ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Văn Dương

KANTS: HỆ KIẾN NHÂN TẠO CHO PHÂN LỚP

KHOÁ LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Công nghệ thông tin

ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Văn Dương

KANTS: Hệ kiến nhân tạo cho phân lớp

KHOÁ LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Công nghệ thông tin
Cán bộ hướng dẫn: PGS.TS Hoàng Xuân Huấn
Đồng hướng dẫn: ThS. Đỗ Đức Đông

HÀ NỘI - 2010

LỜI CẢM ƠN

Tôi muốn bày tỏ sự cảm ơn sâu sắc của mình tới thầy Hoàng Xuân Huấn, thuộc bộ môn Khoa học máy tính, khoa Công nghệ thông tin, trường Đại học Công nghệ, ĐHQGHN. Trong thời gian thực hiện khóa luận, thầy đã nhiệt tình hướng dẫn và giúp đỡ tôi rất nhiều. Ngoài thời gian tìm hiểu và cung cấp tài liệu, thầy cũng chỉ ra những vướng mắc trong qua trình làm, giúp đỡ tôi khắc phục để đạt hiệu quả cao hơn. Ngoài ra tôi còn muốn gửi lời cảm ơn tới thầy đồng hướng dẫn Đỗ Đức Đông, thầy cũng đã nhiệt tình giúp đỡ tôi trong việc tìm hiểu giải quyết những khúc mắc sai lầm khi làm khóa luận này.

Tôi cũng muốn bày tỏ sự cảm ơn của mình tới các các thầy, các cô trong bộ môn, cũng như các thầy, các cô trong khoa, trường đã hết sức tạo điều kiện tốt và giúp đỡ cho tôi hoàn thành khóa luận của mình.

TÓM TẮT NỘI DUNG

Mặc dù đã được nghiên cứu từ rất lâu, nhưng đến nay bài phân lớp mẫu vẫn còn có rất ít công cụ toán học để giải quyết và hiệu quả chưa cao. Mạng Neural nhân tạo là một phương pháp hay để giải quyết bài toán phân lớp mẫu. Năm 1987, Kohonen giới thiệu phương pháp bản đồ tự tổ chức là một loại mạng neural đơn giản và hiệu quả để giải quyết bài toán phân cụm và phân lớp. Năm 1991, Dorigo giới thiệu phương pháp hệ kiến để giải quyết các bài toán tối ưu tổ hợp rất hiệu quả. Từ đó, các mô hình giải quyết các bài toán phức tạp mà tư tưởng dựa trên sự mô phỏng hành vì loài kiến đã đạt được nhiều bước tiến đáng kể. Điển hình là hệ kiến của Chialvo và Millonas.

Nội dung chính của khóa luận là trình bày khảo cứu về thuật toán KANT (một sự kết hợp) để giải quyết bài toán phân lớp sau đó ứng dụng cơ sở lý thuyết trên để xây dựng chương trình kiểm tra độ chính xác của thuật toán so với k láng giềng gần nhất và cải tiến một phần thuật toán bằng học tập hợp (**Ensembler learning**) để thu được kết quả tốt hơn.

Danh mục các hình

BẢNG TỪ VIẾT TẮT

SOM (Self-organizing map)	Bản đồ tự tổ chức
KNN (K nearest neibours)	K láng giềng gần nhất
AS (Ant System)	Phương pháp hệ kiến
ANN (Artificail Neural Network)	Mạng neural nhân tạo
BMU (Best matching unit)	Phần tử gần đúng nhất

MỤC LỤC

MỞ ĐẦU

Sự phát trển mạnh mẽ của công nghệ cao nói chung và khoa học máy tính nói riêng ngày càng thu hút nhiều nhà khoa học và công nghệ quan tâm nghiên cứu bài toán nhận dạng mẫu. Thoạt tiên, bài toán nhận dạng mẫu xuất phát từ nhu cầu tạo nên các thành phần máy có khả năng quan sát môi trường. Cùng với sự phát triển của các ứng dụng công nghệ thông tin, đặc biệt trong lĩnh vực học máy, người ta phải đi sâu phát triển các hệ nhận dạng mẫu có khả năng tìm các mẫu mới trong các cơ sở dữ liệu lớn hay còn gọi là khám phá tri thức từ dữ liệu.

Phân lớp mẫu là bài toán thường gặp nhất trong nhận dạng mẫu và phân thành hai loại có giám sát và không có giám sát. Trong bài toán phân lớp có giám sát, dựa trên một tập dữ liệu đã được gán nhãn, người ta xây dựng một bộ phân lớp để gán nhãn cho các dữ liệu chưa biết. Còn trong bài toán không giám sát, người ta phân một tập dữ liệu chưa được gán nhãn thành các các tập con sao cho các đối tượng dữ liệu trong mỗi tập con thì có đặc tính giống nhau hơn so với đối tượng ở các tập con khác.

