối với riêng mạng perceptron nhiều tầng. Nó cũng có thể được áp dụng với các loại hàm lỗi khác chứ không chỉ là hàm tính sai số bình phương cực tiểu, và để đánh giá các đạo hàm khác này có thể sử dụng các phương pháp khác như phương pháp ma trận Jacobian và Hessian. Và cũng tương tự như vậy thì tại bước thứ hai, việc điều chỉnh trọng số sử dụng các đạo hàm đã được tính trước đó có thể thực hiện với nhiều phương pháp tối ưu hoá khác nhau, và rất nhiều trong số các phương pháp đó cho kết quả tốt hơn phương pháp gradient descent.

Tóm lại, khi giải quyết bài toán bằng mạng nơron theo thủ tục truyền ngược có những vấn đề rút ra là:

- Sẽ có bao nhiêu nơron trong mạng, bao nhiêu *ngõ vào*, bao nhiêu *ngõ ra* và bao nhiêu lớp *ẩn*. Càng nhiều lớp *ẩn* bài toán trở nên phức tạp nhưng có thể giải quyết được những vấn đề lớn.

- Thuật toán *Back propagation* cung cấp một phương pháp “*xấp xỉ*” cho việc tìm trong không gian *trọng số* (nhằm tìm ra những *trọng số* phù hợp cho mạng). Chúng ta càng lấy giá trị của *tham số học* càng nhỏ bao nhiêu thì sự thay đổi *trọng số* càng nhỏ bấy nhiêu và *quỹ đạo không gian học* sẽ càng *trơn*. Tuy nhiên điều này lại làm cho *tốc độ học* chậm đi. Trái lại, nếu chúng ta chọn *tham số tốc độ học* quá lớn, sự thay đổi lớn của các *trọng số* có thể làm cho mạng trở nên không ổn định. Về mặt ý tưởng, tất cả các nơron trong mạng nên chọn cùng một *tốc độ học*, *tham số học* η nên gán một giá trị nhỏ. Các nơron với nhiều *ngõ vào* nên chọn một *tham số tốc độ học* nhỏ hơn để giữ một *thời gian học* tương tự cho nhau cho tất cả các nơron trong mạng.

II.3.2. Hiệu quả của lan truyền ngược

Một trong những đặc tính quan trọng nhất của lan truyền ngược chính là ở khả năng tính toán hiệu quả của nó.

Đặt w là tổng số các trọng số và trọng ngưỡng. Do đó một phép tính hàm lỗi (cho một mẫu nhập nào đó) cần $O(w)$ thao tác với w đủ lớn. Điều này cho phép số lượng trọng số có thể lớn hơn số lượng nút, trừ những mạng có quá ít kết nối. Do vậy, hiệu quả của việc tính toán trong lan truyền ngược sẽ liên quan đến việc tính giá trị của tổng trọng hóa $a_j = \sum_i w_{ji} z_i$, với z_i là giá trị nhập hoặc là giá trị xuất của một nút có cung kết nối với nút j và w_{ji} chính là trọng số của cung kết nối đó; còn việc tính toán các hàm truyền thì tổng phí khá nhỏ. Mỗi lượt tính tổng trên cần đến một phép nhân và một phép cộng, dẫn đến chi phí tính toán toàn bộ sẽ bằng $O(w)$.

Với tất cả w trọng số thì sẽ có w đạo hàm cần tính toán. Với mỗi lần tính đạo hàm như vậy cần phải thực hiện tìm biểu thức hàm lỗi, xác định công thức tính đạo hàm và sau đó tính toán chúng theo giải thuật lan truyền ngược, mỗi công việc đó sẽ đòi hỏi $O(w)$ thao tác. Như vậy toàn bộ quá trình tính toán tất cả các đạo hàm sẽ tỉ lệ với $O(w^2)$. Giải thuật lan truyền ngược cho phép các đạo hàm được tính trong $O(w)$ thao tác. Điều này cũng dẫn đến rằng cả hai pha lan truyền ngược và lan truyền tiến đều cần $O(w)$ thao tác, việc tính đạo hàm để lan truyền ngược cũng cần $O(w)$ thao tác. Như vậy giải thuật lan truyền ngược đã làm giảm độ phức tạp tính toán từ $O(w^2)$ đến $O(w)$ đối với mỗi vec-tor nhập. Vì quá trình luyện mạng, dù có sử dụng lan truyền ngược, có thể cần rất nhiều thời gian, nên việc đạt được hiệu quả như vậy là hết sức quan trọng. Với tổng số N mẫu luyện, số lượng các bước tính toán để đánh giá hàm lỗi trên toàn bộ tập dữ liệu sẽ là N lần bước tính toán của một mẫu.

II.4. Các vấn đề trong xây dựng mạng MLP

II.4.1. Chuẩn bị dữ liệu

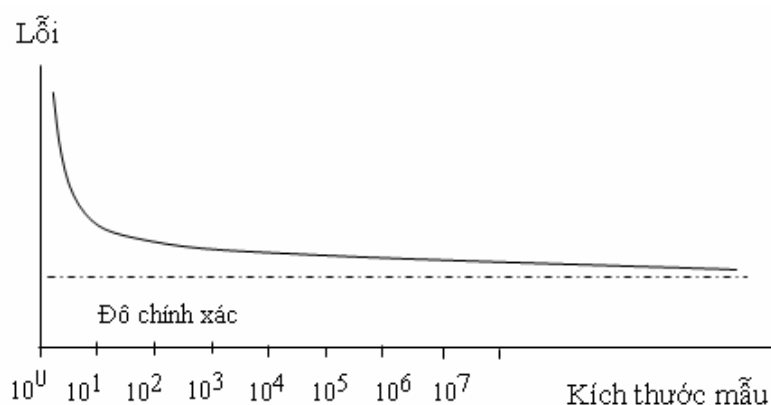
a. Kích thước mẫu

Không có nguyên tắc nào hướng dẫn kích thước mẫu phải là bao nhiêu đối với một bài toán cho trước. 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.

Đối với mạng truyền thẳng, cho hàm đích có độ phức tạp nhất định, kèm một lượng nhiễu nhất định thì độ chính xác của mô hình luôn có một giới hạn nhất định. Có thể cần tập mẫu vô hạn để đạt đến giới hạn chính xác. Nói cách khác độ chính xác của mô hình là hàm theo kích thước tập mẫu. Khi kích thước mẫu tăng, độ chính xác sẽ được cải thiện - lúc đầu nhanh, nhưng chậm dần khi tiến đến giới hạn.

Dạng tổng quát của mối liên hệ giữa sai số và kích thước mẫu như sau:



Hình 3: Mối liên hệ giữa sai số và kích thước mẫu

Trong thực hành thường gặp phải 2 vấn đề sau:

♦ Đối với hầu hết bài toán thực tế, mẫu bị ràng buộc chặt chẽ với dữ liệu có sẵn. Ta thường không có được số lượng mẫu mong muốn.

♦ Kích thước mẫu cũng có thể bị giới hạn bởi bộ nhớ hoặc khả năng lưu trữ của máy tính. Nếu tất cả các dữ liệu đồng thời được giữ trong bộ nhớ suốt thời gian luyện, kích thước bộ nhớ máy tính sẽ bị chiếm dụng nghiêm trọng.

Nếu lưu trữ trên đĩa sẽ cho phép dùng mẫu lớn hơn nhưng thao tác đọc đĩa từ thế hệ này sang thế hệ khác khiến cho tiến trình chậm đi rất nhiều.

Chú ý: việc tăng kích thước mẫu không làm tăng thời gian luyện. Những tập mẫu lớn hơn sẽ yêu cầu ít thế hệ luyện hơn. Nếu ta tăng gấp đôi kích thước của mẫu, mỗi thế hệ luyện sẽ tốn thời gian khoảng gấp đôi, nhưng số thế hệ cần luyện sẽ giảm đi một nửa. Điều này có nghĩa là kích thước mẫu (cũng có nghĩa là độ chính xác của mô hình) không bị giới hạn bởi thời gian luyện.

Luật cơ bản là: Sử dụng mẫu lớn nhất có thể sao cho đủ khả năng lưu trữ trong bộ nhớ trong (nếu lưu trữ đồng thời) hoặc trên đĩa từ (đủ thời gian đọc từ đĩa).

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

Nếu ta tổ chức mẫu sao cho mỗi mẫu trong quần thể đều có cơ hội như nhau thì tập mẫu được gọi là tập mẫu đại diện. Tuy nhiên khi ta xây dựng một mạng để xác định xem một mẫu thuộc một lớp hay thuộc một loại nào thì điều ta mong muốn là các lớp có cùng ảnh hưởng lên mạng, để đạt được điều này ta có thể sử dụng mẫu phân tầng. Xét ví dụ sau:

Giả sử ta xây dựng mô hình nhận dạng chữ cái viết tay tiếng Anh, và nguồn dữ liệu của ta có 100.000 ký tự mà mỗi ký tự được kèm theo một mã cho biết nó là chữ cái nào. Chữ cái xuất hiện thường xuyên nhất là e, nó xuất hiện 11.668 lần chiếm khoảng 12%; chữ cái xuất hiện ít nhất là chữ z, chỉ có 50 lần chiếm 0,05%.

