ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

ĐẶNG THỊ THU HIỀN

BÀI TOÁN NỘI SUY VÀ MẠNG NƠRON RBF

LUẬN ÁN TIẾN SĨ CÔNG NGHỆ THÔNG TIN

Hà nội – 2009

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

ĐẶNG THỊ THU HIỀN

BÀI TOÁN NỘI SUY VÀ MẠNG NƠRON RBF

Chuyên ngành: Khoa học máy tính
Mã số: 62.48.01.01

LUẬN ÁN TIẾN SĨ CÔNG NGHỆ THÔNG TIN

NGƯỜI HƯỚNG DẪN KHOA HỌC:
1. PGS.TS Hoàng Xuân Huấn
2. GS.TSKH Huỳnh Hữu Tuệ

Hà nội - 2009

LỜI CẢM ƠN

Luận án được thực hiện tại trường ĐH Công nghệ - ĐHQG Hà nội, dưới sự hướng dẫn của PGS.TS Hoàng Xuân Huấn và GS.TSKH Huỳnh Hữu Tuệ.

Tôi xin bày tỏ lòng biết ơn sâu sắc tới Thầy Hoàng Xuân Huấn, người đã có những định hướng giúp tôi thành công trong việc nghiên cứu của mình. Thầy cũng đã động viên và chỉ bảo cho tôi vượt qua những khó khăn để tôi hoàn thành được luận án này. Tôi cũng chân thành cảm ơn tới Thầy Huỳnh Hữu Tuệ, Thầy đã cho tôi nhiều kiến thức quý báu về nghiên cứu khoa học. Nhờ sự chỉ bảo của Thầy tôi mới hoàn thành tốt luận án.

Tôi cũng xin cảm ơn tới các Thầy Cô thuộc khoa Công nghệ thông tin – ĐH Công nghệ, đã tạo mọi điều kiện thuận lợi giúp tôi trong quá trình làm nghiên cứu sinh.

Cuối cùng, tôi xin gửi lời cảm ơn sâu sắc tới gia đình, bạn bè nơi đã cho tôi điểm tựa vững chắc để tôi có được thành công như ngày hôm nay.

LỜI CAM ĐOAN

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi. Các kết quả được viết chung với các tác giả khác đều được sự đồng ý của đồng tác giả trước khi đưa vào luận án. Các kết quả nêu trong luận án là trung thực và chưa từng được ai công bố trong các công trình nào khác.

Tác giả

DANH MỤC CÁC KÝ HIỆU VÀ TỪ VIẾT TẮT

RBF	Radial Basis Function (Hàm cở sở bán kính)
ANN	Artificial Neural Network (mạng nơ ron nhân tạo)
Feel-forward NN	feel-forward neural network (mạng nơ ron truyền tới)
Recurent NN	Recurent neural network (mạng nơ ron hồi quy)
MLP	Multi-Layer Perceptrons (Perceptron nhiều tầng)
LMS	Least-Mean Square (cực tiểu trung bình bình phương)
BP	Back Propagation (lan truyền ngược)
HDH	Thuật toán lặp hai pha mới phát triển
QHDH	Thuật toán lặp một pha mới phát triển
QTL	Thuật toán huấn luyện nhanh Looney giới thiệu
QTH	Thuật toán huấn luyện nhanh theo gợi ý của Haykin

DANH MỤC CÁC BẢNG

DANH MỤC CÁC HÌNH VẼ

MỤC LỤC

MỞ ĐẦU

Nội suy hàm số là một bài toán cổ điển nhưng quan trọng trong giải tích số, nhận dạng mẫu và có nhiều ứng dụng rộng rãi. Bài toán nội suy được mô tả như sau: một hàm chưa xác định cụ thể $f:D(\subset R^n)\to R^m$ nhưng đã xác định được một tập mẫu $\{x^k, y^k\}_{k=1}^N$ trong đó $x^k \in R^n$, $y^k \in R^m$ ($\forall k = 1,..,N$) thỏa mãn $f(x^k)=y^k$, cần tìm hàm g có dạng *đủ tốt* đã biết thỏa mãn hệ phương trình nội suy $g(x^k) = y^k \ \forall k$.

Trường hợp một chiều, bài toán đã được Lagrange (thế kỷ 18) nghiên cứu giải quyết khá đầy đủ nhờ dùng hàm nội suy đa thức. Cùng với sự phát triển các ứng dụng nhờ sử dụng máy tính trong nửa sau thế kỷ 20, sự phát triển của lý thuyết nội suy Spline và sóng nhỏ (wavelet)… đã tạo nên cơ sở lý thuyết và thực tiễn khá hoàn thiện cho nội suy hàm một biến.

Tuy nhiên, đa số các bài toán nội suy trong các ứng dụng thực tiễn lại là bài toán nội suy nhiều biến. Do các khó khăn trong xử lý toán học và nhu cầu ứng dụng trước đây chưa nhiều nên bài toán nội suy nhiều biến mới được quan tâm nhiều trong 50 năm gần đây. Thoạt tiên, người ta phát triển nội suy nhiều biến theo hướng sử dụng đa thức. Các sơ đồ chính được Franke(1982) và Boor(1987) đúc kết (có thể xem [9]). Các sơ đồ này có độ phức tạp cao và kết quả ứng dụng không tốt.

Phương *pháp k- láng giềng gần nhất* được Cover và Hart (1967) đề xuất khá sớm về mặt lý thuyết, nhưng chỉ đến khi Duda và Hart (1973) đưa ra tổng quan đầy đủ thì phương pháp này mới được ứng dụng rộng rãi và được phát triển thêm theo hướng *hồi qui trọng số địa phương*. Cách tiếp cận này cho ra một phương pháp đơn giản dễ sử dụng. Tuy nhiên, nhược điểm cơ bản của nó là chỉ xác định thu hẹp địa phương của hàm nội suy khi biết điểm cần tính giá trị hàm, nên không dùng được cho bài toán cần xác định trước hàm nội suy để nội suy hàm số tại điểm tùy ý.

Trong 30 năm gần đây. Mạng nơron nhân tạo là cách tiếp cận tốt để khắc phục những nhược điểm trên. Mô hình đầu tiên về mạng nơron nhân tạo được McCelland và Pit (1943) đề xuất để nhận dạng mẫu. Rosenblatt (1953) và Widrow

(1960) đã xây dựng các perceptron một tầng theo mô hình này, với các luật học sửa lỗi và bình phương tối thiểu. Việc nghiên cứu tiếp theo bị chững lại gần một thập niên do nhận xét của Minsky và Papett(1969) về các nhược điểm của perceptron một tầng. Đến giữa những năm 1980 mạng MLP được nghiên cứu và ứng dụng lại nhờ thuật toán lan truyền ngược sai số (Rumelhart và McCelland 1986; Parker 1985) và trở thành công cụ mạnh để xấp xỉ hàm nhiều biến. Tuy vậy, mạng MLP có thời gian huấn luyện lâu, chất lượng mạng tùy thuộc vào hiệu quả giải bài toán cực trị và đến nay chưa có phương pháp tốt nào để xác định kiến trúc *đủ tốt* cho mạng. Hơn nữa chủ yếu nó chỉ dùng để xấp xỉ hàm chứ không bảo đảm có thể giải được bài toán nội suy.

Powell (1987) đề xuất dùng các hàm cơ sở bán kính để giải quyết bài toán nội suy nhiều biến [49]. Kỹ thuật này được Broomhead và Low (1988) giới thiệu như là mạng nơron [16]. Mạng RBF với hàm cơ sở bán kính có thể xem là dạng lai của các phương pháp *học dựa trên mẫu* (k-lân cận gần nhất và hồi quy trọng số địa phương) và mạng nơron MLP. Như mạng nơron MLP, hàm nội suy của mạng xác định từ dữ liệu huấn luyện sau khi học, chất lượng mạng tùy thuộc vào thuật toán huấn luyện. Mạng RBF giống với các phương pháp học dựa trên mẫu ở chỗ hàm nội suy là tổ hợp tuyến tính của các hàm RBF, các hàm này chỉ có ảnh hưởng địa phương nên có thể xử lý chúng mà không ảnh hưởng nhiều lên toàn miền xác định.

Mạng RBF chủ yếu dùng để xấp xỉ hàm (mà nội suy là bài toán riêng). Ưu điểm của mạng RBF là thời gian huấn luyện nhanh và luôn đảm bảo hội tụ đến cực trị toàn cục của sai số trung bình phương. Việc xác định tâm, số hàm cơ sở thế nào là tốt vẫn là bài toán mở. Trường hợp số dữ liệu huấn luyện ít (Looney khuyên là nhỏ hơn 200 [38]) thì có thể lấy các mốc nội suy là tâm hàm RBF ở tầng ẩn, mạng này có thể cho nghiệm đúng của hệ phương trình nội suy nên gọi là *mạng nội suy RBF*. Khi số mốc nhiều, do các hạn chế trong kỹ thuật giải hệ phương trình tuyến tính, nên các phương pháp xây dựng mạng và huấn luyện hiện có vẫn tốn thời gian và hiệu quả chưa cao. Mặc dù so với mạng MLP thì việc huấn luyện chúng vẫn nhanh và dễ hơn nhiều [38]. Đến nay cùng với mạng MLP, mạng RBF tỏ ra là một

phương pháp hiệu quả và được ứng dụng rộng rãi để nội suy và xấp xỉ hàm nhiều biến (xem [14,15,30]).

Thuật toán huấn luyện mạng có vai trò rất quan trọng, nó ảnh hưởng trực tiếp đến tính hội tụ và tổng quát của mạng. Trong những năm gần đây đã có nhiều tác giả đề xuất cải tiến thuật toán huấn luyện mạng RBF. Như N. Benoudjit và các cộng sự (2002) [10] đã cải tiến bằng cách tối ưu tham số độ rộng bán kính. M. Lazaro và cộng sự (2003)[37] đề xuất thuật toán mới để tăng tốc độ hội tụ. K.Z Mao và cộng sự (2005) [41] đưa ra cách chọn số nơron tầng ẩn dựa trên cấu trúc dữ liệu để tăng tính tổng quát của mạng. Ta thấy rằng hầu hết những thuật toán này vẫn xác định trọng số tầng ra theo phương pháp Gradient hoặc biến thể của nó. Thời gian huấn luyện lâu, chưa ước lượng được sai số. Khi số lượng dữ liệu lớn sẽ gặp nhiều khó khăn.

Một vấn đề có tính thời sự đang đặt ra trong các bài toán điều khiển học và khai thác tri thức từ dữ liệu (Data mining) là cần giải tốt các bài toán nội suy thời gian thực, trong đó việc xác định lại hàm nội suy khi có dữ liệu bổ sung phải hoàn thành kịp thời.

Nhiệm vụ đặt ra cho tác giả luận án này là nghiên cứu và đề xuất các thuật toán huấn luyện mạng nội suy hiệu quả cho các trường hợp có số mốc nội suy lớn và tìm giải pháp cho bài toán thời gian thực.

Trong luận án chúng tôi đề xuất một thuật toán lặp hai pha huấn luyện mạng nội suy RBF. Pha thứ nhất xác định tham số độ rộng cho các hàm cơ sở bán kính sao cho các trọng số tầng ra được tìm nhờ phép lặp xác định điểm bất động của một ánh xạ co trong pha sau. Phân tích toán học và kết quả thực nghiệm cho thấy thuật toán có những ưu điểm vượt trội so với những thuật toán thông dụng: dùng được khi số mốc nội suy lớn (hàng chục ngàn mốc), dễ ước lượng sai số huấn luyện, thời gian huấn luyện ngắn, tính tổng quát cũng tốt hơn và dễ song song hoá. Trong trường hợp bài toán nội suy có mốc cách đều, thay cho khoảng cách Euclide, chúng tôi dùng khoảng cách Mahalanobis thích hợp và cải tiến thuật toán hai pha thành

thuật toán một pha. Phân tích toán học và kết quả thực nghiệm cho thấy thuật toán này cải thiện đáng kể chất lượng mạng so với thuật toán hai pha cả về thời gian huấn luyện và tính tổng quát. Đối với bài toán thời gian thực, đặc biệt là bài toán động, chúng tôi đề xuất kiến trúc mạng địa phương. Mạng này chia miền xác định thành các miền con chứa số mốc nội suy tương đối bằng nhau, nhờ phương pháp phỏng theo thuật toán xây dựng cây *k-d* quen biết. Sau đó dùng thuật toán huấn luyện hai pha để huấn luyện mạng RBF trên mỗi miền con và ghép chúng lại theo ý tưởng nội suy spline. Phân tích toán học và kết quả thực nghiệm cho thấy chất lượng mạng có nhiều ưu điểm nổi trội.

Các kết quả trên được công bố trong tạp chí khoa học quốc tế Signal Processing và International Journal of Data Mining, Modelling and Management Science (IJDMMM). Hội nghị quốc tế của IEEE và hội thảo trong nước.

Ngoài phần kết luận, luận án được tổ chức như sau. Chương 1 giới thiệu những điểm cơ bản của bài toán nội suy hàm số và mạng nơron nhiều tầng truyền tới cần cho nội dung chính của luận án bao gồm: nội suy đa thức cho hàm một biến, các khái niệm và tiếp cận chính đối với bài toán nội suy hàm nhiều biến, giới thiệu tóm tắt về mạng nơron nhân tạo và các mạng nơron nhiều tầng truyền tới. Chương 2 giới thiệu các khái niệm cơ bản về mạng nơron RBF và mạng nội suy với hàm cơ sở bán kính dạng Gauss. Sau đó chúng tôi mô tả các thuật toán thông dụng để huận luyện mạng. Chương 3 trình bày thuật toán hai pha mới (gọi là thuật toán HDH) để huấn luyện mạng nội suy RBF bao gồm cả phân tích toán học và kết quả thực nghiệm. Chương 4 trình bày thuật toán một pha mới áp dụng cho bài toán nội suy với mốc cách đều. Chương 5 trình bày mạng địa phương RBF áp dụng cho bài toán động, hay bài toán thời gian thực. Cuối cùng chúng tôi đưa ra một số kết luận và đề xuất các nghiên cứu tiếp theo.

CHƯƠNG 1. NỘI SUY HÀM SỐ VÀ MẠNG NƠRON

Nội suy hàm số là một bài toán quan trọng trong giải tích số và nhận dạng mẫu [5,22,30,36,38] đang được ứng dụng rộng rãi. Bài toán nội suy hàm một biến đã được nghiên cứu từ rất sớm gắn liền với các tên tuổi lớn như Lagrange và Newton. Nhưng trong các ứng dụng thực tế ta thường phải giải quyết bài toán nội suy nhiều biến và nó chỉ mới được quan tâm nghiên cứu trong năm mươi năm gần đây cùng với sự phát triển mạnh mẽ của khoa học máy tính.

Đầu tiên, người ta phát triển nội suy nhiều biến theo hướng sử dụng đa thức nhưng không hiệu quả do phức tạp trong tính toán và kết quả ứng dụng không tốt.

Các phương pháp *k- lân cận gần nhất* Cover và Hart (1967) và *hồi quy trọng số địa phương* cho một giải pháp đơn giản, dễ sử dụng với bài toán này và đang là một công cụ tốt. Tuy nhiên các phương pháp này không thể huấn luyện trước được, mà chỉ xác định khi biết điểm cần nội suy. Như vậy, việc xác định giá trị hàm nội suy tại mẫu mới thực hiện khi đã biết mẫu để xác định láng giềng (lân cận). Cách tiếp cận này sẽ gặp khó khăn khi áp dụng cho các bài toán cần xác định trước hàm nội suy.

Mạng nơron nhân tạo là cách tiếp cận tốt để khắc phục những nhược điểm trên. Mặc dù còn vướng nhiều vấn đề mở về lý thuyết, nhưng hiện nay mạng nơron nhân tạo là một công cụ hữu hiệu để giải các bài toán nội suy hàm nhiều biến trong các bài toán ứng dụng thực tiễn. Trong đó thông dụng nhất là mạng MLP và mạng RBF (xem [14,15,30]).

Chương này giới thiệu những điểm cơ bản của bài toán nội suy hàm số và mạng nơron nhiều tầng truyền tới (MLP) cần cho nội dung chính của luận án. Mục 1.1 giới thiệu về bài toán nội suy bao gồm nội suy đa thức cho hàm một biến và các khái niệm và tiếp cận chính đối với bài toán nội suy hàm nhiều biến. Mục 1.2 trình

bày tổng quan về mạng nơron nhân tạo và giới thiệu về các mạng nơron nhiều tầng truyền tới.

1.1. Nội suy hàm số

Trong nhiều bài toán, ta cần tính giá trị của một hàm số tại những điểm của đối số trong miền *D* nào đó của không gian *n*-chiều, nhưng không có biểu diễn tường minh hàm số mà chỉ xác định được giá trị của hàm số trên một tập hữu hạn điểm của *D*. Việc xác định gần đúng hàm này dẫn tới bài toán nội suy và xấp xỉ hàm số.

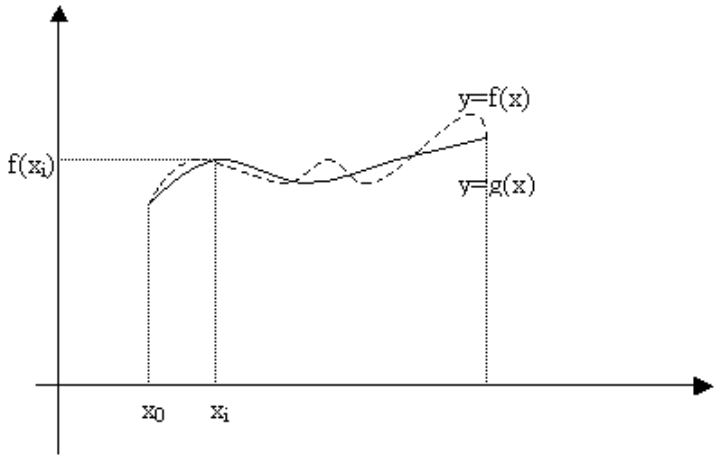
1.1.1. Bài toán nội suy tổng quát

Bài toán nội suy tổng quát được phát biểu như sau. Xét hàm nhiều biến chưa biết f : D ($\subset R^n$)$\rightarrow R^m$ nhưng xác định được một tập mẫu gồm *N* phần tử $\{x^k, y^k\}_{k=1}^N$ trong đó $x^k \in R^n$, $y^k \in R^m$ ($\forall k=1,..,N$) thỏa mãn $f(x^k) = y^k$. Ta cần tìm hàm *g* có dạng *đủ tốt* đã biết thỏa mãn:

$$g(x^i) = y^i, \ \forall i = 1,...,N \tag{1.1}$$

Các điểm x^k được gọi là các mốc nội suy còn hàm *g* gọi là hàm nội suy của *f* .

Hàm nội suy thường được dùng để xấp xỉ hàm *f* trên miền D, giá trị hàm nội suy tính được tại điểm *x* bất kỳ trên miền D gọi là giá trị nội suy của hàm *f* tại *x* (hay gọn hơn là giá trị nội suy tại *x* nếu không có sự nhầm lẫn). Hình 1.1 minh họa hàm nội suy trong trường hợp một biến.



Hình 1.1 Minh họa bài toán nội suy hàm một biến

Những giá trị y^k tại mốc nội suy tương ứng x^k có thể chứa nhiễu và nhiều trường hợp việc giải hệ phương trình (1.1) không có nghiệm đúng đối với dạng hàm *g* đã biết hoặc cho kết quả nội suy không tốt. Một cách tiếp cận khác là thay đòi hỏi thỏa mãn hệ phương trình (1.1) bởi một tiêu chuẩn xấp xỉ tốt nhất (*đủ tốt*) nào đó, thông dụng nhất là tiêu chuẩn cực tiểu tổng bình phương sai số (gọi là bình phương tối thiểu cho gọn). Với cách tiếp cận này ta có bài toán xấp xỉ.

1.1.2. Nội suy hàm một biến

Bài toán nội suy hàm một biến đã được nghiên cứu từ hơn ba thế kỷ đến nay và khá hoàn thiện, đặc biệt là nội suy bằng đa thức. Trước khi đi vào trường hợp đa thức, ta xét lược đồ giải quyết tổng quát.

a) Lược đồ giải quyết cho nội suy hàm một biến.

Trường hợp hàm một biến, bài toán nội suy được phát biểu như sau:

Một hàm số $y = f(x)$ chỉ xác định được tại các điểm $x_0 = a < x_1 < ... < x_n = b$ và $y_i = f(x_i)$ $\forall i \leq n$. Ta cần tìm một biểu thức giải tích đủ đơn giản $g(x)$ để xác định giá trị gần đúng của $y : y \approx g(x)$ tại các điểm $x \in [a,b]$ sao cho tại các điểm x_i ta có: $g(x_i) = y_i$

Lược đồ giải quyết : Giả sử đã biết các giá trị y_i của hàm số tại các mốc nội suy x_i tương ứng. Chọn trước một hàm phụ thuộc $(n+1)$ tham số độc lập $\{c_j\}_{j=0}^{n}$ $\Phi(c_0,c_1,...,c_n,x)$ thoả mãn các điều kiện nhất định. Người ta xác định các c_j cho biểu thức nội suy nhờ hệ phương trình.

$$\Phi(c_0,c_1,...,c_n, x_k) = y_k \ \forall k = 0,...,n \tag{1.2}$$

Nếu $\Phi(c_0,c_1,...,c_n,x)$ là hàm phi tuyến thì hệ phương trình (1.2) không đảm bảo duy nhất nghiệm nên người ta thường chọn Φ có dạng tuyến tính:

$$\Phi(c_0,c_1,...,c_n, x) = \sum_{k=0}^{n} c_k \varphi_k(x) \tag{1.3}$$

Trong đó c_j *(j=1,...,n)* là các tham số cần tìm và $\{\varphi_k(x)\}_{k=0}^{n}$ là họ hàm độc lập tuyến tính cho trước thoả mãn điều kiện định thức ma trận.

$$|\varphi_i(x_k)| \neq 0 \tag{1.4}$$

Khi đó các c_j trong hệ (1.2) luôn giải được duy nhất nghiệm.

Các hàm số $\varphi_k(x)$ thường được chọn theo kinh nghiệm hoặc đơn giản là hàm lũy thừa x^k để dễ tính toán.

Với các $\{c_j\}_{j=0}^{n}$ đã xác định nhờ điều kiện (1.2). Hàm $g(x)=\Phi(c_0,c_1,...,c_n, x)$ là hàm nội suy và dùng làm công thức để tính giá trị $f(x)$.

Khi g lấy trong lớp đa thức bậc n ta dễ dàng xác định được nhờ đa thức nội suy Lagrange mà không phải thực hiện thủ tục giải hệ phương trình tuyến tính.

b) Đa thức nội suy Lagrange

Trường hợp f là hàm một biến với $n+1$ mốc nội suy và hàm nội suy dạng đa thức thì nó phải là đa thức bậc n để hệ phương trình (1.2) có duy nhất nghiệm

(xem [5,22,36]). Khi đó hàm nội suy *g(x)* là đa thức nội suy Lagrange $L_n(x)$ và được xây dựng như sau.

Xây dựng đa thức nội suy Lagrange.

Ký hiệu $L_n(x)$ là đa thức nội suy bậc *n* cần tìm. Ta xây dựng đa thức này dưới dạng

$$L_n(x) = \sum_{k=0}^{n} y_k L_n^k(x) \tag{1.5}$$

Trong đó, $L^k_n(x)$ có *n* nghiệm $x = x_j$; $j \neq k$ và $L^k_n(x_j) = \begin{cases} 1 & khi\ k = j \\ 0 & khi\ k \neq j \end{cases}$

hay $L^k_n(x_k) = \delta^k_j \ \forall j \leq n$; Dễ dàng kiểm tra được

$$L_n^k(x) = \frac{\prod_{i \neq k}(x - x_i)}{\prod_{i \neq k}(x_k - x_i)} \tag{1.6}$$

Thỏa mãn các điều kiện đã nêu và hàm $g(x) = L_n(x)$ thoả mãn hệ phương trình (1.1) và là đa thức nội suy cần tìm.

Sai số nội suy tại điểm *x* được ước lượng bằng công thức:

$$R_n(x) \approx \frac{f^{(n+1)}(c)}{(n+1)!} \prod_{k=0}^{n} (x - x_k)$$

Với *c* là điểm thích hợp thuộc khoảng [a,b].

c) Công thức nội suy Newton cho trường hợp mốc cách đều

Trường hợp các mốc nội suy thỏa mãn điều kiện:

$x_{i+1} - x_i = \Delta x_i = h = \frac{(b-a)}{n}$ $(\forall i = 0,1,...,n-1)$ ta nói các mốc này cách đều.

Khi đó với phép biến đổi $\frac{x - x_0}{h} = t$ các đa thức L^k_n là đa thức bậc *n* theo *t* . Đa thức này chỉ phụ thuộc vào số mốc *n* và giá trị hàm tại các mốc nên có nhiều cách biểu diễn đơn giản, dễ sử dụng. Ở đây chúng tôi giới thiệu công thức Newton tiến để biểu diễn đa thức này. Trước hết ta xây dựng công thức tổng quát.

Công thức tổng quát

$$\text{Đặt,} \quad x - x_0 = th \tag{1.7}$$

$$\text{Ta có,} \quad \begin{cases} x - x_k = (t-k)h & \forall k \le n \\ x - x_k = (t-k)h \end{cases} \tag{1.8}$$

Thay vào (1.6) ta được:

$$L_n^k(x) = P_n^k(t) = \frac{\prod_{i \neq k}(t-i)}{\prod_{i \neq k}(k-i)} = \frac{t(t-1)...(t-k+1)(t-k-1)...(t-n)}{(-1)^{n=k}\,k!(n-k)!} \tag{1.9}$$

$$\text{Hay,} \quad P_n^k(t) = (-1)^{n-k}\frac{C_n^k}{n!}t(t-1)...(t-k+1)(t-k-1)...(t-n) \tag{1.10}$$

Biểu thức này của $P_n^k(t)$ là hàm không phụ thuộc vào các mốc nội suy nên hàm nội suy $P_n(t) = L_n(x)$ cũng vậy. Các biểu diễn công thức này qua các sai phân hữu hạn cho ta các dạng công thức nội suy Newton. Ở đây sẽ trình bày công thức nội suy Newton tiến. Trước khi giới thiệu công thức, ta cần định nghĩa sai phân hữu hạn của hàm số.

Sai phân hữu hạn

Trường hợp các mốc cách đều, tức là: $x_{i+1} - x_i = \Delta x_i = h = const\ (i=1,2,...,n-1)$. Các sai phân hữu hạn của hàm $y = f(x)$ được xác định như sau:

Sai phân cấp một: $\Delta y_i = y_{i+1} - y_i$

Sai phân cấp hai: $\Delta^2 y_i = \Delta y_{i+1} - \Delta y_i$

Sai phân cấp k: $\Delta^k y_i = \Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i$

Với các sai phân được xác định như trên ta có công thức nội suy Newton như sau.

Công thức nội suy Newton

Với phép biến đổi x-$x_0 = th$ như trên đa thức nội suy được biểu diễn bởi công thức :

$$L_n^k(x) = P_n(t) = y_0 + t\Delta y_0 + \frac{t(t-1)}{2!}\Delta^2 y_0 + \frac{t(t-1)...(t-n+1)}{n!}\Delta^n y_0 \quad (1.11)$$

với sai số với hàm f là :

$$R_n(x) = h^{n+1}\frac{t(t-1)...(t-n)}{(n+1)!}f^{(n+1)}(c) \quad (1.12)$$

Sai số này cũng có thể ước lượng thô nhờ thêm vào mốc x_{n+1}:

$$R_n(x) \approx \frac{\Delta^{n+1}y_0}{(n+1)!}t(t-1)...(t-n) \quad (1.13)$$

Khi có nhiều mốc nội suy, hàm nội suy sẽ là đa thức bậc cao. Chúng thuộc loại hàm không ổn định (sai số đối số bé nhưng sai số hàm số lớn), và dễ xảy ra hiện tượng phù hợp trội (overfitting). Tức là cho giá trị nội suy có sai số lớn tại các điểm khác mốc nội suy. Để khắc phục hiện tượng này, phương pháp thông dụng là dùng hàm nội suy Spline.

d) Nội suy Spline

Để khắc phục hiện tượng phù hợp trội khi có nhiều mốc nội suy, người ta dùng các đa thức bậc thấp trên mỗi đoạn con của đoạn *[a,b]* và ghép trơn đến mức cần thiết trên toàn đoạn thành hàm nội suy, các hàm này có tên gọi là hàm Spline.

Hàm Spline

Định nghĩa: Hàm Spline bậc *(m,k)* trên đoạn *[a,b]* là hàm số có các tính chất sau :

1. Tồn tại phân hoạch $a = x_0 < x_1 <....< x_n = b$ của *[a,b]* sao cho trên mỗi đoạn $\Delta_j = [x_j, x_{j+1}]$ $\forall j=0,...,n-1$, nó là đa thức bậc m
2. Trên *[a,b]* nó có đạo hàm cấp k liên tục.

Từ định nghĩa ta thấy để hàm thoả mãn điều kiện 2 thì chỉ cần đạo hàm các cấp $\le k$ ở hai phía của mỗi điểm chia x_i ($i=1,...,n-1$) bằng nhau là đủ. Vì vậy nó còn được gọi là hàm ghép trơn.

Tập các hàm Spline bậc (m,k) trên đoạn $[a,b]$ được ký hiệu là $SP^k_m[a,b]$ nếu $k = m-1$ ta gọi Spline bậc m và ký hiệu $SP_m[a,b]$.

Xây dựng hàm nội suy Spline bậc m

Giả sử $y = f(x)$ đo được tại $n+1$ mốc nội suy $a = x_0 < x_1 <....< x_n = b$ là $y_i = f(x_i)$ và $S_m \in SP_m[a,b]$ là hàm Spline được xác định bởi phân hoạch này. Ta ký hiệu P_k là thu hẹp của S_m trên đoạn $\Delta_k = (x_k, x_{k+1})$, $k=0,...,n-1$.

Bởi vì P_k là đa thức bậc m nên cần xác định $m+1$ hệ số của đa thức trên mỗi đoạn. Vì có n đoạn nên hệ số cần tìm là $n(m+1)$ ẩn số. Tại mỗi điểm ghép x_i ($i=1,...,n-1$) ta có m phương trình :

$$P^{(k)}_{i-1}(x_i) = P^{(k)}_i(x_i), \quad \forall k=0,1,...,m-1 \tag{1.14}$$

Bới vì có $n-1$ điểm ghép trơn nên có $(n-1)m$ phương trình như vậy. Ngoài ra có thêm $n+1$ phương trình từ số liệu ban đầu:

$S_m(x_i) = y_i$, $\forall i=0,1,...,n$ nên ta có $(n+1)+(n-1)m$ phương trình. Vậy còn $n(m+1) – (n+1) – (n-1)m = m-1$ bậc tự do. Do đó để S_m được xác định ta cần có thêm $m-1$ điều kiện nữa.

Trong thực hành ta có thể tìm Spline nội suy bậc m dựa trên mệnh đề sau [5]:

Mệnh đề : Nội suy bậc m của hàm $y = f(x)$ với mốc nội suy $a= x_0 < x_1 <....< x_n =b$: $y_k = f(x_k)$ hoàn toàn xác định nếu biết đa thức nội suy của nó trên đoạn tuỳ ý $\Delta_j = (x_j, x_{j+1})$.

1.1.3. Nội suy hàm nhiều biến

Bài toán nội suy nhiều biến là bài toán thường gặp trong ứng dụng thực tiễn. Để tiện cho trình bày, ta ký hiệu hàm nội suy là $\varphi(x)$ thay cho $g(x)$ và bài toán được phát biểu lại như sau.

a) Bài toán nội suy nhiều biến.

Xét miền giới nội D trong R^n và $f : D\ (\subset R^n) \rightarrow R^m$ là một hàm liên tục xác định trên D. Người ta chỉ mới xác định được tại N điểm $x^1, x^2 \ldots . x^N$ trong D là $f(x^i) = y^i$ với mọi $i=1,2\ldots,N$ và cần tính giá trị của $f(x)$ tại các điểm x khác trong D.

Ta tìm một hàm $\varphi(x)$ xác định trên D có dạng đã biết sao cho:

$$\varphi(x^i)=y^i\ ,\ \forall i=1,\ldots N. \tag{1.15}$$

và dùng $\varphi(x)$ thay cho $f(x)$. Khi m >1, bài toán nội suy tương đương với m bài toán nội suy m hàm nhiều biến giá trị thực, nên để đơn giản ta chỉ cần xét với m=1.

b) Các cách tiếp cận chính giải quyết bài toán

Các hàm nội suy $\varphi(x)$ được chọn dưới dạng thuận tiện cho sử dụng, nên mặc dù các chuyên gia giải tích số đã có những nỗ lực để phát triển lý thuyết nội suy đa thức, nhưng tính toán thường phức tạp và dễ có hiện tượng phù hợp trội.