Trong các bài toán nhận dạng mẫu, bài toán phân lớp có giám sát là bài toán được ứng dụng rộng rãi nhất. Việc xây dựng bộ phân lớp trong bài toán này được thực hiện bởi các thuật toán học máy (học có giám sát). Với học có giám sát truyền thống, con người thường phải bỏ ra rất nhiều công sức để gán nhãn cho tập dữ liệu đào tạo nếu muốn có một bộ học tốt.

Phương pháp đơn giản và thông dụng hiện nay để giải bài toán phân lớp là k láng giềng gần nhất. Gần đây, phương pháp KANTS mô phỏng hành vi loài kiến kết hợp với bản đồ tự tổ chức (SOM) của Kohonen. Nội dung của khóa luận này là trình bày khái quát về phương pháp phân lớp KANTS, trên cơ sở đó xây dựng chương trình thử nghiệm thuật toán bằng C++ đánh giá hiệu quả với các k khác nhau. Ngoài ra, chúng tôi xây dựng bộ phân lớp mới nhờ phương pháp học tập hợp của các bộ học với k khác nhau đã có. Kết quả thực nghiệm cho thấy, chất lượng bộ học mới được cải tiến đáng kể so với từng bộ học thành phần

Trong các phương pháp kinh điển để giải bài toán phân lớp có giám sát, mô hình mạng neural nhân tạo và phương pháp k-láng giềng gần nhất đã chứng tỏ được tính hiệu

quả. Xong, hiệu suất và độ chính xác của các phương pháp/mô hình này chưa cao như kì vọng. Khóa luận này xin được trình bày thuật toán KANTS: một sự kết hợp giữa bản đồ tự tổ chức (một loại mạng neural nhân tạo) của Kohonen và phương pháp hệ kiến của Chialvo và Milonas.

Bố cục của khóa luận gồm các phần sau:

Chương 1: Giới thiệu về bài toán phân lớp và hai phương pháp kinh điển để giải bài toán này là: mạng neural nhân tạo và phương pháp k-láng giềng gần nhất

Chương 2: Giới thiệu về bản đồ tự tổ chức của Kohonen bao gồm kiến trúc và luật học

Chương 3: Phương pháp hệ kiến và thuật toán KANTS

Chương 4: Kết quả thực nghiệm và sự mở rộng của KANTS.

Chương 5: Kết luận

CHƯƠNG 1:

BÀI TOÁN PHÂN LỚP VÀ MỘT SỐ PHƯƠNG PHÁP TIẾP CẬN

Chương này trình bày về khái niệm bài toán phân lớp trong học máy và hai phương pháp kinh điển để giải bài toán này hiện nay: mạng neural và k-láng giềng gần nhất.

1.1. PHÁT BIỂU BÀI TOÁN PHÂN LỚP

1.1.1. Mẫu (pattern/sample):

Có thể phân làm hai hoại: mẫu trừu tượng và mẫu cụ thể. Các ý tưởng, lập luận và khái niệm... là những ví dụ về mẫu trừu tượng, nhận dạng các mẫu như vậy thuộc về lĩnh vực nhận dạng khái niệm.

Các mẫu cụ thể bao gồm các đối tượng có tính không gian, thời gian và hình ảnh, hoặc các đối tượng vật lý, chữ ký, chữ viết, ký hiệu, ảnh, đoạn sóng âm thanh, điện não đồ hoặc điện tâm đồ, hàm số...là những ví dụ về mẫu cụ thể.

1.1.2. Nhận dạng mẫu là gì?

Không có một định nghĩa thống nhất nào về nhận dạng mẫu (Pattern recognition viết tắt là PR) nhưng điều này cũng không gây ra tranh cãi gì trong giới nghiên cứu. Sau đây là một số định nghĩa theo ngữ cảnh nghiên cứu:

- Duda et al: Nhận dạng mẫu là việc quy những đối tượng vật lí hay sự kiện vào một loại (nhóm) nào đó đã xác định từ trước.
- Jürgen Schürmann: Nhận dạng mẫu là việc gán nhãn w cho một quan sát x.
- Selim Aksoy: Nhận dạng mẫu là việc nghiên cứu cách làm cho một máy có thể thực hiện:

  + Quan sát môi trường.
  + Học cách phân biệt được các mẫu cần quan tâm.
  + Đưa ra các quyết định đúng đắn về loại (nhóm) của các mẫu.

Như vậy thay cho việc tìm định nghĩa chính xác cho khái niệm nhận dạng mẫu ta sẽ liệt kê các bài toán chính trong lĩnh vực này.