Trước hết do giới hạn của bộ nhớ máy tính, giả sử bộ nhớ chỉ có thể xử lý được 1300 mẫu. Ta tạo hai dạng tập mẫu: tập mẫu đại diện và tập mẫu phân tầng. Với tập mẫu đại diện, chữ e sẽ xuất hiện 152 lần (11,67% của 1300) trong khi đó chữ z chỉ xuất hiện một lần (0,05% của 1300). Ngược lại ta có thể tạo tập mẫu phân tầng để mỗi chữ có 50 mẫu. Ta thấy rằng nếu chỉ có thể dùng 1300 mẫu thì tập mẫu phân tầng sẽ tạo ra mô hình tốt hơn. Việc tăng số mẫu của z từ 1 lên 50 sẽ cải thiện rất nhiều độ chính xác của z, trong khi nếu giảm số mẫu của e từ 152 xuống 50 sẽ chỉ giảm chút ít độ chính xác của e.

Bây giờ giả sử ta dùng máy tính khác có bộ nhớ đủ để xử lý một lượng mẫu gấp 10 lần, như vậy số mẫu sẽ tăng lên 13000. Rõ ràng việc tăng kích thước mẫu sẽ giúp cho mô hình chính xác hơn. Tuy nhiên ta không thể dùng tập mẫu phân tầng như trên nữa vì lúc này ta sẽ cần tới 500 mẫu cho chữ z trong khi ta chỉ có 50 mẫu trong nguồn dữ liệu. Để giải quyết điều này ta tạo tập mẫu như sau: tập mẫu gồm tất cả các chữ hiếm với số lần xuất hiện của nó và kèm thêm thông tin về chữ có nhiều mẫu nhất. Chẳng hạn ta tạo tập mẫu có 50 mẫu của chữ z (đó là tất cả) và 700 mẫu của chữ e (chữ mà ta có nhiều mẫu nhất).

Như vậy trong tập mẫu của ta, chữ e có nhiều hơn chữ z 14 lần. Nếu ta muốn các chữ z cũng có nhiều ảnh hưởng như các chữ e, khi học chữ z ta cho chúng trọng

số lớn hơn 14 lần. Để làm được điều này ta có thể can thiệp chút ít vào quá trình lan truyền ngược trên mạng. Khi mẫu học là chữ z, ta thêm vào 14 lần đạo hàm, nhưng khi mẫu là chữ e ta chỉ thêm vào 1 lần đạo hàm. Ở cuối thể hệ, khi cập nhật các trọng số, mỗi chữ z sẽ có ảnh hưởng hơn mỗi chữ e là 14 lần, và tất cả các chữ z gộp lại sẽ có bằng có ảnh hưởng bằng tất cả các chữ e.

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 suất cao nhất. Xét ví dụ về bài toán dự đoán một người có mắc bệnh ung thư hay không. Khi đó ta có trường thông tin về người này là “ngày tháng năm sinh”. Mạng sẽ đạt được hiệu quả cao hơn khi ta biến đổi trường thông tin này sang thành “tuổi”. Thậm chí ta có thể quy tuổi về một trong các giá trị: 1 = “trẻ em” (dưới 18), 2 = “thanh niên” (từ 18 đến dưới 30), 3 = “trung niên” (từ 30 đến dưới 60) và 4 = “già” (từ 60 trở lên).

- ♦ 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. Trong ví dụ dự đoán người có bị ung thư hay không ở trên, những thuộc tính như “nghề nghiệp”, “nơi sinh sống”, “tiểu sử gia đình”,... là những thông tin có ích. Tuy nhiên những thông tin như “thu nhập”, “số con cái”,... là những thông tin không cần thiết.

II.4.2. Xác định các tham số cho mạng

a. Chọn hàm truyền

Không phải bất kỳ hàm truyền nào cũng cho kết quả như mong muốn. Để trả lời cho câu hỏi «*hàm truyền như thế nào được coi là tốt ?* » là điều không hề đơn giản. Có một số quy tắc khi chọn hàm truyền như sau:

- ♦ Không dùng hàm truyền tuyến tính ở tầng ẩn. Vì nếu dùng hàm truyền tuyến tính ở tầng ẩn thì sẽ làm mất vai trò của tầng ẩn đó: Xét tầng ẩn thứ i:

$$\text{Tổng trọng số } n_i = w_i a_{i-1} + b_i$$

$$a_i = f(n_i) = w_f n_i + b_f \text{ (hàm truyền tuyến tính)}$$

Khi đó: tổng trọng số tại tầng thứ (i + 1)

$$n_{i+1} = w_{i+1} a_i + b_{i+1}$$

$$\begin{aligned}
 &= w_{i+1}[w_f n_i + b_f] + b_{i+1} \\
 &= w_{i+1} [w_f(w_i a_{i-1} + b_i) + b_f] + b_{i+1} \\
 &= W a_{i-1} + b
 \end{aligned}$$

Như vậy $n_{i+1} = W a_{i-1} + b$, và tầng i đã không còn giá trị nữa.

♦ Chọn các hàm truyền sao cho kiến trúc mạng nơron là đối xứng (tức là với đầu vào ngẫu nhiên thì đầu ra có phân bố đối xứng). Nếu một mạng nơron không đối xứng thì giá trị đầu ra sẽ lệch sang một bên, không phân tán lên toàn bộ miền giá trị của output. Điều này có thể làm cho mạng rơi vào trạng thái bão hòa, không thoát ra được.

Trong thực tế người ta thường sử dụng các hàm truyền dạng – S. Một hàm $s(u)$ được gọi là hàm truyền dạng – S nếu nó thỏa mãn 3 tính chất sau:

– $s(u)$ là hàm bị chặn: tức là tồn tại các hằng số $C1 \leq C2$ sao cho: $C1 \leq s(u) \leq C2$ với mọi u .

– $s(u)$ là hàm đơn điệu tăng: giá trị của $s(u)$ luôn tăng khi u tăng. Do tính chất thứ nhất, $s(u)$ bị chặn, nên $s(u)$ sẽ tiệm cận tới giá trị cận trên khi u dần tới dương vô cùng, và tiệm cận giá trị cận dưới khi u dần tới âm vô cùng.

– $s(u)$ là hàm khả vi: tức là $s(u)$ liên tục và có đạo hàm trên toàn trục số.

Có 3 dạng hàm kích hoạt thường được dùng trong thực tế:

***Hàm dạng bước:**

$$\text{step}(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \qquad \text{step}(x) = \begin{cases} 1 & x \geq \theta \\ 0 & x < \theta \end{cases}$$

***Hàm dấu:**

$$\text{step}(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases} \qquad \text{step}(x) = \begin{cases} 1 & x \geq \theta \\ -1 & x < \theta \end{cases}$$

***Hàm sigmoid:**
$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-\alpha(x+\theta)}}$$

Một hàm truyền dạng - S điển hình và được áp dụng rộng rãi là hàm Sigmoid.

b. Xác định số nơron tầng ẩn

Câu hỏi chọn số lượng nơron trong tầng ẩn của một mạng MLP thế nào là khó, nó phụ thuộc vào bài toán cụ thể và vào 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 nơron 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. Có nhiều đề nghị cho việc chọn số lượng nơron tầng ẩn h trong một mạng MLP. Chẳng hạn h phải thỏa mãn $h > (p-1)/(n+2)$, trong đó p là số lượng mẫu huấn luyện và n là số lượng đầu vào của mạng. Càng nhiều nút ẩn trong mạng, thì càng nhiều đặc tính của dữ liệu huấn luyện sẽ được mạng nắm bắt, nhưng thời gian học sẽ càng tăng.

Một kinh nghiệm khác cho việc chọn số lượng nút ẩn là số lượng nút ẩn bằng với số tối ưu các cụm mờ (*fuzzy clusters*)[8]. Phát biểu này đã được chứng minh bằng thực nghiệm. Việc chọn số tầng ẩn cũng là một nhiệm vụ khó. Rất nhiều bài toán đòi hỏi nhiều hơn một tầng ẩn để có thể giải quyết tốt.