Mặt khác, đòi hỏi hàm nội suy thoả mãn chặt điều kiện (1.15) gây nên các khó khăn trong tính toán và chọn lớp hàm nội suy mà vẫn có thể cho kết quả xấp xỉ tồi (phù hợp trội). Một hướng đang được ứng dụng rộng rãi là tìm hàm xấp xỉ tốt nhất (hoặc đủ tốt) và dùng nó để nội suy giá trị hàm số tại các điểm khác mốc nội suy. Trong trường hợp này ta xem nó là nghĩa suy rộng của bài toán nội suy. Mặc dù các kết quả trong luận án là xét bài toán nội suy phát biểu ở trên nhưng để so sánh với các phướng pháp được ứng dụng thông dụng, chúng tôi xét cả bài toán xấp xỉ và cách tiếp cận giải quyết.

Sau đây là các phương pháp được ứng dụng rộng rãi nhất.

- *Học dựa trên mẫu*. Thuật ngữ này được T.Mitchell [54] dùng để chỉ các phương pháp *k-lân cận gần nhất*, phương pháp *hồi quy trọng số địa phương*.
- *Mạng Nơron MLP*.
- *Mạng Nơron RBF* (mặc dù T. Mitchell cũng xếp vào lớp học dựa trên mẫu nhưng chúng tôi tách riêng dựa theo đặc tính huấn luyện)

Trong đó phương pháp *k*-lân cận gần nhất với trọng số nghịch đảo khoảng cách cho kết quả là hàm nội suy, còn hồi quy trọng số địa phương và mạng nơron MLP chỉ cho kết quả là hàm xấp xỉ. Các phương pháp học dựa trên mẫu không thể huấn luyện trước được mà chỉ xác định được khi biết điểm cần nội suy. Chất lượng của các mạng nơron tuỳ thuộc vào phương pháp huấn luyện, mạng RBF huấn luyện đầy đủ với số hàm cơ sở ít hơn số mốc nội suy chỉ cho kết quả là hàm xấp xỉ.

Dưới đây sẽ giới thiệu tóm tắt phương pháp *k*-lân cận gần nhất với trọng số nghịch đảo khoảng cách và bài toán xấp xỉ nhiều biến xác dịnh theo phương pháp bình phương tối thiểu. Mạng MLP được giới thiệu ở mục 1.2 còn mạng nơron RBF sẽ được trình bày trong chương 2.

c) Phương pháp k-lân cận gần nhất

Trong phương pháp này, người ta chọn trước số tự nhiên *k*. Với mỗi $x \in D$, ta xác định giá trị $\varphi(x)$ qua giá trị của f tại *k* mốc nội suy gần nó nhất như sau.

Ký hiệu $z_1,...,z_k$ là *k* mốc nội suy gần *x* nhất và *d(u,v)* là khoảng cách của hai điểm *u,v* bất kỳ trong *D*, khi đó $\varphi(x)$ xác định như sau:

$$\varphi(x) = \sum_{j=1}^{k} \rho_j f(z_j) \tag{1.16}$$

Trong đó ρ_i được xác định bởi:

$$\rho_i = \frac{\left[d(x,z_i)\right]^{-1}}{\sum_{j=1}^{k}\left[d(x,z_j)\right]^{-1}} \tag{1.17}$$

Dễ thấy rằng khi *x* dần tới các mốc nội suy thì $\varphi(x)$ xác định như trên hội tụ đến giá trị của f tại mốc nội suy tương ứng khi đối số hội tụ tới mốc này.

Phương pháp này có ưu điểm là cách tính toán đơn giản và dễ thực hiện, tuy nhiên trên thực tế việc xác định giá trị *k* phù hợp là một vấn đề khó (phụ thuộc rất nhiều vào kinh nghiệm đánh giá bài toán thực tế). Hơn nữa, mỗi khi cần xác định giá trị của một điểm, phương pháp này lại tìm trong tất cả các giá trị đã biết để tìm được các mốc gần nhất mới xác định được hàm nội suy, chứ không tính trước hàm để dùng được như mạng nơron. Tuy vậy, phương pháp không đánh giá chặt chẽ được nhưng nó vẫn được ưa dùng trong thực nghiệm.

d) Xấp xỉ bình phương tối thiểu hàm nhiều biến

Bài toán xấp xỉ hàm nhiều biến được xem là bài toán tổng quát mà nội suy là một trường hợp đặc biệt. Phương pháp xấp xỉ bình phương tối thiểu là phương pháp hiệu quả để giải bài toán này.

Bài toán xấp xỉ hàm nhiều biến.

Giả sử $y = f(x)$ là một hàm nào đó, đo được tại *N* điểm $\{x^k\}_{k=1}^{N}$ thuộc miền *D* giới nội trong R^n là $y^k = f(x^k); \forall k = \overline{1..N}$ với $x^k = (x_1^k,...,x_n^k) \in D$ và $y^k \in R^m$. Ta cần tìm một hàm $\varphi(x)$ có dạng cho trước sao cho sai số tại các điểm đã biết là tốt nhất có thể được và dùng nó để xấp xỉ hàm $f(x)$.

Người ta thường dùng tiêu chuẩn cực tiểu tổng bình phương sai số để tìm hàm $\varphi(x)$, trong đó hàm này được chọn dưới dạng $\varphi(x) = \Phi(x, c_1, c_2,..., c_k)$ theo cách xác định các tham số $c_1, c_2,..., c_k$ để cho tổng sai số $\sum_{i=1}^{N}\left\|\varphi(x^i) - y^i\right\|^2$ nhỏ nhất. Chuẩn

thường dùng là chuẩn Euclide: $\left\|\varphi(x^i) - y^i\right\| = \sqrt{\sum_{j=1}^{m}(\varphi_j(x^i) - y_j^{\ i})^2}$ để dễ tìm cực trị theo phương pháp nhân tử Lagrange hoặc Gradient. Nếu $\varphi(x)$ là hàm tuyến tính thì bài toán có duy nhất nghiệm và dễ dàng xác định được.

Khi đó ta nói $\varphi(x)$ là hàm xấp xỉ tốt nhất của $f(x)$ theo bình phương tối thiểu, hay gọn hơn $\varphi(x)$ là hàm xấp xỉ bình phương tối thiểu của $f(x)$. Về mặt hình học, đồ thị hàm $y = \varphi(x)$ không đòi hỏi phải đi qua các điểm mốc như trong phép nội suy.

Trong nhiều trường hợp khi số mốc nội suy lớn (N lớn), để giảm bớt chi phí tính toán, thay vì tính toán với $i = \overline{1..N}$ người ta có thể cho $i = \overline{1..M}$ với $M<N$ để tính $\sum_{i=1}^{M}\left\|\varphi(z^i) - f(z^i)\right\|^2$ nhỏ nhất, với tập $\{z^k\}_{k=1}^{M}$ là những điểm gần x nhất. Phương pháp này thường được gọi là phương pháp địa phương và hàm $\Phi(x, c_1, c_2, ..., c_k)$ thường được chọn theo dạng hàm tuyến tính, và khi đó ta gọi là phương pháp hồi quy trọng số địa phương. Mạng MLP và mạng RBF huấn luyện đầy đủ được trình bày về sau thuộc cũng loại này.

1.2. Giới thiệu về mạng nơron nhân tạo

Bí ẩn về cấu tạo cũng như cơ chế hoạt động của bộ não và hệ thần kinh con người luôn được các nhà khoa học quan tâm nghiên cứu, nhưng hiểu biết về nó vẫn rất hạn chế.

Tuy vậy, từ đầu thế kỷ 20 người ta đã phát hiện ra rằng bộ não con người là mạng lưới chằng chịt các nơron liên kết với nhau, và kích hoạt lẫn nhau theo định đề Hebb (1949) khi có tín hiệu môi trường. Cơ chế này là cơ sở để mô phỏng trên máy tính và hình thành nên cấu trúc của mạng nơron nhân tạo. Lĩnh vực khoa học về *mạng nơron nhân tạo* (Artificial Neural Network- ANN) đã ra đời từ những nỗ lực đó [28,30,31]. Trước hết cần giới thiệu tóm tắt về mạng nơron sinh học.

1.2.1. Mạng nơron sinh học

Đến nay người ta biết rằng các quá trình tính toán trong bộ não sinh học khác với các máy tính số. Công trình nghiên cứu về não đầu tiên thuộc về Ramón Y Cajál (1911). Ông cho rằng hệ thần kinh cấu tạo từ các nơron kết nối với nhau bởi các khớp kết nối. Bộ não xử lý chậm hơn (10^{-3}s) so với các thiết bị vật lý (10^{-9}s), tuy nhiên tốn ít năng lượng hơn và thực hiện được nhiều chức năng hơn. Con người có khoảng 10^{11} nơron, 10^{15} khớp kết nối. Khi con người mới sinh, rất ít khớp kết nối được kết nối với nhau, nhờ quá trình học mà theo thời gian chúng được kết nối ngày càng nhiều. Dưới đây chúng tôi trình bày mô hình của nơron sinh học.

Các thành phần chính của một nơron sinh học.

Các nơron sinh học có kiến trúc giống nhau, bao gồm thân (hay bộ tổng hợp), các xúc tu, trục cảm ứng và các khớp kết nối để truyền tín hiệu qua các xúc tu của các tế bào như mô tả trong hình 1.2.

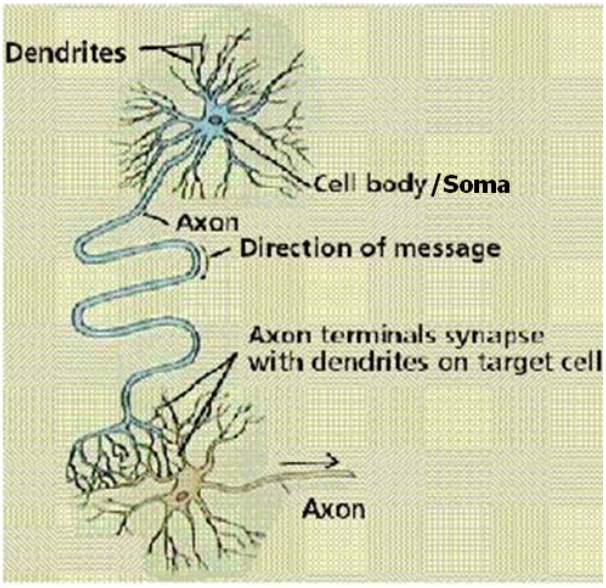
- Thân (Cell body/Soma) của nơron: là nơi tiếp nhận và xử lý các tín hiệu điện nhận được từ các xúc tu. Bên trong Soma các tín hiệu được tổng hợp lại. Một cách thô, có thể xem gần đúng sự tổng hợp ấy như là một phép lấy tổng tất cả các dữ liệu mà nơron nhận được.

- Các xúc tu (Dendrites) dài gắn liền với soma, chúng truyền tín hiệu điện (dưới dạng xung điện thế) nhận từ môi trường hoặc các nơron khác đến cho soma xử lý.

- Trục cảm ứng (Axons): Khác với dendrites, axons có khả năng phát các xung điện thế, chúng là các dây dẫn tín hiệu từ nơron đi các nơi khác. Chỉ khi nào điện thế trong soma vượt quá một giá trị ngưỡng (threshold) nào đó thì axon mới phát một xung điện thế, còn nếu không thì nó ở trạng thái nghỉ.

- Khớp nối (Synapse): axon nối với các dendrites của các nơron khác thông qua những mối nối đặc biệt gọi là khớp (synapse). Khi điện thế của

synapse tăng lên do các xung phát ra từ axon thì synapse sẽ nhả ra một số chất hoá học (neurotransmitters); các chất này mở "cửa" trên dendrites để cho các ions truyền qua. Chính dòng ions này làm thay đổi điện thế trên dendrites, tạo ra các xung dữ liệu lan truyền tới các nơron khác.



Hình 1.2 : Cấu tạo của nơron sinh học

Có thể tóm tắt hoạt động của một nơron như sau: nơron lấy tổng tất cả các điện thế vào mà nó nhận được, và phát ra một xung điện thế nếu tổng ấy lớn hơn một ngưỡng nào đó. Các nơron nối với nhau ở các synapses. Synapse được gọi là mạnh khi nó cho phép truyền dẫn dễ dàng tín hiệu qua các nơron khác. Ngược lại, một synapse yếu sẽ truyền dẫn tín hiệu rất khó khăn.

Các synapses đóng vai trò rất quan trọng trong sự học tập. Khi chúng ta học tập thì hoạt động của các synapses được tăng cường, tạo nên nhiều liên kết mạnh giữa các nơron. Có thể nói rằng người nào học càng giỏi thì càng có nhiều synapses và các synapses ấy càng mạnh mẽ, hay nói cách khác, thì liên kết giữa các nơron càng nhiều, càng nhạy bén. Đây cũng là nguyên tắc chính để ứng dụng nó trong việc học của các mạng nơron [28].

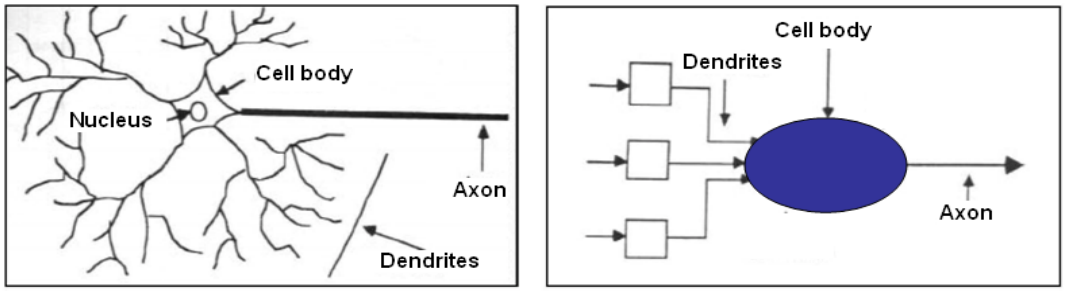
Mặc dù W. Mculloch và W.Pitts (1940) đề xuất mô hình nơron khá sớm nhưng định đề Heb (1949) mới là nền tảng lý luận cho mạng nơron.

Định đề Heb. Khi một tế bào A ở gần tế bào B, kích hoạt thường xuyên hoặc lặp lại việc làm cháy nó thì phát triển một quá trình sinh hoá ở các tế bào làm tăng tác động này.

1.2.2. Mạng nơron nhân tạo

a) Mô phỏng các nơron sinh học

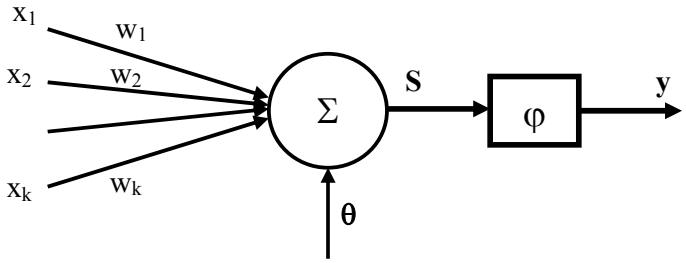
Các nơron nhân tạo (sẽ gọi gọn lại là nơron) là mô phỏng kiến trúc của nơron sinh học, chúng được mô tả trong hình 1.3. Mỗi nơron nhận các giá trị từ các nơron khác như là tín hiệu vào hay từ đầu vào để xử lý. Nếu giá trị tổng hợp các tín hiệu nhận được vượt quá một ngưỡng nào đó, nơron này sẽ gửi tín hiệu đến các nơron khác.



Hình 1.3. Mô phỏng một nơron sinh học

b) Cấu tạo của một nơron

Mỗi nơron là một đơn vị xử lý thông tin có kiến trúc như hình 1.4.



Hình 1.4. Mô hình một nơron nhân tạo

Một nơron bao gồm các liên kết nhận tín hiệu vào bằng số có các trọng số kết nối w_i tương ứng với tín hiệu x_i, một hàm tổng S và một hàm chuyển (hay hàm kích hoạt) φ, để tạo tín hiệu ra dựa trên giá trị hàm tổng và giá trị ngưỡng θ.

Liên kết: Mỗi liên kết thứ i sẽ nhận giá trị vào x_i tương ứng và có trọng số kết nối w_i tương ứng.

Trọng số kết nối: Các trọng số kết nối w_i của mỗi đường liên kết là yếu tố then chốt của nơron, chúng sẽ được xác định tuỳ theo tập dữ liệu nhờ quá trình huấn luyện.

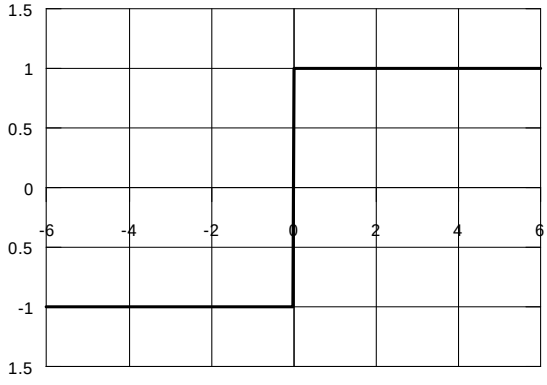
Hàm tổng: Hàm tổng S tích hợp các trọng số kết nối với các tín hiệu vào trên các liên kết tương ứng, trường hợp đơn giản nó có dạng: $S=\sum_{i=1}^{k} w_i x_i$ hoặc $S=\|x-w\|^2$ như là thông tin tổng hợp từ tín hiệu vào và trọng số kết nối.

Hàm chuyển / hàm kích hoạt.

Hàm chuyển φ lấy giá trị s-θ và cho giá trị ra φ(s- θ). Giá trị đầu ra y của nơron có thể biểu diễn theo công thức toán học như sau: $y=\varphi(\sum_{i=1}^{k} w_i x_i - \theta)$. Hàm chuyển φ là một hàm cụ thể nào đó được chọn tuỳ theo bài toán thực tế và thường có các dạng sau [28, 30]:

1) Hàm ngưỡng

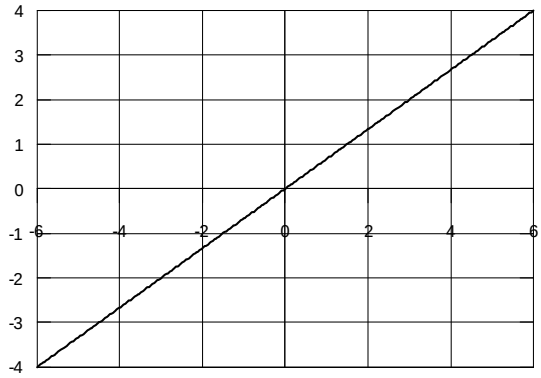
$$\varphi(x)=\begin{cases} -1 & \forall x<0 \\ 1 & \forall x\geq 0 \end{cases}$$



Hình 1.5: Đồ thị hàm ngưỡng

2) Hàm tuyến tính

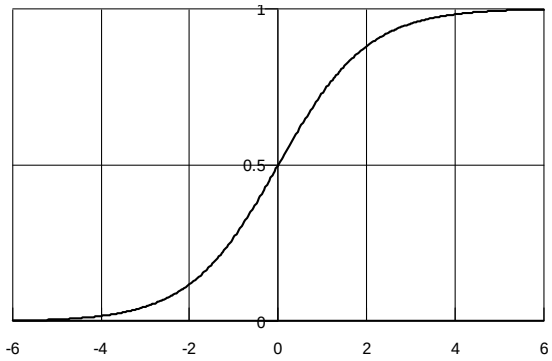
$$\varphi(x) = ax$$



Hình 1.6: Đồ thị hàm tuyến tính

3) Hàm sigmoid

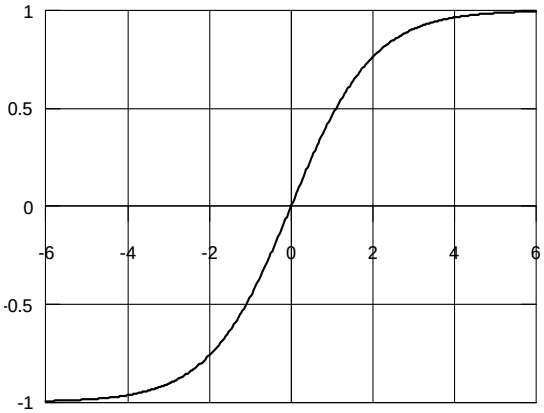
$$\varphi(x) = \frac{1}{1+e^{-x}}$$



Hình 1.7: Đồ thị hàm sigmoid

4) Hàm tanh

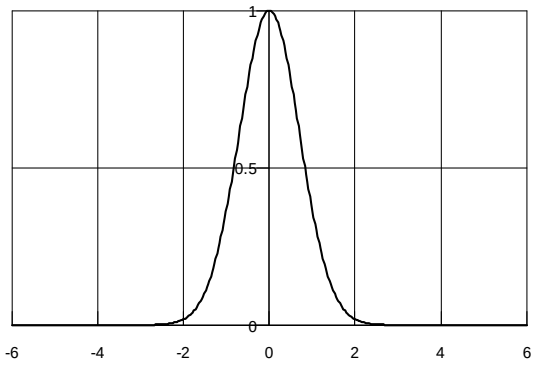
$$\varphi(x) = \frac{1-e^{-x}}{1+e^{-x}}$$



Hình 1.8: Đồ thị hàm tanh

5) Hàm bán kính (Gauss)

$$\varphi(x) = e^{-x^2}$$



Hình 1.9: Đồ thị hàm Gauss

c) Kiến trúc và thiết kế mạng nơron

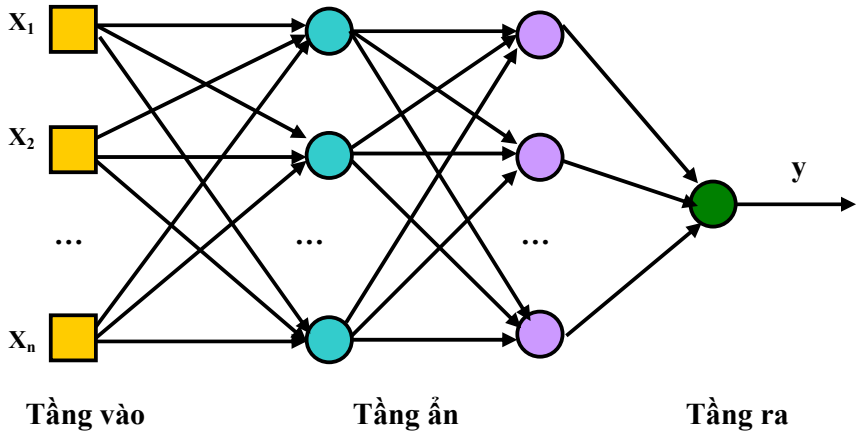
Mặc dù hiểu biết của con người về kiến trúc và hoạt động của não còn chưa đầy đủ, nhưng người ta tạo ra được các máy có một số tính năng tương tự như bộ não. Mạng nơron nhân tạo là máy mô phỏng cách bộ não hoạt động thực hiện các nhiệm vụ của nó. Một mạng nơron là bộ xử lý song song phân tán lớn nó giống bộ não người về 2 mặt:

- Tri thức được nắm bắt bởi nơron thông qua quá trình học.
- Độ lớn của trọng số kết nối nơron đóng vai trò khớp nối cất giữ tri thức học.

Phần trên đã trình bày về kiến trúc của một nơron. Các nơron này sẽ kết nối với nhau theo một quy tắc nào đó để tạo thành một mạng nơron. Tính năng của hệ thống này tuỳ thuộc vào cấu trúc của hệ, các trọng số liên kết nơron và quá trình tính toán tại các nơron đơn lẻ. Mạng nơron có thể học từ dữ liệu mẫu và tổng quát hoá dựa trên các dữ liệu mẫu học.

Có rất nhiều mạng nơron trong đó mạng nơron nhiều tầng truyền tới là dạng thông dụng nhất. Chúng bao gồm tầng vào là các nút nhận tín hiệu bằng số, tầng ẩn (gồm một hoặc nhiều tầng nơron) và tầng ra là một tầng nơron. Khi gọi có thể gọi theo số tầng nơron và thêm tầng vào hoặc tính theo số tầng nơron. Hình 1.10 mô tả

một mạng nơron 4 tầng truyền tới (gồm tầng vào, hai tầng nơron ẩn và tầng ra) hoặc là mạng 3 tầng nơron.



Hình 1.10: Mô hình một mạng nơron 4 tầng truyền tới

Mỗi một mạng nơron khác nhau sẽ có số lượng nơron khác nhau cũng như sự kết nối giữa chúng là không giống nhau.

Các trọng số kết nối

Dùng cho liên kết các tầng nơron khác nhau và có vai trò quan trọng trong hoạt động của một mạng nơron, nó cũng là sự mô tả đặc tính riêng của mỗi mạng.

Quá trình học/huấn luyện của mạng nơron

Trong số nhiều đặc tính của mạng nơron đặc tính quan trọng nhất là tự cải tiến làm việc của nó thông qua việc học. Việc cải tiến này theo thời gian phù hợp với một độ đo được quy định trước. Mạng nơron học nhờ điều chỉnh trọng số kết nối w_i và ngưỡng θ. Một cách lý tưởng mạng có tri thức tốt hơn sau mỗi lần lặp của quá trình học.

Có nhiều định nghĩa về sự học, ở đây ta theo định nghĩa của Mendel và Mc Cloren (1970) về học trong mạng nơron: "Học là quá trình qua đó các tham số tự do của một mạng nơron được sửa đổi thích ứng qua một quá trình tích luỹ kinh nghiệm liên tục trong một môi trường mà mạng bị tác động".

Xây dựng mạng nơron

Khi xây dựng mạng nơron ta thường theo các bước sau:

Xây dựng kiến trúc mạng: Xem xét có bao nhiêu tầng mà mạng chứa đựng, và chức năng của chúng là gì. Kiến trúc cũng cho biết có bao nhiêu kết nối được tạo ra giữa các nơron trong mạng, và chức năng của những kết nối này để làm gì.

Huấn luyện mạng: Tại bước này các trọng số kết nối giữa các nơron sẽ liên tục thay đổi giá trị trong quá trình huấn luyện mạng, và chúng sẽ có giá trị cố định khi quá trình huấn luyện thành công.

Kiểm tra hoạt động của mạng: Đây là bước cuối cùng nhưng cũng rất quan trọng trong quá trình xây dựng một mạng nơron. Người ta sẽ đưa vào tập các dữ liệu thử và chờ đợi kết quả ở đầu ra. Mạng nơron được xác định là tốt nếu như kết quả dữ liệu ở đầu ra đúng như những gì mà người thiết kế mong đợi.

d) Phân loại mạng nơron

Các mạng nơron có kiến trúc khác nhau và cách huấn luyện khác nhau cho các tính năng khác nhau. Ngoài những loại mạng với tính năng điển hình được trình bày trong nhiều cuốn sách (xem [28,30,31,38]), các kiểu mạng được kết nối với kiến trúc phong phú trong các ứng dụng cụ thể. Để có thuật ngữ chung, người ta thường phân loại mạng nơron dựa trên kiểu liên kết truyền tín hiệu hoặc số tầng.

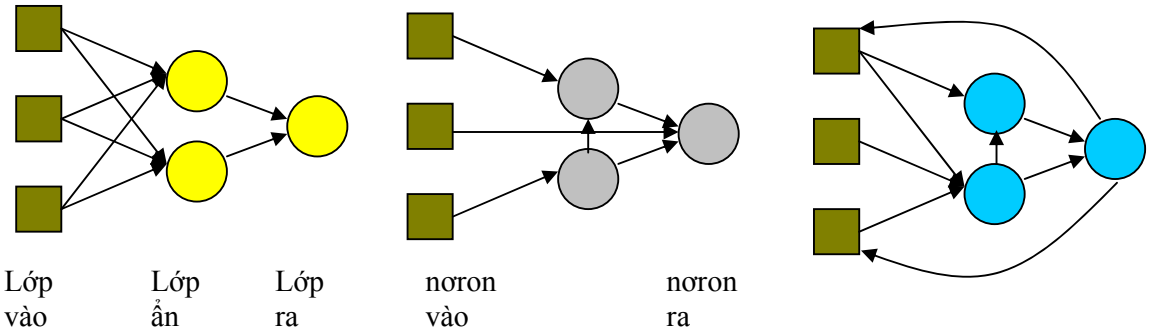
Theo kiểu liên kết nơron

Người ta phân biệt tín hiệu được truyền tới hay có hồi quy. Trong mạng nơron truyền tới (feel-forward neural network) các liên kết nơron truyền tín hiệu theo một hướng nhất định không tạo thành đồ thị có chu trình (Directed Acyclic Graph) với các đỉnh là các nơron, các cung là các liên kết giữa chúng.

Trong mạng nơron hồi quy (recurent neural network) có các liên kết nơron tạo thành chu trình. Các thông tin ra của các nơron được truyền lại cho nó hoặc các nơron đã góp phần kích hoạt chúng.

Theo số lớp/tầng

Các nơron có thể tổ chức lại thành các lớp sao cho mỗi nơron của lớp này chỉ được nối với các nơron ở lớp tiếp theo, không cho phép các liên kết giữa các nơron trong cùng một lớp, hoặc từ nơron lớp sau lên nơron lớp trước. Ở đây cũng không cho phép các liên kết nơron nhảy qua một lớp.



a) Mạng nơron nhiều lớp b)Mạng nơron truyền tới c) Mạng nơron hồi qui

Hình 1.11 Mô hình các loại mạng nơron

Dễ dàng nhận thấy rằng các nơron trong cùng một lớp nhận được tín hiệu từ lớp có thể xử lý song song.

e) Luật học của mạng nơron

Mạng nơron như một hệ thống thích nghi có khả năng học/huấn luyện để tinh chỉnh các trọng số liên kết cũng như cấu trúc của mình sao cho phù hợp với các mẫu học. Việc xác định cấu trúc mạng hiện nay vẫn theo kinh nghiệm còn các trọng số liên kết được xác định nhờ các luật học.

Luật học là một thủ tục sửa đổi trọng số và ngưỡng của mạng để cho mạng có thể đảm nhận được một nhiệm vụ nào đó. Nó có thể xem là thuật toán huấn luyện mạng. Luật học của mạng nơron có thể phân làm 3 loại [28]: học có giám sát (supervised learning), không giám sát (unsupervised learning) và học tăng cường (Reinforcement learning).

Học có giám sát

Trong học có giám sát, người ta dựa vào một tập mẫu huấn luyện: $\{p_1,t_1\}$, $\{p_2,t_2\}$, ..., $\{p_q,t_q\}$**,** trong đó p_q là véctơ tín hiệu vào t_q là đầu ra đích tương ứng để huấn luyện mạng. Các luật học có giám sát sẽ so sánh tín hiệu ra của mạng với tín hiệu đích của tín hiệu vào tương ứng để hiệu chỉnh trọng số kết nối và các giá trị ngưỡng sao cho tín hiệu ra ngày càng gần với tín hiệu đích (chương 7-12 của [28]).

Học không giám sát

Trong học không giám sát các trọng số và ngưỡng được sửa đổi dựa vào tín hiệu vào (lấy trong tập dữ liệu huấn luyện) mà không biết giá trị đích. Hầu hết các thuật toán loại này là thuật toán phân cụm. Trong đó ta phân tập dữ liệu *D* thành *k* cụm con sao cho mỗi phần tử trong cùng một cụm thì giống nhau hơn là các phần tử khác cụm. Cách phân cụm có thể là rõ hoặc mờ, số cụm có thể biết trước hoặc không, chi tiết xem chương 13-16 của [28].

Học tăng cường

Học tăng cường gần giống với học có giám sát ở chỗ có thể đánh giá chất lượng mạng (cho điểm) khi biết tín hiệu vào và ra của nó. Do vậy khi đưa các tín hiệu vào và quan sát tín hiệu ra ta có thể điều chỉnh để chất lượng mạng tốt dần lên. Tuy vậy ta không biết trước giá trị đích của mỗi mẫu tương ứng. Kiểu học này hiện nay ít thông dụng như học có giám sát. Nó thường phù hợp với những ứng dụng điều khiển hệ thống [28].

1.2.3. Mạng perceptron nhiều tầng MLP (Multi-Layer Perceptrons)

Phần này giới thiệu mạng nơron Perceptrons nhiều tầng, loại mạng thông dụng nhất trong các mạng nhiều tầng truyền tới (Feed-forward Neural network). Nó được sử dụng nhiều trong thực tế hiện nay.