1.1.3. Các bài toán nhận dạng mẫu thường gặp

Các bài toán nhận dạng mẫu thường gặp có thể quy về các dạng sau.

- **Phân lớp có giám sát hay phân loại (categorize)**: Dựa trên một tập con (tập đào tạo) đã biết nhãn, đưa ra một cách gán nhãn cho các đối tượng mới để phân tập các đối tượng thành các lớp. Ví dụ: nhận dạng chữ viết tay nhờ các chữ đã biết, nhận dạng loài hoa nhờ các thông tin về độ dài, độ rộng, màu sắc.
- Phân lớp không giám sát hay phân cụm (cluster): Chia tập đối tượng thành nhóm sao cho các đối tượng trong mỗi nhóm tương đối giống nhau còn các đối tượng khác nhóm thì khác nhau.
- Phân tích hồi quy (regression) hay nhận dạng hàm: Xác định một biến (hàm) qua tập các biến khác.
- Nhận thực (Identify): Xác định đối tượng trong tập đã cho có là đối tượng đang quan tâm hay không. Chẳng hạn như nhận thực vân tay, nhận thực mặt người...
- Mô tả: Mô tả các đối tượng dưới hình thức dễ phân tích. Chẳng hạn mô tả điện tâm đồ dưới dạng biểu đồ đặc trưng hoặc xâu mã.

Khóa luận này sẽ đề cập đến bài toán đầu tiên: **Phân lớp có giám sát hay phân loại (categorize).** Để hiểu rõ hơn yêu cầu của bài, xem ví dụ ở phần 1.3 phương pháp k láng giềng gần nhất.

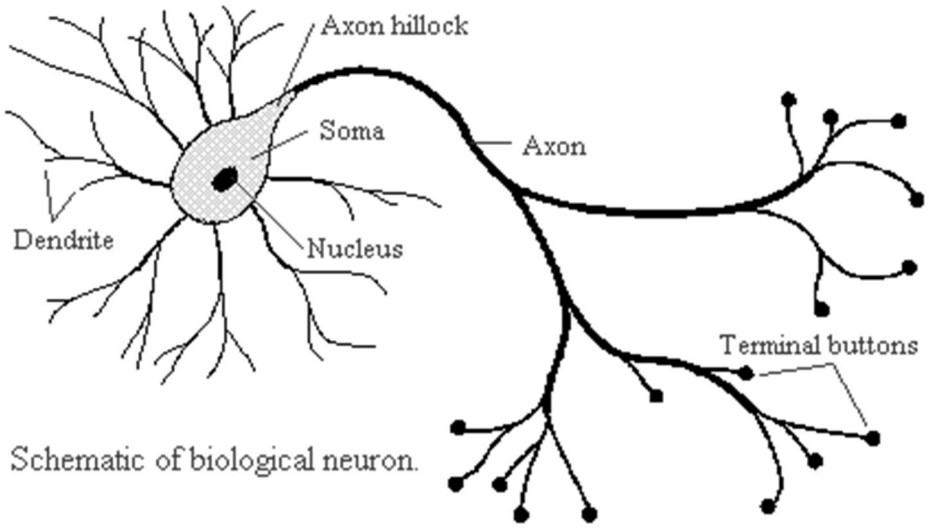
1.2. MẠNG NEURAL NHÂN TẠO

Bộ não con người chứa đựng những bí mật mà đến bây giờ khoa học vẫn chưa giải đáp được, chính nhờ có bộ não hoàn chỉnh mà con người đã trở thành động vật bậc cao thống trị muôn loài. Đã từ lâu con người đã nghiên cứu cấu trúc đặc biệt của bộ não từ đó ứng dụng để giải quyết những bài toán khoa học kỹ thuật. Người ta đã phát hiện ra rằng bộ não con người là mạng lưới chằng chịt các Neural liên kết với nhau, đây là cơ sở hình thành nên cấu trúc của mạng Neural nhân tạo.

Về bản chất toán học thì mạng Neural nhân tạo như là một mặt trong không gian đa chiều để xấp xỉ một hàm chưa biết nào đấy. Nhưng mạng Neural nhân tạo lại giống mạng Neural sinh học ở chỗ đó là khả năng có thể huấn luyện(học), đây là đặc điểm quan trọng nhất của mạng Neural nhân tạo. Chính vì đặc điểm này mà mạng Neural nhân tạo có khả năng thực hiện tốt các công việc sau khi đã được huấn luyện, và đến khi môi trường thay đổi ta lại có thể huấn luyện lại mạng Neural nhân tạo để nó thích nghi với điều kiện mới.