Để tìm ra mô hình mạng nơron tốt nhất, Ishikawa and Moriyama (1995) sử dụng học cấu trúc có quên (*structural learning with forgetting*), tức là trong thời gian học cắt bỏ đi các liên kết có trọng số nhỏ. Sau khi huấn luyện, chỉ các nơron có đóng góp vào giải quyết bài toán mới được giữ lại, chúng sẽ tạo nên bộ xương cho mô hình mạng nơron.

c. Khởi tạo trọng số

Trọng thường được khởi tạo bằng phương pháp thử sai, nó mang tính chất kinh nghiệm và phụ thuộc vào từng bài toán. Việc định nghĩa thế nào là một bộ trọng tốt cũng không hề đơn giản. Một số quy tắc khi khởi tạo trọng:

♦ Khởi tạo trọng sao cho mạng nơron thu được là cân bằng (với đầu vào ngẫu nhiên thì sai số lan truyền ngược cho các ma trận trọng số là xấp xỉ bằng nhau):

$$|\Delta \mathbf{W}_1 / \mathbf{W}_1| = |\Delta \mathbf{W}_2 / \mathbf{W}_2| = |\Delta \mathbf{W}_3 / \mathbf{W}_3|$$

Nếu mạng nơron không cân bằng thì quá trình thay đổi trọng số ở một số ma trận là rất nhanh trong khi ở một số ma trận khác lại rất chậm, thậm chí không đáng kể. Do đó để các ma trận này đạt tới giá trị tối ưu sẽ mất rất nhiều thời gian.

♦ Tạo trọng sao cho giá trị kết xuất của các nút có giá trị trung gian. (0.5 nếu hàm truyền là hàm Sigmoid). Rõ ràng nếu ta không biết gì về giá trị kết xuất thì giá trị ở giữa là hợp lý. Điều này cũng giúp ta tránh được các giá trị thái quá.

Thủ tục khởi tạo trọng thường được áp dụng:

- **B1:** Khởi tạo các trọng số nút ẩn (và các trọng số của các cung liên kết trực tiếp giữa nút nhập và nút xuất, nếu có) giá trị ngẫu nhiên, nhỏ, phân bố đều quanh 0.
- **B2:** Khởi tạo một nửa số trọng số của nút xuất giá trị 1, và nửa kia giá trị -1.

Giải thuật di truyền

Giải thuật di truyền (Genetic Algorithms - GA) đã được đề cập trong rất nhiều tài liệu, trong đó có các công trình của D.E. Goldberg [26] và Thomas Back [27]. Trong phần này chỉ trình bày các khái niệm cơ bản về giải thuật di truyền và khả năng ứng dụng của nó.

I.2. Tóm tắt về giải thuật di truyền

Từ trước tới nay, trong các nghiên cứu và ứng dụng tin học đã xuất hiện nhiều bài toán chưa tìm ra được phương pháp giải nhanh và hợp lý. Phần lớn đó là các bài toán tối ưu nảy sinh trong các ứng dụng. Để giải các bài toán này người ta thường phải tìm đến một giải thuật hiệu quả mà kết quả thu được chỉ là xấp xỉ tối ưu. Trong nhiều trường hợp chúng ta có thể sử dụng giải thuật xác suất, tuy không bảo đảm kết quả tối ưu nhưng cũng có thể chọn các giá trị sao cho sai số đạt được sẽ nhỏ như mong muốn.

Theo lời giải xác suất, việc giải bài toán quy về quá trình tìm kiếm trên không gian tập hợp các lời giải có thể. Tìm được lời giải tốt nhất và quá trình được hiểu là tối ưu. Với miền tìm kiếm nhỏ, một số thuật toán cổ điển được sử dụng. Tuy nhiên đối với các miền lớn, phải sử dụng các kỹ thuật trí tuệ nhân tạo đặc biệt, giải thuật di truyền là một trong những công cụ đó. Ý tưởng của giải thuật di truyền là mô phỏng những gì mà tự nhiên đã thực hiện. GA hình thành dựa trên quan niệm cho rằng: quá trình tiến hóa tự nhiên là quá trình hoàn hảo nhất, hợp lý nhất và tự nó đã mang tính tối ưu.

Giải thuật di truyền áp dụng quá trình tiến hóa tự nhiên để giải các bài toán tối ưu trong thực tế (từ tập các lời giải có thể ban đầu thông qua nhiều bước

tiến hóa hình thành các tập hợp mới với lời giải tốt hơn và cuối cùng sẽ tìm được lời giải gần tối ưu).

Giải thuật di truyền là một kỹ thuật của khoa học máy tính nhằm tìm kiếm giải pháp thích hợp cho các bài toán tối ưu tổ hợp (*combinatorial optimization*). Giải thuật di truyền là một phân ngành của giải thuật tiến hóa vận dụng các nguyên lý của tiến hóa như di truyền, đột biến, chọn lọc tự nhiên, và trao đổi chéo.

Giải thuật di truyền thường được ứng dụng nhằm sử dụng ngôn ngữ máy tính để mô phỏng quá trình tiến hoá của một tập hợp những đại diện trừu tượng (gọi là những nhiễm sắc thể) của các giải pháp có thể (gọi là những *cá thể*) cho bài toán tối ưu hóa vấn đề. Tập hợp này sẽ tiến triển theo hướng chọn lọc những giải pháp tốt hơn.

Thông thường, những giải pháp được thể hiện dưới dạng nhị phân với những chuỗi 0 và 1, nhưng lại mang nhiều thông tin mã hóa khác nhau. Quá trình tiến hóa xảy ra từ một tập hợp những cá thể hoàn toàn ngẫu nhiên ở tất cả các thế hệ. Trong từng thế hệ, tính thích nghi của tập hợp này được ước lượng, nhiều cá thể được chọn lọc định hướng từ tập hợp hiện thời (dựa vào *thể trạng*), được sửa đổi (bằng đột biến hoặc tổ hợp lại) để hình thành một tập hợp mới. Tập hợp này sẽ tiếp tục được chọn lọc lặp đi lặp lại trong các thế hệ kế tiếp của giải thuật.

Giải thuật di truyền cũng như các thuật toán tiến hoá đều được hình thành dựa trên một quan niệm được coi là một tiên đề phù hợp với thực tế khách quan. Đó là quan niệm "Quá trình tiến hoá tự nhiên là quá trình hoàn hảo nhất, hợp lý nhất và tự nó đã mang tính tối ưu". Quá trình tiến hoá thể hiện tính tối ưu ở chỗ thế hệ sau bao giờ cũng tốt hơn thế hệ trước.

Quá trình phát triển của giải thuật di truyền có thể được chỉ ra qua các mốc thời gian sau:

1960: Ý tưởng đầu tiên về Tính toán tiến hoá được Rechenberg giới thiệu trong công trình "Evolution Strategies" (Các chiến lược tiến hoá). Ý tưởng này sau đó được nhiều nhà nghiên cứu phát triển.

1975: Giải thuật gen do John Holland phát minh và được phát triển bởi ông cùng với các đồng nghiệp và những sinh viên. Cuốn sách "Adaption in Natural and Artificial Systems" (Sự thích nghi trong các hệ tự nhiên và nhân tạo) xuất bản năm 1975 đã tổng hợp các kết quả của quá trình nghiên cứu và phát triển đó.

1992: John Koza đã dùng GA để xây dựng các chương trình giải quyết một số bài toán và gọi phương pháp này là "lập trình gen".

Ngày nay giải thuật di truyền càng trở nên quan trọng, đặc biệt là trong lĩnh vực tối ưu hoá, một lĩnh vực có nhiều bài toán thú vị, được ứng dụng nhiều trong thực tiễn nhưng thường khó và chưa có giải thuật hiệu quả để giải.

I.3. Các khái niệm cơ bản

Giải thuật di truyền dựa vào quá trình tiến hoá trong tự nhiên nên các khái niệm và thuật ngữ của nó đều có liên quan đến các thuật ngữ của di truyền học.

I.3.1. Cá thể, nhiễm sắc thể

Một cá thể trong giải thuật di truyền, biểu diễn một giải pháp của bài toán. Tuy nhiên không giống với trong tự nhiên, một cá thể có nhiều nhiễm sắc thể (NST), có 1 thì gọi là thể đơn bội, còn nếu có nhiều thì là thể đa bội, ở đây để giới hạn trong giải thuật di truyền ta quan niệm một cá thể có một nhiễm sắc thể. Do đó khái niệm cá thể và nhiễm sắc thể trong giải thuật di truyền coi như là tương đương.

Một NST được tạo thành từ nhiều gen, mỗi gen có thể có các giá trị khác nhau để quy định một tính trạng nào đó. Trong GA, một gen được coi như một phần tử trong chuỗi NST.

I.3.2. Quần thể

Quần thể là một tập hợp các cá thể có cùng một số đặc điểm nào đấy. Trong giải thuật di truyền ta quan niệm quần thể là một tập các lời giải của một bài toán.