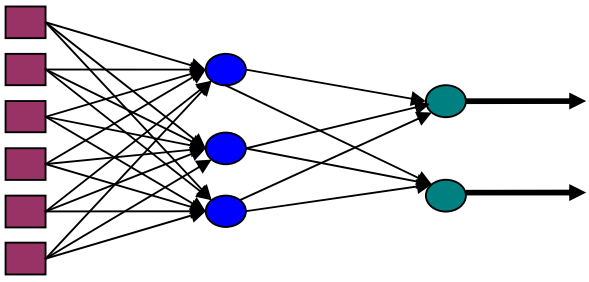
a) Kiến trúc mạng

Mạng MLP là mạng truyền tới nhiều tầng bao gồm các nút nhận tín hiệu vào bằng số, một hoặc nhiều tầng nơron ẩn và tầng nơron ra, trong đó tín hiệu tầng trước được đưa tới tầng sau. Hàm tổng trong các nơron đều có dạng: $s=\sum_{i=1}^{k} w_i x_i$, các hàm chuyển có thể có dạng khác nhau. Hình 1.12 mô tả mạng nơron 2 tầng nơron với 6 nút vào, 3 nơron tầng ẩn và 2 nơron tầng ra (có thể gọi là mạng 3 tầng).

Tầng vào: Nếu hàm đang xét có *n* biến thì có *n+1* nút trong đó nút đầu ứng với giá trị x_0 = -1và trọng số là ngưỡng θ, mỗi nút còn lại ứng với một biến của đối.

Tầng ra: Mỗi một nơron ở tầng ra sẽ ứng với một hàm. Nếu hàm cần xét xấp xỉ có giá trị là véc tơ *M* chiều thì có *M* nơron ở tầng ra.

Tầng ẩn: Số tầng ẩn và lượng nơron của mỗi tầng tuỳ thuộc vào mục đích thiết kế.



Hình 1.12 Kiến trúc mạng nơron truyền tới nhiều tầng

Mạng này có thể dùng để nhận dạng mẫu và tổng quát hơn là xấp xỉ hàm nhiều biến.

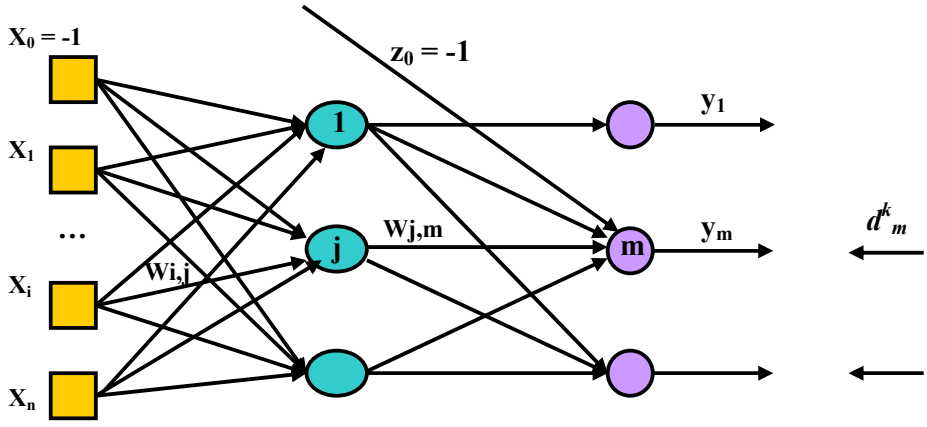
b) Thuật toán huấn luyện lan truyền ngược (Back-Propagation)

Huấn luyện mạng nơron là xác định các trọng số kết nối cho các liên kết của mỗi nơron. Quá trình xác định các trọng số này gọi là quá trình huấn luyện. Mục này trình bày phương pháp truyền ngược sai số để xác định các trọng số kết nối nhờ thuật toán Gradient cực tiểu hoá sai số trung bình phương (Least-Mean Square).

Ta xét một mạng 2 tầng như hình 1.13. Mạng này có các tín hiệu vào vô hướng $\{x_0,x_1...,x_n\}$ trong đó $x_0 = -1$. Tầng ẩn có j nơron với tín hiệu ra của chúng là $\{z_1...,z_n\}$ và tín hiệu $z_0 = -1$ chuyển đến các nơron tầng ra. Tầng ra có M nơron với đầu ra tương ứng là $\{y_1,...,y_M\}$. Mạng này dùng để xấp xỉ hàm có giá trị véc tơ M chiều $y = (y_1,...,y_M)$ của n biến.

Giả sử ta có các giá trị của y tại các điểm x thuộc tập X với $y(x^k)=d^k \ \forall x^k \in X$. Ta chia ngẫu nhiên tập X thành 2 tập: tập huấn luyện X_h và tập kiểm tra X_t. Thường tập huấn luyện có số lượng gấp đôi tập kiểm tra. Quá trình huấn luyện có thể thực hiện theo mớ hoặc theo từng dữ liệu, để đơn giản ta xét thuật toán huấn luyện theo từng dữ liệu.

Ta xét nơron m của tầng ra có trọng số liên kết hiện thời với nơron j của tầng ẩn là $w^c_{j,m}$ và đầu ra y_m. Nơron j của tầng ẩn có trọng số kết nối hiện thời với nút vào i là $w^c_{i,j}$. Ký hiệu φ_0 là hàm chuyển của nơron tầng ra và φ_h là hàm chuyển của nơron tầng ẩn, ta có các quy tắc học cho tầng ra và tầng ẩn như sau.



Hình 1.13 Huấn luyện mạng lan truyền ngược

Quy tắc học của tầng ra

Cho dữ liệu (x^k, d^k), ký hiệu y^k là đầu ra của mạng tương ứng với x^k ta hiệu chỉnh các trọng số kết nối để cực tiểu hoá tổng bình phương sai số của mạng:

$$E(w)=\frac{i}{2}\sum_{i=1}^{M}(y_i^k - d_i^k)^2 \quad (1.17)$$

Ký hiệu $w_{j,m}^{new}$ là trọng số kết nối mới, ρ là số dương bé:

$$w_{j,m}^{new} = w_{j,m}^{c} + \Delta w_{j,m} \ , \ m=1,..,M \quad (1.18)$$

Với:

$$\Delta w_{j,m} = -\rho \frac{\partial E}{\partial w_{j,m}} = \rho(d_m^k - y_m^k)\varphi_0^{'}(s_m)z_j \quad (1.19)$$

Trong đó $s_m = \sum_{i=0}^{J} w_{i,m} z_i$ và z_j được tính bởi:

$$z_j = \varphi_h(\sum_{i=0}^{n} w_{i,j} x_j) = \varphi_h(s_j) \quad (1.20)$$

Quy tắc học của tầng ẩn

Ký hiệu $w_{i,j}^{new}$ là trọng số kết nối mới của nơron j với nút vào i ta có:

$$w_{i,j}^{new} = w_{i,j}^{c} + \Delta w_{i,j} \quad j=1,...,J \quad (1.21)$$

Trong đó:

$$\Delta w_{i,j} = -\rho \frac{\partial E}{\partial w_{j,m}} \quad (1.22)$$

Ta có:

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial z_j}\frac{\partial z_j}{\partial s_j}\frac{\partial s_j}{\partial w_{i,j}} \quad (1.23)$$

với:

$$\frac{\partial s_j}{\partial w_{i,j}} = x_i \text{ và } \frac{\partial z_j}{\partial s_j} = \varphi_h^{'}(s_j) \quad (1.24)$$

và: $$\frac{\partial E}{\partial z_j} = \frac{\partial}{\partial z_j}\left\{\frac{1}{2}\sum_{m=1}^{M}[d_m^k - \varphi_0(s_m)]^2\right\} = -\sum_{m=1}^{M}[d_m^k - \varphi_0(s_m)\frac{\partial \varphi_0(s_m)}{\partial z_j}$$

$$= -\sum_{m=1}^{M}(d_m^k - y_m)\varphi_0^{'}(s_m)w_{j,m} \tag{1.25}$$

Thay (1.24) và (1.25) vào (1.23) ta có:

$$\Delta w_{i,j} = \rho[\sum_{m=1}^{M}(d_m^k - y_m)\varphi_0^{'}(s_m)w_{j,m}]\varphi_h^{'}(s_j)x_j \tag{1.26}$$

Thủ tục học lan truyền ngược

Thủ tục học sẽ bắt đầu từ tầng ra đến tầng ẩn như sau :

- **Bước 1**: Tạo các trọng số kết nối w^c ban đầu cho mọi nơron và chọn giá trị ρ cho tốc độ học (có thể thay đổi theo mỗi phép lặp).
- **Bước 2**: Lấy một mẫu x^k từ tập huấn luyện (có thể lấy ngẫu nhiên) và chuyển qua mạng để được y^k.
- **Bước 3**: Kết hợp y^k với d^k để cập nhật trọng số tầng ra mới theo (1.18) đến (1.20).
- **Bước 4**: Cập nhật trọng số tầng ẩn theo (1.21) và (1.26)
- **Bước 5**: Kiểm tra điều kiện dừng, nếu chưa thoả mãn thì trở lại *bước 2*, nếu tốt thì sang *bước 6* để thử kiểm tra.
- **Bước 6**: Kiểm tra dữ liệu thử nếu đạt yêu cầu thì huấn luyện xong nếu chưa thì trở lại bước 1.

Quá trình huấn luyện mạng có thể rơi vào cực trị địa phương nên có thể dùng thuật toán di truyền để tìm trọng số ban đầu. Người ta cũng có thể huấn luyện theo mớ khi đó hàm E cho bởi: $E(w)=\sum_{k=1}^{T}\sum_{m=1}^{M}(y_m^k - d_m^k)^2$

Trong đó các mẫu huấn luyện đánh số từ 1 đến T và y^k là đầu ra tương ứng của mạng với đầu vào x^k.

c) Nhận xét

Hagan [28] đã đưa ra các nhận định sau.

Về kiến trúc mạng:

Với mạng MLP, nếu có đủ nơron trong tầng ẩn thì có thể dùng để xấp xỉ một hàm liên tục tùy ý, số tham số trong mạng không vượt quá số ràng buộc có được từ mẫu mới có thể tổng quát hoá được. Tuy nhiên việc xác định số nơron tối ưu vẫn là bài toán mở. Một mạng đã cho thích hợp với hàm xấp xỉ như thế nào cũng chưa biết đầy đủ. Đặc biệt, việc ước lượng sai số đến nay vẫn chủ yếu dựa vào thực nghiệm.

Về sự hội tụ:

Thuật toán lan truyền ngược huấn luyện mạng truyền tới có tốc độ hội tụ chậm. Với mạng một tầng nơron thì đảm bảo hội tụ tối ưu toàn cục nhưng với mạng có nhiều tầng nơron thường chỉ hội tụ đến cực trị địa phương. Để khắc phục có thể huấn luyện nhiều lần với các giá trị khởi tạo ngẫu nhiên hoặc khởi tạo theo kinh nghiệm (Heuristic), hay kết hợp với thuật toán di truyền (Genetic Algorithm) sau đó dùng thuật toán tối ưu hoá. Thông thường người ta hay dùng biến thể của thuật toán gradient như thuật toán Newton và Gradient liên hợp để tăng tốc độ hội tụ cho thuật toán.

CHƯƠNG 2. MẠNG NƠRON RBF

Ở chương 1 đã trình bày về mạng perceptron nhiều tầng (MLP) dùng để nội suy và xấp xỉ hàm nhiều biến. Mạng này đang được sử dụng rộng rãi để xấp xỉ hàm số, nhưng nó không đảm bảo giải quyết được bài toán nội suy và khó chọn số nơron ẩn phù hợp. Nhược điểm cơ bản nữa của mạng MLP là thời gian huấn luyện lâu và thường chỉ tìm được gần đúng của cực trị địa phương. Mạng RBF (Radial Basis Function – RBF) là một lựa chọn để khắc phục nhược điểm này.

Mạng RBF là một mạng truyền tới 3 tầng (2 tầng nơron). Mỗi nơron tầng ẩn là một hàm phi tuyến của khoảng cách giữa véctơ vào $\overrightarrow{X}$ và véc tơ tâm $\overrightarrow{C^j}$ kết hợp với nơron *j* có bán kính tương ứng là σ_j. Cơ sở toán học của mạng RBF là kỹ thuật hàm cơ sở bán kính. Kỹ thuật này được Powell (1987) đề xuất để giải quyết bài toán nội suy nhiều biến [49]. Sau đó được Broomhead và Lowe (1988) giới thiệu như là mạng nơron [16]. Ưu điểm của mạng RBF là thời gian huấn luyện nhanh và luôn đảm bảo hội tụ đến cực trị toàn cục của sai số trung bình phương. Với các hàm cơ sở bán kính có tâm là các mốc nội suy thì có thể cho lời giải của bài toán nội suy. Vì vậy cùng với mạng MLP, mạng RBF tỏ ra là một phương pháp hiệu quả và được ứng dụng rộng rãi để nội suy và xấp xỉ hàm nhiều biến (xem [14,15,30]).

Chương này giới thiệu tóm tắt các kiến thức cơ bản về mạng nơron RBF cần cho nội dung chính của luận án bao gồm kiến trúc; đặc điểm huấn luyện và các thuật toán huấn luyện mạng thông dụng hiện nay, chi tiết hơn xem [30,38,50]. Mục 2.1 giới thiệu về kỹ thuật hàm cơ sở bán kính (RBF), mục 2.2 mô tả kiến trúc mạng RBF. Các thuật toán thông dụng huấn luyện mạng RBF được giới thiệu ở mục 2.3. Những so sánh về đặc tính của mạng RBF với mạng MLP được điểm ra ở mục 2.4.

2.1. Hàm cơ sở bán kính RBF và bài toán nội suy

2.1.1. Bài toán nội suy nhiều biến với cách tiếp cận RBF

Bài toán nội suy nhiều biến được giới thiệu chi tiết trong chương trước, Powell (1988) đề xuất dùng hàm nội suy dưới dạng tổ hợp tuyến tính của các hàm cơ sở bán kính [49]. Với cách tiếp cận này, bài toán nội suy được phát biểu lại như sau.

Xét hàm nhiều biến $f : D(\subset R^n) \to R^m$ cho bởi tập N mẫu $\{x^k, y^k\}_{k=1}^N$ $(x^k \in R^n; y^k \in R^m)$ sao cho: $f(x^k) = y^k; k = 1,..., N$.

Bài toán này tương đương với nội suy *m* hàm giá trị thực độc lập nên ta có thể giả thiết *m=1*. Powell đề xuất tìm hàm φ có dạng đã biết thỏa mãn:

$$\varphi(x) = \sum_{k=1}^{M} w_k h(\|x - v^k\|, \sigma_k) + w_0, \text{ sao cho } \varphi(x^k) = y^k; \forall k = 1,..., N \tag{2.1}$$

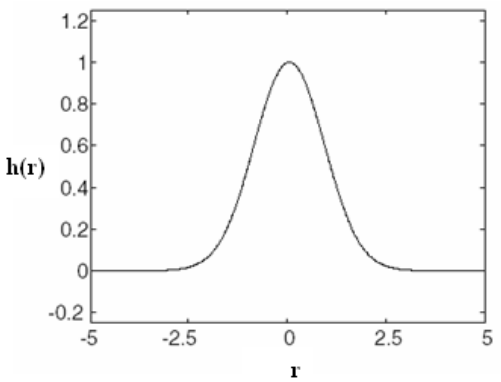
trong đó: $\{x^k\}_{k=1}^N$ là tập vectơ trong không gian *n*-chiều (các mốc nội suy) và $y^k = f(x^k)$ là giá trị đo được của hàm *f* cần nội suy; hàm thực $h(\|x - v^k\|, \sigma_k)$ được gọi là *hàm cơ sở bán kính* với tâm là v^k và $M(\leq N)$ là số hàm bán kính sử dụng để xấp xỉ hàm *f*; w_k và σ_k là các giá trị tham số cần tìm.

Một số hàm cơ sở bán kính thông dụng

Có nhiều dạng hàm cơ sở bán kính, thông dụng nhất là các dạng sau:

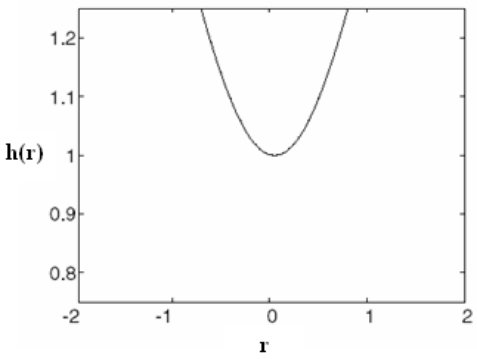
Hàm Gauss: $h(r, \sigma) = e^{\frac{-r^2}{2\sigma^2}}$

Hàm cơ sở bán kính Gauss có giá trị tăng dần đến giá trị 1 khi *r* tiến dần đến 0 và dần tới 0 khi *r* dần ra vô hạn, điều này được miêu tả cụ thể ở hình 2.1dưới đây.



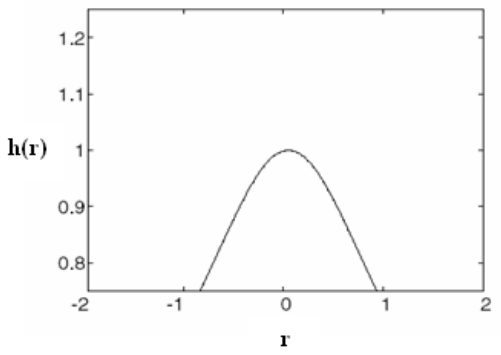
Hình 2.1. Hàm cơ sở bán kính Gauss với σ=1

Hàm đa bậc hai (Multiquadric): $h(r,\sigma) = (r^2 + \sigma^2)^{1/2}$



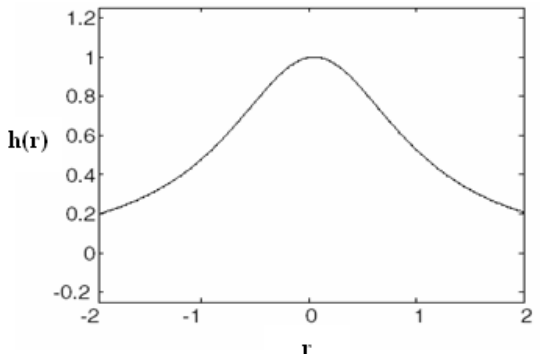
Hình 2.2. Hàm cơ sở bán kính Multiquadric với σ=1

Hàm đa bậc hai nghịch đảo(Inverse Multiquadric): $h(r,\sigma) = (r^2 + \sigma^2)^{-1/2}$



Hình 2.3. Hàm cơ sở bán kính Inverse Multiquadric với r =1 và c = 0

Hàm Côsi (Cauchy): $h(r,\sigma) = \frac{(r^2+\sigma^2)^{-1}}{\sigma}$



Hình 2.4. Hàm cơ sở bán kính Cauchy với r =1 và c = 0

Nếu số hàm bán kính bằng số mốc nội suy và các σ_k đã cho thì hệ phương trình 2.1 luôn tồn tại duy nhất nghiệm $\mathbf{w} = (w_0, w_1, \ldots, w_n)^T$, trong đó chỉ số trên ^{T} để chỉ vectơ hoặc ma trận chuyển vị.

2.1.2. Kỹ thuật hàm cơ sở bán kính

Như đã nói trong chương 1, không giảm tổng quát, ta xét bài toán nội suy với *m = 1*, và số mốc nội suy không quá nhiều ta tìm hàm φ dưới dạng sau:

$$\varphi(x) = \sum_{k=1}^{N} w_k \varphi_k(x) + w_0 \tag{2.2}$$

trong đó $\varphi_k(x)$ là hàm cơ sở bán kính. Như đã nói ở mục trước, có nhiều dạng hàm cơ sở bán kính, trong đó dạng hàm thông dụng nhất là hàm Gauss và về sau ta chỉ xét các hàm bán kính dạng này:

$$\varphi_k(x) = e^{-\left\|x-v^k\right\|^2/\sigma_k^2} \forall k = 1,\ldots N \tag{2.3}$$

Trong công thức (2.2) và (2.3) ta có:

- $\|.\|$ là kí hiệu chuẩn Euclide được tính $\|u\| = \sqrt{\sum_{i=1}^{N} u_i^{\,2}}$

- v^k gọi là tâm của hàm cơ sở bán kính φ_k. Đối với bài toán nội suy, ta lấy tâm chính là các mốc nội suy $v^k = x^k \quad \forall k$, khi đó $M = N$ (xem chương 3 của [38]). Với bài toán xấp xỉ thì $M<N$, việc xác định tâm tối ưu còn là bài toán đang được quan tâm.
- Các tham số w_k và σ_k cần tìm để φ cực tiểu tổng bình phương sai số hoặc thỏa mãn các điều kiện nội suy:

$$\varphi(x^i) = \sum_{k=1}^{N} w_k \varphi_k(x^i) + w_0 = y^i \ ; \ \forall i = 1,..,N \tag{2.4}$$

Với mỗi k, tham số σ_k được gọi là tham số *độ rộng* của hàm cơ sở bán kính vì nó dùng để điều khiển độ rộng miền ảnh hưởng của hàm cơ sở φ_k, khi $\left\|x - v^k\right\| > 3\sigma_k$ thì giá trị hàm $\varphi_k(x)$ là rất nhỏ, không có ý nghĩa vì nó gần triệt tiêu. Chính vì vậy ta nói hàm bán kính này chỉ có ảnh hưởng địa phương. Trường hợp số hàm bán kính bằng số mốc nội suy ($N=M$) ta lấy tâm của chúng trùng với mốc tương ứng. Xét ma trận vuông cấp N:

$$\Phi = (\varphi_{k,i})_{N\times N} \quad \text{trong đó} \quad \varphi_{k,i} = \varphi_k(x^i) = e^{-\left\|x^i - x^k\right\|^2 / \sigma_k^2} \tag{2.5}$$

và các tham số σ_k đã chọn. Micchelli [42] đã chứng minh rằng nếu các mốc x^k khác nhau thì Φ là ma trận khả nghịch. Vì vậy, với w_0 cho trước tùy ý hệ phương trình (2.4) luôn tồn tại duy nhất nghiệm $w_1, ..., w_N$. Trường hợp $M < N$, tổng bình phương sai số được xác định theo công thức:

$$E = \sum_{i=1}^{N} \left(\varphi\left(x^i\right) - y^i\right)^2 \tag{2.6}$$

E có duy nhất cực trị toàn cục.

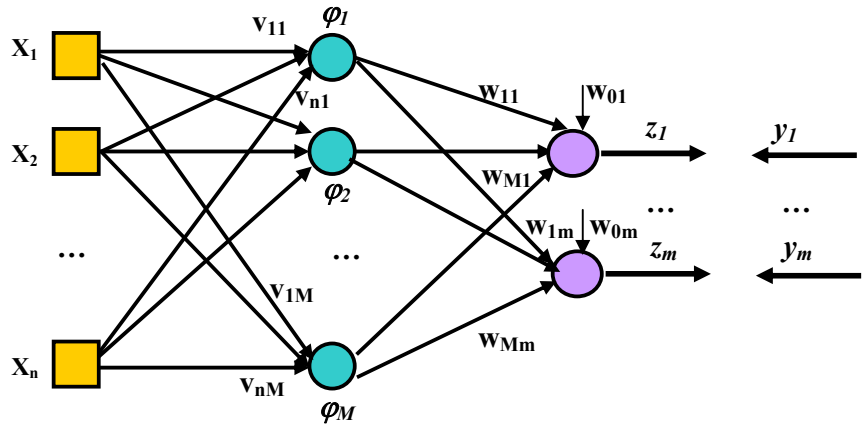
Tính xấp xỉ tổng quát và xấp xỉ tốt nhất nhờ các hàm cơ sở bán kính điển hình đã được khảo sát trong [26,47,48]. Hàm nội suy hoặc xấp xỉ tốt nhất có thể xác định gần đúng nhờ các phương pháp cực tiểu hoá dựa trên ưu điểm là tổng các bình phương sai số E của nó không có cực tiểu địa phương [13].

2.2. Kiến trúc mạng RBF

Mạng nơron RBF có kiến trúc để nội suy hàm $f : D(\subset R^n) \to R^m$ là một mạng 3 tầng truyền tới (có hai tầng nơron), cấu trúc như sau:

- Tầng vào gồm n nút cho véctơ tín hiệu vào $x \in R^n$.
- Tầng ẩn là tầng cơ sở bán kính, gồm M nơron trong đó nơron thứ k có tâm v^k và đầu ra của nó là hàm bán kính tương ứng φ_k.
- Tầng ra có hàm kích hoạt tuyến tính, gồm m nơron xác định m hàm nội suy (xấp xỉ) thành phần.

Kiến trúc của một mạng RBF tổng quát được mô tả trong hình 2.5.



Hình 2.5: Mô tả kiến trúc mạng nơron RBF

Giả sử với dữ liệu $\{x^k\}_{k=1}^N$ là tập véctơ trong không gian n chiều, $\{y^k\}_{k=1}^N$ là véctơ đích tương ứng với véctơ vào x^k. Ký hiệu w_{0j} là giá trị ngưỡng của nơron ra thứ j. Khi đó đầu ra của nơron j ở tầng ra là z_j được xác định theo công thức sau:

$$z_j = w_{1j}\varphi_1 + \ldots + w_{Mj}\varphi_M + w_{0j} \quad j=1,\ldots,m \tag{2.7}$$

Trong đó $w_{i,j}$ là các trọng số của nơron ra thứ j kết nối với nơron ẩn thứ i và φ_k là đầu ra của nơron ẩn thứ k, về sau ta sẽ dùng dạng Gauss:

$$\varphi_k = e^{\frac{-\|x-v^k\|^2}{2\sigma_k^2}}, \; k = 1, \ldots, M \tag{2.8}$$

Với các w_{0j} đã cho, các trọng số tầng ra có thể xác định nhờ tìm tổng bình phương sai số.

Về sau ta quy ước gọi các mạng RBF có số hàm bán kính bằng số mốc nội suy và có tâm trùng với mốc tương ứng là *mạng nội suy RBF* (vì trong trường hợp này hệ phương trình (2.4) luôn có nghiệm đúng).

2.3. Huấn luyện mạng RBF

Huấn luyện mạng nơron RBF là quá trình tìm các tham số $(w_{j,k}, v^k, \sigma_k)$ của các hàm bán kính và trọng số tầng ra, để đầu ra ứng với đầu vào là mốc nội suy (xấp xỉ) phù hợp với yêu cầu bài toán. Có rất nhiều phương pháp huấn luyện mạng, hầu hết các phương pháp này đều có đặc điểm chung là có xu hướng cực tiểu hóa giá trị bình phương sai số E bằng một biến thể của phương pháp Gradient hoặc giải hệ phương trình tuyến tính theo phương pháp giả nghịch đảo. So với các mạng nơron MLP mạng nơron RBF có điểm mạnh hơn hẳn là có thời gian huấn luyện ngắn [13,38,47]. Như đã nói ở trên, tổng bình phương sai số $\sum_{i=1}^{N}\left\|\varphi\left(x^i\right)-y^i\right\|^2$ chỉ có một cực trị duy nhất nên khi số mốc lớn thì phương pháp bình phương tối thiểu là thông dụng nhất. Có thể chia các kiểu huấn luyện mạng RBF ra thành ba loại: huấn luyện một pha, huấn luyện hai pha và huấn luyện đầy đủ [38,50].

2.3.1. Phương pháp huấn luyện một pha

Phương pháp huấn luyện một pha hay còn gọi là phương pháp huấn luyện nhanh. Phương pháp này khởi gán các tham số bán kính là một hằng số. Sau đó huấn luyện trọng số kết nối w của tầng ra bằng phương pháp giả nghịch đảo hoặc bình phương tối thiểu.

Với tập dữ liệu huấn luyện $\left\{x^k, y^k\right\}_{k=1}^{N}; \left(x^k \in R^n, y^k \in R^m\right)$ đã cho, ở phương pháp này, người ta thường chọn tâm v^k của các hàm bán kính là một tập con của tập

dữ liệu huấn luyện $\left\{x^k\right\}_{k=1}^{N}$, với bài toán nội suy thì $M=N$ và các tâm là mốc nội suy tương ứng.

Các bán kính σ_k được gán giá trị là một hằng số, trên cơ sở thực nghiệm Looney khuyên nên đặt $\sigma_k = \frac{1}{(2M)^{1/n}}$ (trong đó M là số hàm cơ sở bán kính, n là số nơron đầu vào) còn Haykin thì khuyên lấy $\sigma_k = \frac{D\max}{\sqrt{2N}}$ trong đó $Dmax$ là khoảng cách cực đại giữa các mốc nội suy. Các trọng số tầng ra thường được tìm ra bằng các phương pháp học có giám sát như là phương pháp giả nghịch đảo hoặc phương pháp bình phương tối thiểu theo phương pháp cực trị hóa Gradient. Về bản chất thì hai phương pháp này đều tìm các trọng số để giá trị bình phương sai số E đạt cực tiểu.

1) Phương pháp giả nghịch đảo

Với tập huấn luyện $\left\{x^k, y^k\right\}_{k=1}^{N}; \left(x^k \in R^n, y^k \in R^m\right)$, giả sử mạng RBF có M nơron ở tầng ẩn. Ta xét ma trận H_{NxM} như sau $H(i,k) = \varphi_k(x^i) = e^{\frac{-\left\|x^i - v_k\right\|^2}{\sigma_k}}$ và ma trận Y là ma trận hàng thứ k là ma trận y^k chuyển vị khi đó giá trị của các w_k được tính như sau :

$W = H^+ Y$ trong đó $H^+ = (H^T H)^{-1} H^T$.

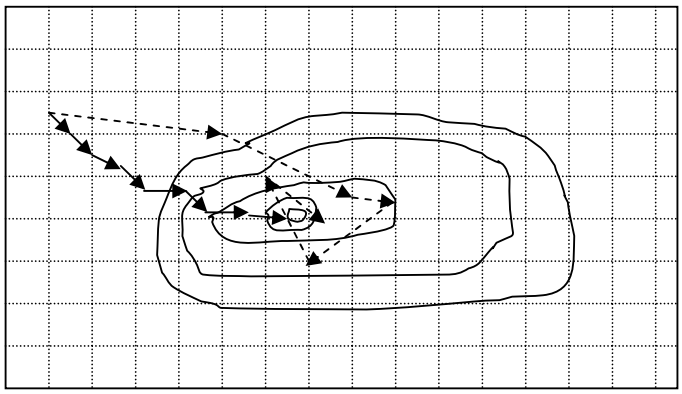
2) Phương pháp Gradient

Với phương pháp này ta tìm cực trị tổng bình phương sai số cho các nơron ra một cách độc lập nên xem $m=1$. Ban đầu các tham số w_k được tạo ra ngẫu nhiên sau đó các tham số này được cập nhật bằng công thức sau:

$$w_k(i+1) = w_k(i) + \eta \Delta w_k$$

trong đó $\Delta w_k = \sum_{i=1}^{N} \varphi_k\left(x^i\right)\left(y^i - \varphi\left(x^i\right)\right)$. Hệ số η được gọi là tốc độ học, nếu η nhỏ thì giá trị của các trọng số w tiến chậm đến điểm cực tiểu, nhưng nếu η lớn thì giá

trị của các trọng số *w* thường có xu hướng dao động quanh điểm cực tiểu, cách xử lý hợp lý có thể tham khảo [28]. Các dáng điệu hội tụ của nghiệm được minh họa ở hình 2.6. Thông thường người ta vẫn chọn η có giá trị nhỏ để đảm bảo quá trình lặp sẽ hội tụ về giá trị cực tiểu cho dù hơi mất thời gian.



Hình 2.6: Quá trình hội tụ đến giá trị cực tiểu của thuật toán Gradient, đường nét đứt ứng với giá trị η lớn, đường nét liền ứng với giá trị η nhỏ

3) Thuật toán huấn luyện nhanh (Quick Training)

Chúng tôi trình bày thuật toán Gradient cụ thể với huấn luyện nhanh (*Quick Training*) đã được Looney giới thiệu chi tiết trong chương 3 của [38].