1.2.1. Mạng Neural sinh học

Mạng Neural sinh học là một mạng lưới (plexus) các Neuron có kết nối hoặc có liên quan về mặt chức năng trực thuộc hệ thần kinh ngoại biên (peripheral nervous system) hay hệ thần kinh trung ương (central nervous system).



Hình 1: Minh họa một Neuron thần kinh sinh học

Trên đây là hình ảnh của một tế bào thần kinh(Neural thần kinh), ta chú ý thấy rằng một tế bào thần kinh có ba phần quan trọng:

-Phần đầu cũng có nhiều xúc tu (Dendrite) là nơi tiếp xúc với các với các điểm kết nối(Axon Terminal) của các tế bào thần kinh khác

-Nhân của tế bào thần kinh (Nucleus) là nơi tiếp nhận các tín hiệu điện truyền từ xúc tu. Sau khi tổng hợp và xử lý các tín hiệu nhận được nó truyền tín hiệu kết quả qua trục cảm ứng (Axon) đến các điểm kết nối (Axon Terminal) ở đuôi.

-Phần đuôi có nhiều điểm kết nối (Axon Terminal) để kết nối với các tế bào thần kinh khác.

Khi tín hiệu vào ở xúc tu kích hoạt nhân nhân Neuron có tín hiệu ra ở trục cảm ứng thì Neuron được gọi là cháy. Mặc dù W. Mculloch và W.Pitts (1940) đề xuất mô hình mạng neural nhân tạo khá sớm nhưng định đề Heb (1949) mới là nền tảng lý luận cho mạng neural nhân tạo.

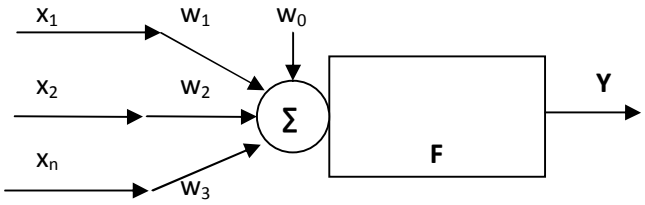
Định đề Heb: Khi một neuron(thần kinh) A ở gần neuron B, kích hoạt thường xuyên hoặc lặp lại việc làm cháy nó thì phát triển một quá trình sinh hoá ở các neuron làm tăng tác động này.

1.2.2. Mạng Neural nhân tạo

Mạng Neural nhân tạo được thiết kế để mô hình một số tính chất của mạng Neural sinh học, tuy nhiên, khác với các mô hình nhận thức, phần lớn các ứng dụng lại có bản chất kỹ thuật. Mạng Neural nhân tạo (ANN) là máy mô phỏng cách bộ não hoạt động thực hiên các nhiệm vụ của nó. Một mạng Neural là bộ xử lý song song phân tán lớn nó giống bộ não người về 2 mặt:

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

a) Cấu tạo một Neuron trong mạng Neural nhân tạo



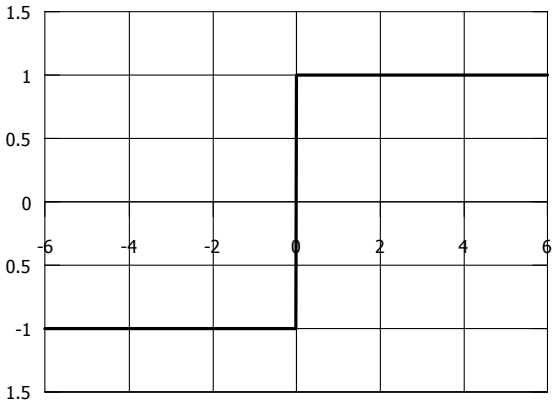
Cấu tạo một Neural nhân tạo

Một neuron 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, hàm F gọi là hàm kích hoạt để tạo tín hiệu ra dựa trên giá trị

hàm tổng có trọng số của các giá trị đầu vào, Y là giá trị đầu ra của Neural. Ta có thể biểu diễn một Neural nhân tạo theo công thức toán học như sau: $Y = F\left(\mathrm{w}_0 + \sum_{i=1}^{n} x_i w_i \right)$

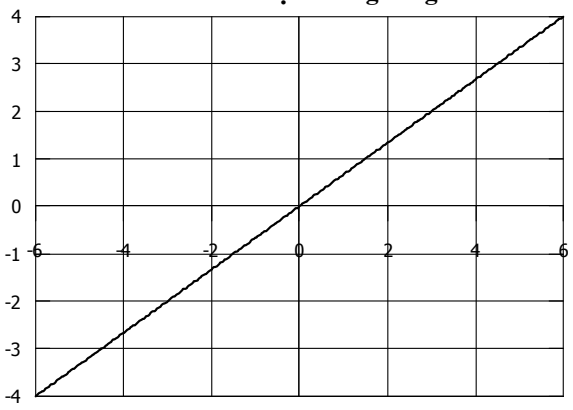
Tùy vào thực tế bài toán hàm F là một hàm cụ thể nào đấy, trong quá trình huấn luyện(học) thì các tham số w_i được xác định. Trên thực thế F thường được chọn trong những hàm sau:

1) Hàm ngưỡng $F(x) = \varphi(x) = \begin{cases} -1; \forall x < 0 \\ 1; \forall x \geq 0 \end{cases}$



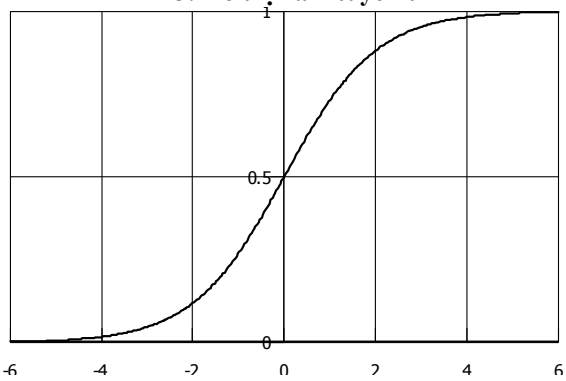
Hình 2: Đồ thị hàm ngưỡng

2) Hàm tuyến tính $F(x) = ax$



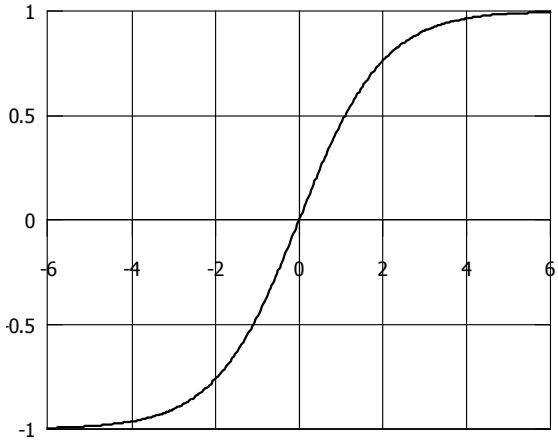
Hình 3: Đồ thị hàm tuyến tính

3) Hàm sigmoid $F(x) = \dfrac{1}{1 + e^{-x}}$



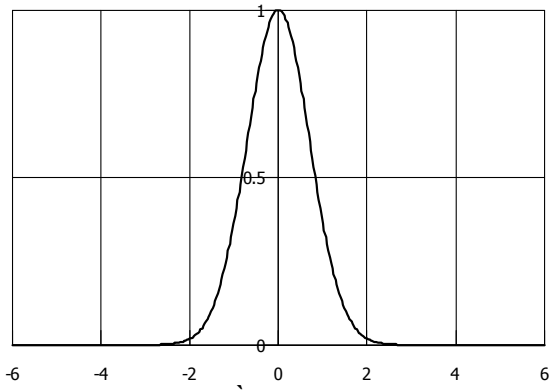
Hình 4: Đồ thị hàm sigmoid

4) Hàm tanh $F(x) = \frac{1 - e^{-x}}{1 + e^{-x}}$



Hình 5: Đồ thị hàm tanh

5) Hàm bán kính (Gauss) $F(x) = e^{-x^2}$

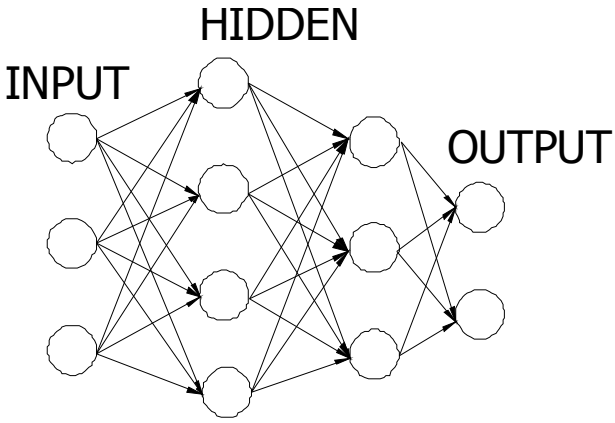


Hình 6: Đồ thị hàm Gauss

Trên thực tế thì các họ hàm sigmoid thường dùng cho mạng Neural truyền thẳng nhiều tầng MLP vì các hàm này dễ tính đạo hàm: $f'(x) = f(x)(1 - f(x))$, trong khi đó mạng Neural RBF lại dùng hàm kích hoạt là hàm bán kính.

b) Kiến trúc của mạng Neural nhân tạo

Kiến trúc của mạng Neural nhân tạo lấy tư tưởng chính của mạng Neural sinh học đó là sự kết nối của các Neural. Tuy nhiên, mạng Neural nhân tạo có kiến trúc đơn giản hơn nhiều, về cả số lượng Neuron và cả kiến trúc mạng, trong khi ở mạng Neural tự nhiên một Neuron có thể kết nối với một Neuron khác bất kỳ ở trong mạng thì ở mạng Neural nhân tạo các Neuron được kết nối sao cho nó có thể dễ dàng được biểu diễn bởi một mô hình toán học nào đấy. Ví dụ trong mạng Neural truyền tới các Neuron được phân thành nhiều lớp, các Neuron ở lớp trước chỉ được kết nối với các Neuron ở lớp sau.