I.3.3. Các toán tử di truyền

a. Chọn lựa

Trong tự nhiên, quá trình chọn lọc và đấu tranh sinh tồn đã làm thay đổi các cá thể trong quần thể. Những cá thể tốt, thích nghi được với điều kiện sống thì có

khả năng đấu tranh lớn hơn, do đó có thể tồn tại và sinh sản. Các cá thể không thích nghi được với điều kiện sống thì dần mất đi. Dựa vào nguyên lý của quá trình chọn lọc và đấu tranh sinh tồn trong tự nhiên, chọn lựa các cá thể trong GA chính là cách chọn các cá thể có độ thích nghi tốt để đưa vào thế hệ tiếp theo hoặc để cho lai ghép, với mục đích là sinh ra các cá thể mới tốt hơn. Có nhiều cách để lựa chọn nhưng cuối cùng đều nhằm đáp ứng mục tiêu là các cá thể tốt sẽ có khả năng được chọn cao hơn.

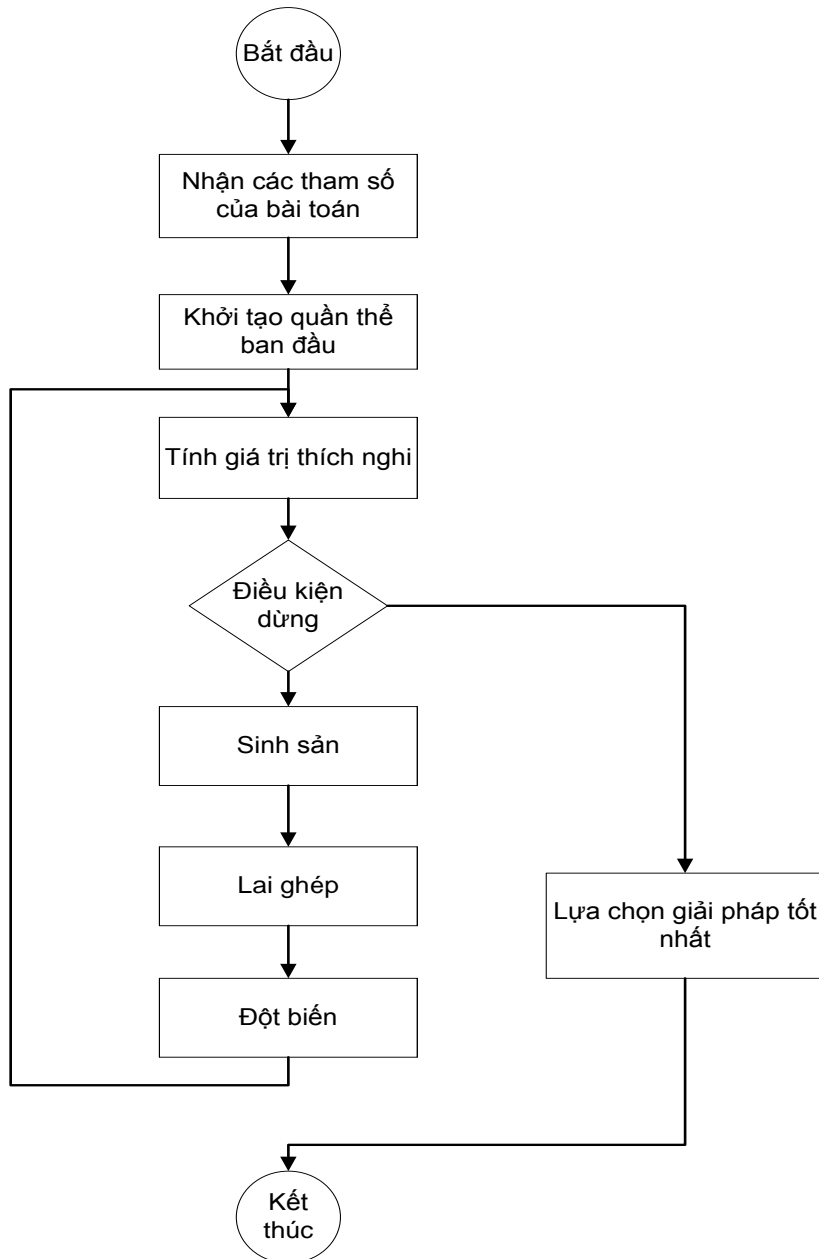
b. Lai ghép

Lai ghép trong tự nhiên là sự kết hợp các tính trạng của bố mẹ để sinh ra thế hệ con. Trong giải thuật di truyền, lai ghép được coi là một sự tổ hợp lại các tính chất (thành phần) trong hai lời giải cha mẹ nào đó để sinh ra một lời giải mới mà có đặc tính mong muốn là tốt hơn thế hệ cha mẹ. Đây là một quá trình xảy ra chủ yếu trong giải thuật di truyền.

c. Đột biến

Đột biến là một sự biến đổi tại một (hay một số) gen của nhiễm sắc thể ban đầu để tạo ra một nhiễm sắc thể mới. Đột biến có xác suất xảy ra thấp hơn lai ghép. Đột biến có thể tạo ra một cá thể mới tốt hơn hoặc xấu hơn cá thể ban đầu. Tuy nhiên trong giải thuật di truyền thì ta luôn muốn tạo ra những phép đột biến cho phép cải thiện lời giải qua từng thế hệ.

I.4. Mô hình giải thuật di truyền



Hình 4: Mô hình của giải thuật di truyền

Với các khái niệm được nêu ở trên, giải thuật di truyền được mô tả như sau:

1. **[Bắt đầu]** Nhận các tham số cho thuật toán.
2. **[Khởi tạo]** Sinh ngẫu nhiên một quần thể gồm n cá thể (là n lời giải cho bài toán).
3. **[Quần thể mới]** Tạo quần thể mới bằng cách lặp lại các bước sau cho đến khi quần thể mới hoàn thành.
 - a. **[Thích nghi]** Ước lượng độ thích nghi $eval(x)$ của mỗi cá thể.

b.[**Kiểm tra**] Kiểm tra điều kiện kết thúc giải thuật.

c.[**Chọn lọc**] Chọn hai cá thể bố mẹ từ quần thể cũ theo độ thích nghi của chúng (cá thể có độ thích nghi càng cao thì càng có nhiều khả năng được chọn)

d.[**Lai ghép**] Với một xác suất lai ghép được chọn, lai ghép hai cá thể bố mẹ để tạo ra một cá thể mới.

e.[**Đột biến**] Với một xác suất đột biến được chọn, biến đổi cá thể mới

4. [**Chọn kết quả**] Nếu điều kiện dừng được thỏa mãn thì thuật toán kết thúc và trả về lời giải tốt nhất trong quần thể hiện tại.

Mặc dù GA có khả năng đạt tới cực trị toàn cục cho quá trình tìm kiếm nhưng do có kết hợp những yếu tố ngẫu nhiên nên tốc độ tìm kiếm nói chung là rất chậm. Mặt khác nó không thể hoàn toàn đạt được tới cực trị toàn cục mà chỉ cho những kết quả xung quanh đó. Đối lập với GA, giải thuật lan truyền ngược sai số (BP) lại cho phép đạt được những cực trị nếu như điểm xuất phát của quá trình tìm kiếm nằm trong vùng cực trị toàn cục.

**PHỤ LỤC 2: MÃ NGUỒN CHƯƠNG TRÌNH LUYỆN MẠNG NƠON VỚI
BƯỚC HỌC VƯỢT KHE NHẬN DẠNG ĐỐI TƯỢNG**

```
% Lap trình tong quat cho thuat toan vuot khe
% Date:15-10-2011
% Version 1
%-----
a=ap;
u0=up;
% huongtim
d1=subs(Ju1,u1);%d1=diff(J,u1)
d2=subs(Ju2,u2);%d2=diff(J,u2)
Ju=[d1 d2];
s0=-Ju;
PPThammuctieu
%J=(u1-5)^2+(u2-5)^2;
Jtruoc=J;
%Buoc 1 -----
%XL=anpha
%XU=beta
XL=a;
u01=u0+XL*s0;
u1=u01(1);
u2=u01(2);
%J=(u1-5)^2+(u2-5)^2;
PPThammuctieu
Jsau=J;
if Jsau > Jtruoc %Jsau=Jsau; Jtruoc=Jtruoc
    XL=0;
    XU=a;
else
    XL=a;
    b=1.5*a;
    XU=b;
    Jtruoc=Jsau;
end
u02=u0+XU*s0;
u1=u02(1);
u2=u02(2);
PPThammuctieu
Jsau=J;
while Jsau<Jtruoc
%     if Jsau > Jtruoc`
%         F=Jtruoc;
%         break
%     else
        XL=a;
        b=1.5*a;
        XU=b;
        Jtruoc=Jsau;
        u03=u0+XU*s0;
```

```

        u1=u03(1);
        u2=u03(2);
        PPThammuctieu
        Jsau=J;
    end
%end
    if Jsau > Jtruoc
        F = Jtruoc;
    end
% Buoc 2 -----
    % FL1=abs(XU-XL)
while (abs(XU-XL)>FD)
    %FL=abs(XU-XL);
    %FL1=FL;
%    if FL <=FD
%        break
%    else
        TH=XL+gama*(XU-XL);
        teta = TH;
        u04=u0+TH*s0;
        u1=u04(1);
        u2=u04(2);
        PPThammuctieu
        Jsau=J;
        if Jsau<F
            XU =TH;
            Jtruoc = F;
        else
            XL=TH;
            u05=u0+XL*s0;
            u1=u05(1);
            u2=u05(2);
            PPThammuctieu
            Jtruoc=J;
        end
    end
%    end
end
tocdohoc=TH;
% if abs(XU-XL)<=FD
%     XL=XU
% end
c=tocdohoc
%Gradient-----
e{3}=t(:,k)-x{3};
J=J+sum(e{3}.^2);
Ju1=diff(J,u1);
Ju2=diff(J,u2);
%Ham muc tieu
e{3}=t(:,k)-x{3};
J=J+sum(e{3}.^2);