Trong thuật toán này, khởi tạo ngẫu nhiên tập trọng số $\{w_{kj}^{(0)}\}$ ứng với mỗi nơron ở tầng ra và tinh chỉnh chúng theo thuật toán Grandient $-\nabla E = -(\frac{\partial E}{\partial w_{11}},...,\frac{\partial E}{\partial w_{MJ}})$. Véctơ tâm $\{v^k\}$ được xác định bằng với tập dữ liệu vào nên ta sẽ có số nơron ẩn là *M* chính bằng tập dữ liệu vào *N*. Tổng sai số bình phương được tính theo công thức:

$$E = \sum_{i=1}^{N}\sum_{j=1}^{m}(z_j^i - y_j^i)^2 \tag{2.9}$$

Thuật toán được mô tả trong hình 2.7 sau đây:

Bước 1: Cho tập dữ liệu vào $\{x^i\}$ gồm N véctơ. Khởi gán các tham số

Input N; M=N; //M=N Nơron ẩn
for i=1 to M do $v^i=x^i$; // xác định véctơ tâm
Input l; $h = l$; // l là số vòng lặp mong muốn
E= 99999.9; e=0.0001; //khởi gán tổng sai số bình phương E, sai số e cho điều kiện dừng.
η_l =1.0; // khởi gán tốc độ học

Bước 2: Khởi gán trọng số kết nối

For i =1 to M do
For j =1 to m do
w_{ij} = random(0,1) – 0.5; //khởi gán w trong đoạn [-0.5,0.5]

Bước 3: Tính toán tham số bán kính σ

$\sigma = \frac{1}{(2M)^{1/n}}$; // tính giá trị tham số σ
For k=1 to M do $\sigma_k = \sigma$, //đặt σ_k = σ cho mỗi nơron thứ k

Bước 4: Tính giá trị $\varphi_k = f_k(x^i, v^k)$ k=1…M cho mỗi véctơ vào x^i

For i=1 to N do // mỗi véctơ vào thứ i
For k = 1 to M do //mỗi nơron ẩn thứ k
If $i = k$ then φ_k^i =1 // khi $x^i = v^k$, φ_k^i = exp(0)=1
Else $\varphi_k^i = \exp(-\|x^i - v^k\|^2 / (2\sigma_k^2))$;

Bước 5: Cập nhật giá trị đầu ra z_j^i của nơron ra

For i=1 to N do // với mỗi véctơ đầu vào
For j= 1 to m // với mỗi nơron ra
$z_j^i = (1/M)\sum_{k=1}^{M} w_{kj}\varphi_k^i$ // cập nhật cho tầng ra

$E_{new} = \sum_{i=1}^{N}\sum_{j=1}^{m}(y_j^i - z_j^i)^2$ //tính tổng sai số

If $E_{new} < E$ then $\eta_l = \eta_l * 1.04$
Else $\eta_l := \eta_l * 0.92$;
$E = E_{new}$;

Bước 6: Chỉnh lại trọng số w

For k =1 to M do // với mỗi trọng số w_{kj}
For j= 1 to m

$$w_{kj} := w_{kj} + (2\eta_l/M)\sum_{i=1}^{N}(y_j^i - z_j^i)\varphi_k^i;$$

Bước 7: Dừng huấn luyện

If ($h \geq l$) or ($E<e$) then stop

Else $h:=h+1$; go to bước 5;

Hình 2.7 Thuật toán huấn luyện nhanh (Quick Training)

Thuật toán *Quick Training* chỉ điều chỉnh trọng số w của các nơron ở tầng ra, trong đó sử dụng một nơron ẩn cho mỗi véctơ dữ liệu đã huấn luyện x^i. Trong trường hợp nếu tập dữ liệu vào lớn (N lớn), chẳng hạn Looney khuyên là lớn hơn 200 thì nên dùng số nơron ẩn M nhỏ hơn số mốc ($M<N$). Điều này sẽ nói tới trong các thuật toán hai pha sau đây.

Khi sử dụng thuật toán *Quick Training* cho kết quả nhanh hơn nhiều lần so với thuật toán lan truyền ngược Back-propagation [13]. Chúng không có cực tiểu địa phương.

2.3.2. Phương pháp huấn luyện hai pha

Phương pháp này thường tách biệt hai pha, pha thứ nhất tìm các tham số bán kính. Pha thứ hai huấn luyện tìm các trọng số kết nối tầng ra w. Đây là phương pháp thông dụng được sử dụng rộng rãi hiện nay.

Với phương pháp huấn luyện hai pha khi số mẫu lớn, thông thường các giá trị tâm v^k và bán kính σ_k của hàm cơ sở bán kính φ_k được tính bằng các thuật toán phân cụm như thuật toán k-mean, k-mean có ngưỡng… Sau đó giá trị của các trọng số w_k được tính bằng các phương pháp giả nghịch đảo, hay Gradient như đã nêu ở trên.

Thuật toán phân cụm k-mean

- Phát biểu bài toán: Cho tập dữ liệu $X= \{x^1,...,x^N\} \subset R^n$ chúng ta cần phân tập dữ liệu này thành k tập $\{S_1, S_2, ..S_k\}: \left(k < N; S_j \bigcap_{i \neq j} S_i = \varnothing; \bigcup_{i=1}^{k} S_i = X \right)$ sao cho thỏa mãn: $\arg\min_{S} \sum_{i=1}^{k} \sum_{x_j \in S_i} \left\| x_j - \mu_i \right\|^2$ với μ_i là tâm của tập S_i.

Về mặt toán học thì bài toán phân cụm trên thuộc loại NP-khó tuy nhiên trên thực tế thì người ta thường giải bài toán bằng phương pháp heuristic như sau:

Đầu tiền ta khởi tạo ngẫu nhiên tập $\{\mu_1, \mu_2, ...\mu_k\}$ sau đó thực hiện vòng lặp qua hai bước sau:

+ Bước 1: tạo các cụm $S_i^{(t)} = \left\{x^j : \left\|x^j - \mu_i\right\| \le \left\|x^j - \mu_{i*}\right\|; \forall i^* = \overline{1,k}\right\}$

+ Bước 2: điều chỉnh lại tâm $\mu_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{x^j \in S_i^{(t)}} x^j$

Thuật toán sẽ dừng cho đến khi tập $\{\mu_1, \mu_2, ...\mu_k\}$ không còn có sự thay đổi giá trị (chi tiết hơn xem [24]).

Sau khi chạy thuật toán k-mean ta sẽ chọn tâm v^k của các hàm cơ sở bán kính φ_k chính là tập $\{\mu_1, \mu_2, ...\mu_k\}$ còn bán kính $\sigma_k = \frac{1}{|S_k|} \sum_{x_j \in S_k} \| x_j - \mu_k \|$.

2.3.3. Phương pháp huấn luyện đầy đủ

Phương pháp này thường dùng cho mạng xấp xỉ. Tất cả các tham số của mạng đều được hiệu chỉnh để cực tiểu hoá tổng sai số bình phương. Thường sử dụng phương pháp Gradient để tìm kiếm cả ba tham số (tâm *v*, bán kính σ, trọng số tầng ra *w*) dùng các công thức lặp dưới đây:

$$w_{k+1\,j+1} = w_{kj} - \eta\left(\frac{\partial E}{\partial w_{kj}}\right) = w_{kj} + (2\eta / M)\sum_{i=1}^{N}\left[\sum_{j=1}^{m}(y_j^i - z_j^i)\varphi_k^i\right] \tag{2.10}$$

$$v_{q+1}^{k+1} = v_q^k + \frac{2\eta_2}{(M\sigma^2)}\sum_{i=1}^{N}\left\{\sum_{j=1}^{m}(y_j^i - z_j^i)w_{kj}\right\}\varphi_{\mathrm{k}}^i(x_q^i - v_q^k) \tag{2.11}$$

$$\sigma_{k+1}{}^2 = \sigma_k^2 + (2\eta_3/M)\sum_{i=1}^{N}\{\sum_{j=1}^{m}(y_j^i - z_j^i)w_{kj}\sum_{\mathrm{k}=1}^{\mathrm{M}}[w_{k\,j}\varphi_{\mathrm{k}}^i \| (x^i - v^k \|^2 /(2\sigma_k^4)]\} \tag{2.12}$$

Trong đó $\{x^i, y^i\}_{i=1}^N$; $(x^i \in R^n, y^i \in R^m)$ là tập dữ liệu huấn luyện, *z* là giá trị đầu ra ở bước tương ứng. Nhìn chung phương pháp này cho kết quả xấp xỉ khá tốt, song

nhược điểm là thời gian huấn luyện lâu. Sau đây chúng tôi giới thiệu thuật toán huấn luyện đầy đủ (*Full Training)* được Looney giới thiệu trong [38].

Thuật toán huấn luyện Full Training

Thuật toán *Quick Training* đã cho thấy hiệu quả với dữ liệu huấn luyện ít và số nơron ẩn bằng số mốc nội suy, khi dữ liệu huấn luyện nhiều và phân bố không đều thì tính tổng quát của mạng kém. Thuật toán *Full Training* tốt hơn trong các trường hợp đó. Thuật toán này chọn số nơron ẩn M nhỏ hơn N, thông thường chọn ngẫu nhiên hoặc theo các thuật toán phân cụm. Theo đó các tham số bán kính, tâm cũng được điều chỉnh, các trọng số ở tầng ra vẫn được huấn luyện để cực tiểu hoá tổng sai số bình phương E. Thuật toán được mô tả chi tiết trong hình 2.8 và hình 2.9.

Bước 1: Cho tập dữ liệu vào $\{x^i\}$ gồm N véctơ, n chiều.
Khởi gán các tham số
Input N; Input M; //M nơron ẩn, M<N
for i=1 to M do // xác định véctơ tâm ngẫu nhiên từ 0 đến 1
for j=1 to n do
v_j^i=(random(0,1);
Input l; $h = l$; // l là số vòng lặp mong muốn
E= 99999.9; e=0.001; //khởi gán tổng sai số bình phương E, sai số e cho điều kiện dừng.
η_1 =1.6; η_2 =1.0; η_3 =1.0; // khởi gán tốc độ học

Bước 2: Khởi gán trọng số kết nối của mỗi nơron ở tầng ra
For i =1 to M do
For j =1 to m do
w_{ij} = random(0,1) – 0.5; // khởi gán w trong đoạn [-0.5,0.5]

Bước 3: Khởi gán tham số bán kính σ
σ = 0.005; // dùng σ cho σ_k

Bước 4: Tính giá trị $\varphi_k = f_k(x^i, v^k)$ k=1…M cho mỗi véctơ vào x^i
For i=1 to N do // mỗi véctơ vào thứ i
For k = 1 to M do //mỗi nơron ẩn thứ k
If $i = k$ then $\varphi_k^i = 1$ // khi $x^i = v^k$, φ_k^i = exp(0)=1
Else $\varphi_k^i = \exp(-\|x^i - v^k\|^2 / (2\sigma^2))$;

Bước 5: Gọi thủ tục **Update(k)** để điều chỉnh mạng và tốc độ học η_k
Update(0); // Cập nhật mạng, nhưng không điều chỉnh η_i

Bước 6: Chỉnh các tham số *w, v, σ*

For k =1 to M do // với mỗi trọng số w_{kj}

For j= 1 to m

$$w_{kj} := w_{kj} + (2\eta_1/M)\sum_{i=1}^{N}(y_j^i - z_j^i)\varphi_k^i ;$$

Update(1); // cập nhật tốc độ học η_1

For k =1 to M do

For q = 1 to n

$$v_q^k := v_q^k + \frac{2\eta_2}{(M\sigma^2)}\sum_{i=1}^{N}\left\{\sum_{j=1}^{m}(y_j^i - z_j^i)w_{kj}\right\}\varphi_k^i(x_q^i - v_q^k) ;$$

Update(2);

For k =1 to M do

$$\sigma_k^2 := \sigma_k^2 + (2\eta_3/M)\sum_{i=1}^{N}\{\sum_{j=1}^{m}(y_j^i - z_j^i)w_{kj}\sum_{k=1}^{M}[w_{kj}\varphi_k^i \| (x^i - v^k \|^2 /(2\sigma_k^4)]\} ;$$

Update(3);

Bước 7: Dừng huấn luyện

If ($h \geq l$) or ($E<e$) then stop

Else h:= $h+1$; go to bước 4;

Hình 2.8 Thuật toán huấn luyện đầy đủ Full Training

Procedure Update(k); Cập giá trị đầu ra, tính tổng sai số bình phương, hiệu chỉnh tốc độ học η_r.

$r = k$;

E_{new} =0;

For i =1 to N do // với mỗi véctơ đầu vào

For j= 1 to m // với mỗi Nơron ra

$z_j^i = (1/M)\sum_{k=1}^{M} w_{kj}\varphi_k^i$ // cập nhật cho tầng ra

$E_{new} := E_{new} + (y_j^i - z_j^i)^2$ //tính tổng sai số

If $E_{new} < E$ then $\eta_r = \eta_r * 1.04$

Else $\eta_r := \eta_r * 0.92$;

$E = E_{new}$;

Hình 2.9 Thủ tục Update(k) của thuật toán huấn luyện đầy đủ

2.3.4. Nhận xét chung các thuật toán huấn luyện

Trong phương pháp một pha có thể tìm w_k bằng phương pháp giải đúng hệ phương trình (2.4), nhưng khi số mốc lớn thì các phương pháp này cho nghiệm có sai số lớn và không ổn định (sai số dữ liệu nhỏ nhưng sai số của nghiệm lớn).

Các phương pháp huấn luyện mạng RBF một pha, hai pha hoặc huấn luyện đầy đủ đều đưa đến tìm cực tiểu tổng sai số bình phương E theo phương pháp Gradient hoặc biến thể của nó. Tuy nhiên các phương pháp tìm cực trị hàm nhiều biến vẫn khá chậm, khó điều khiển tốc độ hội tụ, ước lượng sai số và song song hoá. Mặt khác, các tham số độ rộng bán kính σ_k cũng ảnh hưởng lớn tới chất lượng mạng và thời gian huấn luyện. Nếu chúng được chọn bé thì các điểm xa tâm ít chịu ảnh hưởng của các hàm cơ sở còn ngược lại thì tốc độ hội tụ chậm.

2.4. So sánh mạng RBF với mạng MLP

Mạng RBF và mạng MLP cùng là mạng truyền tới đều dùng để xấp xỉ hàm. Sau đây là một số điểm cần lưu ý khi chọn kiểu mạng ứng dụng:

1. Mạng RBF là mạng chỉ có một tầng/lớp ẩn, còn mạng MLP có thể có một hoặc nhiều tầng nơron ẩn và khó xác định số nơron đủ tốt cho tầng này.
2. Khi dùng phương pháp Gradient cả hai đều dùng thuật toán truyền ngược sai số (BP), phức tạp hơn đối với mạng MLP có các tầng phi tuyến còn mạng RBF thì chỉ tập trung tính toán ở tầng ẩn còn tầng ra tuyến tính thì véctơ Gradient xác định dễ dàng.
3. Mạng MLP có thể dùng để ngoại suy hàm số. Mạng RBF với các hàm bán kính chỉ có ảnh hưởng địa phương (như hàm Gauss,…) nên không dùng để ngoại suy được.
4. Các thuật toán huấn luyện ảnh hưởng nhiều tới chất lượng của mạng. Thực tiễn cho thấy mạng RBF có thời gian huấn luyện nhanh hơn mạng MLP [13] và không sợ rơi vào cực trị địa phương.

5. Hiện nay sai số nội suy hay xấp xỉ hàm vẫn phải qua thực nghiệm chứ chưa có phương pháp ước lượng hữu hiệu nào.

2.5. Kết luận của chương

Như đã nói trong chương 1 bài toán nội suy hàm một biến đã được nghiên cứu nhiều và giải quyết tương đối hoàn thiện. Với bài toán nội suy nhiều biến thì còn nhiều vấn đề mở. Mặc dù có những hạn chế về kết quả lý thuyết, các phương pháp học dựa trên mẫu và mạng nơron đang là công cụ hữu hiệu để giải quyết bài toán nội suy nhiều biến trong thực tiễn. So với các mạng nơron, phương pháp *k*-lân cận gần nhất và hồi quy tuyến tính địa phương tuy dễ sử dụng nhưng có nhược điểm là không xác định trước hàm nội suy (xấp xỉ) qua dữ liệu huấn luyện được.

Mỗi hàm bán kính chỉ có miền ảnh hưởng địa phương phụ thuộc vào tham số độ rộng nên mạng RBF có thể xem là cầu nối giữa mạng MLP và các phương pháp dựa trên mẫu. Nó giống mạng MLP là xác định trước được hàm nội suy để dùng và giống phương pháp học dựa trên mẫu ở tính ảnh hưởng địa phương. So với mạng MLP, mạng RBF có thời gian huấn luyện nhanh hơn nhiều. Tuy vậy ngày nay có nhiều bài toán ứng dụng thời gian thực, đòi hỏi thời gian huấn luyện càng ngắn càng tốt. Đó là chủ để còn mở hiện nay và luận án tập trung tìm phương pháp huấn luận mạng nội suy, đưa ra các thuật toán dùng được cho số mốc lớn mà chưa xét đến mạng xấp xỉ.

CHƯƠNG 3. THUẬT TOÁN MỚI HUẤN LUYỆN MẠNG NỘI SUY RBF

Trong chương này, chúng tôi đề xuất một thuật toán lặp hai pha huấn luyện mạng nội suy RBF. Pha thứ nhất xác định tham số độ rộng cho các hàm cơ sở bán kính, còn các trọng số tầng ra được tìm nhờ phép lặp xác định điểm bất động của một ánh xạ co trong pha sau. Phân tích toán học và kết quả thực nghiệm cho thấy thuật toán có những ưu điểm vượt trội so với những thuật toán thông dụng: dễ ước lượng sai số huấn luyện, thời gian huấn luyện ngắn, tính tổng quát cũng tốt hơn và dễ song song hoá. Thuật toán này cũng là cơ sở cho các thuật toán ở các chương sau. Mục 3.1 sẽ dành để giới thiệu các phương pháp lặp giải hệ phương trình tuyến tính cần dùng về sau và tóm tắt lại mạng nội suy RBF với hàm bán kính dạng Gauss. Thuật toán lặp hai pha được giới thiệu ở mục 3.2, kết quả thử nghiệm được trình bày ở mục 3.3. Mục 3.4 dành để so sánh hiệu quả của thuật toán mới với thuật toán Gradient và các nhận xét chung được đưa ra trong mục cuối.

Kết quả chủ yếu của chương này đã được công bố trong hội thảo khoa học [4], và tạp chí quốc tế Signal Processing [34].

3.1. Nền tảng lý thuyết của thuật toán

Trước hết chúng tôi trình bày phương pháp lặp đơn và cải tiến của nó là phương pháp seidel [5], sau đó sẽ đưa ra một số nhận xét về hàm Gauss.

3.1.1. Các phương pháp lặp đơn giải hệ phương trình tuyến tính

Giả sử ta cần giải hệ phương trình (3.1)

$$Ax = b \tag{3.1}$$

Nếu đưa được về hệ tương đương dạng (3.2)

$$x = Bx + d \tag{3.2}$$

Trong đó, B là ma trận vuông cấp n thoả mãn (3.3)

$$\|B\| = q < 1 \tag{3.3}$$

Chuẩn của B xác định bởi công thức:

$$\|B\| = \max\{\|Bx\|, \|x\| = 1\} \tag{3.4}$$

Nếu chuẩn $\|x\| = \max\{|x_i|\}$ thì chuẩn của B là:

$$\| B \| = \max\left\{\sum_{j=1}^{n} |c_{i,j}|, i = 1,..,n\right\} \tag{3.5}$$

a) Phương pháp lặp đơn

Khi đó với $x^0 = \begin{pmatrix} x_1^0 \\ x_2^0 \\ \cdots \\ x_n^0 \end{pmatrix}$ tuỳ ý, và dãy nghiệm xấp xỉ được xây dựng bởi công thức lặp:

$$x^{k+1} = Bx^k + d \quad ; \left(x_i^{k+1} = \sum_{j=1}^{n} b_{ij} x_j^k + d_i \right) \tag{3.6}$$

thỏa mãn $\lim_{k\to\infty} x^k = x*$. Trong đó x* là nghiệm đúng duy nhất của (3.2) và ta có ước lượng sai số:

$$\left\|x_i^* - x_i^{(k)}\right\| \le \frac{q}{1-q}\left\|x^k - x^{k-1}\right\| \tag{3.7}$$

$$\left\|x_i^* - x_i^{(k)}\right\| \le \frac{q}{1-q}\max_{j\le n}\left\{\left|x_j^k - x_j^{k-1}\right|\right\}, \forall i \le n$$

Ngoài ra ta có ước lượng tiên nghiệm:

$$\left\|x_i^* - x_i^{(k)}\right\| \le \frac{q^k}{1-q}\max_{j\le n}\left\{\left|x_j^1 - x_j^0\right|\right\} \tag{3.8}$$

Thông thường ta có thể chọn $x^0 = d$, khi đó coi như ta đã tính xấp xỉ ban đầu với $x^0 = 0$ và $x^0 = d$ là bước tiếp theo.

b) phương pháp lặp Seidel

Phương pháp Seidel là cải tiến của phương pháp lặp đơn để có tốc độ hội tụ nhanh hơn. Trong phương pháp này thay vì công thức lặp (3.6) ta dùng công thức sau:

$$\begin{cases} x_1^{(k+1)} = \sum_{j=1}^{n} b_{1j} x_j^{(k)} + d_1 \\ x_i^{(k+1)} = \sum_{j=1}^{i-1} b_{ij} x_j^{(k+1)} + \sum_{j=i}^{n} b_{ij} x_j^{(k)} + d_i \quad \forall i > 1 \end{cases} \tag{3.9}$$

Để ước lượng sai số ta vẫn dùng công thức (3.7), nhưng ở phương pháp này thì vế phải của (3.7) hội tụ về giá trị không, nhanh hơn phương pháp lặp đơn.

3.1.2. Tóm lược về mạng nội suy RBF với hàm RBF dạng Gauss

Bài toán nội suy và kỹ thuật nội suy dùng hàm cơ sở bán kính đã được trình bày trong chương 1, 2. Để tiện theo dõi chúng tôi xin điểm lại một số nhận xét để làm cơ sở cho việc phát triển thuật toán.

Bài toán nội suy. Xét hàm nhiều biến $f : D(\subset R^n) \to R^m$ cho bởi tập mẫu $\{x^k, y^k\}_{k=1}^{N}$ $(x^k \in R^n; y^k \in R^m)$ sao cho $f(x^k) = y^k; k = 1,..., N$. Ta cần tìm hàm φ có dạng đã biết thỏa mãn:

$$\varphi(x^i) = y^i; \forall i = 1,..., N \tag{3.10}$$

Các điểm x^k là các mốc nội suy, φ là hàm nội suy và được dùng để xấp xỉ hàm f trên miền D.

Powell (1987) đề xuất tìm hàm nội suy φ nhờ các hàm cơ sở bán kính, với M là số các hàm cơ sở bán kính ($M \leq N$). Trong khuôn khổ luận án này chúng tôi chỉ xét trường hợp $M=N$ vì chúng tôi quan tâm tới bài toán nội suy hơn là bài toán xấp xỉ. Mà với bài toán nội suy có thể chọn số các hàm bán kính bằng chính số mốc nội suy, và các mốc này sẽ làm tâm cho hàm bán kính tương ứng. Trong đó dạng hàm bán kính thông dụng nhất là hàm Gauss và về sau sẽ dùng dạng hàm này.

Không giảm tổng quát, ta xét bài toán nội suy với $m = 1$, và tìm hàm φ dưới dạng sau:

$$\varphi(x) = \sum_{k=1}^{N} w_k \varphi_k(x) + w_0 \qquad (3.11)$$

$$\text{trong đó} \quad \varphi_k(x) = e^{-\left\|x - x^k\right\|^2 / \sigma_k^2} \forall k = 1,...N$$

x^k là tâm của hàm cơ sở bán kính φ_k. Với mỗi *k*, tham số σ_k dùng để điều khiển độ rộng miền ảnh hưởng của hàm cơ sở φ_k, nếu $\left\|x - v^k\right\| > 3\sigma_k$ thì $\varphi_k(x)$ gần triệt tiêu. Các tham số w_k và σ_k cần tìm để hàm φ dạng (3.11) và thỏa mãn các điều kiện nội suy của (3.10):

$$\varphi(x^i) = \sum_{k=1}^{N} w_k \varphi_k(x^i) + w_0 = y^i \ ; \ \forall i = 1,.., N \qquad (3.12)$$

Từ các công thức (3.11) và (3.12) ta có hệ thức:

$$\sum_{k=1}^{N} w_k e^{-\left\|x^i - x^k\right\|^2 / \sigma_k^2} = y^i - w_0 = z_i; \quad \forall i = 1,.., N \qquad (3.13)$$

Các tham số σ_k đã chọn ta xét ma trận cấp NxN:

$$\Phi = (\varphi_{k,i})_{N \times N} \quad \text{trong đó} \quad \varphi_{k,i} = \varphi_k(x^i) = e^{-\left\|x^i - x^k\right\|^2 / \sigma_k^2} \qquad (3.14)$$

Micchelli [42] đã chứng minh rằng nếu các mốc x^k khác nhau thì Φ là ma trận khả nghịch. Vì vậy, với w_0 cho trước tùy ý hệ phương trình (3.13) luôn tồn tại duy nhất nghiệm $w_1, ..., w_N$.

Để giải quyết bài toán nội suy này, chúng tôi dùng mạng nơron RBF và gọi là *mạng nội suy RBF* cho gọn. Như vậy mạng nội suy RBF dùng để nội suy hàm thực *n* biến $f : D(\subset R^n) \to R^m$ là một mạng 3 tầng truyền tới. Tầng vào gồm *n* nút cho vectơ tín hiệu vào $x \in R^n$, tầng ẩn gồm *N* nơron trong đó nơron thứ *k* có tâm là mốc nội suy x^k và đầu ra của nó là hàm bán kính $\varphi_k(x)$ tương ứng, tầng ra gồm *m* nơron xác định giá trị hàm nội suy của *f*. Giá trị của tầng ra được xác định theo công thức (3.12).

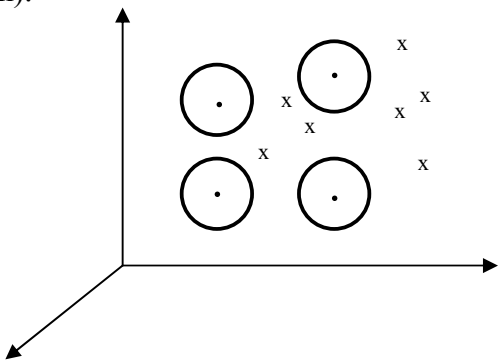
Với tầng ẩn thì thường dùng hàm tổng là hàm Gauss, còn hàm kích hoạt là hàm đồng nhất (dạng đặc biệt của tuyến tính) và cũng có thể xem hàm tổng là bình phương khoảng cách của đầu vào đến tâm và hàm kích hoạt là $f(u) = e^{-\frac{u^2}{2\sigma^2}}$

Tầng ra thì dùng hàm tổng là hàm Σ, hàm kích hoạt là hàm tuyến tính.

Các thuật toán huấn luyện thường xác định trọng số tầng ra *w* bằng phương pháp cực tiểu hoá tổng sai số bình phương theo công thức (3.15) sau:

$$E = \sum_{i=1}^{N} \left(\varphi\left(x^i\right) - y^i \right)^2 \tag{3.15}$$

Tham số độ rộng của hàm bán kính (còn gọi gọn hơn là bán kính) σ có ảnh hưởng rất nhiều tới mạng. Nếu chọn bán kính nhỏ thì có nhiều điểm cách xa tâm của các nơron tầng ẩn (như trên hình 3.1 là dấu x) sẽ ít bị ảnh hưởng bởi các hàm bán kính nên dẫn đến sai số nội suy ở các điểm này lớn, nhưng nếu tăng bán kính σ lên thì tốc độ hội tụ sẽ chậm (thực nghiệm về sau cho thấy cũng không đảm bảo có sai số nội suy tốt hơn).



Hình 3.1 Mô hình minh hoạ miền ảnh hưởng của tham số bán kính σ

Một vấn đề đặt ra là phải chọn được bán kính σ như thế nào để vừa đảm bảo hội tụ nhanh và sai số có thể chấp nhận được. Dưới đây chúng tôi đề xuất thuật toán huấn luyện mạng RBF theo phương pháp lặp để khắc phục những hạn chế trên.

3.2. Thuật toán lặp hai pha huấn luyện mạng nội suy RBF

Ý tưởng chính của thuật toán mới là ở pha đầu tìm các bán kính σ_k được xác định nhờ cân bằng giữa tính tổng quát của mạng và tốc độ hội tụ của pha sau. Ở pha thứ hai, các tham số w_k được xác định nhờ tìm điểm bất động của một ánh xạ co.

Định lý dưới đây là nền tảng cho thuật toán.

3.2.1. Định lý cơ bản

Trở lại xét hệ phương trình (3.13) để tìm các trọng số tầng ra:

$$\sum_{k=1}^{N} w_k e^{-\left\|x^i - x^k\right\|^2 / \sigma_k^2} = y^i - w_0 = \mathrm{z_i}; \quad \forall i = 1,..,N$$

Ký hiệu: I là ma trận đơn vị cấp N; $W = \begin{bmatrix} w_1 \\ ... \\ w_N \end{bmatrix}, Z = \begin{bmatrix} z_1 \\ ... \\ z_N \end{bmatrix}$ là các vectơ trong không gian R^N và Φ là ma trận cho bởi (3.14)

Nếu đặt: $$\Psi = I - \Phi = \left|\Psi_{k,j}\right|_{N\times N} \tag{3.16}$$

Thì ta có

$$\Psi_{k,j} = \begin{cases} 0 & khi: k = j \\ -e^{-\left\|x^j - x^k\right\|^2 / \sigma_k^2} & khi: k \neq j \end{cases} \tag{3.17}$$

Khi đó hệ phương trình (3.12) tương đương với hệ:

$$W = \Psi W + Z \tag{3.18}$$

Với các tham số σ_k đã chọn, w_0 tùy ý thì hệ (3.12), (3.18) luôn có duy nhất nghiệm W.

Để tìm σ_k, trước tiên ta lấy w_0 là trung bình cộng của các giá trị y^k:

$$w_0 = \frac{1}{N}\sum_{k=1}^{N} y^k \tag{3.19}$$

Bây giờ với mỗi $k \le N$, ta có hàm q_k của σ_k xác định theo công thức sau:

$$q_k = \sum_{j=1}^{N} \left|\Psi_{k,j}\right| \tag{3.20}$$

Việc xác định các bán kính σ_k này dựa vào định lý sau.

Định lý 3.1: Hàm q_k là đơn điệu tăng, hơn nữa với mọi số dương $q<1$, luôn tồn tại giá trị σ_k sao cho $q_k(\sigma_k)$ bằng q.

Chứng minh. Từ biểu thức (3.20) và chú ý tới (3.17) ta dễ dàng thấy q_k là hàm đơn điệu tăng của σ_k, hơn nữa ta có:

$$\lim_{\sigma_k \to \infty} q_k = N - 1 \quad \text{và} \quad \lim_{\sigma_k \to 0} q_k = 0 \tag{3.21}$$

Bởi vì q_k là hàm liên tục nên với mọi q thuộc khoảng (0,1) luôn tồn tại giá trị σ_k sao cho $q_k(\sigma_k) = q$. Vậy định lý đã được chứng minh.

Định lý này cho thấy với $q < 1$, chúng ta có thể tìm được tập các $\{\sigma_k\}_{k=1}^{N}$ để nghiệm đúng W^* của (3.18) là điểm bất động của ánh xạ co ứng với $\Psi W + Z$ và hệ số co là q.

3.2.2. Mô tả thuật toán

Với sai số ε và các hằng số dương q, α ($0<q<1$, $0<\alpha<1$) cho trước, thuật toán sẽ gồm 2 pha để xác định các tham số σ_k và W^*. Trong pha thứ nhất, các σ_k được xác định để $q_k \le q$, sao cho q_k gần tới q nhất, nghĩa là nếu thay thế σ_k bằng $\frac{\sigma_k}{\alpha}$ thì $q_k > q$. Khi đó chuẩn của ma trận ψ tương ứng với chuẩn vectơ $\|.\|_*$ cho bởi (3.22) là $\|\psi\|_* = \max_{\|u\|\le 1}\{\|\psi_u\|_*\}$ sẽ không lớn hơn q. Như vậy cho phép ta tìm nghiệm gần đúng W^* của (3.18) bằng phương pháp lặp đơn giản trong pha thứ hai. Thuật toán được đặc tả trong hình 3.2.

Procedure Huấn luyện mạng RBF (HDH)

Begin

Pha 1: Xác định tham số bán kính

For k=1 to N do

Xác định các σ_k để $q_k < q$;

sao cho nếu thay thế $\sigma_k := \frac{\sigma_k}{\alpha}$ thì $q_k > q$

End

Pha 2: Xác định các trọng số tầng ra

Xác định W^* bằng phương pháp lặp ;

End

Hình 3.2 Đặc tả thuật toán lặp hai pha huấn luyện mạng RBF

1) Pha 1: Xác định tham số độ rộng σ_k

Trong pha thứ nhất, xác định các σ_k để $q_k \leq q$, sao cho q_k gần tới q nhất nghĩa là nếu thay thế σ_k bằng $\frac{\sigma_k}{\alpha}$ thì $q_k > q$.