Hình 7: Kiến trúc mạng neural truyền tới

c) Quá trình học

Như đã nói ở trên mạng Neural nhân tạo có khả năng huấn luyện được(học), quá trình huấn luyện là quá trình mà mạng Neural nhân tạo tự thay đổi mình dưới sự kích thích của môi trường(bộ dữ liệu huấn luyện) để phù hợp với điều kiện của môi trường. Quá trình huấn luyện chỉ có thể được thực hiện khi mạng Neural nhân tạo đã xây dựng được kiến trúc cụ thể, và hàm kích hoạt F đã được xác định. Về bản chất quá trình học là quá trình xác định các tham số w_i của các Neuron trong mạng Neural. Có ba kiểu học chính, mỗi kiểu mẫu tương ứng với một nhiệm vụ học trừu tượng. Đó là học có giám sát, học không có giám sát và học tăng cường. Dưới đây xin nêu ra phương pháp học có giám sát, các phương pháp khác xem thêm **phần phụ lục**.

Học có giám sát

Trong học có giám sát, ta được cho trước một tập ví dụ gồm các cặp $(x^i, y^i, i = \overline{1..n}), x \in X, y \in Y$ và mục tiêu là tìm một hàm $f : X \to Y$ (trong lớp các hàm

được phép) khớp với các ví dụ. Trên thực tế người ta thường tìm hàm *f* sao cho tổng bình phương sai số đạt giá trị nhỏ nhất trên tập ví dụ: $E = \sum_{i=1}^{n} \left(f(x^i) - y^i \right)^2$

Chương 2 xin được trình bày về bản đồ tự tổ chức (còn gọi là bản đồ Kohonen) – một loại mạng neural dùng để phân cụm.

1.3. PHƯƠNG PHÁP K LÁNG GIỀNG GẦN NHẤT

1.3.1 Thuật toán k láng giềng gần nhất là gì?

Phương pháp k láng giềng gần nhất (K nearest neibours – KNN) là một trong những phương pháp phân loại đơn giản và cơ bản nhất, là lựa chọn đầu tiên mà ta nghĩ đến khi cần học phân lớp mà không biết rõ về phân bố của dữ liệu.

K láng giềng gần nhất là một thuật toán học có giám sát với kết quả của một truy vấn được phân loại chủ yếu dựa trên nhãn của các lân cận cạnh nó. Chức năng của thuật toán này là để phân lớp một đối tượng dựa trên các thuộc tính và các mẫu huấn luyện. Bộ phân loại không sử dụng một mô hình nào để phân loại mà chỉ dựa trên bộ nhớ. Cho một đối tượng O cần cần lớp, trước hết tìm k số các đối tượng (hay các điểm huấn luyện) gần nhất với đối tượng này. Sự "gần" ở đây được hiểu là gần về khoảng cách Ơclit của các vector dữ liệu (vector thể hiện đặc trưng số đã được chuẩn hóa).

Sự phân lớp được thực hiện theo quy tắc sau: mỗi đối tượng trong k các đối tượng được tìm thấy ở trên mang một nhãn lớp (vì là học có giám sát) và sẽ được "bỏ phiếu". Lá phiếu này chính là nhãn lớp của chúng. Đếm số "phiếu" được bỏ, ta sẽ tìm được nhãn được bỏ phiếu nhiều nhất, nhãn lớp của O sau đó sẽ được gán nhãn này. Đây là cách gán nhãn dựa trên xác suất và sự gần về mặt dữ liệu. Việc chọn đặc trưng số để biểu diễn đặc điểm của dữ liệu dưới dạng vector dữ liệu có ảnh hưởng lớn đến chất lượng của thuật toán phân lớp. Các đặc trưng được chọn phải tạo ra sự phân bố đủ tốt để giảm thiểu tối đa nhiễu, việc chọn đặc trưng cũng phụ thuộc nhiều vào đặc điểm của từng bài toán.