```

```

%-----
% Lap trình tong quat cho thuat toan vuot khe
% Date:15-10-2011
% Version 1
%-----
function y = PPTbexulynuocthai(p,t)
clc
clear;
%Nhap du lieu vao he thong
L=3;%so lop
g=inline('1./(1+exp(-x))');%Activation function
unl=[5 16 3];%The units of each layers
% lt=0.3;% learning rate
J=1;
sum=0;
sum1=0;
%Initital the Weights and biases
for i=1:L-1
    for n=1:unl(i)
        for m=1:unl(i+1)
            w{i}(m,n)=1*rand;%The Weights
        end
    end
    for m=1:unl(i+1)
        b{i}(m,1)=1*rand;% The biases
    end
end
w{L}=eye(unl(L));
while (J>0.00001)& sum<1000
    sum=sum+1;%Number of Loops
    J=0;
    for k=1:length(p)
        x{1}=p(:,k);% Inputs vector
    % Feed forward Propagation
        for i=1:L-1
            s{i}=w{i,1}*x{i}+b{i};
            x{i+1}=g(s{i});
            %x{i+1};
        end
        %The caculatar error
        PPTthammuctieu
    %Feed back Propagation
        for i=L-1:-1:1
            e{i}=(x{i+1}.*(1-
x{i+1})).*(w{i+1}'*e{i+1});
            thuattoanvuotkhe%Su dung thuat toan vuot khe de xac
            %dinh toc do hoc cua mang neural
            deltw{i}=(c.*e{i})*x{i};
            w{i}=w{i}+deltw{i};% The modify
Weights

```



```

deltb{i}=(c.*e{i}).*ones(unl(i+1),1);
                                b{i}=b{i}+deltb{i};% The modify
biases
                                end
                                end
                                J=J/2;
                                if mod(sum,100)==0
                                    sum1=sum1+1;
                                    data(sum1,1)=sum;
                                    data(sum1,2)=J;
                                end
                                end
                                % Example a value
                                x{1} = [22 35 1 6 36]';
                                for i=1:L-1
                                    s{i}=w{i}*x{i}+b{i};
                                    x{i+1}=g(s{i});
                                end
                                x{i+1}
                                %x{L}
                                semilogy(data(:,1),data(:,2)),grid
                                %-----

```

PHỤ LỤC 3: MÃ NGUỒN CHƯƠNG TRÌNH LUYỆN MẠNG NƠON VỚI BƯỚC HỌC VƯỢT KHE ĐỀ NHẬN DẠNG CHỮ VIẾT

dinhnghia.h

```

/*
=====
*-----
*
* DE TAI      : HUAN LUYEN MANG NO-RON VOI BUOC HOC TINH THEO NGUYEN LY
VUOT KHE *
* NGON NGU    : C
* TRINH DICH  : VISUAL C++
* TEN TEP     : dinhnghia.h
*-----
*=====
*/
#define SIGMF(x) 1/(1 + exp(-(double)x))//HAM KICH HOAT NO RON
#define DSIGM(y) (float)(y)*(1.0-y)//DAO HAM CUA HAM KICH HOAT NO RON
#define SLNRLV 35 // SO LUONG NO RON LOP VAO
#define SLNRLA 5 // SO LUONG NO RON LOP AN
#define SLNRLR 10 // SO LUONG NO RON LOP RA
#define EPSILON 0.06 // SAI SO TRUNG BINH BINH PHUONG DE DUNG QUA
TRINH LUYEN MANG
#define SLMHL 18 // SO LUONG MAU HUAN LUYEN MANG
#define STEPinit 0.5 // GIA TRI KHOI TAO BUOC HOC, CO THE DUNG CHO
NHIEU TRUONG HOP
#define DCBH 0.0001 // DIEU CHINH BUOC HOC
#define MSDCBH 5 // MAU SO DIEU CHINH BUOC HOC
#define TSDCBH 5 // TU SO DIEU CHINH BUOC HOC
#define BLTD 30000 // BUOC LAP TOI DA
#define FD 1e-1 //

```

taphuanluyen.h

```

/*
=====
*-----
* DE TAI      : HUAN LUYEN MANG NO-RON VOI BUOC HOC TINH THEO NGUYEN LY
VUOT KHE *
* NGON NGU    : C
* TRINH DICH  : VISUAL C++
* TEN TEP     : taphuanluyen.h
*-----
*=====
*/
#include "dinhnghia.h"
//TAP MAU HUAN LUYEN DAU VAO
int TAPHUANLUYEN[SLMHL][SLNRLV] =
{
    { 0,1,1,1,1,1,0, /* 0 */
      1,0,0,0,0,0,1,
      1,0,0,0,0,0,1,
      1,0,0,0,0,0,1,
      0,1,1,1,1,1,0 },
    { 0,0,0,0,0,0,0, /* 1 */
      0,1,0,0,0,0,1,
      1,1,1,1,1,1,1,
      0,0,0,0,0,0,1,

```

```

0,0,0,0,0,0,0 } ,
{ 0,1,0,0,0,0,1, /* 2 */
  1,0,0,0,0,1,1,
  1,0,0,0,1,0,1,
  1,0,0,1,0,0,1,
  0,1,1,0,0,0,1 } ,

{ 1,0,0,0,0,1,0, /* 3 */
  1,0,0,0,0,0,1,
  1,0,0,1,0,0,1,
  1,1,1,0,1,0,1,
  1,0,0,0,1,1,0 } ,

{ 0,0,0,1,1,0,0, /* 4 */
  0,0,1,0,1,0,0,
  0,1,0,0,1,0,0,
  1,1,1,1,1,1,1,
  0,0,0,0,1,0,0 } ,

{ 1,1,1,0,0,1,0, /* 5 */
  1,0,1,0,0,0,1,
  1,0,1,0,0,0,1,
  1,0,1,0,0,0,1,
  1,0,0,1,1,1,0 } ,

{ 0,0,1,1,1,1,0, /* 6 */
  0,1,0,1,0,0,1,
  1,0,0,1,0,0,1,
  1,0,0,1,0,0,1,
  0,0,0,0,1,1,0 } ,

{ 1,0,0,0,0,0,0, /* 7 */
  1,0,0,0,0,0,0,
  1,0,0,1,1,1,1,
  1,0,1,0,0,0,0,
  1,1,0,0,0,0,0 } ,

{ 0,1,1,0,1,1,0, /* 8 */
  1,0,0,1,0,0,1,
  1,0,0,1,0,0,1,
  1,0,0,1,0,0,1,
  0,1,1,0,1,1,0 } ,

{ 0,1,1,0,0,0,0, /* 9 */
  1,0,0,1,0,0,1,
  1,0,0,1,0,0,1,
  1,0,0,1,0,1,0,
  0,1,1,1,1,0,0 } ,

{ 1,1,1,1,0,0,0, /* 4 */
  0,0,0,1,0,0,0,
  0,0,0,1,0,0,0,
  0,0,0,1,0,0,0,
  1,1,1,1,1,1,1 } ,

{ 1,1,1,1,0,1,0, /* 5 */
  1,0,0,1,0,0,1,
  1,0,0,1,0,0,1,
  1,0,0,1,0,0,1,
  1,0,0,0,1,1,0 } ,