Với một số dương $\alpha < 1$ và giá trị khởi tạo σ_0. Có thể khởi gán σ_0 bằng $\sigma_k = \frac{1}{(2N)^{1/n}}$ theo gợi ý của Looney [38]. Chi tiết của thủ tục lặp được thể hiện trong hình 3.3.

Procedure Pha 1 của thuật toán HDH;

Begin

Khởi tạo $k=0$;

Bước 0: k$\leftarrow$ k+1;

if $k \leq N$ then

Bước 1: Khởi tạo $\sigma \leftarrow \sigma_0$; //khởi tạo σ_k

Tính $p = \sum_{j=1; j\neq k}^{N} e^{\frac{-\|x^j - x^k\|^2}{(\sigma)^2}}$; //tính q_k

Bước 2: Nếu $p=q$ thì $\sigma_k \leftarrow \sigma$; Quay trở lại Bước 0

Bước 3: Nếu $p>q$ thì điều chỉnh tham số bán kính để $q_k \leq q$ và gần với q nhất.

3.1. $\sigma^1 \leftarrow \alpha\sigma$; //giảm tham số bán kính

Tính $p = \sum_{j=1; j\neq k}^{N} e^{\frac{-\|x^j - x^k\|^2}{(\sigma^1)^2}}$; //tính lại q_k

3.2. Nếu $p \leq q$ thì $\sigma_k \leftarrow \sigma^1$; Quay trở về Bước 0

3.3. Nếu $p>q$ thì $\sigma \leftarrow \sigma^1$ và trở lại 3.1;

Bước 4: Nếu $p<q$ thì

4.1.$\sigma^1 \leftarrow \sigma/\alpha$; //tăng tham số bán kính;

Tính $p = \sum_{j=1; j\neq k}^{N} e^{\frac{-\|x^j - x^k\|^2}{(\sigma^1)^2}}$; //Tính lại q_k;

4.2. Nếu $p>q$ thì $\sigma_k \leftarrow \sigma$; Quay trở về Bước 0;

4.3. Nếu $p<q$ thì $\sigma \leftarrow \sigma^1$ và trở lại 4.1

End

Hình 3.3. Pha 1: xác định tham số bán kính

2) Pha 2: Xác định trọng số tầng ra w_k

Để tìm nghiệm W^* của (3.18) ta thực hiện thủ tục lặp trong hình 3.4.

Procedure Pha 2 của thuật toán HDH;

Begin

Bước 1: Khởi tạo $W^0 = Z$;

Bước 2: Tính $W^1 = \psi W^0 + Z$;

Bước 3: Nếu điều kiện dừng chưa thỏa mãn thì gán $W^0 := W^1$ và trở lại Bước 2;

Bước 4: $W^* \approx W^1$.

End

Hình 3.4. Pha 2: xác định trọng số tầng ra

Với mỗi vectơ N-chiều u, ta ký hiệu chuẩn $||u||_*$ tính theo công thức sau [19]:

$$|| u ||_* = \max_{j \le N} \{ | u_j | \} \tag{3.22}$$

điều kiện kết thúc có thể chọn một trong biểu thức sau:

$$\text{a)} \ \frac{q}{1-q} \left\| W^1 - W^0 \right\|_* \le \varepsilon \tag{3.23}$$

$$\text{b)} \ t \ge \frac{\ln \dfrac{\varepsilon(1-q)}{\|Z\|_*}}{\ln q} = \frac{\ln q - \ln \|Z\|_* + \ln(1-q)}{\ln q} \tag{3.24}$$

với t là số lần lặp.

Các điều kiện dừng này được suy từ định lý về tính hội tụ được trình bày sau đây.

3.2.3. Đặc tính hội tụ

Định lý sau đảm bảo tính hội tụ và cho phép ước lượng sai số của thuật toán.

Định lý 3.2. Thuật toán huấn luyện mạng HDH luôn kết thúc sau hữu hạn bước và đánh giá sau là đúng.

$$\left\|W^1 - W^*\right\|_* \le \varepsilon \tag{3.25}$$

Chứng minh: Trước hết, từ kết luận của Định lý 3.1 ta dễ dàng suy ra pha thứ nhất của thuật toán sẽ kết thúc sau hữu hạn bước và $q_k \le q$ với mọi *k*. Mặt khác chuẩn $\|\Psi\|_*$ của ma trận Ψ được xác định tương ứng với chuẩn vectơ (3.22) sẽ là: (Định lý 2 mục 9.6 chương 1 trang 177 của [19]).

$$\|\Psi\|_* = \max_{k \le N}\{q_k\} \le q \tag{3.26}$$

Vì vậy, pha thứ hai là thủ tục tìm điểm bất động $u = \Psi u + Z$ của ánh xạ co có hệ số co *q* và xấp xỉ ban đầu $u^0 = 0$, $u^1 = Z$. Với *t* bước lặp ở pha 2 sẽ tương ứng với *t+1* bước lặp $u^{k+1} = \Psi u^k + Z$ của thuật toán tìm điểm bất động (trong định lý 1 mục 12.2 chương II của [19]). Vì vậy ta có đánh giá:

$$\left\|W^1 - W^*\right\|_* \le \frac{q^{t+1}}{1-q}\left\|u^1 - u^0\right\|_* = \frac{q^{t+1}}{1-q}\|Z\|_* \tag{3.27}$$

Biểu thức (3.24) tương đương với vế phải của (3.27) nhỏ hơn hoặc bằng ε. Có nghĩa là ($\frac{q^{t+1}}{1-q}\|Z\|_* \le \varepsilon$). Mặt khác nếu áp dụng (3.27) cho *t=0;* $u^0 = W^0$; $u^1 = W^1$ thì ta có:

$$\left\|W^1 - W^*\right\|_* \le \frac{q}{1-q}\left\|w^1 - w^0\right\|_* \tag{3.28}$$

Kết hợp (3.23) với (3.28) ta có (3.25) suy ra điều cần chứng minh. Định lý đã được chứng minh.

3.2.4. Độ phức tạp của thuật toán

Mục này sẽ phân tích và tính toán độ phức tạp của thuật toán.

Pha 1: Bên cạnh tham số số chiều *n* và số mốc nội suy *N*. Độ phức tạp của Pha 1 còn phụ thuộc vào phân bố của tập mốc nội suy $\{x^k\}_{k=1}^N$ và nó không phụ thuộc vào hàm *f*. Ngoài ra còn phụ thuộc vào việc khởi gán giá trị σ_0 có thể làm cho $p>q$ (xem bước 3 của hình 3.3) hoặc $p<q$ (xem bước 4 của hình 3.3). Với những

trường hợp trước, với mỗi giá trị $k \le N$, đặt m_k là số lần lặp trong bước 3 sao cho $q_k > q$ với $\sigma_k = \alpha^{m_k - 1}\sigma_0$ nhưng $q_k \le q$ với $\sigma_k = \alpha^{m_k}\sigma_0$. Vì vậy,

$$m_k \le \log_\alpha \frac{\sigma_{\min}}{\sigma_0} \text{ , với } \sigma_{\min} = min\{\sigma_k\} \ (\le \sigma_0) \tag{3.29}$$

Cũng tương tự như vậy, nếu m_k là số lần lặp trong bước 4.

$$m_k \le \left|\log_\alpha \frac{\sigma_{\max}}{\sigma_0}\right| \text{ , với } \sigma_{\max} = max\{\sigma_k\} \ (\ge \sigma_0) \tag{3.30}$$

$$\text{Đặt } c = max\left\{\left|\log_\alpha \frac{\sigma_{\min}}{\sigma_0}\right|, \left|\log_\alpha \frac{\sigma_{\max}}{\sigma_0}\right|\right\} \tag{3.31}$$

Khi đó ta có độ phức tạp của Pha 1 là O(cnN^2) (hình 3.3).

Pha 2: Đặt T là số lần lặp trong Pha 2 của thuật toán. T phụ thuộc vào chuẩn $||Z||_*$ của véctơ z và giá trị q. Theo công thức (3.24) và theo chứng minh ở định lý 3.2 , T có thể được ước lượng theo công thức sau:

$$T = \frac{\ln \dfrac{\varepsilon(1-q)}{\|Z\|_*}}{\ln q} = \log_q \frac{\varepsilon(1-q)}{\|Z\|_*} \tag{3.32}$$

Như vậy độ phức tạp của Pha 2 là $O(T.nN^2)$

Vậy tổng độ phức tạp của thuật toán mới phát triển là $O((T+c)nN^2)$.

Trong thuật toán này thời gian huấn luyện pha thứ nhất chỉ phụ thuộc vào phân bố của các mốc nội suy còn thời gian ở pha thứ hai phụ thuộc vào độ lớn của chuẩn $\|Z\|_*$ mà không phụ thuộc trực tiếp vào hàm f cần nội suy.

3.3. Thử nghiệm thuật toán

Để đánh giá hiệu quả của thuật toán huấn luyện mạng nơron. Thông thường người ta hay quan tâm tới các tiêu chí sau:

Tốc độ hội tụ của thuật toán biểu hiện qua thời gian chạy của và sai số huấn luyện.

Tính tổng quát của mạng biểu hiện qua chính sai số kiểm tra các điểm mới sau khi huấn luyện.

Trong phần này chúng tôi thực nghiệm trên mạng RBF với hàm 3 biến cho trong công thức (3.33) để thử nghiệm thời gian huấn luyện và tính tổng quát của thuật toán. Đặc biệt chúng tôi có so sánh với thuật toán Gradient ở mục sau.

$$y = x_1^2 x_2 + \sin(x_2 + x_3 + 1) + 4 \tag{3.33}$$

Thử nghiệm tính tổng quát của mạng bằng cách sau: Đầu tiên chọn một số điểm không nằm trong tập mốc nội suy, sau khi huấn luyện mạng, nội suy các điểm đó rồi so sánh với giá trị đúng của hàm đã biết và ước lượng sai số.

Dữ liệu thử nghiệm được lấy cách đều theo hai chiều và tích hợp ngẫu nhiên với chiều thứ ba.

Thực nghiệm chạy trên máy tính có cấu hình: Intel Pentium 4 Processor, 3.0GHz, 512MB DDR RAM. Kết quả và những nhận xét của thuật toán được trình bày sau đây:

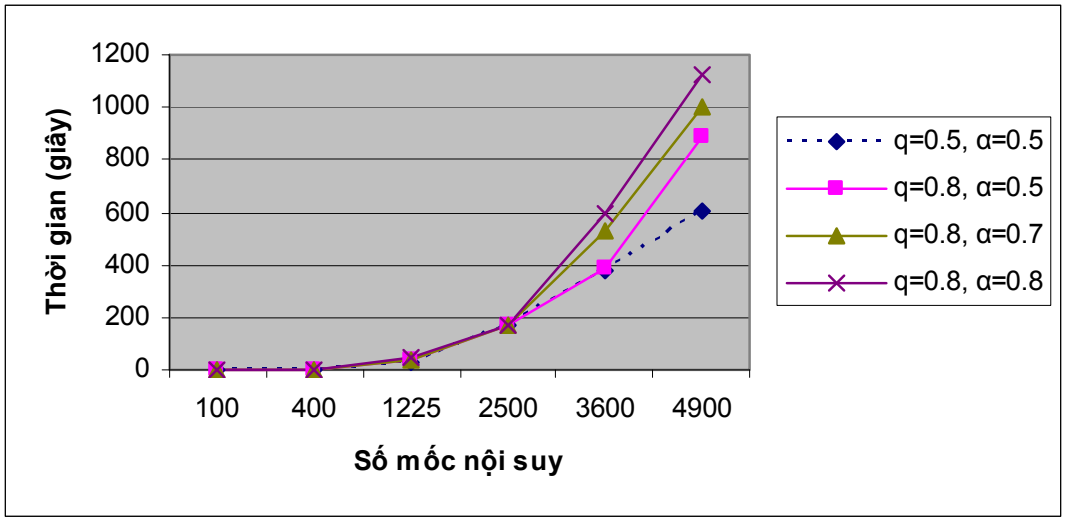
3.3.1. Tốc độ hội tụ

Thời gian huấn luyện phản ánh tốc độ hội tụ của thuật toán. Nó được thử nghiệm với số mốc khác nhau với các tham số q, α và ε thay đổi tuần tự.

Kết quả thử nghiệm trên hàm (3.33) với $x_1 \in [0,3]$, $x_2 \in [0,4]$, $x_3 \in [0,5]$. Tham số dừng thuật toán $\varepsilon = 10^{-6}$. Tham số q và α nhận các giá trị lần lượt là 0.5 và 0.5; 0.8 và 0.5; 0.8 và 0.7; 0.8 và 0.8 lần lượt với các mốc biến đổi từ 100 đến 5000. Kết quả được trình bày trong bảng 3.1 và hình 3.5.

Bảng 3.1: Thời gian huấn luyện với tham số dừng $\varepsilon = 10^{-6}$

Số mốc	$\varepsilon=10^{-6}$, q=0.5, α=0.5	$\varepsilon=10^{-6}$, q=0.8, α=0.5	$\varepsilon=10^{-6}$, q=0.8, α=0.7	$\varepsilon=10^{-6}$, q=0.8, α=0.8
	Thời gian huấn luyện	Thời gian huấn luyện	Thời gian huấn luyện	Thời gian huấn luyện
100	1''	1''	1''	1''
400	2''	3''	4''	4''
1225	30''	35''	37''	45''
2500	167''	170''	173''	174''
3600	378''	390''	530''	597''
4900	602''	886''	1000''	1125''



Hình 3.5 Đồ thị thời gian huấn luyện khi các tham số q và α thay đổi

Trong bảng 3.2 là kết quả trong trường hợp $q = \alpha = 0.7$ với 2500 mốc và các tham số dừng ε khác nhau.

Bảng 3.2 : Thời gian huấn luyện của 2500 mốc, q=α=0.7 và ε thay đổi.

Số mốc	$\varepsilon=10^{-9}$	$\varepsilon=10^{-8}$	$\varepsilon=10^{-6}$	$\varepsilon=10^{-5}$
	Thời gian huấn luyện	Thời gian huấn luyện	Thời gian huấn luyện	Thời gian huấn luyện
2500	177''	175''	172''	170''

Nhận xét: Từ những kết thực nghiệm trên ta có nhận xét sau:

1. Thời gian huấn luyện của thuật toán là ngắn so với các thuật toán khác (thậm trí chỉ vài phút cho khoảng 3000 mốc). Thời gian huấn luyện tăng khi q hoặc α tăng, có nghĩa là nó sẽ giảm khi các tham số q hoặc α giảm. Nhìn đồ thị hình 3.5 ta thấy trong trường hợp q=0.5 và α=0.5 thì thời gian huấn luyện là (đường nét đứt) thấp nhất, còn (đường có dấu x) là trường hợp q=0.8 và α=0.8 lớn nhất. Tuy nhiên, tham số α nhạy cảm hơn, thời gian huấn luyện bị ảnh hưởng bởi tham số α hơn tham số q.
2. Nhìn trong bảng 3.2, khi tham số dừng ε giảm dần, ta thấy thời gian huấn luyện thay đổi không nhiều. Điều này có nghĩa là độ chính xác đòi hỏi khi huấn luyện của một ứng dụng không ảnh hưởng mạnh tới thời gian huấn luyện.

3.3.2. Tính tổng quát

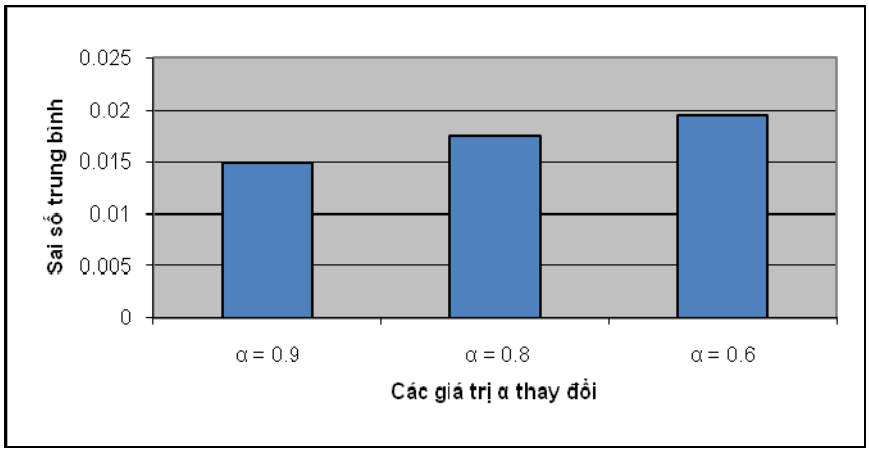
Lấy 400 mốc trong miền $\{x_1 \in [0,3], x_2 \in [0,4], x_3 \in [0,5]\}$ của hàm (3.33) và huấn luyện mạng với các giá trị khác nhau của tham số q và α, với tham số dừng $\varepsilon = 10^{-6}$. Sau khi huấn luyện xong, lấy 8 điểm ngẫu nhiên không nằm trong tập mẫu huấn luyện để kiểm tra. Kết quả kiểm tra cho trường hợp q và α thay đổi được trình bày trong bảng 3.3 và bảng 3.4.

1) Kiểm tra với q=0.8 và α thay đổi

Thử nghiệm với trường hợp $\varepsilon=10^{-6}$, q=0.8 và α lần lượt nhận các giá trị 0.9; 0.8; 0.6 được trình bày trong bảng 3.3.

Bảng 3.3. Kiểm tra các điểm với q=0.8; $\varepsilon = 10^{-6}$ và α thay đổi nhận các giá trị 0.9 ;0.8 ;0.6

Điểm kiểm tra			**Giá trị hàm**	**q=0.8, α=0.9 (Thời gian =5'')**		**q=0.8, α=0.8 (Thời gian =4'')**		**q=0.8, α=0.6 (Thời gian =4'')**	
X_1	**X_2**	**X_3**		**Giá trị nội suy**	**Sai số**	**Giá trị nội suy**	**Sai số**	**Giá trị nội suy**	**Sai số**
2.68412	2.94652	3.329423	26.065739	26.0679	21.6E-04	26.06879	30.502E-04	26.0691	33.802E-04
2.21042	1.052145	0.040721	10.007523	10.0024	51.24E-04	10.0144	68.763E-04	10.0146	71.163E-04
2.842314	2.525423	0.048435	23.983329	24.01001	266.81E-04	24.0201	367.706E-04	24.0251	417.90E-04
2.842315	3.789123	3.283235	35.587645	35.5818	58.45E-04	35.5799	77.452E-04	35.5963	86.548E-04
2.05235	3.78235	1.63321	20.063778	20.05203	117.48E-04	20.0803	165.219E-04	20.0812	174.21E-04
2.84202	3.789241	3.283023	35.582265	35.5986	163.34E-04	35.5621	201.655E-04	35.561	212.65E-04
2.051234	3.15775	0.59763	16.287349	16.28183	55.16E-04	16.294	66.505E-04	16.295	78.505E-04
2.52621	3.36832	0.86412	24.627938	24.67451	465.72E-04	24.58628	416.584E-04	24.5798	481.38E-04
Sai số trung bình					**149.97E-04**		**174.298E-04**		**194.522E-04**



Hình 3.6 Đồ thị kiểm tra sai số khi α thay đổi

Nhận xét: Từ bảng 3.3 và hình 3.6 ta thấy khi α tăng thì sai số giảm đi rất nhanh. Cụ thể với trường hợp α=0.6 thì sai số trung bình là 194.97E-04 còn khi α = 0.9 thì là 149.97E-04. Khi α nhỏ thì tham số độ rộng bán kính σ_k sẽ giảm nên dẫn đến ảnh hưởng tính tổng quát của mạng.

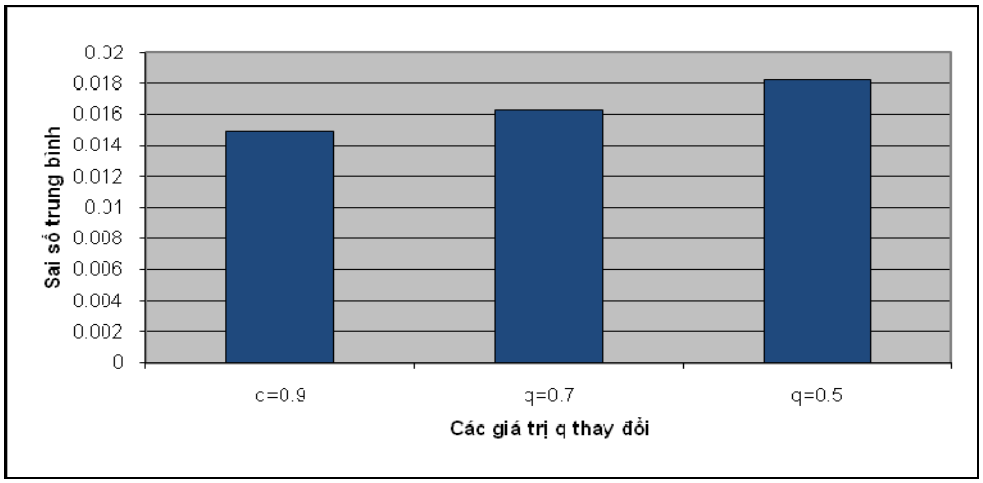
Trong thí nghiệm của chúng tôi, thì α tốt nhất trong khoảng từ [0.7,…,0.9] nhưng việc chọn giá trị α như thế nào còn phụ thuộc vào việc cân bằng giữa thời gian huấn luyện và tính tổng quát của mạng.

2) Kiểm tra với α= 0.9 và *q* thay đổi.

Thử nghiệm với trường hợp ε=10^{-6}, α=0.9 và *q* lần lượt nhận các giá trị 0.9; 0.7; 0.5 được trình bày trong bảng 3.4.

Bảng 3.4: Kiểm tra các điểm với α=0.9; ε=10^{-6} và q thay đổi nhận các giá trị 0.9; 0.7; 0.5

Điểm kiểm tra			Giá trị hàm	q=0.9, α=0.9		q=0.7, α=0.9		q=0.5, α=0.9	
X_1	X_2	X_3		Giá trị nội suy	Sai số	Giá trị nội suy	Sai số	Giá trị nội suy	Sai số
2.68412	2.94652	3.32942	26.06573	26.0655	2.22E-04	26.0654	3.12E-04	26.0693	35.46E-04
2.21042	1.052145	0.04072	10.00752	10.0217	141.79E-04	10.0196	120.33E-04	10.0224	149.06E-04
2.842314	2.525423	0.04843	23.98332	24.0112	279.17E-04	24.0204	370.87E-04	24.0221	387.53E-04
2.842315	3.789123	3.28323	35.58764	35.5818	58.03E-04	35.5819	57.27E-04	35.5818	58.08E-04
2.05235	3.78235	1.63321	20.06377	20.1105	467.62E-04	20.1159	520.95E-04	20.1135	497.7E-04
2.84202	3.7892411	3.28302	35.58226	35.5881	58.26E-04	35.5884	61.45E-04	35.5886	63.11E-04
2.051234	3.15775	0.59763	16.28734	16.2853	20.73E-04	16.2852	21.13E-04	16.2775	98.93E-04
2.52621	3.36832	0.86412	24.62793	24.6117	162.8E-04	24.6133	146.16E-04	24.6108	171.74E-04
Sai số trung bình					**148.83E-04**		**162.67E-04**		**182.71E-04**



Hình 3.7 Đồ thị kiểm tra sai số khi q thay đổi

Nhận xét: Từ kết quả trên bảng 3.4 và hình 3.7 chỉ ra rằng tính tổng quát của mạng tăng nhanh khi q tăng. Cụ thể khi q = 0.5 thì sai số trung bình là 182.71E-04, khi q = 0.9 là 148.83E-04. Cho dù q thay đổi ảnh hưởng ít tới thời gian huấn luyện như đã thảo luận trong mục trên hơn là ảnh hưởng tới tỉnh tổng quát.

3.4. So sánh với phương pháp Gradient

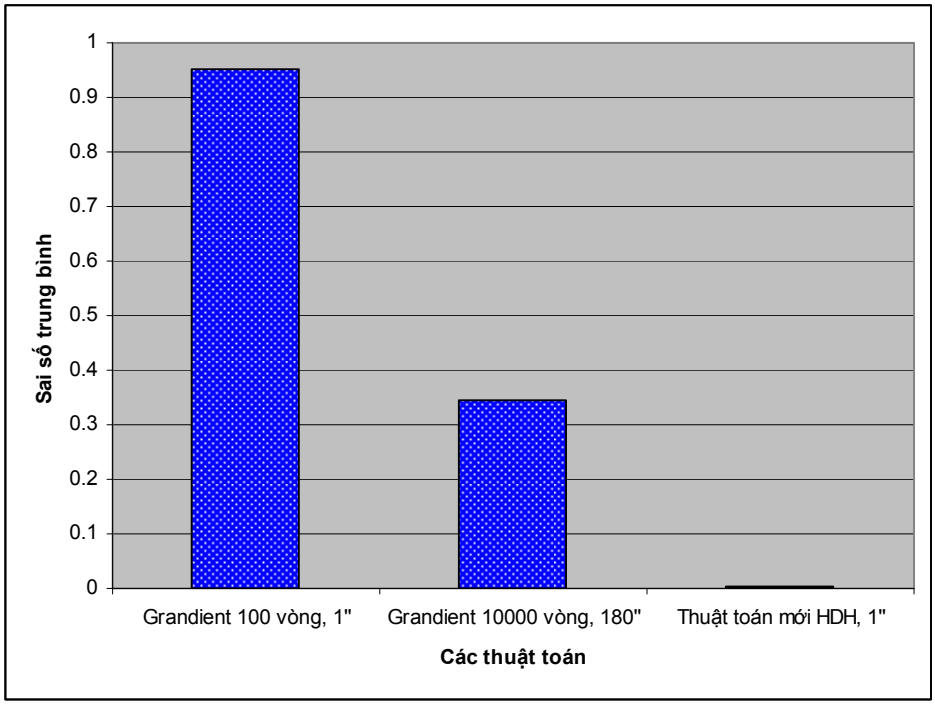
Chúng tôi thực nghiệm trên hàm (3.33) với 100 mốc nội suy $x_1 \in [0,3]$, $x_2 \in [0,4]$, $x_3 \in [0,5]$ để so sánh hiệu quả của thuật toán mới và thuật toán Gradient. Trong thuật toán Gradient rất khó để đạt được độ chính xác cao trong huấn luyện và trong điều khiển tính tổng quát của mạng. Bên cạnh thời gian huấn luyện, chúng tôi còn so sánh hai đặc tính của thuật toán đó là độ chính xác của những điểm huấn luyện và sai số của những điểm chưa huấn luyện chính là tính tổng quát của mạng. Dùng Matlab 6.5 để cài đặt thuật toán Gradient.

3.4.1. So sánh thời gian và độ chính xác của những điểm huấn luyện.

Chúng tôi lấy ngẫu nhiên 8 điểm trong 100 mốc nội suy. Sau khi huấn luyện mạng bằng thuật toán mới với $\varepsilon=10^{-6}$, q= 0.8, α=0.9. Bằng thuật toán Gradient trong hai trường hợp 100 vòng lặp và 10000 vòng lặp. Chúng tôi so sánh thời gian và độ chính các của thuật toán. Kết quả kiểm tra được chỉ ra trong bảng 3.5 và hình 3.8.

Bảng 3.5: Kiểm tra sai số của 8 mốc huấn luyện để so sánh độ chính xác

Điểm kiểm tra			**Giá trị hàm**	**Thuật toán Gradient với 100 vòng lặp Thời gian: 1''**		**Thuật toán Gradient với 10000 vòng lặp Thời gian: 180''**		**Thuật toán mới HDH $\varepsilon=10^{-6}$, q= 0.8, α=0.9 Thời gian:1''**	
X_1	**X_2**	**X_3**		**Giá trị nội suy**	**Sai số**	**Giá trị nội suy**	**Sai số**	**Giá trị nội suy**	Sai số
1.666667	0.000000	0.000000	4.841471	4.4645	3769.7E-04	5.0959	2544.2E-04	4.84146	0.1E-04
0.333333	0.444444	1.379573	4.361647	3.5933	7683.4E-04	3.6708	6908.4E-04	4.36166	0.09E-04
2.666667	0.444444	1.536421	7.320530	8.7058	13852.7E-04	7.2647	558.2E-04	7.32052	0.08E-04
0.666667	1.333333	0.128552	5.221158	4.0646	11565.5E-04	4.9517	2694.5E-04	5.22117	0.1E-04
2.666667	1.333333	1.589585	12.77726	12.5041	2731.6E-04	12.1965	5807.6E-04	12.7772	0.7E-04
1.666667	1.777778	0.088890	9.209746	6.6682	25415.4E-04	9.2944	846.5E-04	9.20972	0.2E-04
2.333333	0.444444	0.039225	7.415960	6.7228	6931.5E-04	7.48	640.4E-04	7.41596	0.005E-04
2.666667	3.555556	0.852303	28.51619	28.0927	4234.9E-04	29.2798	7636.0E-04	28.5162	0.09E-04
Sai số trung bình					**9523.1E-04**		**3454.5E-04**		**0.19E-04**



Hình 3.8 So sánh độ chính xác và thời gian của thuật toán mới và thuật toán Grandient

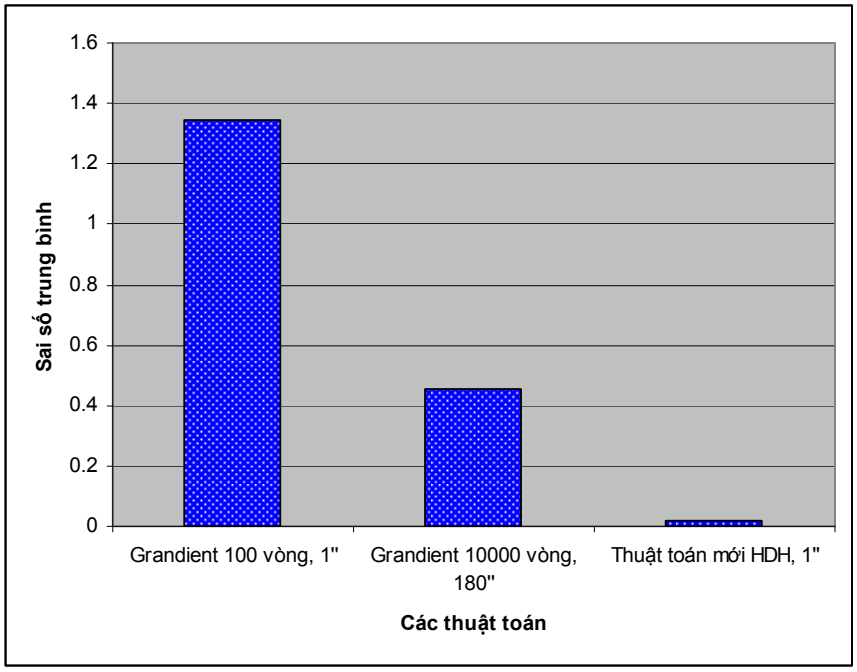
Nhận xét: Trong hình 3.8 và bảng 3.5 ta thấy, cùng thời gian huấn luyện là 1 giây thuật toán Gradient có sai số trung bình là 9523.1E-04 còn thuật toán mới HDH là 0.19E-04, sai số trung bình của thuật toán mới nhỏ hơn rất nhiều. Còn nếu tăng thời gian huấn luyện của thuật toán Gradient lên 180 giây thì sai số trung bình là 3454.5E-04 có giảm hơn so với huấn luyện 1 giây nhưng vẫn lớn hơn thuật toán mới rất nhiều. Như vậy, theo thực nghiệm ta thấy thuật toán mới tốt hơn so với thuật toán Gradient trong cả hai trường hợp về thời gian huấn luyện và tính chính xác.

3.4.2. So sánh tính tổng quát

Chúng tôi lấy ra 8 mốc không đưa vào huấn luyện để dùng làm tập điểm kiểm tra (các điểm này xa các tâm hàm bán kính hơn). Sau khi huấn luyện bằng hai thuật toán với các tham số như mục 3.4.1. So sánh sai số ở tập kiểm tra để đánh giá tính tổng quát. Kết quả kiểm tra được thể hiện trong bảng 3.6 và hình 3.9.