Để hiểu rõ hơn KNN, ta hãy xét ví dụ sau:
Ví dụ:

Ta có các dữ liệu là những câu hỏi khảo sát ý kiến một số người và trắc nghiệm khách quan với hai thuộc tính: độ bền axit và độ dẻo dai, để phân loại xem một loại giấy nào đó là tốt hay không:

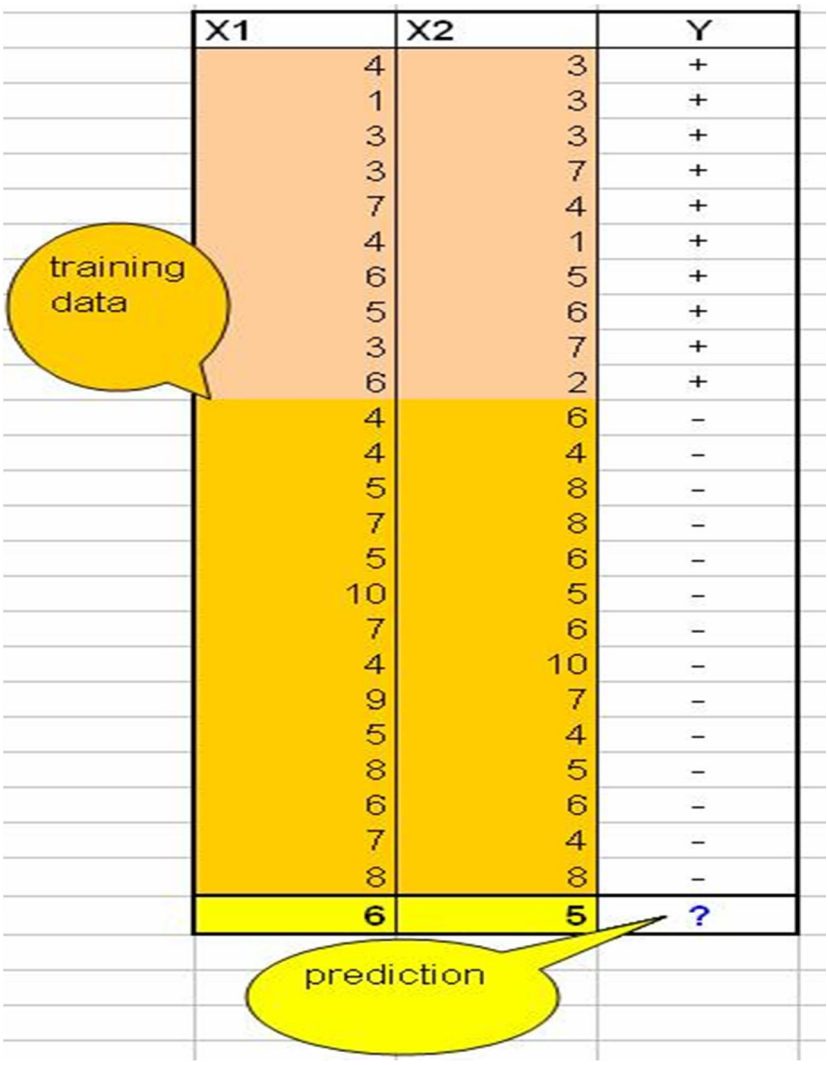
X_1 = độ bền axit (giây)	X_1 = độ dẻo dai (kq/m^2	Phân lớp
7	7	Kém
7	4	Kém
3	4	Tốt
1	4	Tốt

Từ đây, khi các nhà máy sản xuất giấy, nếu họ thu được 1 mẫu giấy có X1 = 3 và X2 = 7, sẽ rất khó để xác định chất lượng giấy nếu phải tiến hành điều tra lấy ý kiến người tiêu dùng. Nhưng KNN là một phương pháp đủ tốt để xác định chất lượng mà không cần khảo sát tốn kém mà chỉ dựa vào một số kết quả khảo sát tin cậy đã tiến hành.

1.3.2 Thuật toán KNN

Ta có thể biểu diễn mỗi đối tượng huấn luyện và các đối tượng cần phần lớp bằng 1 vector n nhiều. Khi đó mỗi vector lại có thể đặt trọng không gian n chiều.

Thuật toán KNN rất đơn giản. Nó làm việc dựa trên khoảng cách nhỏ nhất từ điểm cần phân lớp tới các dữ liệu huấn luyện để xác định k lân cận gần nhất. Sau khi tìm được k lân cận này, ta thực hiện bỏ phiếu để tìm ra nhãn lớp được bỏ nhiều nhất làm kết quả.
Dữ liệu cho thuật toán KNN gồm nhiều thuộc tính đa chiều X_i để phân các đối tượng vào các nhãn trong tập Y. Các dữ liệu của KNN là những vector thể hiện đặc trưng số của đối tượng.