```

```

{ 1,0,0,0,0,0,0, /* 7 */
  1,0,0,0,0,0,0,
  1,0,0,1,0,0,0,
  1,1,1,1,1,1,1,
  0,0,0,1,0,0,0 },

{ 0,1,0,0,0,1,0, /* 3 */
  1,0,0,0,0,0,1,
  1,0,0,1,0,0,1,
  1,0,1,0,1,0,1,
  0,1,1,0,1,1,0 },

{ 1,0,0,0,0,1,1, /* 2 */
  1,0,0,0,1,0,1,
  1,0,0,1,0,0,1,
  1,0,1,0,0,0,1,
  1,1,0,0,0,0,1 },

{ 1,1,1,1,0,0,0, /* 4 */
  0,0,0,1,0,0,0,
  0,0,0,1,0,0,0,
  1,1,1,1,1,1,1,
  0,0,0,1,0,0,0 },

{ 1,1,1,1,1,1,1, /* 0 */
  1,0,0,0,0,0,1,
  1,0,0,0,0,0,1,
  1,0,0,0,0,0,1,
  1,1,1,1,1,1,1 },

{ 0,1,1,0,0,0,1, /* 9 */
  1,0,0,1,0,0,1,
  1,0,0,1,0,0,1,
  1,0,0,1,0,0,1,
  0,1,1,1,1,1,1 } };

//DAU RA MONG MUON TUONG UNG
int  DRMM[SLMHL][SLNRLR] =
{
    { 1,0,0,0,0,0,0,0,0,0,0 }, /* 0 */
    { 0,1,0,0,0,0,0,0,0,0,0 }, /* 1 */
    { 0,0,1,0,0,0,0,0,0,0,0 }, /* 2 */
    { 0,0,0,1,0,0,0,0,0,0,0 }, /* 3 */
    { 0,0,0,0,1,0,0,0,0,0,0 }, /* 4 */
    { 0,0,0,0,0,1,0,0,0,0,0 }, /* 5 */
    { 0,0,0,0,0,0,1,0,0,0,0 }, /* 6 */
    { 0,0,0,0,0,0,0,1,0,0,0 }, /* 7 */
    { 0,0,0,0,0,0,0,0,1,0,0 }, /* 8 */
    { 0,0,0,0,0,0,0,0,0,1,0 }, /* 9 */
    { 0,0,0,0,0,1,0,0,0,0,0 }, /* 4 */
    { 0,0,0,0,0,1,0,0,0,0,0 }, /* 5 */
    { 0,0,0,0,0,0,1,0,0,0,0 }, /* 7 */
    { 0,0,0,1,0,0,0,0,0,0,0 }, /* 3 */
    { 0,0,1,0,0,0,0,0,0,0,0 }, /* 2 */
    { 0,0,0,0,1,0,0,0,0,0,0 }, /* 4 */
    { 1,0,0,0,0,0,0,0,0,0,0 }, /* 0 */
    { 0,0,0,0,0,0,0,0,0,0,1 } }; /* 9 */

```

backprop5.c

/*

=====

```

*-----
*
* DE TAI      : HUAN LUYEN MANG NO-RON VOI BUOC HOC TINH THEO NGUYEN LY
VUOT KHE *
* NGON NGU    : C
* TRINH DICH   : VISUAL C++
* TEN TEP     : backprop.c
*-----
*
*=====
*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "taphuanluyen.h"
#include "dinhnghia.h"
/***** DINH NGHIA CAC BIEN TOAN CUC *****/
float MTTSLA[SLNRLV][SLNRLA]; //MA TRAN TRONG SO LOP AN
float MTTSLR[SLNRLA][SLNRLR]; //MA TRAN TRONG SO LOP RA
float BTMTTSLA[SLNRLV][SLNRLA]; //BIEN THIEN MA TRAN TRONG SO LOP AN
float BTMTTSLR[SLNRLA][SLNRLR]; //BIEN THIEN MA TRAN TRONG SO LOP RA
float x[SLNRLV]; //VEC-TO DAU VAO LOP VAO
float y[SLNRLA]; //VEC TO DAU RA LOP AN
float z[SLNRLR]; //VEC TO DAU RA LOP RA
float HW1[SLNRLV][SLNRLA]; //KHONG SU DUNG
float HW2[SLNRLV][SLNRLA]; //KHONG SU DUNG
float OW1[SLNRLA][SLNRLR]; //KHONG SU DUNG
float OW2[SLNRLA][SLNRLR]; //KHONG SU DUNG
float SSLA[SLNRLA]; //SAI SO LOP AN
float SSLR[SLNRLR]; //SAI SO LOP RA
int PATR[SLMHL];
float ECM[SLMHL];
float TOCDOHOC=2; //TOC DO HOC
int SOBUOCLAP=0;
float BVK=0; //BUOC VUOT KHE
int NBS=0;
float FX[SLMHL];
float F[SLMHL];
float A,GAMA;
float QUANTINH=0.1; //TOAN HANG QUAN TINH
int MTDVKT[35]; //MA TRAN DAU VAO KIEM TRA
long int itr;
int HTHL; //HOAN THANH HUAN LUYEN
int LCBH; //LUA CHON BUOC HOC
int RESET; //RESET MANG
int RES=1; //RESET MANG
int SDM; //SU DUNG MANG
/***** KET THUC DINH NGHIA CAC BIEN TOAN CUC *****/

/***** CAC NGUYEN MAU HAM *****/

int KHOITAOMANG();
void QUATRINH HUANLUYEN();
void DAPUNGDAURA(int afer[]);
float GIATRIHAMMUCTIEU(int x[],float y[],int SIZE);
void DIEUCHINHTRONGSO(int k);
void HAMMUCTIEU();
void BUOCLAP();

/***** CAC HAM VA CAC THU TUC *****/

```

```

/*-----
Ten Ham:          KHOITAOMANG
Mo ta:            1. KHOI TAO MA TRAN TRONG SO LOP AN, VOI CACS GIA TRI
NGAU NHIEN BI CHAN
                  2. KHOI TAO MA TRAN TRONG SO LOP RA, VOI CACS GIA
TRI NGAU NHIEN BI CHAN
Cac dau vao:      KHONG CO
Gia tri tra ve:   1
-----*/

int KHOITAOMANG()
{
    int i,j;
    int ch;
    int num;

    NBS=0;
    HTHL=2;
    RESET=0;
    RES=1;
    MHL=0;
    srand(time(0));
    for(i=0;i<SLNRLV;i++)
        for(j=0;j<SLNRLA;j++)
        {
            MTTSLA[i][j] = -0.5+ (float) rand()/RAND_MAX;
            BTMTTSLA[i][j] = 0;
        }

    for(i=0;i<SLNRLA;i++)
        for(j=0;j<SLNRLR;j++)
        {
            MTTSLR[i][j] = -0.5 + (float) rand()/RAND_MAX;
            BTMTTSLR[i][j] = 0;
        }

    for(i=0;i<SLMHL;i++)
        PATR[i] = 0;

    for(i=0;i<35;i++)
        MTDVKT[i]=0;
    return 1;
}

/*-----
Ten Thu tuc:      TOCDOHOCZERO
Mo ta:            1. CAC DAU RA CUA MANG VAN LA ZERO, VI TOC DO HOC LA
ZERO.
                  2. CAC SAI SO HOAN TOAN BANG VOI CAC DAU RA MONG
MUON.
                  3. THU TUC NAY DUNG DE PHUC VU CHO VIEC TINH
TOAN BUOC VUOT KHE.
Cac dau vao:      KHONG CO
Gia tri tra ve:   KHONG CO
-----
*/

void TOCDOHOCZERO(void)
{
    int t;
    TOCDOHOC=0;

```

```

        BUOCLAP();
        HAMMUCTIEU();
        for(t=0;t<SLMHL;t++)
            FX[t] = ECM[t];
    }

/*-----
Ten Thu tuc:      TINHBUOCHOCVUOTKHE
Ngay sua   :      03-11-2007
Phien ban   :      V2
Mo ta:         THU TUC TINH BUOC HOC THEO NGUYEN LY VUOT KHE
Cac dau vao:     KHONG CO
Gia tri tra ve:  KHONG CO
-----*/

void TINHBUOCHOCVUOTKHE(void)
{
    float XL,XU,FL[SLMHL],temp;
    int i,t,j;
    if(NBS==0)
    {
        A=0.5;
        GAMA=0.1;
        TOCDOHOC=A;
    }
    for(t=0;t<SLMHL;t++)
        FL[t]=FX[t];
    XL=0;

BUOC1:

    TOCDOHOC=A;
    BUOCLAP();
    HAMMUCTIEU();
    for(t=0;t<SLMHL;t++)
        F[t] = ECM[t];
        if(F[t]>FL[t])
        {
            XU=A;
            goto BUOC2;
        }
        //XL=A;
    for(t=0;t<SLMHL;t++)
        FL[t]=F[t];
    XL=A;
    A=1.5*A;
    XU=A;
    goto BUOC1;
BUOC2:

    if(FD>=(XU-FL))
        goto BUOC3;
    A=XL+GAMA*(XU-XL);
    temp=TOCDOHOC;
    TOCDOHOC=A;
    BUOCLAP();
    HAMMUCTIEU();
    TOCDOHOC=temp;
    for(t=0;t<SLMHL;t++)
        F[t] = ECM[t];
    for(t=0;t<SLMHL;t++)