Bảng 3.6:Kiểm tra 8 điểm chưa được huấn luyện và so sánh tính tổng quát

Điểm kiểm tra			**Giá trị hàm**	**Thuật toán Gradient với 100 vòng lặp Thời gian: 1''**		**Thuật toán Gradient với 10000 vòng lặp Thời gian: 180''**		**Thuật toán mới HDH $\varepsilon=10^{-6}$, q= 0.8, α=0.9 Thời gian:1''**	
X_1	**X_2**	**X_3**		**Giá trị nội suy**	**Sai số**	**Giá trị nội suy**	**Sai số**	**Giá trị nội suy**	**Sai số**
0.32163	0.45123	1.38123	4.350910	2.1394	22115.1E-04	3.9309	4200.1E-04	4.32214	287.7E-04
0.67123	0.8912	1.4512	4.202069	2.8529	13491.6E-04	4.7884	5863.3E-04	4.20115	9.1E-04
1.68125	1.34121	0.27423	8.293276	6.1078	21854.7E-04	8.3869	936.2E-04	8.30878	155.0E-04
0.34312	1.78123	2.56984	3.406823	3.2115	1953.2E-04	4.1438	7369.7E-04	3.399	78.2E-04
2.65989	3.56012	0.8498	28.42147	27.5174	9040.7E-04	29.1648	7433.2E-04	28.429	75.2E-04
1.67013	2.23123	0.29423	9.84913	8.6415	12076.3E-04	9.5863	2628.3E-04	9.79204	570.9E-04
2.65914	3.56123	0.85612	28.41991	27.5147	9052.1E-04	29.1634	7434.8E-04	28.419	9.1E-04
1.3163	0.44925	1.12987	5.311670	3.5188	17928.7E-04	5.3729	612.2E-04	5.28737	243.0E-04
Sai số trung bình					**13439.0E-04**		**4559.7E-04**		**178.56E-04**



Hình 3.9 So sánh tính tổng quát của thuật toán mới và thuật toán Grandient

Nhận xét: Nhìn các kết quả trên bảng 3.6 và hình 3.9 ta thấy cùng thời gian huấn luyện là 1 giây, thuật toán Gradient có sai số trung bình là 13439.0E-04 lớn hơn rất nhiều so với 178.56E-04 của thuật toán mới với cùng thời gian. Trong trường hợp tăng thời gian huấn luyện cho thuật toán Gradient lên 180 giây thì sai số trung bình là 4559.7E-04 vẫn lớn hơn rất nhiều so với thuật toán mới trong thời gian 1 giây. Như vậy, qua thực nghiệm ta thấy thuật toán mới HDH có thời gian huấn luyện và sai số nhỏ hơn nhiều cũng như tính tổng quát tốt hơn so với thuật toán Gradient.

3.5. Nhận xét chung về thuật toán lặp hai pha HDH

Trong chương này chúng tôi đề xuất thuật toán hai pha đơn giản để huấn luyện mạng nội suy RBF. Pha thứ nhất xác định tham số độ rộng bán kính phù hợp với từng mốc nội suy, pha thứ 2 dùng phương pháp lặp để tính trọng số tầng ra. Trong phần trên đã chỉ ra rằng thuật toán luôn hội tụ, thời gian chạy chỉ phụ thuộc

vào việc khởi gán giá trị ban đầu q, α, ε, phân bố của mốc nội suy và chuẩn của véctơ.

Qua kết quả thí nghiệm ta thấy thuật toán có thời gian huấn luyện mạng rất nhanh kể cả khi số mốc lớn. Với thuật toán cực tiểu sai số tổng các bình phương, Looney cho rằng chỉ nên chạy với số mốc nhỏ hơn 200 [38] còn thuật toán này vẫn thực hiện được khi số mốc gần 5000.

Thuật toán mới đề xuất dễ thực hiện và có hiệu quả cao. Thực nghiệm cho thấy nó giải quyết tốt bài toán nội suy với số mốc lớn. Dễ dàng điều khiển cân bằng giữa tốc độ hội tụ và tính tổng quát của mạng bằng việc điều chỉnh các tham số. Một ưu việt nữa của thuật toán là các bán kính tầng ẩn có thể huấn luyện độc lập và ở pha hai trọng số tầng ra có thể huấn luyện độc lập, điều này làm cho chúng có thể song song hoá thuật toán. Hơn nữa, khi số mốc nội suy lớn, việc đánh giá chuẩn của vectơ N- chiều sẽ tốn thời gian. Trong trường hợp đó điều kiện dừng của thuật toán nên theo tiêu chuẩn (3.24) để xác định trước số bước lặp ở pha thứ hai, như vậy sẽ giảm thời gian tính toán trong mỗi bước.

Khi số mốc nội suy rất lớn, có thể phân dữ liệu thành những cụm con có kích thước nhỏ hơn. Theo đó việc huấn luyện có thể thực hiện song song cho từng cụm, như vậy sẽ giảm đáng kể thời gian huấn luyện. Kiến trúc mạng như vậy gọi là *mạng RBF địa phương* và sẽ được trình bày chi tiết trong chương 5.

Trong trường hợp số mốc nội suy N là rất lớn, khi đó có thể dùng một cách tiếp cận khác, đó là dùng mạng xấp xỉ RBF. Trong mạng này có thể thiết kế số nơron tầng ẩn ít hơn N dựa trên cách thức sau: Trước tiên, dữ liệu được chia thành K cụm $\{C_i\}_{i=1}^{K}$ bằng một thuật toán phân cụm (như k-mean,..), sau đó véctơ tâm v^i của mỗi nơron ẩn thứ i có thể được chọn là tâm của cụm Ci hoặc là gần với tâm của cụm nhất. Mạng sẽ được huấn luyện với tập mốc nội suy mới $\{v^i\}_{i=1}^{k}$.

Một đặc điểm của mạng RBF là có tính ảnh hưởng địa phương [43]. Vì vậy tham số độ rộng bán kính hay được chọn nhỏ (mục 6.1, trang 288-289 của [31]),

đặc biệt, trong [38] khuyên rằng nên đặt σ = 0.05 hoặc 0.1. Trong mục 3.11 trang 99 của [38] khuyên nên đặt $\sigma = 1/(2N)^{\frac{1}{n}}$. Tất cả các tham số bán kính này đều nhỏ, nên q_k phải nhỏ. Thực tế, điều kiện $q_k<1$ thể hiện một số giới hạn, nhưng không ảnh hưởng đến hiệu quả của thuật toán.

Mặc dù chúng tôi không so sánh với thuật toán giả nghịch đảo và huấn luyện đầy đủ, nhưng Lê Tiến Mười [7] đã thử nghiệm áp dụng cho bài toán nhận dạng chữ viết tay và cho thấy ưu điểm nổi trội của thuật toán HDH.

Tuy vậy với thuật toán mới HDH khi áp dụng nhận dạng mẫu, tính tổng quát của mạng tốt khi các mốc phân bố tương đối đều và không được tốt khi các mốc nội suy phân bố không tốt. Chương 4 chúng tôi sẽ xét trường hợp mốc cách đều.

CHƯƠNG 4. BÀI TOÁN NỘI SUY VỚI MỐC CÁCH ĐỀU

Thuật toán lặp hai pha huấn luyện mạng nội suy RBF mới đề xuất đã rút ngắn đáng kể thời gian huấn luyện mạng nơron nội suy RBF so với các thuật toán khác. Từ đây về sau ta gọi là thuật toán HDH cho gọn.

Tuy nhiên, trong thuật toán HDH, thời gian huấn luyện của pha thứ nhất chiếm phần lớn thời gian huấn luyện của cả mạng và nó tăng rất nhanh khi số mốc tăng. Trong nhiều ứng dụng ở các lĩnh vực đồ hoạ máy tính, phương trình toán lý, các bài toán kỹ thuật, và nhận dạng mẫu thường gặp bài toán nội suy với mốc cách đều [8,32,36,54]. Trong trường hợp này, các mốc nội suy có thể được biểu diễn dưới dạng $x^{i1,i2,...in} = (x_1^{i1},...,x_n^{in})$, trong đó $x_k^{ik} = x_k^0 + ik.h_k$; với h_k (k =1,…,n) là các hằng số bước cho trước (bước thay đổi của biến x_k), ik nhận giá trị từ 0 đến N_k (N_k +1 là số mốc chia của từng chiều). Khi đó thay cho chuẩn Euclide, ta xét chuẩn Mahalanobis: $\|x\|_A = \sqrt{x^T Ax}$, với A là ma trận đường chéo chọn thích hợp với các bước thay đổi của các biến, thì việc xác định các tham số bán kính σ_k trong pha thứ nhất chỉ còn phụ thuộc vào hằng số bước h_k. Nên có thể xác định trước σ_k theo số chiều và miền xác định của hàm cần nội suy. Thuật toán đã nêu trở thành thuật toán một pha. Trong chương này chúng tôi trình bày một cách giải quyết hiệu quả hơn cho những bài toán có mốc nội suy cách đều. Trên cơ sở thuật toán lặp hai pha mới đề xuất trong chương trước. Bố cục của chương như sau: Mục 4.1 trình bày các phân tích toán học để biểu diễn bài toán mốc cách đều dưới dạng dễ khảo sát về sau, định lý cơ sở và thuật toán được trình bày ở mục 4.2. Mục 4.3 dành để giới thiệu các kết quả thí nghiệm và những nhận xét chung được đưa ra ở mục 4.4 thay cho kết luận của chương.

Kết quả chính của chương này được công bố trong kỷ yếu Hội thảo quốc gia các vấn đề chọn lọc của CNTT lần thứ X, Đại Lải 9/2007 [3].

4.1. Biểu diễn bài toán

Bài toán mốc cách đều: Khi các mốc nội suy là cách đều thì có thể biểu diễn các mốc nội suy theo bộ chỉ số dạng (*i1,i2,..,in*) với $x^{i1,i2,...in} = (x_1^{i1},...,x_n^{in})$, trong đó:

$$x_k^{ik} = x_k^0 + ik.h_k \tag{4.1}$$

h_k (k = 1,…,n) là các hằng số bước cho trước (bước thay đổi của biến x_k), n là số chiều, và ik nhận giá trị từ 0 đến N_k (N_k +1 là số mốc chia của từng chiều).

Ta nhận thấy các hàm cơ sở bán kính xác định theo công thức (3.11) nhận giá trị bằng nhau ở những điểm có cùng khoảng cách với tâm và các mặt mức của hàm này có dạng siêu cầu. Việc chọn hàm cơ sở như vậy không thích hợp với các h_k khác nhau nhiều. Vì vậy, thay cho chuẩn Euclide, ta xét chuẩn Mahalanobis: $\|x\|_A = \sqrt{x^T Ax}$; với A là ma trận đường chéo có dạng:

$$A = \begin{pmatrix} \frac{1}{a_1^2} & 0 & ... & 0 \\ 0 & \frac{1}{a_2^2} & ... & 0 \\ ... & ... & . & .... \\ 0 & 0 & ... & \frac{1}{a_n^2} \end{pmatrix}$$

Các tham số a_k sẽ được chọn thích hợp với các bước h_k (dưới đây sẽ cho thấy là a_k=h_k với mọi k). Khi đó biểu thức (3.10) và (3.11) được viết lại như (4.2) và (4.3).

$$\varphi(x) = \sum_{i1,...in=1}^{N_1,...N_n} w_{i1,...in}\varphi_{i1,...in}(x) + w_0 \tag{4.2}$$

trong đó, $$\varphi_{i1,...in}(x) = e^{-\left\|x - x^{i1,...in}\right\|_A^2 / \sigma_{i1,...in}^2} \tag{4.3}$$

Ma trận $\Phi = \left(\varphi_{i1,...,in}^{j1,...,jn}\right)_{NxN}$ là ma trận vuông cấp $N = N_1...N_n$. Với

$$\varphi_{i1,..,in}^{j1,...,jn} = \varphi_{i1,...in}(x^{j1,...,jn}) = e^{-\left\|x^{j1,...,jn}-x^{i1,...in}\right\|_A^2/\sigma_{i1,...in}^2} \tag{4.4}$$

Ma trận $\Psi = I - \Phi$ sẽ là :

$$\Psi_{i1,..,in}^{j1,...,jn} = \begin{cases} 0; khi: j1,...,jn = i1,...,in \\ -e^{-\left\|x^{j1,...,jn}-x^{i1,...in}\right\|_A^2/\sigma_{i1,...,in}^2} \end{cases} \tag{4.5}$$

Các bán kính $\sigma_{i1,...,in}$ được xác định sao cho ma trận ψ thoả mãn điều kiện của ánh xạ co để thực hiện pha hai của thuật toán HDH.

Có nghĩa là với số dương $q < 1$, chọn các $\sigma_{i1,...,in}$ để cho

$$q_{i1,...,in} = \sum_{j1,...,jn} \left|\psi_{i1,...,in}^{j1,...,jn}\right| \le q < 1 \tag{4.6}$$

Thay biểu thức (4.1) vào ta có:

$$\Psi_{i1,..,in}^{j1,...,jn} = \begin{cases} 0; khi: j1,...,jn = i1,...,in \\ -e^{-\sum_{p=1}^{n}(jp-ip)^2\frac{h_p^2}{a_p^2}/\sigma_{i1,...,in}^2} \end{cases} \tag{4.7}$$

Và $q_{i1,...,in}$ có thể viết lại như sau:

$$q_{i1,..,in} = \sum_{j1,...,jn \neq i1,...,in} e^{-\sum_{p=1}^{n}(jp-ip)^2\frac{h_p^2}{a_p^2}/\sigma_{i1,...,in}^2} \tag{4.8}$$

Do tính chất hàm mũ (e^x), khai triển tích ở vế phải của (4.9) cho thấy (4.8) có thể viết lại là:

$$q_{i1,...,in} = \prod_{p=1}^{n}\sum_{jp=1}^{N_p} e^{-\frac{h_p^2}{\sigma_{i1,...,in}^2 a_p^2}(jp-ip)^2} - 1 \tag{4.9}$$

Với cách biểu diễn mới, ta sẽ đưa ra định lý cơ sở của thuật toán:

4.2. Định lý cơ sở và mô tả thuật toán

4.2.1. Định lý cơ sở

Định lý 4.1. Với mọi $q \in (0,1)$, nếu $\sigma_{i1,\ldots,in} = \sqrt{\dfrac{1}{\ln\left(\dfrac{\sqrt[n]{1+q}-1}{8}\right)^{-1}}}$ thì $q_{i1,i2,\ldots,in} < q$

Chứng minh: ta đặt $a_p = h_p ; \forall p$, công thức (4.9) có ước lượng như sau:

$$q_{i1,\ldots,in} < \left(1 + 2\sum_{k=1}^{\infty} e^{-\frac{k^2}{\sigma^2_{i1,\ldots,in}}}\right)^n - 1 \le q \tag{4.10}$$

$$\Leftrightarrow \quad 1 + 2\sum_{k=1}^{\infty} e^{-\frac{k^2}{\sigma^2_{i1,\ldots,in}}} \le \sqrt[n]{1+q} \tag{4.11}$$

$$\Leftrightarrow \quad \sum_{k=1}^{\infty} e^{-\frac{k^2}{\sigma^2_{i1,\ldots,in}}} \le \frac{\sqrt[n]{1+q}-1}{2} \tag{4.12}$$

Ta sẽ chọn các tham số $\sigma_{i1,\ldots,1n}$ bằng nhau và bằng σ để (4.12) thỏa mãn.

Bởi vì vế phải của (4.12) < 1/2 khi $q < 1$ nên để (4.12) đúng thì số hạng đầu tiên của vế trái phải nhỏ hơn 1/2:

$$\Leftrightarrow \sigma < \frac{1}{\sqrt{\ln 2}} \tag{4.13}$$

Mặt khác ta ước lượng vế trái của (4.12) như sau:

$$\sum_{k=1}^{\infty} e^{-\frac{k^2}{\sigma^2}} = e^{-\frac{1}{\sigma^2}} \sum_{k=1}^{\infty} e^{-\frac{k^2-1}{\sigma^2}} = e^{-\frac{1}{\sigma^2}} \left(1 + \sum_{k=2}^{\infty} e^{-\frac{k^2-1}{\sigma^2}}\right) = e^{-\frac{1}{\sigma^2}} \left(1 + \sum_{k=1}^{\infty} e^{-\frac{(k+1)^2-1}{\sigma^2}}\right) \tag{4.14}$$

$$= e^{-\frac{1}{\sigma^2}} \left(1 + \sum_{k=1}^{\infty} e^{-\frac{k(k+2)}{\sigma^2}}\right) < e^{-\frac{1}{\sigma^2}} \left(1 + \sum_{k=1}^{\infty} e^{-\frac{k^2}{\sigma^2}}\right) < e^{-\frac{1}{\sigma^2}} \left(1 + \int_0^{\infty} e^{-\frac{t^2}{\sigma^2}} dt\right) = e^{-\frac{1}{\sigma^2}} \left(1 + \sigma\sqrt{\Pi}\right) \tag{4.15}$$

Chú ý tới công thức (4.13) ta có:

$$\sum_{k=1}^{\infty} e^{-\frac{k^2}{\sigma^2}} < e^{-\frac{1}{\sigma^2}}\left(1+\sqrt{\frac{\Pi}{\ln 2}}\right) \approx 3{,}13 e^{-\frac{1}{\sigma^2}} < 4\sum_{k=1}^{\infty} e^{-\frac{k^2}{\sigma^2}} \tag{4.16}$$

Nên (4.12) thỏa mãn khi

$$4e^{-\frac{1}{\sigma^2_{i1,\ldots,in}.}} \le \frac{\sqrt[n]{1+q}-1}{2} \quad hay \quad e^{-\frac{1}{\sigma^2_{i1,\ldots,in}.}} \le \frac{\sqrt[n]{1+q}-1}{8} \tag{4.17}$$

$$\Leftrightarrow \quad -\frac{1}{\sigma^2_{i1,\ldots,in}.} \le \ln\left(\frac{\sqrt[n]{1+q}-1}{8}\right) \tag{4.18}$$

$$\Leftrightarrow \quad \sigma^2_{i1,\ldots,in} \le \frac{1}{\ln\left(\frac{\sqrt[n]{1+q}-1}{8}\right)^{-1}} \tag{4.19}$$

$$\Leftrightarrow \sigma_{i1,\ldots jn} \le \sqrt{\frac{1}{\ln\left(\frac{\sqrt[n]{1+q}-1}{8}\right)^{-1}}} \tag{4.20}$$

Để (4.12) thoả mãn ta có thể $\sigma_{i1,\ldots,in}$ như nhau theo công thức :

$$\sigma_{i1,\ldots,in} = \sqrt{\frac{1}{\ln\left(\frac{\sqrt[n]{1+q}-1}{8}\right)^{-1}}} \tag{4.21}$$

Định lý đã được chứng minh.

4.2.2. Mô tả thuật toán một pha

Dựa trên những phân tích và tính toán trong phần trên. Với hằng số dương $q<1$ ta có thể chọn trước các $\sigma_{i1,..,in}$ theo công thức (4.21) để thoả mãn điều kiện của ánh xạ co trong công thức (4.6). Sau đó áp dụng phương pháp lặp đơn trong pha thứ hai của thuật toán HDH để xác định trọng số tầng ra. Thuật toán được đặc tả trong hình 4.1 dưới đây.

Proceduce Thuật toán 1 pha huấn luyện mạng

Bước 1: Xác định bán kính

Xác định các $\sigma_{i1,..,in}$ theo công thức (4.21) ;

Bước 2: Tính trọng số tầng ra

Tìm W^* bằng phương pháp lặp đơn; // pha 2 của thuật toán HDH;

End

Hình 4.1 : Thuật toán 1 pha huấn luyện mạng RBF với mốc cách đều

Nhận xét:

1) Trong [34], thuật toán 2 pha HDH yêu cầu $q_{i1,...,in} < q$ và nếu thay $\sigma_{i1,...,in} = \frac{\sigma_{i1,...,in}}{\alpha}$ thì $q_{i1,...,in} > q$. Nhưng trong thuật toán mới 1 pha thì những tham số bán kính $\sigma_{i1,...,in}$ được xác định trước theo công thức (4.21) và thoả mãn $q_{i1,...,in} < q$.

2) Những mạng được huấn luyện bằng thuật toán một pha là khác so với những mạng được huấn luyện bằng thuật toán 2 pha HDH.

4.3. Thực nghiệm

Mục tiêu của phần này là so sánh thời gian, sai số huấn luyện và tính tổng quát của thuật toán một pha mới phát triển (gọi tắt là thuật toán QHDH cho gọn) với thuật toán HDH và các thuật toán một pha Gradient khác.

Thực nghiệm trên hai hàm khác nhau: Trước tiên với hàm 2 biến:

$$y = \frac{x_1^2 + x_1 x_2}{5} + \frac{x_2}{3} + \frac{1}{2},\ x_1 \in [0,10],\ x_2 \in [0,20] \tag{4.22}$$

Với các mốc nội suy là cách đều cho mỗi chiều, sử dụng để so sánh thời gian huấn luyện với thuật toán HDH. Sau đó là với hàm 3 biến phi tuyến phức tạp hơn:

$$y = x_1^2 x_2 + \sin(x_2 + x_3 + 1) + 1,\ x_1 \in [0,3],\ x_2 \in [0,4],\ x_3 \in [0,5] \quad (4.23)$$

Dùng để so sánh sai số huấn luyện và tính tổng quát của mạng với thuật toán HDH, QTL, QTH. Trong đó QTL là thuật toán Quick Training do Looney [38] giới thiệu với tham số độ rộng bán kính $\sigma = \frac{1}{(2N)^{1/n}}$ và QTH là thuật toán Quick Training với $\sigma = \frac{D\max}{\sqrt{2N}}$ theo gợi ý của Haykin[30]; *Dmax* là khoảng cách lớn nhất giữa các mốc; *N* là số lượng các mốc nội suy, còn *n* là số chiều của mốc nội suy.

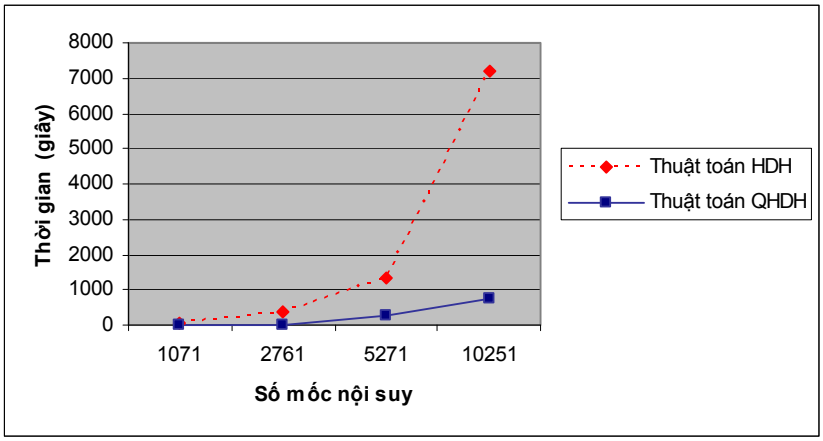
Thực nghiệm thực hiện trên máy Intel Pentium IV Processor, 3.0GHz, 512MB DDR RAM. Sai số cho điều kiện dừng là $\varepsilon = 10^{-6}$.

4.3.1. So sánh thời gian huấn luyện

Kết quả được chỉ ra trong bảng 4.1 với hàm hai biến để so sánh thời gian huấn luyện giữa thuật toán QHDH và thuật toán HDH.

Bảng 4.1 : So sánh thời gian huấn luyện giữa thuật toán 2 pha HDH và 1 pha QHDH

Số mốc	**Thuật toán 2 pha HDH q=0.9, α=0.9**	**Thuật toán 1 pha QHDH q=0.9**
1071 ;N_1=51, h_1=0.2, N_2=21, h_2=1	32''	10''
2761 ;N_1=251, h_1=0.04, N_2=11, h_2=2	365''	25''
5271 ;N_1=251, h_1=0.04, N_2=21, h_2=1	1315''	275''
10251;N_1=201,h_1=0.05,N_2=51,h_2=0.4	>2h	765''

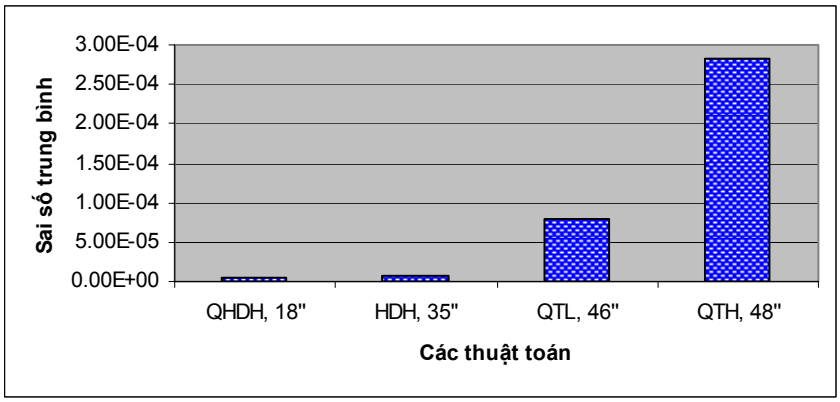


Hình 4.2: Đồ thị so sánh thời gian huấn luyện giữa thuật toán HDH và QHDH

Nhận xét: Nhìn bảng 4.1, hình 4.2 ta thấy thời gian huấn luyện của thuật toán QHDH giảm đáng kể so với thuật toán 2 pha HDH. Khi số mốc càng lớn thì sự chênh nhau về thời gian càng rõ rệt. Cụ thể với 1071 mốc thuật toán HDH hết 32 giây, còn QHDH hết 10 giây. Nhưng với 10251 thuật toán HDH hết >2 giờ trong khi đó thuật toán QHDH chỉ mất 765 giây. Bởi vì phần lớn thời gian huấn luyện của thuật toán HDH là do pha một. Còn trong thuật toán một pha với những mốc cách đều ta ước lượng được các bán kính trước sau đó chỉ còn huấn luyện pha hai.

4.3.2. So sánh sai số huấn luyện

Kết quả thực nghiệm ở bảng 4.2 và hình 4.3 cho hàm 3 biến như công thức (4.22) với 1331 mốc nội suy, có $N_1=11$, $h_1=0.3$; $N_2=11$; $h_2=0.4$; $N_3=11$, $h_3 = 0.5$. Sau khi huấn luyện, chúng tôi lấy 10 mốc nội suy để tính toán và so sánh sai số huấn luyện của các thuật toán QHDH, HDH, QTL, QTH.



Hình 4.3: So sánh sai số và thời gian huấn luyện của các thuật toán QHDH, HDH, QTL, QTH với 1331 mốc của hàm 3 biến.

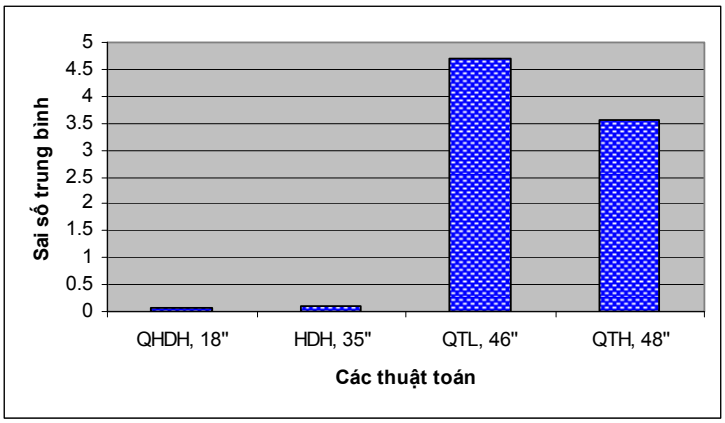
Nhận xét: Nhìn bảng 4.2 và hình 4.3 ta thấy sai số huấn luyện và thời gian huấn luyện của thuật toán QHDH là tốt nhất. Cụ thể với thuật toán QHDH chỉ mất 18 giây đã có sai số trung trình là 5.04E-06, còn với thuật toán QTH là 48 giây có sai số là 2.82E-04 lớn hơn rất nhiều.

Bảng 4.2: So sánh sai số và thời gian huấn luyện của các thuật toán QHDH, HDH, QTL và QTH với 1331 mốc của hàm 3 biến.

Mốc kiểm tra			Giá trị hàm	Thuật toán QHDH q=0.9, σ=0.5335655 Thời gian=18”		Thuật toán HDH q=0.9, α=0.9, σmax=0.3 Thời gian =35’’		Thuật toán QTL σ = 0.07215459 Với 140 vòng lặp SSE=0.00174 Thời gian =46’’		Thuật toán QTH σ=0.137050611 với 140 vòng lặp SSE=0.001552 Thời gian =48’’	
x_1	x_2	x_3		Giá trị nội suy	Sai số	Giá trị nội suy	Sai số	Giá trị nội suy	Sai số	Giá trị nội suy	Sai số
1.5	0.4	0.5	2.84630009	2.8463029	2.77E-06	2.8463	3.52E-06	2.84614	1.63E-04	2.84598	3.22E-04
0.6	0.4	1	1.81946318	1.8194664	3.23E-06	1.81945	8.66E-06	1.8193	1.59E-04	1.81961	1.43E-04
2.1	1.2	1	6.23362586	6.2336304	4.57E-06	6.23362	5.98E-06	6.23323	3.99E-04	6.23417	5.48E-04
1.5	0.8	1.5	2.64225431	2.6422579	3.57E-06	2.64225	5.81E-06	2.64204	2.17E-04	2.64235	9.35E-05
2.1	0.8	2	3.91614211	3.9161457	3.54E-06	3.91613	7.78E-06	3.91584	3.05E-04	3.91637	2.26E-04
1.5	0.8	2.5	1.88383406	1.8838388	4.76E-06	1.88382	9.58E-06	1.88367	1.64E-04	1.88362	2.16E-04
1.5	1.2	3	2.81654534	2.8165506	5.23E-06	2.81654	9.39E-06	2.81631	2.33E-04	2.81609	4.55E-04
0.9	1.2	3.5	1.42131446	1.4213279	1.35E-05	1.4213	1.05E-05	1.42125	6.50E-05	1.42091	4.01E-04
1.2	2	3.5	4.09511999	4.0951257	5.68E-06	4.09511	8.23E-06	4.0948	3.22E-04	4.09488	2.41E-04
0.9	3.6	3.5	4.88588981	4.8858934	3.54E-06	4.88588	9.47E-06	4.88552	3.72E-04	4.88606	1.74E-04
Sai số trung bình					**5.04E-06**		**7.89E-06**		**8.01E-05**		**2.82E-04**

4.3.3. So sánh tính tổng quát

Kết quả thực nghiệm trong bảng 4.3 và hình 4.4 cho hàm 3 biến (như công thức 4.22) với 1331 mốc nội suy, có $N_1=11$, $h_1=0.3$; $N_2=11$; $h_2=0.4$; $N_3=11$, $h_3 = 0.5$. Sau khi huấn luyện, chúng tôi lấy 10 điểm ngẫu nhiên xa tâm mốc nội suy để tính toán và so sánh.



Hình 4.4: So sánh tính tổng quát của mạng huấn luyện bởi các thuật toán QHDH, HDH, QTL và QTH với 1331 mốc của hàm 3 biến.

Nhận xét: Nhìn bảng 4.3 và hình 4.4 thấy rằng tính tổng quát của thuật toán QHDH tốt hơn nhiều với thuật toán khác và thời gian cũng ít hơn. Cụ thể với thuật toán QHDH có sai số trung bình là 0.07993729 trong 18 giây, còn thuật toán QTL là 4.689752 trong 46 giây.

Bảng 4.3: So sánh tính tổng quát của mạng huấn luyện bởi các thuật toán QHDH, HDH, QTL và QTH với 1331 mốc của hàm 3 biến.

Mốc kiểm tra			**Giá trị hàm**	**Thuật toán QHDH q=0.9, σ=0.5335655 Thời gian=18”**		**huật toán HDH q=0.9, α=0.9, σmax=0.3 Thời gian =35’’**		**Thuật toán QTL σ = 0.07215459 Với 140 vòng lặp SSE=0.00174 Thời gian =46’’**		**Thuật toán QTH σ=0.137050611 với 140 vòng lặp SSE=0.001552 Thời gian =48’’**	
x_1	**x_2**	**x_3**		**Giá trị nội suy**	**Sai số**	**Giá trị nội suy**	**Sai số**	**Giá trị nội suy**	**Sai số**	**Giá trị nội suy**	**Sai số**
1.65	0.6	0.75	3.34497335	3.370238053	0.0252647	3.39041135	0.045438	0.0001625	3.34481	0.787788	2.55719
1.35	2.2	0.75	4.28631188	4.310879376	0.0245675	4.29900687	0.012695	0.0002138	4.2861	1.0316	3.25471
2.25	1.4	1.25	7.60071335	7.524131051	0.0765823	7.52335295	0.077361	0.0003744	7.60034	1.8153	5.78541
1.65	1	1.75	3.15093868	3.196305881	0.0453672	3.20572068	0.054782	0.0001566	3.15078	0.757311	2.39363
1.05	2.6	1.75	3.06297984	3.110876244	0.0478964	3.12253884	0.059559	0.0001546	3.06283	0.74289	2.32009
2.25	1	2.25	5.16751064	5.527266742	0.3597561	5.74917064	0.58166	0.000256	5.16725	1.24075	3.92676
1.65	1.4	3.25	4.21978442	4.298019119	0.0782347	4.37204441	0.15226	0.0002094	4.21957	1.01239	3.20739
1.35	2.2	3.75	5.62798613	5.677770128	0.049784	5.68750142	0.059515	0.0002761	5.62771	1.33495	4.29303
1.05	3.8	3.75	5.95690116	6.011258157	0.054357	6.00281715	0.045916	0.0002936	5.95661	1.42524	4.53166
2.55	0.6	4.25	4.48173598	4.519298982	0.037563	4.52936198	0.047626	0.0002209	4.48152	1.06504	3.4167
Sai số trung bình					**0.07993729**		**0.113681**		**4.689752**		**3.568657**

4.4. Nhận xét chung về thuật toán một pha mới

Cho dù thuật toán hai pha HDH đã cải thiện đáng kể chất lượng của mạng, nhưng trong trường hợp các mốc nội suy cách đều nhau, thì thuật toán này chưa khai thác được ưu điểm phân bố đều của các mốc nội suy. Theo thuật toán HDH việc xác định giá trị hàm cơ sở bán kính là giống nhau với những điểm có khoảng cách đến tâm bằng nhau. Điều này không thực sự phù hợp với những tập dữ liệu có độ lớn của mỗi chiều quá chênh lệnh. Khắc phụ nhược điểm đó, thay cho chuẩn Euclide chúng tôi dùng chuẩn Mahalanobis để xác định giá trị của hàm cơ sở bán kính Gauss. Khi đó các tham số độ rộng bán kính sẽ được xác định trước, và sử dụng pha 2 của thuật toán HDH để huấn luyện mạng. Lúc này, thuật toán hai pha HDH được cải tiến thành thuật toán một pha. Thực nghiệm cho thấy thuật toán mới một pha không những giảm đáng kể thời gian huấn luyện mà tính tổng quát của mạng cũng tốt hơn. Thuật toán này thực sự có ý nghĩa cho bài toán có mốc nội suy cách đều nhau và chạy tốt với số mốc lớn.

CHƯƠNG 5. MẠNG RBF ĐỊA PHƯƠNG

Thuật toán lặp hai pha mới đề xuất trong chương trước đã rút ngắn đáng kể thời gian huấn luyện mạng nơron nội suy RBF. Tuy nhiên thời gian huấn luyện mạng tăng rất nhanh khi số mốc tăng nên khó sử dụng mạng trong các bài toán thường xuyên có dữ liệu bổ sung trong thời gian thực. Đến nay, chưa có một phương pháp hiệu quả nào để xấp xỉ hàm nhiều biến hoặc phân lớp mẫu cho các bài toán thời gian thực đòi hỏi thời gian huấn luyện ngắn, đặc biệt với các bài toán động. Mạng nội suy RBF địa phương giới thiệu trong bài này là kết hợp cách tiếp cận mạng nơron nhân tạo và phương pháp học dựa trên mẫu (instance-based learning) để giải quyết vấn đề mở đó. Chúng tôi dùng ý tưởng của nội suy Spline: nội suy bằng các dạng hàm đơn giản trên từng đoạn con, sau đó ghép trơn các hàm này thành hàm nội suy. Trong mạng này, các dữ liệu huấn luyện được phân cụm dựa trên miền xác định thành các cụm con có cỡ đủ nhỏ và dùng thuật toán lặp huấn luyện mạng nội suy RBF vừa đề xuất để huấn luyện mạng trên mỗi cụm con.

Chương này trình bày như sau. Mục 5.1 đưa ra nhận xét gợi mở ý tưởng xây dựng mạng địa phương. Kiến trúc mạng và các thuật toán huấn luyện được trình bày ở mục 5.2. Mục 5.3 giới thiệu một chứng minh cho tính tổng quát của mạng này, mô hình cho bài toán động và các kết quả thực nghiệm được giới thiệu trong các mục 5.3 và 5.4. Các nhận xét chung được đưa ở mục 5.6.

Các kết quả chính của chương này được công bố trong hội thảo quốc tế của IEEE [20], và tạp chí quốc tế *International Journal of Data Mining, Modelling and Management Science (IJDMMM)*[21].

5.1. Giới thiệu

Như đã nói trong các chương trước, các mạng nơron nhân tạo như mạng MLP, mạng RBF và các phương pháp học dựa trên mẫu như k-lân cận gần nhất và hồi quy địa phương có trọng số, đang là những giải pháp thông dụng để xấp xỉ hàm

nhiều biến và phân lớp mẫu. Các mạng nơron đòi hỏi nhiều thời gian huấn luyện khi cỡ mẫu huấn luyện lớn và khi có mẫu mới bổ sung cũng mất nhiều thời gian học lại, còn các phương pháp học dựa trên mẫu dựa vào thông tin của mẫu mới nên không học trước theo tập dữ liệu huấn luyện được. Vì vậy, với các bài toán thời gian thực đòi hỏi thời gian huấn luyện ngắn, đặc biệt với những bài toán động (khi thường xuyên có dữ liệu huấn luyện mới được bổ sung) thì các phương pháp này khó áp dụng và cũng chưa có một phương pháp mới nào thích hợp cho chúng.

Trong các phương pháp này, mạng RBF được xem là cầu nối giữa mạng MLP và các phương pháp học dựa trên mẫu với ưu điểm chính là:

1) Thời gian học nhỏ hơn nhiều so với mạng MLP.

2) Mặc dù các hàm cơ sở bán kính có ảnh hưởng địa phương nhưng việc học của mạng không phụ thuộc vào mẫu mới như các phương pháp dựa trên mẫu.

Khi cỡ mẫu nhỏ, người ta thường dùng mạng nội suy RBF, trong đó các mốc nội suy được dùng làm tâm của các hàm cơ sở bán kính. Trường hợp cỡ mẫu lớn thì dùng mạng xấp xỉ RBF với số hàm cơ sở bán kính ít hơn số mốc nội suy. Nhưng mạng này có sai số lớn, khó tìm tâm thích hợp cho mỗi hàm cơ sở và huấn luyện lại khi có dữ liệu bổ sung. Chính vì vậy nó rất khó ứng dụng cho các bài toán thời gian thực đã nêu.

Thuật toán lặp hai pha HDH huấn luyện mạng được giới thiệu trong chương trước có nhiều ưu điểm như: cải thiện đáng kể thời gian huấn luyện mạng, đạt được sai số bé, dễ ước lượng sai số, dễ song song hóa để giảm thời gian tính toán và đặc biệt là tốn ít thời gian huấn luyện lại khi bổ sung thêm các dữ liệu huấn luyện mới. Tuy vậy, thời gian huấn luyện của thuật toán tăng rất nhanh khi số mốc nội suy tăng. Đặc tính này gợi nên ý tưởng phân miền dữ liệu thành các miền con chứa mốc nội suy gần bằng nhau và không vượt quá M cho trước, rồi xây dựng mạng nội suy RBF trên mỗi cụm con này. Mạng như vậy được gọi mạng nội suy RBF địa phương. Việc chia miền con có thể thực hiện nhờ cải tiến thuật toán phân cụm nhờ cây k-d [12,17,23]. Thuật toán HDH huấn luyện mạng trên mỗi cụm con thực hiện rất nhanh

và thực nghiệm cho thấy tính tổng quát của mạng địa phương tốt hơn mạng toàn cục. Đối với các bài toán động, nếu có thêm các mốc nội suy mới trong thời gian hoạt động của mạng, ta chỉ cần huấn luyện tăng cường mạng trên miền con chứa nó nhờ thuật toán HDH. Thời gian huấn luyện tăng cường này tốn rất ít so với huấn luyện từ đầu. Các đánh giá toán học và kết quả thực nghiệm cho thấy loại mạng này có nhiều ưu điểm và thích hợp cho các bài toán thời gian thực.

Bảng 5.1 giới thiệu kết quả thực nghiệm với hàm 3 biến như công thức (4.22) có các mốc nội suy trên hình hộp [0,3]x[0,2]x[0,1]với sai số ε=10^{-6} và q = 0.9; $\alpha = 0.9$. Thực hiện trên máy Intel Pentium IV, Processor 2.2GHz, 256MB DDR RAM.

Bảng 5.1: Thời gian huấn luyện mạng với hàm 3 biến với $\varepsilon=10^{-6}$, $q=0.9$; $\alpha=0.9$.

Số mốc	**Thời gian tính bán kính σ**	**Thời gian huấn luyện tính W**	**Tổng thời gian**
100	1''	1''	2''
500	8''	1''	9''
1000	39''	2''	41''
2000	3'31	5''	3'36

Nhìn bảng 5.1 cho thấy thời gian huấn luyện ở pha thứ nhất tăng rất nhanh khi số mốc tăng. Để giảm thời gian huấn luyện và sử dụng cho các bài toán thời gian thực hoặc bài toán động ta có thể dùng mạng RBF địa phương.

5.2. Mạng RBF địa phương

5.2.1. Kiến trúc và thủ tục xây dựng mạng

Giả sử tập mốc nội suy $\{x^k\}_{k=1}^N$ nằm trong miền đóng giới nội D=$\prod_{i=1}^{n}[a_i,b_i]$ và N lớn. Ta chọn trước số nguyên dương M cho số điểm trong mỗi cụm con và chia miền D thành các hình hộp n chiều D_j *(j=1,2,...,k)* với số mốc trong mỗi cụm nhỏ hơn M theo thuật toán phân cụm nhờ cây *k-d* giới thiệu trong mục

sau[12,17,23]. Sau đó sử dụng thuật toán lặp để huấn luyện các mạng RBF cho mỗi miền con D_j. Xây dựng thủ tục để xác định mỗi x trong D thuộc miền con D_j và mạng RBF địa phương sẽ là kết nối giữa thủ tục này với các mạng RBF con. Với mỗi dữ liệu mới thuộc D_j thì chỉ có mạng nội suy địa phương của miền D_j phải huấn luyện lại. Khi bổ sung dữ liệu mới, nếu số mốc nội suy trong miền con lớn hơn M thì thuật toán cây *k-d* sẽ được sử dụng để phân chia thành hai cụm có kích cỡ nhỏ hơn. Cụ thể, thủ tục xây dựng mạng như sau.

Procedure Xây dựng mạng RBF địa phương;

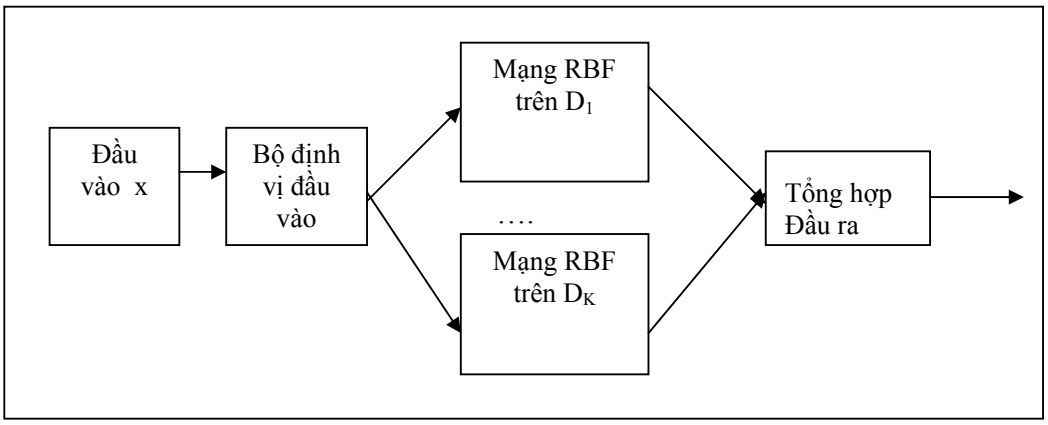
Begin

1. Phân D thành các miền con $D_1,..,D_k$; // sử dụng thuật toán phân cụm cây *k-d* để số mốc trong mỗi cụm con không vượt quá M.
2. Xây dựng bộ định vị đầu vào cho các mạng RBF con.
3. Huấn luyện các mạng RBF con; // Dùng thuật toán HDH.
4. Kết nối bộ định vị đầu vào với các mạng con để được mạng RBF địa phương.

End;

Hình 5.1 Thủ tục xây dựng mạng RBF địa phương

Kiến trúc mạng được mô tả trong hình 5.2. Trong pha huấn luyện, thành phần *tổng hợp đầu ra* không hoạt động. Mỗi mạng con được huấn luyện độc lập. Trong khi nội suy, với mỗi giá trị vào x, *bộ định vị* sẽ xác định mạng con tương ứng, còn tất cả các mạng con khác có giá trị vào rỗng. Và giá trị đầu ra của tất cả các mạng con đều là rỗng ngoại trừ mạng con có đầu vào x. Sau đó thành phần *tổng hợp đầu ra* là tổng đầu ra của các mạng RBF con.



Hình 5.2. Mô hình kiến trúc mạng RBF địa phương

Với cấu trúc mạng RBF địa phương mới này, hàm nội suy khả vi liên tục trên từng miền con. Ở phần tiếp theo sẽ chỉ ra rằng khi *M* nhỏ thì thời gian huấn luyện và huấn luyện lại cũng như sai số giảm. Nhưng nếu số mạng con RBF nhiều thì sẽ làm cho mạng phức tạp hơn và các miền khả vi sẽ giảm. Đặc điểm này chính là điều không mong muốn trong giải tích số. Như thế, việc lựa chọn *M* như thế nào để cân bằng giữa các điều kiện đó.

Ở bước 1 của thuật toán, có thể xác định số miền con thay cho chọn trước ngưỡng *M* cho tập mốc của các miền con.

5.2.2. Thuật toán phân cụm nhờ cây k-d.

Cây K-d được Bentley giới thiệu đề xuất trong [11] và các biến thể của nó là công cụ hiệu quả để phân cụm nhanh dữ liệu [12,17,23]. Thuật toán sau đây sửa đổi nhỏ kỹ thuật *k-d* trong [11,12], để phân hình hộp *n*-chiều *D* chứa *N* đối tượng dữ liệu thành các hình hộp con $D_1,..., D_k$ sao cho mỗi hình hộp D_j chứa không quá *M* đối tượng dữ liệu.

Thuật toán có thể mô tả bởi quá trình xây dựng một cây nhị phân *n*-chiều có gốc là hình hộp *D* chứa *N* đối tượng dữ liệu, mỗi nút ở độ sâu *k* là hình hộp *n*-chiều có được nhờ một nhát cắt trực giao với cạnh lớn nhất của hình hộp cha chia thành hai hình hộp con để chúng chứa số lượng dữ liệu xấp xỉ nhau.

Thủ tục chia đôi hình hộp *n*-chiều $D_j = \prod_{i=1}^{n} [a_i^j, b_i^j]$ như hình 5.3 sau:

Procedure chia đôi hình hộp *n*-chiều $D_j = \prod_{i=1}^{n} [a_i^j, b_i^j]$

Begin

Bước 1. Chọn *i* sao cho cạnh $[a_i^j, b_i^j]$ của D_j là lớn nhất // Chọn phương cho mặt cắt.

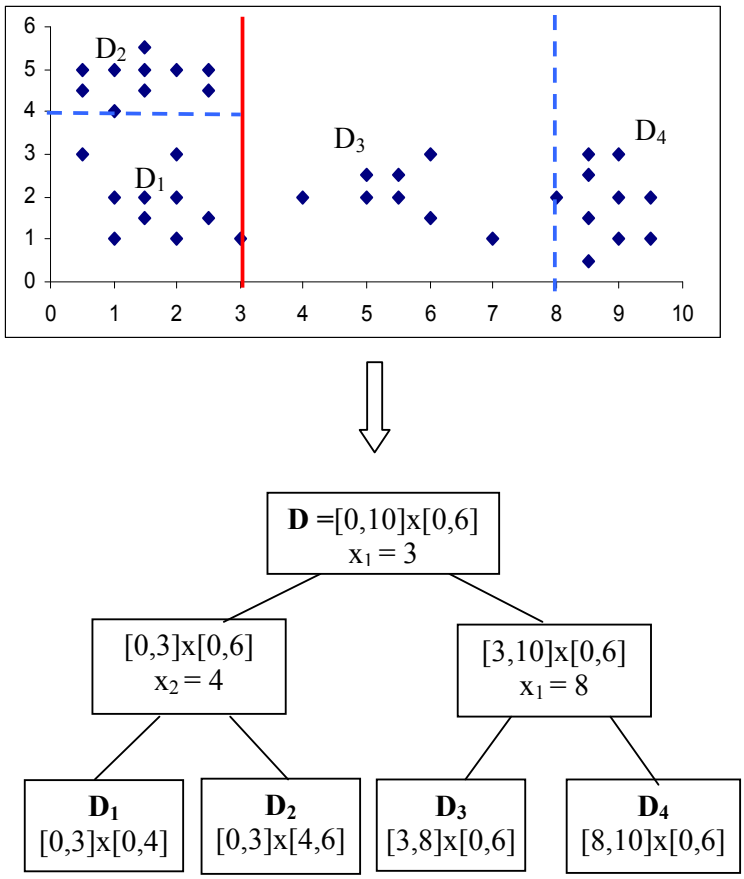
Bước 2. Chia cạnh $[a_i^j, b_i^j]$ bởi nhát cắt trực giao và qua ít nhất một điểm dữ liệu. Sao cho hai hình hộp nhận được từ nó chứa số đối tượng dữ liệu bằng nhau với quy ước các điểm dữ liệu thuộc nhát cắt có thể đồng thời tính là thuộc hai hình hộp.

Thủ tục chia đôi kết thúc khi số lượng đối tượng trong mỗi hình hộp con không vượt quá *M* chọn trước. Như vậy số mốc nội suy trong mỗi hình hộp con ở độ sâu *d* được xem là bằng nhau, và xấp xỉ bằng $\frac{N}{2^d}$ do có những mốc có thể được tính vào nhiều hình hộp. Khi thủ tục kết thúc thì số lượng dữ liệu trong mỗi miền con thuộc khoảng $\left[\frac{M}{2}, M\right]$ và có khoảng từ $\frac{N}{M}$ đến $2\frac{N}{M}$ miền con.

End;

Hình 5.3 Thủ tục chia đôi hình hộp n-chiều

Hình 5.4 mô tả kết quả chia hình hộp 2 chiều [0,10]x[0,6] chứa 38 dữ liệu (*N*=38) với *M*=10. Kết quả có 4 hình hộp con đều chứa 10 dữ liệu, trong đó các điểm (1,4) và (8,2) được tính đồng thời ở hai hình hộp chứa nó còn điểm (3,1) chỉ thuộc D_3.



Hình 5.4: Cây K-D mô tả tập dữ liệu trong không gian 2 chiều, với N=38, M=10.

Như đã nhận xét ở mục trên, có thể chọn trước số miền con làm điều kiện dừng ở bước 2.

5.2.3. Độ phức tạp thuật toán huấn luyện mạng

Các tính toán huấn luyện mạng chủ yếu gồm thực hiện phân miền dữ liệu nhờ cấu trúc cây *k-d* và huấn luyện mạng RBF trên các hình hộp con. Quá trình xây dựng cây *K-d* ở đây tương tự như thuật toán đã đánh giá trong [12,17,23], có độ phức tạp là *O(NlogN)* với *N* là số mốc nội suy. Thời gian huấn luyện mỗi mạng RBF trên mỗi miền con là $O((T+c)nM^2)$ với *n* là số chiều của mốc nội suy, *M* là số mốc nội suy trong mỗi cụm. Nếu huấn luyện các cụm con được thực hiện tuần tự thì độ phức tạp tính toán để huấn luyện các mạng RBF là *O((T+c)nMN)*. Như đã được

chỉ ra trong [12,17,23]. Nhờ cấu trúc cây *k-d*, khi bổ sung dữ liệu mới, độ phức tạp tính toán để tìm ra cụm thích hợp cho nó là $O\left(\log\frac{N}{M}\right)$.

5.3. Tính xấp xỉ tổng quát của mạng nội suy RBF địa phương

Tính xấp xỉ tổng quát của mạng nội suy RBF đã được Hartman và Park chứng minh trong [29,47] với tham số độ rộng bán kính tuỳ ý. Trong thuật toán HDH, các tham số này bị hạn chế để $q_k \in [\alpha q, q]$ nên mặc dù không ảnh hưởng nhiều tới việc áp dụng nhưng tính tổng quát của mạng được xây dựng là không khẳng định được. Định lý dưới đây cho ta đánh giá tính xấp xỉ tổng quát của mạng nội suy RBF địa phương.

Để phát biểu định lý, ta cần đến khái niệm δ-lưới của miền *D*. Tập mốc gọi là δ-lưới trên *D* nếu $\forall x \in D$ tồn tại mốc x^i sao cho $d(x,x^i)<\delta$, trong đó $d(x,x^i)$ là khoảng cách sinh bởi chuẩn xác định bởi (5.1)

$$\|u\|_* = \max_{j\le N}\{|u_j|\} \tag{5.1}$$

Định lý 5.1. (tính xấp xỉ tổng quát của mạng nội suy RBF địa phương).

Giả sử *f* là hàm liên tục trên miền *D* và *M* là số phần tử cực đại trong mỗi cụm con cho trước. Khi đó với mọi $\varepsilon>0$ tồn tại $\delta>0$ sao cho với bất kỳ tập dữ liệu δ-lưới nào trên *D*, thì một mạng địa phương huấn luyện đủ tốt với tập dữ liệu này sẽ thỏa mãn:

$$\forall x \in D, \qquad |f(x)-\varphi(x)| < \varepsilon \tag{5.2}$$

$\varphi(x)$ là hàm nội suy cần tìm.

Chứng minh. Với hàm *f* liên tục trên *D* và ε đã xác định, ta phải chứng minh rằng tồn tại δ sao cho nếu tập mốc nội suy là δ-lưới thì với mọi $x \in D$ ta phải có bất đẳng thức (5.2).

Thực vậy, vì f liên tục trên D nên f liên tục đều trên đó và tồn tại δ_1 để ∀x,y mà $d(x,y) < \delta_1$ thì:

$$|f(x) - f(y)| < \frac{\varepsilon}{3} \tag{5.3}$$

Mặt khác từ (3.11) ta thấy với mỗi miền D_i hàm φ có dạng:

$$\varphi(x) = \sum_{k=1}^{M_i} w_k \varphi_k(x) + w_0 \tag{5.4}$$

Trong đó M_i là số mốc thuộc cụm D_i còn $\varphi_k(x)$ là giá trị hàm bán kính của mốc x^k trong tập D_i (đã được đánh số lại). Dễ thấy rằng:

$$|\varphi_k(x)| \le 1; \forall x, k \tag{5.5}$$

Chú ý tới (5.1) ta có:

$$\forall x, y \in D_i, \quad |\varphi(x) - \varphi(y)| = \left| \sum_{k}^{M_i} w_k [\varphi_k(x) - \varphi_k(y)] \right| \le 2\|W\|_* \tag{5.6}$$

Mặt khác, vì $M_i \le M$ cho trước nên tồn tại δ_2 đủ nhỏ để khi tập mốc là δ_2-lưới thì đường kính của mỗi tập D_i đủ nhỏ sao cho độ lệch của hàm f trên mỗi tập D_i so với giá trị trung bình trên đó nhỏ hơn (1-q)$\frac{\varepsilon}{6}$. Chú ý tới các biểu thức (3.13),(3.19) và thủ tục hình 3.4 ta có:

$$\|W\|_* < \left\| \frac{1}{1-q} \|Z\|_* \right\| \le \frac{\varepsilon}{6} \tag{5.7}$$

Nếu chọn sai số huấn luyện để với mọi mốc nội suy x^k ta đều có :

$$|f(x^k) - \varphi(x^k)| < \frac{\varepsilon}{3} \tag{5.8}$$

Bây giờ ta đặt $\delta = \min\{\delta_1, \delta_2\}$ và xét $x \in D$ tuỳ ý, khi đó x $\in D_k$ và có mốc nội suy x^i trong D_k để $d(x,x^i) < \delta$. Từ các biểu thức (5.3)(5.6)(5.8) ta có ước lượng:

$$|f(x) - \varphi(x)| = |f(x) - f(x^i) + f(x^i) - \varphi(x^i) + \varphi(x) - \varphi(x^i)| < \varepsilon \tag{5.9}$$

Định lý được chứng minh.

Nhận xét. Các biểu thức (3.11) và (5.7) cho ta thấy hàm nội suy dao động quanh giá trị trung bình của các mốc nội suy và các hàm có biên độ dao động bé trên mỗi miền con sẽ hứa hẹn có tính tổng quát tốt hơn.

5.4. Bài toán động

Bài toán động là các bài toán sau khi huấn luyện xong thường được bổ sung các mốc nội suy mới. Đối với các bài toán này, khi có các mốc nội suy mới, ta kiểm tra lại số lượng mốc trong hình hộp con tương ứng, nếu vượt quá *M* mốc thì ta thực hiện tách đôi hình hộp con này và huấn luyện lại mạng RBF địa phương tương ứng. Nhờ cấu trúc cây *k-d*, việc bổ sung dữ liệu vào cụm thích hợp mất rất ít thời gian và với cách xử lý đã nêu ta không phải huấn luyện lại toàn mạng. Kết quả thực nghiệm ở mục sau cho thấy thời gian huấn luyện tăng cường khi có dữ liệu bổ sung rất ngắn.

5.5. Kết quả thực nghiệm

Thực nghiệm nhằm so sánh thời gian huấn luyện, sai số huấn luyện và tính tổng quát của mạng địa phương so với mạng nội suy RBF toàn cục (ở đây dùng từ toàn cục để phân biệt với mạng địa phương). Về sai số huấn luyện, ở điều kiện kết thúc của thuật toán HDH ở biểu thức (3.23) ta xác định sai số cho vectơ *W* mà chưa biết sai số của hàm *f* nên cần so sánh hiệu quả của hai mạng. Ngoài ra thực nghiệm cũng so sánh thời gian huấn luyện tăng cường bằng thuật toán HDH khi có dữ liệu mới và thời gian huấn luyện từ đầu để dùng cho bài toán động.

Các hàm được xét là hàm 2 biến như công thức (5.10) với $x_1 \in [0,8]$, $x_2 \in [0,10]$:

$$y = \frac{(x_1^2 + x_1 x_2)}{5} + \frac{x_2}{3} + \frac{1}{2} \tag{5.10}$$

và hàm ba biến như công thức (5.11) với $x_1 \in [0,3]$, $x_2 \in [0,4]$, $x_3 \in [0,5]$:

$$y = x_1^2 x_2 + \sin(x_2 + x_3 + 1) + 4 \tag{5.11}$$

Việc chọn hàm hai biến nhằm mục đích làm cho đường kính miền xác định giảm nhanh khi tăng số cụm như đã nhận xét ở mục trên, còn hàm ba biến chọn dạng phức tạp hơn để kiểm tra chúng khi số cụm giảm mà không quan tâm tới số lượng đối tượng trong cụm con. Vì đã đánh giá độ phức tạp thuật toán theo số chiều nên các ví dụ ở đây đủ để kiểm tra hiệu quả mạng.

Thực nghiệm thực hiện trên máy Intel Pentium IV Processor, 3.0GHz, 512MB DDR RAM. Các tham số chọn thống nhất với $q=0.8$, $\alpha=0.9$ và sai số cho điều kiện dừng là $\varepsilon=10^{-6}$.

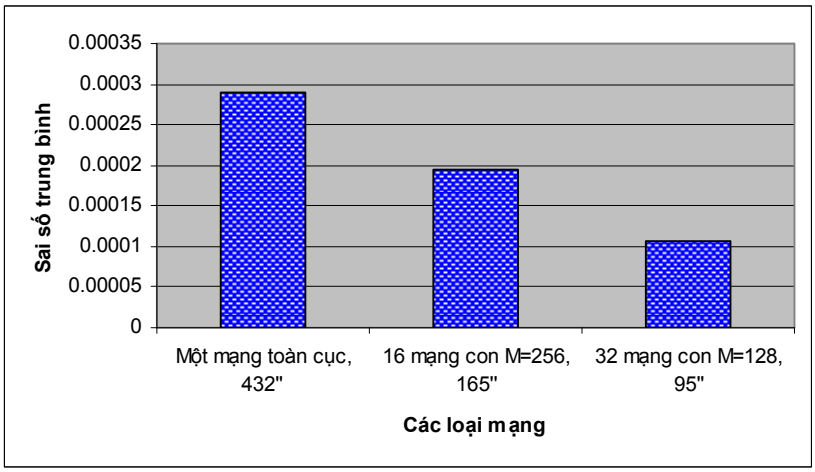
Việc huấn luyện các mạng RBF địa phương thực hiện tuần tự, nếu xử lý song song thì thời gian huấn luyện sẽ nhanh hơn.

5.5.1. So sánh thời gian và sai số huấn luyện

Kết quả thực nghiệm trình bày ở bảng 5.2, hình 5.5 cho hàm 2 biến, bảng 5.3, hình 5.6 tương ứng cho hàm 3 biến. Các mạng sau khi được huấn luyện sẽ lấy 10 mốc nội suy để so sánh sai số huấn luyện. Hàm 2 biến với 4096 mốc nội suy cách đều nhau, còn hàm 3 biến với 19683 mốc nội suy cách đều nhau. Trong bảng 5.2 cho trước số mốc trong từng cụm M còn Bảng 5.3 thì điều kiện dừng là số cụm con chọn trước.

Bảng 5.2: So sánh thời gian và sai số huấn luyện của hàm 2 biến có 4096 mốc nội suy

Điểm kiểm tra		**Giá trị hàm gốc**	**Một mạng toàn cục, Thời gian=432''**		**Chia 16 cụm M=256 Thời gian=165''**		**Chia 32 cụm M=128 Thời gian=95''**	
X_1	**X_2**		**Giá trị nội suy**	**Sai số**	**Giá trị nội suy**	**Sai số**	**Giá trị nội suy**	**Sai số**
0.126984	2.222222	1.3004031	1.3003962	0.0683E-04	1.3004014	0.0175E-04	1.300402	0.00676E-04
0.380952	4.444444	2.3491307	2.3491123	0.1837E-04	2.349125	0.0574E-04	2.349126	0.04706E-04
4.571429	3.333333	8.8383219	8.8382503	0.716E-04	8.8382803	0.417E-04	8.838289	0.3288E-04
2.031746	4.285714	4.4956664	4.4956489	0.1743E-04	4.4956643	0.0208E-04	4.495665	0.01487E-04
3.936508	4.285714	8.4019400	8.4014976	4.424E-04	8.4016351	3.05E-04	8.401831	1.0919E-04
5.333333	1.746032	8.6333333	8.6331865	1.467E-04	8.6332876	0.457E-04	8.633297	0.3674E-04
6.47619	8.571429	22.847392	22.846471	9.21E-04	22.846686	7.06E-04	22.84688	5.1614E-04
7.111111	8.888889	26.2185185	26.2182106	3.079E-04	26.21831	2.08E-04	26.21837	1.5116E-04
7.619048	9.047619	28.9126984	28.9124999	1.985E-04	28.9125	1.99E-04	28.91263	0.6698E-04
8	6.666667	26.1888888	26.1881325	7.56E-04	26.188449	4.39E-04	26.18874	1.444E-04
Sai số trung bình				**2.887E-04**		**1.95E-04**		**1.0644E-04**



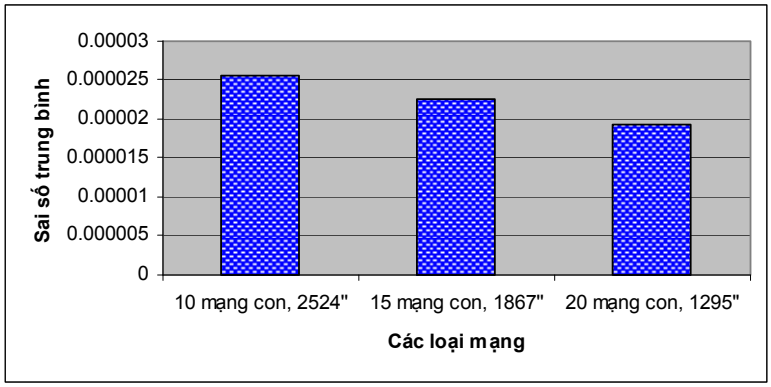
Hình 5.5: Đồ thị so sánh thời gian và sai số huấn luyện của hàm 2 biến có 4096 mốc.

Nhìn bảng 5.2 và hình 5.5 kiểm tra với 4096 hàm hai biến, ta thấy thời gian huấn luyện của "mạng toàn cục" là 432 giây, lớn hơn hẳn so với "mạng địa phương khi *M*=265" là 165 giây, và khi *M*=128 là 95 giây. Cũng trong bảng này sai số huấn luyện giảm dần từ 2.887E-04 xuống còn 1.95E-04 và 1.0644E-04.

Như vậy mạng cục bộ gồm 23 mạng con là tốt nhất về thời gian và sai số huấn luyện.