```

```

        if (FL[t]>F[t])
        {
            XU=A;
            for (j=0; j<SLMHL; j++)
                FL[j]=F[j];
            goto BUOC2;
        }
    for (t=0; t<SLMHL; t++)
        if (F[t]>FL[t])
        {
            for (j=0; j<SLMHL; j++)
                FL[j]=F[j];
            XL=A;
            goto BUOC2;
        }
    BUOC3:
        for (t=0; t<SLMHL; t++)
            FX[t]=F[t];
        NBS=NBS+1;
        TOCDOHOC=A;
    }
/*-----
-----
Ten Thu tuc:    TINHTOANBUOCHOC
Mo ta:         TINH TOAN TOC DO HOC SAU MOI BUOC LAP
Cac dau vao:   KHONG CO
Gia tri tra ve: KHONG CO
-----*/

void TINHTOANBUOCHOC()
{
    TOCDOHOC=TSDCBH/ (SOBUOCLAP*0.001+MSDCBH);
}

/*-----
-----
Ten Thu tuc:    BUOCLAP
Mo ta:         DIEU CHINH CAC TRONG SO CUA MANG
Cac dau vao:   KHONG CO
Gia tri tra ve: KHONG CO
-----*/

void BUOCLAP()
{
    int i;
    i=0;
    //LUA CHON MAU HUAN LUYEN NGAU NHIEN: i = (int) (SLMHL*rnd), VOI
    0<rnd<1
    do
    {
        i = (int) (SLMHL*(float) rand() / RAND_MAX);
    }while(PATR[i]);
    DAPUNGDAURA(TAPHUANLUYEN[i]);
    DIEUCHINHTRONGSO(i);
}
/*-----
-----
Ten Thu tuc:    HUANLUYENCODINH
Mo ta:         HUAN LUYEN MANG DE TB-BINH PHUONG SAI SO DAU RA NHO HON
EPSILON CHO TRUOC
                BUOC HOC LA MOT HANG SO.
Cac dau vao:   KHONG CO
Gia tri tra ve: KHONG CO

```



```

-----*/

void HUANLUYENCODINH()
{
    int t,l;
    printf("\nDANG HUAN LUYEN MANG VOI BUOC HOC CO DINH...\n");
START:
    if(RES==0)
        printf("\nKHOI DONG VA HUAN LUYEN LAI MANG!");
    KHOITAOMANG();
    //Luyen mang
    do
    {
        TOCDOHOC=0.2;
        BUOCLAP();
        HAMMUCTIEU();
        l = 1;
        for(t=0;t<SLMHL;t++)
        {
            PATR[t] = ECM[t] < EPSILON;
            l = l && (PATR[t]);
        }
        SOBUOCLAP++;
        RESET++;
        if(RESET>15000)break;
        if(SOBUOCLAP>BLTD)
        {
            HTHL=0;
            printf("\nQUA TRINH HUAN LUYEN MANG THAT BAI! \n");
            printf("\nDE NGHI HUAN LUYEN LAI! \n");
            break;
        }
    }while(!l);//while(!l && !kbhit());
    if(RESET>15000)
    {
        RES=0;
        goto START;
    }
    if(SOBUOCLAP<BLTD)
    {
        printf("\n\nMANG DA DUOC HUAN LUYEN XONG SAU: ");
        printf("%ld",RESET);
        printf(" BUOC LAP!\n\n");
    }
}
/*-----
-
Ten Thu tuc:      HUANLUYENGIAMDAN
Mo ta:           HUAN LUYEN MANG DE TB-BINH PHUONG SAI SO DAU RA NHO HON
EPSILON CHO TRUOC
                  BUOC HOC DUOC TINH THEO PHUONG PHAP GIAM DAN DON
GIAN
Cac dau vao:     KHONG CO
Gia tri tra ve:  KHONG CO
-----
-*/

void HUANLUYENGIAMDAN()
{
    int t,l;
    printf("\n\nDANG HUAN LUYEN MANG VOI BUOC HOC GIAM DAN...\n");

```

```

START:
    if (RES==0)
        printf("\n\nKHOI DONG VA HUAN LUYEN LAI MANG!");
    KHOITAOMANG();
    //Luyen mang
    do
    {
        TINHTOANBUOCHOC();
        BUOCLAP();
        HAMMUCTIEU();
        l = 1;
        for (t=0; t<SLMHL; t++)
        {
            PATR[t] = ECM[t] < EPSILON;
            l = l && (PATR[t]);
        }
        SOBUOCLAP++;
        RESET++;
        if (RESET>3000) break;
        if (SOBUOCLAP>BLTD)
        {
            HTHL=0;
            printf("\nQUA TRINH HUAN LUYEN MANG THAT BAI! \n");
            printf("\nDE NGHI HUAN LUYEN LAI! \n");
            break;
        }
    } while (!l); //while (!l && !kbhit());
    if (RESET>3000)
    {
        if (SOBUOCLAP<BLTD)
        {
            HTHL=1;
            printf("\n\nMANG DA DUOC HUAN LUYEN XONG SAU: ");
            printf("%ld", RESET);
            printf(" BUOC LAP!\n\n");
        }
    }
}
/*-----
-
Ten Thu tuc:      HUANLUYENVUOTKHE
Mo ta:           HUAN LUYEN MANG DE TB-BINH PHUONG SAI SO DAU RA NHO HON
EPSILON CHO TRUOC
                  BUOC HOC DUOC TINH THEO NGUYEN LY VUOT KHE
Cac dau vao:     KHONG CO
Gia tri tra ve:  KHONG CO
-----
*/

void HUANLUYENVUOTKHE()
{
    int t, l;
    printf("\n\nDANG HUAN LUYEN MANG THEO BUOC VUOT KHE...\n");
START:
    if (RES==0) {
        printf("\nKHOI DONG VA HUAN LUYEN LAI MANG!");
    }
    KHOITAOMANG();
    TOCDOHOCZERO();
    TINHBUOCHOCVUOTKHE();
    //Luyen mang

```

```

do
{
    TINHBUOCHOCVUOTKHE();
    BUOCLAP();
    HAMMUCTIEU();

    l = 1;
    for(t=0;t<SLMHL;t++)
    {
        PATR[t] = ECM[t] < EPSILON;
        l = l && (PATR[t]);
    }
    SOBUOCLAP++;
    RESET++;
//    printf("lan lap thu %d\n",RESET);
    if(RESET>100)break;
    if(SOBUOCLAP>BLTD)
    {
        HTHL=0;
        printf("\nQUA TRINH HUAN LUYEN MANG THAT BAI! \n");
        printf("\nDE NGHI HUAN LUYEN LAI! \n");
        break;
    }

}while(!l); //while(!l && !kbhit());
if(RESET>100)
{
    RES=0;
    goto START;
}
if(SOBUOCLAP<BLTD)
{
    HTHL=1;
    printf("\n\nMANG DA DUOC HUAN LUYEN XONG SAU: ");
    printf("%ld",RESET);
    printf(" BUOC LAP!\n\n");
}
}

/*-----
-
Ten Thu tuc:    DAPUNGDAURA
Mo ta:         TINH CAC DAU RA LOP AN, y, VA DAU RA LOP RA, z, TU DAU
VAO LOPA VAO, x.
Cac dau vao:   VEC-TO DAU VAO x[i]
Gia tri tra ve: KHONG CO
-----
*/

void DAPUNGDAURA(int afer[])
{
    int i,j;
    float totin;

    for(i=0;i<SLNRLV;i++)
        x[i] = (float)afer[i];
    for(j=0;j<SLNRLA;j++)
    {
        totin = 0;
        for(i=0;i<SLNRLV;i++)
            totin = totin + x[i]*MTTSLA[i][j];
    }
}

```

```

        y[j] = SIGMF(totin);
    }
    for(j=0;j<SLNRLR;j++)
    {
        totin = 0;
        for(i=0;i<SLNRLA;i++)
            totin = totin + y[i]*MTTSLR[i][j];
        z[j] = SIGMF(totin);
    }
}

/*-----
Ten Ham:      TB_BINHPHUONGSAISO
Mo ta:        TINH BINH PHUONG SAI SO DAU RA CUA MANG THEO VEC-TO DAU
RA VA VEC-TO DICH
Cac dau vao:  1. VEC-TO DAU RA MONG MUON DRMM[i],
                2. VEC-TO DAU RA CU LOP RA z[i],
                3. SO LUONG NO-RON CUA LAOP DAU RA, SLNRLR
Gia tri tra ve: TRUNG BINH BINH PHUONG SAI SO DAU RA CUA LOP RA
-----*/

float TB_BINHPHUONGSAISO(int a[],float b[],int SIZE)
{
    int i;
    float e=0;

    for(i=0;i<SIZE;i++)
        e = e + ((float)a[i] - b[i])*(a[i] - b[i]);
    e = 0.5 * e;
    return e;
}

/*-----
-
Ten Thu tuc:  SAISODAURA
Mo ta:        TINH SAI SO DAU RA CUA MANG THEO VEC-TO DAU RA VA VEC-TO
DICH
                1. VEC-TO DAU RA MONG MUON DRMM[i],
                2. VEC-TO DAU RA CU LOP RA z[i],
Cac dau vao:  SO THU TU MAU
Gia tri tra ve: KHONG CO
-----*/

void SAISODAURA(int i)
{
    int j;
    for(j=0;j<SLNRLR;j++)
        SSLR[j] = 0;

    for(j=0;j<SLNRLR;j++)
        SSLR[j] = z[j] - (float)DRMM[i][j];
}

/*-----
Ten Thu tuc:  SAISOLOPAN
Mo ta:        TINH SAI SO DAU RA CUA LOP AN THEO SAI SO LOP RA VA DAO
HAM CUA SIGMF
Cac dau vao:  KHONG CO
Gia tri tra ve: KHONG CO
-----*/

```

```

void SAISOLOPAN()
{
    int i,j;
    for(i=0;i<SLNRLA;i++)
        SSLA[i] = 0;

    for(i=0;i<SLNRLA;i++)
        for(j=0;j<SLNRLR;j++)
            SSLA[i] = SSLA[i] + MTTSLR[i][j]*z[j]*(1-z[j])*SSLR[j];
}

/*-----
Ten Thu tuc:    DIEUCHINHTRONGSO
Mo ta:          CAP NHAT MA TRAN TRONG SO LOP AN VA LOP RA
Cac dau vao:    SO THU TU MAU
Gia tri tra ve: KHONG CO
-----*/

void DIEUCHINHTRONGSO(int k)
{
    int i,j;
    float temp;

    SAISODAU(k);
    SAISOLOPAN();

    for(i=0;i<SLNRLA;i++)
    {
        for(j=0;j<SLNRLR;j++)
        {
            temp = -TOCDOHOC*y[i]*z[j]*(1-z[j])*SSLR[j];
            MTTSLR[i][j] = MTTSLR[i][j] + temp +
QUANTINH*BTMTTSLR[i][j];
            BTMTTSLR[i][j] = temp;
        }
    }
    for(i=0;i<SLNRLV;i++)
    {
        for(j=0;j<SLNRLA;j++)
        {
            temp = -TOCDOHOC*x[i]*y[j]*(1-y[j])*SSLA[j];
            MTTSLA[i][j] = MTTSLA[i][j] + temp +
QUANTINH*BTMTTSLA[i][j];
            BTMTTSLA[i][j] = temp;
        }
    }
}

/*-----
Ten Thu tuc:    HAMMUCTIEU
Mo ta:          TINH BINH PHUONG SAI SO DAU RA CUA MANG THEO VEC-TO DAU
RA VA VEC-TO DICH
Cac dau vao:    KHONG CO
Gia tri tra ve: KHONG CO
-----*/

void HAMMUCTIEU()
{
    int i;

```

```

        for(i=0;i<SLMHL;i++)
        {
            DAPUNGDAURA(TAPHUANLUYEN[i]);
            ECM[i]=TB_BINHPHUONGSAISO(DRMM[i],z,SLNRLR);
        }
    }

/*-----
Ten Thu tục:      KIEMTRAMANG
Mo ta:           NGUOI SU DUNG NHAP MA TRAN DAU VAO, TINH TOAN DAP UNG
DAU RA CUA MANG
Cac dau vao:     KHONG CO
Gia tri tra ve:  KHONG CO
-----*/

void KIEMTRAMANG()
{
    int i,j,t,ind;
    float temp;
    float tempz1[SLNRLR];
    float tempz2[3];
    ind=0;

    for(;;)
    {
        printf("\n\nXIN MOI DUA VEC-TO DAU VAO DE KIEM TRA
MANG\n\n");
        for(i=0;i<7;i++)
        {
            for(j=0;j<5;j++)
            scanf("%d",&MTDVKT[j*7+i]);
            printf("\n");
        }
        printf("\n\nDAP UNG DAU RA CUA MANG:\n");
        DAPUNGDAURA(MTDVKT);
        for(i=0;i<SLNRLR;i++)
            printf("\nNO-RON %d, Z = %f",i,z[i]);
        for(i=0;i<SLNRLR;i++)
            tempz1[i]=z[i];
        printf("\n\nBA DAU RA CO DAP UNG CAO NHAT UNG VOI VEC-TO DAU
VAO:\n");
        for(t=0;t<3;t++)
        {
            temp=0;
            ind=0;
            for(i=0;i<SLNRLR;i++)
                if(temp<=tempz1[i])
                {
                    temp=tempz1[i];
                    ind=i;
                }
            tempz2[t]=temp;
            printf("\nDAU RA THU %d LA %3.0f%%",ind,
tempz2[t]*100);
            tempz1[ind]=0;
        }

        temp=z[0];
        ind=0;
    }
}

```

```

        for(i=0;i<SLNRLR;i++)
            if(temp<=z[i])
            {
                temp=z[i];
                ind=i;
            }
        printf("\n\nKET LUAN CUA MANG NO-RON:\n");
        printf("\nVEC-TO DAU VAO MANG LA MA CUA KY TU:
%d",ind);

    }
}

/*-----
Ten Thu tuc:    MATRANTRONGSOLOPAN
Mo ta:          IN MA TRAN TRONG SO LOP AN RA MAN HINH
Cac dau vao:    KHONG CO
Gia tri tra ve: KHONG CO
-----*/

void MATRANTRONGSOLOPAN()
{
    int i,j;
    printf("\n\nMA TRAN TRONG SO LOP AN MTTSLA[slnrlv][slnrla]:\n\n");
    for(i=0;i<SLNRLV;i++)
    {
        for(j=0;j<SLNRLA;j++)
        {
            if(MTTSLA[i][j]>=0)
                printf("+%f ",MTTSLA[i][j]);
            else
                printf("%f ",MTTSLA[i][j]);
        }
        printf("\n");
    }
}

/*-----
Ten Thu tuc:    MATRANTRONGSOLOPRA
Mo ta:          IN MA TRAN TRONG SO LOP RA RA MAN HINH
Cac dau vao:    KHONG CO
Gia tri tra ve: KHONG CO
-----*/

void MATRANTRONGSOLOPRA()
{
    int i,j;
    printf("\n\nMA TRAN TRONG SO LOP RA MTTSLR[slnrla][slnrlr]:\n\n");
    for(j=0;j<SLNRLR;j++)
    {
        for(i=0;i<SLNRLA;i++)
        {
            if(MTTSLR[i][j]>=0)
                printf("+%f ",MTTSLR[i][j]);
            else
                printf("%f ",MTTSLR[i][j]);
        }
        printf("\n");
    }
}

```

```

/***** HAM CHINH *****/

void main(int argc, char *argv[])
{
    int read;
    char c;

    KHOITAOMANG();

    printf("*****\n\n");
    printf("*   CHUONG TRINH HUAN LUYEN MANG NO-RON   *\n\n");
    printf("*   BUOC HOC TINH THEO NGUYEN LY VUOT KHE   *\n\n");
    printf("*****\n\n");
/*
    printf("/Vao\\ /~~~~Lop an~~~~\\ /~~~~Lop ra~~~~\\n");
    printf("  --      -----      ----      -----      ----  \n");
    printf("|  |  |      |  |      |Y|      |  |      |Z\n");
    printf("|X  |->|  W11  |->|  f1  |->|  W21  |->|  f2  |->\n");
    printf("|  |  |  (35x5)|  |      |  |  (5x10)  |  |      |\n");
    printf("|  |  |      |  |      |  |      |  |      |\n");
    printf("  --      -----      ----      -----      ----  \n");
    printf("\\\\35/  \\\\_Y=f1(W11.X)\\_/  \\\\_Z=f1(W11.X)\\_/  \n\n");
    printf("                                f=sigm(net)                                \n\n");

    printf("\nSU DUNG MANG: TRUOC KHI HOC, t, SAU KHI HOC, s.\n");
    SDM = getchar();
    if(SDM=='t')
        KIEMTRAMANG();// SU DUNG MANG TRUOC KHI HUAN LUYEN

    else if(SDM=='s')
    {
        printf("\n\nLUA CHON LOAI BUOC HOC\n");
        printf("\nCO DINH: c, GIAM DAN: g, NGUYEN LY VUOT KHE:
v\n\n");
        c = getchar();
    }
    LCBH = getchar();
    switch(LCBH)
    {
        case 'c':
            HUANLUYENCODINH();
            break;
        case 'g':
            HUANLUYENGIAMDAN();
            break;
        case 'v':
            HUANLUYENVUOTKHE();
            break;
    }
    */
    HUANLUYENVUOTKHE();
    if(HTHL==1)//NEU MANG DA DUOC HUAN LUYEN XONG
    {
        MATRANTRONGSOLOPAN();
        MATRANTRONGSOLOPRA();
//        KIEMTRAMANG();
    }

}

```