Bảng 5.3: So sánh thời gian và sai số huấn luyện của hàm 3 biến có 19683 mốc nội suy

Điểm kiểm tra			**Giá trị hàm gốc**	**Số cụm 10 Thời gian =2524''**		**Số cụm 15 Thời gian =1867''**		**Số cụm 20 Thời gian =1295''**	
X_1	**X_2**	**X_3**		**Giá trị nội suy**	**Sai số**	**Giá trị nội suy**	**Sai số**	**Giá trị nội suy**	**Sai số**
0.807692	0.153846	0.192308	5.075238	5.0752268	0.111E-04	5.075236	0.0169E-04	5.075237	0.0129E-04
2.076923	3.846154	0.576923	19.83289	19.8328513	0.41E-04	19.83287	0.248E-04	19.83287	0.265E-04
0.461538	1.230769	1.346154	3.840466	3.8404115	0.541E-04	3.840405	0.611E-04	3.84042	0.457E-04
1.269231	1.076923	1.923077	4.978063	4.97803538	0.279E-04	4.978033	0.302E-04	4.978039	0.239E-04
0.576923	0.461538	2.5	3.42251	3.42248839	0.213E-04	3.422489	0.209E-04	3.422493	0.163E-04
0.115385	0.153846	3.076923	3.115802	3.11579124	0.112E-04	3.115792	0.101E-04	3.115786	0.16E-04
0.230769	1.538462	3.461538	3.802514	3.8025003	0.14E-04	3.802501	0.132E-04	3.8025	0.146E-04
1.846154	3.846154	3.846154	17.77749	17.7774338	0.593E-04	17.77744	0.514E-04	17.77746	0.356E-04
2.192308	3.384615	4.230769	20.99105	20.9910433	0.0795E-04	20.99105	0.0469E-04	20.99105	0.0477E-04
0.576923	3.384615	4.807692	5.356918	5.35691083	0.0739E-04	5.35691	0.0819E-04	5.356909	0.095E-04
Sai số trung bình					**0.255E-04**		**0.226E-04**		**0.194E-04**



Hình 5.6: So sánh thời gian và sai số huấn luyện của hàm 3 biến có 19683 mốc

Nhìn bảng 5.3 và hình 5.6 kiểm tra với 19683 mốc hàm 3 biến, ta thấy khi số cụm con càng lớn (có nghĩa là số mốc trong mỗi cụm giảm) thì thời gian huấn luyện càng giảm. Cụ thể thời gian huấn luyện khi 10 cụm là 2524 giây lớn hơn hẳn so với trường hợp 15 cụm là 1867 giây và khi 20 cụm là 1295 giây. Cũng trong bảng này sai số huấn luyện giảm dần từ 0.255E-04 xuống còn 0.226E-04 và 0.194E-04.

Như vậy qua các bảng 5.2 và hình 5.5, bảng 5.3 và hình 5.6 ta thấy rằng:

1) Thời gian huấn luyện mạng giảm nhanh khi số cụm tăng, đặc biệt khi cỡ dữ liệu của cụm con thực sự giảm (trường hợp hàm 2 biến). Nếu song song hoá việc huấn luyện cụm con thì thời gian huấn luyện giảm đáng kể.

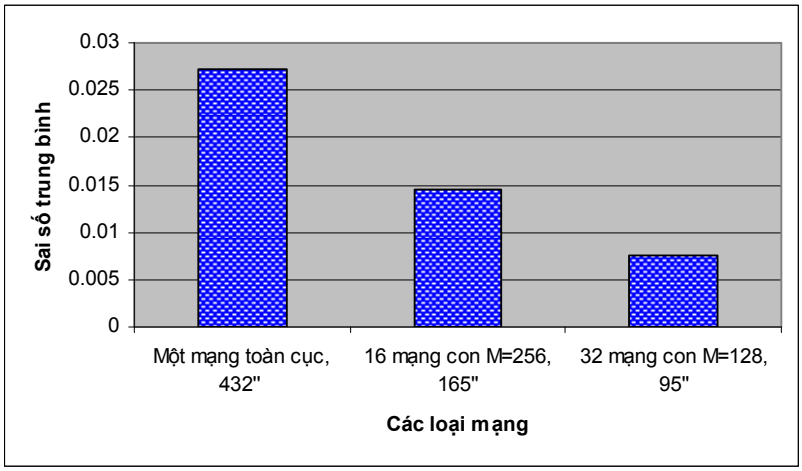
2) Sai số huấn luyện cũng giảm khi số cụm tăng và cũng giảm nhiều hơn khi cỡ dữ liệu của cụm con thực sự giảm.

5.5.2 Tính tổng quát

Kết quả thực nghiệm trình bày ở bảng 5.4 và hình 5.7, bảng 5.5 và hình 5.8 tương ứng cho hàm 2 biến và 3 biến. Các mạng sẽ bỏ đi 10 mốc nội suy để huấn luyện và so sánh giá trị hàm ở các mốc này (là những điểm xa tâm). Hàm 2 biến với 4096 mốc nội suy, còn hàm 3 biến với 19683 mốc nội suy cách đều nhau (kể cả các mốc bỏ đi). Điều kiện dừng khi tính bảng 5.5 là số cụm con chọn trước.

Bảng 5.4. So sánh tính tổng quát với hàm 2 biến có 4086 mốc tại 10 điểm xa tâm

Điểm kiểm tra		**Giá trị hàm gốc**	**Một mạng toàn cục Thời gian =432''**		**Chia 16 cụm M=256 Thời gian =165''**		**Chia 32 cụm M=128 Thời gian =95''**	
X_1	**X_2**		**Giá trị nội suy**	**Sai số**	**Giá trị nội suy**	**Sai số**	**Giá trị nội suy**	**Sai số**
0.126984	2.222222	1.3004031	1.299662	7.41E-04	1.299811	5.92E-04	1.300205	1.98E-04
0.380952	4.444444	2.3491307	2.348308	8.231E-04	2.348199	9.32E-04	2.348599	5.32E-04
4.571429	3.333333	8.8383219	8.837086	12.357E-04	8.83731	10.12E-04	8.83679	15.32E-04
2.031746	4.285714	4.4956664	4.495234	43.257E-04	4.495285	23.81E-04	4.495343	13.23E-04
3.936508	4.285714	8.4019400	8.376309	256.31E-04	8.400373	75.67E-04	8.400987	39.53E-04
5.333333	1.746032	8.6333333	8.631347	198.65E-04	8.632521	171.28E-04	8.63321	81.23E-04
6.47619	8.571429	22.847392	22.83505	123.454E-04	22.83916	92.34E-04	22.84216	62.36E-04
7.111111	8.888889	26.2185185	26.19958	189.353E-04	26.21117	73.45E-04	26.21396	45.63E-04
7.619048	9.047619	28.9126984	28.77724	1354.63E-04	28.85032	623.77E-04	28.88015	325.47E-04
8	6.666667	26.1888888	26.13533	535.615E-04	26.15321	356.75E-04	26.17164	172.53E-04
Sai số trung bình				**272.927E-04**		**144.243E-04**		**76.26E-04**

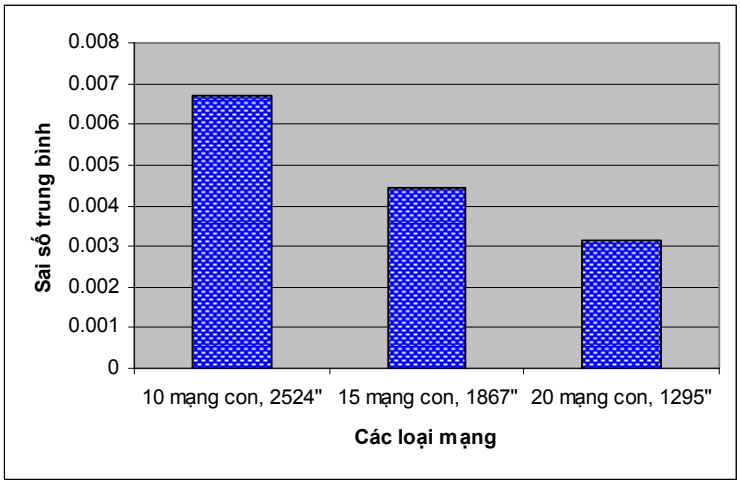


Hình 5.7: So sánh tính tổng quát với hàm 2 biến có 4086 mốc tại 10 điểm xa tâm.

Nhìn bảng 5.4 và hình 5.7 kiểm tra với 4096 hàm hai biến, ta thấy sai số trung bình của "mạng toàn cục" là 271.927E-04 lớn hơn hẳn so với "mạng địa phương khi *M*=265" là 144.243E-04 và khi *M*=128 là 76.26E-04. Như vậy mạng địa phương gồm 32 mạng con là tốt nhất.

Bảng 5.5. So sánh tính tổng quát với hàm 3 biến có 19673 mốc tại 10 điểm xa tâm

Điểm kiểm tra			**Giá trị hàm gốc**	**Số cụm 10 Thời gian =2524''**		**Số cụm 15 Thời gian =1867''**		**Số cụm 20 Thời gian =1295''**	
X_1	**X_2**	**X_3**		**Giá trị nội suy**	**Sai số**	**Giá trị nội suy**	**Sai số**	**Giá trị nội suy**	**Sai số**
0.807692	0.153846	0.192308	5.075238	5.065402	98.357E-04	5.069114	61.24E-04	5.0711023	41.36E-04
2.076923	3.846154	0.576923	19.83289	19.82266	102.35E-04	19.82457	83.26E-04	19.826029	68.63E-04
0.461538	1.230769	1.346154	3.840466	3.836924	35.42E-04	3.83815	23.16E-04	3.8385425	19.23E-04
1.269231	1.076923	1.923077	4.978063	4.976829	12.345E-04	4.977228	8.36E-04	4.9773809	6.82E-04
0.576923	0.461538	2.5	3.42251	3.413817	86.923E-04	3.416657	58.52E-04	3.4179485	45.61E-04
0.115385	0.153846	3.076923	3.115802	3.113437	23.654E-04	3.114587	12.16E-04	3.1147202	10.82E-04
0.230769	1.538462	3.461538	3.802514	3.795283	72.313E-04	3.797301	52.13E-04	3.8008321	16.82E-04
1.846154	3.846154	3.846154	17.77749	17.77584	16.532E-04	17.77625	12.45E-04	17.77624	12.53E-04
2.192308	3.384615	4.230769	20.99105	20.9712	198.52E-04	20.9787	123.52E-04	20.982539	85.12E-04
0.576923	3.384615	4.807692	5.356918	5.354554	23.644E-04	5.356005	9.14E-04	5.3559969	9.21E-04
Sai số trung bình					**67.007E-04**		**44.39E-04**		**31.62E-04**



Hình 5.8: So sánh tính tổng quát với hàm 3 biến có 19673 mốc tại 10 điểm xa tâm.

Nhìn bảng 5.5 và hình 5.8 kiểm tra với 19673 hàm ba biến, ta thấy sai số trung bình của mạng được chia thành 10 mạng con là 67.007E-04 lớn hơn hẳn so với 15 cụm là 44.39E-04 và 20 cụm là 31.62E-04.

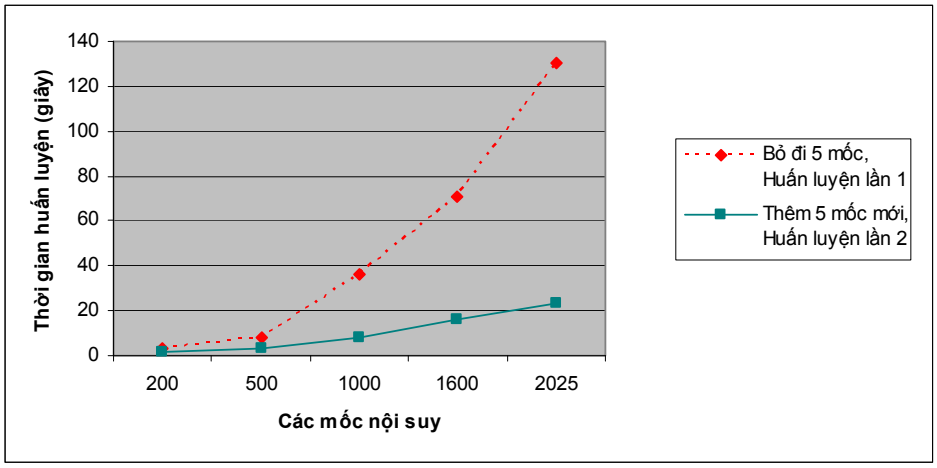
Như vậy nhìn cả các bảng 5.4 và hình 5.7, bảng 5.5 và hình 5.8 ta đều thấy rằng tính tổng quát của mạng tốt hơn khi tăng số cụm, đặc biệt khi cỡ dữ liệu ở cụm con thực sự giảm (trường hợp hàm 2 biến).

5.5.3. Huấn luyện tăng cường ở bài toán động

Thời gian huấn luyện tăng cường (enhenced training time) một mạng cục bộ khi có mốc nội suy bổ sung thực hiện với hàm ba biến như công thức (5.11) với $x_1 \in [0,3]$, $x_2 \in [0,4]$, $x_3 \in [0,5]$. Ta lấy x_1, x_2 cách đều nhau còn x_3 ngẫu nhiên trong khoảng [0,5]. Bỏ đi 5 mốc và huấn luyện số mốc còn lại, sau đó bổ sung thêm 5 mốc mới, lấy giá trị bán kính σ của lần huấn luyện trước làm giá trị khởi tạo cho σ của lần huấn luyện sau. Với 5 mốc mới thêm thì ta khởi gán σ theo như thuật toán HDH. Bảng 5.6 và hình 5.9 cho kết quả để so sánh về thời gian huấn luyện (trường hợp 200 mốc, chỉ tính đến đơn vị 1").

Bảng 5.6. So sánh thời gian huấn luyện tăng cường khi có mốc mới.

Số mốc	Bỏ đi 5 mốc, Thời gian huấn luyện lần 1			Thêm 5 mốc mới, Thời huấn luyện lần 2		
	Pha 1	Pha 2	Tổng	Pha 1	Pha 2	Tổng
200	2”	1”	3”	1”	1”	2”
500	7”	1”	8”	2”	1”	3”
1000	34”	2”	36”	6”	2”	8”
1600	68”	3”	71”	13”	3”	16”
2025	126”	4”	130”	19”	4”	23”



Hình 5.9: Đồ thị so sánh thời gian huấn luyện tăng cường khi có mốc mới.

Nhìn bảng 5.6 và hình 5.9 ta thấy thời gian huấn luyện tăng cường rất nhỏ so với huấn luyện lại, như trường hợp 1600 mốc thời gian huấn luyện lần 1 là 71 giây, nhưng huấn luyện tăng cường lần 2 chỉ là 16 giây. Tương tự cho trường hợp 2025 mốc, thời gian huấn luyện lần 1 là 130 giây, còn lần 2 là 23 giây. Nếu số mốc càng lớn thì sự chênh lệch giữa lần 1 và lần 2 càng nhiều. Ưu điểm này là do thuật toán HDH mang lại. Nhìn bảng 5.6 ta thấy thời gian tính toán của thuật toán phần lớn là thời gian của Pha 1. Mà khi huấn luyện tăng cường lần 2 thì gần như pha 1 không phải tính toán nhiều chỉ tính những mốc mới bổ sung.

Vì đã đánh giá độ phức tạp thuật toán theo số chiều nên chúng tôi dẫn ra thực nghiệm với hàm hai biến, ba biến để đường kính cụm con và do đó biên độ dao động của hàm trên cụm con giảm nhanh khi số cụm con tăng, hàm được chọn cũng nhằm mục đích này. Thực nghiệm cho thời gian huấn luyện tăng cường khi có dữ liệu bổ sung chúng tôi dùng hàm ba biến.

5.6. Nhận xét chung

Ta thấy trong mạng RBF, mỗi hàm bán kính chỉ có ảnh hưởng địa phương nên thông tin từ dữ liệu xa tâm ít ảnh hưởng tới chất lượng mạng nhưng lại làm tăng thời gian tính toán. Với mạng RBF địa phương như trên, thời gian huấn luyện mạng rất nhanh và tính xấp xỉ của mạng cũng tăng. Thuật toán huấn luyện đơn giản và dễ song song hoá.

Loại mạng này thích hợp cho các bài toán thời gian thực, trong đó đòi hỏi thời gian huấn luyện ngắn và đặc biệt thích hợp với các bài toán động, trong đó các mốc nội suy thường xuyên được bổ sung. Ngoài việc sử dụng thuật toán xây dựng cây *k-d* đã nêu để phân miền dữ liệu, ta có thể chia nhanh hình hộp D thành các hình hộp con và sau đó ghép các hình hộp chứa ít dữ liệu hoặc chia các hình hộp chứa nhiều dữ liệu rồi huấn luyện các mạng địa phương để giảm thời gian tính toán.

KẾT LUẬN

Các kết quả đạt được

Trong thời gian qua, mặc dù có những hạn chế về thời gian và điều kiện làm việc, chúng tôi đã hoàn thành mục tiêu luận án. Các kết quả cụ thể đạt được như sau.

1) Đề xuất thuật toán hai pha đơn giản để huấn luyện mạng nội suy RBF. Pha thứ nhất xác định tham số độ rộng bán kính phù hợp với từng mốc nội suy, pha thứ hai dùng phương pháp lặp để tính trọng số tầng ra. Phân tích toán học và thực nghiệm chỉ ra rằng thuật toán luôn hội tụ, thời gian chạy chỉ phụ thuộc vào việc khởi gán giá trị ban đầu q, α, ε, … , phân bố của mốc nội suy và chuẩn của véctơ.

Qua kết quả thực nghiệm ta thấy thuật toán có ưu điểm nổi trội so với các phương pháp thông dụng hiện nay: thời gian huấn luyện mạng rất nhanh kể cả khi số mốc lớn, dễ dàng thực hiện và có hiệu quả cao, đánh giá sai số huấn luyện, điều khiển cân bằng giữa tốc độ hội tụ và tính tổng quát của mạng bằng việc điều chỉnh các tham số. Một ưu việt nữa của thuật toán là các bán kính tầng ẩn có thể huấn luyện độc lập và ở pha hai trọng số tầng ra cũng có thể huấn luyện độc lập, điều này làm cho chúng có thể song song hoá thuật toán.

2) Trong trường hợp các mốc nội suy cách đều nhau, để khai thác được ưu điểm phân bố này chúng tôi dùng metric Mahalanobis và cải tiến thuật toán hai pha thành thuật toán một pha. Nhờ các phân tích toán học, chất lượng mạng nội suy RBF được cải thiện rõ rệt so với mạng huấn luyện bằng thuật toán HDH và các thuật toán huấn luyện nhanh thông dụng. Không những có ưu thế về thời gian huấn luyện và tính tổng quát mà một hiệu quả dẫn xuất của mạng là có thể dùng cho trường hợp số mốc nội suy lớn hơn nhiều so với thuật toán HDH(và do đó với các thuật toán khác).

3) Đề xuất kiến trúc mạng mới, chúng được gọi là mạng RBF địa phương. Với kiến trúc này, thời gian huấn luyện mạng rất nhanh và tính xấp xỉ của mạng

cũng tăng, thuật toán huấn luyện đơn giản và dễ song song hoá. Loại mạng này thích hợp cho các bài toán thời gian thực, trong đó đòi hỏi thời gian huấn luyện ngắn. Đặc biệt, đối với bài toán động, các mốc nội suy thường xuyên được bổ sung thì nhờ kỹ thuật cây *k-d* ta dễ dàng và nhanh chóng tái huấn luyện mạng.

Hướng nghiên cứu tiếp theo

Bài toán nội suy luôn là một bài toán bắt nguồn từ các bài toán thực tế và đang có nhiều lĩnh vực ứng dụng. Việc vận dụng kiến trúc mạng và các thuật toán phải tùy thuộc vào tính đặc thù của từng bài toán, trên cơ sở đã nghiên cứu và hiểu rõ nó, để có thể cài đặt và hiệu chỉnh thích hợp. Theo hướng này, trong thời gian tới chúng tôi tìm hiểu các bài toán thực tế, bắt đầu từ các bài toán đã sử dụng mạng nơron RBF có hiệu quả đến các bài toán mới để nâng cao hiệu quả giải quyết ứng dụng. Bên cạnh đó, nhờ phát triển ứng dụng, chúng tôi hy vọng có các cải tiến và đề xuất các thuật toán, kiến trúc mạng mới thích hợp cho từng loại bài toán được nghiên cứu.

DANH MỤC CÁC CÔNG TRÌNH CÔNG BỐ CỦA TÁC GIẢ

1. Đặng Thị Thu Hiền và Hoàng Xuân Huấn (2008), "Thuật toán một pha huấn luyện nhanh mạng nội suy RBF với mốc cách đều", *kỷ yếu Hội thảo quốc gia các vấn đề chọn lọc của CNTT lần thứ X*, Đại Lải 9/2007, pp. 532-542.
2. Hoàng Xuân Huấn và Đặng Thị Thu Hiền (2006), "Phương pháp lặp huấn luyện mạng nội suy RBF", *kỷ yếu hội thảo quốc gia các vấn đề chọn lọc của CNTT lần thứ VIII*, Hải phòng 2005, pp. 314-323.
3. Dang Thi Thu Hien, H.X. Huan and H.T.Huynh (2009), "Multivariate Interpolation using Radial Basis Function Networks", *International Journal of Data Mining, Modelling and Management Science (IJDMMM)*, Vol.1, No.3, pp.291-309.
4. Dang Thi Thu Hien, H.X. Huan and H.T. Huynh (2008), "Local RBF Neural Networks for Interpolating Multivariate Functions", *Addendum Contributions to the 2008 IEEE International Conference on Research, Innovation and Vision for the Future in Computing & Communication Technologies*, ENST 2008 S 001, pp. 70-75.
5. Hoang Xuan Huan, D.T.T. Hien and H.T. Huynh (2007), "A Novel Efficient Algorithm for Training Interpolation Radial Basis Function Networks", *Signal Processing*, vol. 87, Issue 11, pp. 2708 – 2717.

TÀI LIỆU THAM KHẢO

[1] Lương Mạnh Bá và Nguyễn Thanh Thuỷ (1999), *Nhập môn xử lý ảnh số*, NXB Khoa học và kỹ thuật.

[2] Hoàng Tiến Dũng (2006), *Mạng nơron RBF và ứng dụng*, Luận văn thạc sĩ, Đại học Công nghệ - ĐH Quốc Gia Hà nội.

[3] Đặng Thị Thu Hiền và Hoàng Xuân Huấn (2008), "Thuật toán một pha huấn luyện nhanh mạng nội suy RBF với mốc cách đều", *Kỷ yếu Hội thảo quốc gia các vấn đề chọn lọc của CNTT lần thứ X*, Đại Lải 9/2007, pp. 532-542.

[4] Hoàng Xuân Huấn và Đặng Thị Thu Hiền (2006), "Phương pháp lặp huấn luyện mạng nội suy RBF", *Kỷ yếu hội thảo quốc gia các vấn đề chọn lọc của CNTT lần thứ VIII*, Hải phòng 2005, pp. 314-323.

[5] Hoàng Xuân Huấn (2004), *Giáo trình các phương pháp số*, NXB Đại học quốc gia Hà Nội.

[6] Lê Tấn Hùng và Huỳnh Quyết Thắng (2000), *Kỹ thuật đồ hoạ máy tính*, NXB Khoa học và kỹ thuật.

[7] Lê Tiến Mười (2009), *Mạng neural RBF và ứng dụng nhận dạng chữ viết tay*, Khoá luận tốt nghiệp Đại học, ĐH Công nghệ - ĐH Quốc Gia Hà nội.

[8] R. H. Bartels, John C. Beatty and Brian A. Barsky (1987), *An introduction to Splines for uses in Computer graphics & geometric modeling*, Morgan Kaufmann Publishers, Inc, USA.

[9] B.J.C. Baxter (1992), *The interpolation theory of Radial basis functions*, Ph.D, Cambridge University.

[10] N. Benoudjit, C. Archambeau, A. Lendasse, J. Lee and M. Verleysen (2002), "Width optimization of the Gaussian kernels in radial basis function networks", *European Symposium on Artificial Neural Networks (ESANN'2002)*, Bruges, April 24-25-26, pp. 425–432

[11] J. L. Bentley (1975), "Multidimensional binary search trees used for associative searching", *Commun, ACM* 18(9), pp. 509–517.

[12] S. Berchold, H.P. Kriegel (2000), "Indexing the Solution Space: A New Technique for Nearest Neighbor Search in High-Dimensional Space", *IEEE Transactions on Knowledge and Data Engineering* vol. 12(1), pp. 45-57.

[13] Bianchini, P. Frasconi, M. Gori (1995), "Learning without local minima in radial basis function networks", *IEEE Transactions on Neural Networks* 30 (3), pp. 136–144.

[14] C. M. Bishop (2006), *Parttern recognition and Machine learning*, Springer, Singapore.

[15] E. Blazieri (2003), *Theoretical interpretations and applications of radial basis function networks*, Technical Report DIT-03- 023, Informatica e Telecomunicazioni, University of Trento.

[16] D.S. Broomhead and D. Lowe (1988), "Multivariable functional interpolation and adaptive networks", *Complex Syst.* vol. 2, pp. 321-355.

[17] A.Chmielewski, S.T.Wierzchon (2006), "V-Dectector algorithm with tree – based structures", *Proceedings of International Multiconference on Cumputer Science and Information Technology*, pp. 11-16.

[18] Cohen and N. Intrator (2002), "A hybrid projection-based and radial basis function architecture: initial values and global optimization", *Pattern Analysis and Applications* 5(2), pp. 113–120.

[19] L. Collatz (1966), *Functional analysis and numerical mathematics*, Academic press, New York and London.

[20] Dang Thi Thu Hien, H.X. Huan and H.T. Huynh (2008), "Local RBF Neural Networks for Interpolating Multivariate Functions", *Addendum Contributions to the 2008 IEEE International Conference on Research, Innovation and Vision for the Future in Computing & Communication Technologies*, ENST 2008 S 001, pp. 70-75.

[21] Dang Thi Thu Hien, H.X. Huan and H.T.Huynh (2009), "Multivariate Interpolation using Radial Basis Function Networks", *International Journal of Data Mining, Modelling and Management Science (IJDMMM)* Vol.1(3), pp.291-309.

[22] B.P. Demidovich (1973), *Computational Mathematics*, Mir Publishers, Moscow.

[23] M. Dikaiakos and J. Stadel (1996), "A Performance Study of Cosmological Simulation on Message-Passing and Shared-Memory Multiprocessors", *In Proceedings of the 10th ACM International Conference on Supercomputing*, ACM, pp. 94-101.

[24] R. O. Duda and P. E. Hart (2001), *Pattern classification and scene analysis*, John Wiley & Sons.

[25] J. B. Gomm, and D.L.Yu (2000), "Selecting Radial Basis Function Network Centers with Recursive Orthogonal Least Squares Training", *IEEE Transaction on Neural Networks* Vol.11(2), pp. 306-314.

[26] Guang-Bin Huang, P. Saratchandran, N. Sundararajan (2005), "A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation", *IEEE Transaction on Neural Networks* Vol.16(1), pp. 57-67.

[27] J. Haddadnia and M. Ahmadi (2003), "Design of RBF neural network using an efficient hybrid learing algorithm with application in human face recognition with pseudo zernike moment", *IEICE TRANS. INF. & SYST.* vol.E86-D (2).

[28] M.T. Hangan, H.B Demuth and M. Beale (1996), *Neural network design*, PWS Publishing Company, USA.

[29] E.J. Hartman, J.D. Keeler and J.M. Kowalski (1990), "Layered neural networks with Gaussian hidden units as universal approximations", *Neural Comput.* Vol. 2(2), pp. 210-215.

[30] S. Haykin (1998), *Neural Networks: A Comprehensive Foundation (second edition)*, Prentice Hall International, Inc.

[31] M. H. Hasoun (1995), *Fundamentals of Artificical Neural Networks*, MIT, Press, Cambridge, MA.

[32] D. Hearn and M. P. Bake (1997), *Computer graphics*, Prentice hall.

[33] T. Hiroki and T. Koichi (2008), "Midpoint-validation method of neural networks for pattern classification proplems", *International Journal of Innovative Computing, Information and Control* Vol. 4(10), pp. 2475-2482.

[34] Hoang Xuan Huan, D.T.T. Hien and H.T. Huynh (2007), "A Novel Efficient Algorithm for Training Interpolation Radial Basis Function Networks", *Signal Processing* vol.87 Issue11, pp. 2708 – 2717.

[35] Insoo Sohn (2007), "RBF Neural Network Based SLM Peak-to-Average Power Ratio Reduction in OFDM Systems", *ETRI Journal* Vol.29(3), pp. 402-404.

[36] N.V. Kopchenova, I.A.Maron (1987), *Computational Mathematics worked examples and problems with elements of theory*, Mir Publishers Moscow.

[37] M. Lazaro, I. Santamaria, and C. Pantaleon (2003), "A new EM-based training algorithm for RBF networks", *Neural Networks* Vol.16, pp. 69–77.

[38] C.G. Looney (1997), *Pattern recognition using neural networks: Theory and algorithm for engineers and scientist*, Oxford University press, New York.

[39] J. Luo, C. Zhou, Y. Leung (2001), "A Knowledge-Integrated RBF Network for Remote Sensing Classification", *LREIS, Institute of Geographical Sciences and Natural Resources Research*, CAS, Beijin, China.

[40] M.Y. Mashor (2000), "Hybrid training algorithms for RBF network", *International Journal of the Computer* 8 (2), pp. 50–65.

[41] K.Z Mao, Guang-Bin Huang (2005), "Neuron selection for RBF neural network classifier based on data structure preserving criterion**",** *Neural Networks, IEEE Transactions* Vol. 16, Issue 6, pp. 1531 – 1540.

[42] C.Micchelli (1986), "Interpolation of scattered data: Distance matrices and conditionally positive definite functions", *Constructive approximations* vol.2, pp. 11-22.

[43] T.M. Mitchell (1997), *Machine learning*, McGraw-Hill, New York.

[44] Moore (1999), "Very fast EM-based mixture model clustering using multiresolution k-d trees". *In Advances in Neural Information Processing Systems 11*, pp. 543-549.

[45] E. K. Murphy and V.V. Yakovlev (2006), "RBF Network Optimization of Complex Microwave Systems Represented by Small FDTD Modeling Data Sets", *IEEE Transaction on Microwave theory and techniques* Vol 54(4), pp. 3069-3083.

[46] Nguyen Mai-Duy, T. Tran-Cong (2001), "Numerical solution of differential equations using multiquadric radial basis function networks", *Neural Networks* Vol.14(2), pp.185-99.

[47] J. Park and I.W. Sandberg (1993), "Approximation and radial-basis-function networks", *Neural Comput.* vol 5(3), pp. 305-316.

[48] T. Poggio and F. Girosi (1990), "Networks for approximating and learning", *Proc. IEEE* vol.78(9), pp. 1481-1497.

[49] M.J.D. Powell (1988), "Radial basis function approximations to polynomials", *Proceedings of the Numerical analysis 1987, Dundee*, UK, pp. 223-241.

[50] F. Schwenker. H.A. Kesler, Günther Palm (2001), "Three learning phases for radial-basis-function networks", *Neural networks* Vol.14, pp. 439-458.

[51] Z. J. Shao, H. S. Shou (2008), "Output feedback tracking control for a class of MIMO nonlinear minimum phase systems based on RBF neural networks". *International Journal of Innovative Computing, Information and Control* Vol 4(4), pp. 803-812.

[52] Shu, H. Ding and K.S. Yeo (2003), "Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible Navier-Stokes equations", *Computer Methods in Applied Mechanics and Engineering* Vol.192, pp. 941-54.

[53] Y. F. Sun, Y. C. Liang, W. L. Zhang, H. P. Lee, W. Z. Lin and L. J. Cao (2005), "Optimal partition algorithm of the RBF neural network and its application to financial time series forecasting", *Neural Computing & Applications*, Springer London, vol.14 (1), pp. 36-44.

[54] S. Tejen, J. Jyunwei (2008), "A Hybrid artificial neural networks and particle swarm optimization for function approximation", *International Journal of Innovative Computing, Information and Control* Vol. 4(9), pp. 2363-2374.

[55] S.Theodoridis, K.Koutroumbas (2003), *Pattern recognition*, Second edition, Elsevier.

[56] P. H. Winston (1993), *Artificial intelligence*, third edition, Addison-Wesley Publishing company, USA.

[57] H.S. Yazdi, J. Haddadnia, M. Lotfizad (2007)**, "**Duct modeling using the generalized RBF neural network for active cancellation of variable frequency narrow band noise**",** *EURASIP Journal on Applied Signal Processing* Vol 2007, issue 1, pp.22-22.