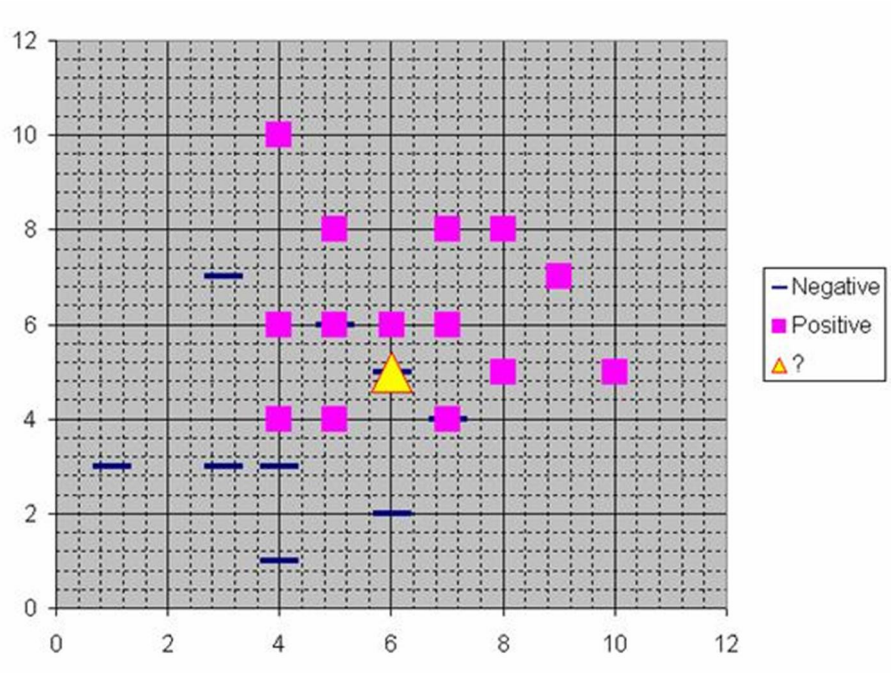


X1	X2	Y
4	3	+
1	3	+
3	3	+
3	7	+
7	4	+
4	1	+
6	5	+
5	6	+
3	7	+
6	2	+
4	6	-
4	4	-
5	8	-
7	8	-
5	6	-
10	5	-
7	6	-
4	10	-
9	7	-
5	4	-
8	5	-
6	6	-
7	4	-
8	8	-
6	5	?

Hình 8: Mẫu dữ liệu ví dụ cho KNN

Dòng cuối cùng là đối tượng với $X_1 = 6$ và $X_2 = 5$ là đối tượng mà ta cần biết nó thuộc phân lớp + hay lớp –

Ta coi 2 thuộc tính X_1 và X_2 của các mẫu như là những tọa độ trên không gian 2 chiều, khi đó có thể trực quan các mẫu như sau:



Hình 9: Trực quan hóa các mẫu trên mặt phẳng

Giả sử ta ta chọn tham số trong thuật toán này là K = 8 (tức là 8 lân cận gần nhất). Sau đó ta tính toán khoảng cách giữa điểm cần tính q và tất cả các mẫu huấn luyện. Ta sẽ sử dụng khoảng cách Ơ clit để tính khoảng cách giữa các mẫu với các đại lượng X_i. Giả sử điểm cần phân lớp q có tọa độ (x_{1q}, x_{2q}) và tọa độ của các mẫu huấn luyện là (x_{1t}, x_{2t}). Bình phương khoảng cách Ơ clit giữa $d^2_{tp} = (x_{1t} - x_{1p})^2 + (x_{2t} - x_{2p})^2$ vậy $d_{tp} = sqrt((x_{1t} - x_{1p})^2 + (x_{2t} - x_{2p})^2)$. Trường hợp có nhiều hơn 2 thuộc tính cách tính cũng tương tự như công thức Ơ clit cho vector n chiều:

Bước tiếp theo là tìm k lân cận gần nhất, với mỗi mẫu đào tạo t cho một khoảng cách là d_t, ta chọn k mẫu có khoảng cách d_t nhỏ nhất. Nói cách khác, ta sắp xếp các khoảng cách của q với tất cả các mẫu đào tạo, sắp xếp theo thứ tự tăng dần vào chọn k điểm cho k khoảng cách nhỏ nhất đầu tiên.

Computation		
Distance	Nearest Neighbor sign	
8		
29		
13		
13		
2	+	
20		
0	+	
2	+	
13		
9		
5		
5		
10		
10		
2	-	
16		
2	-	
29		
13		
2	-	
4		
1	-	
2	-	
13		

Hình 10: Bỏ phiếu các mẫu dữ liệu trong KNN

Trong bảng trên, nhãn + được bỏ phiếu 3 lần còn nhãn – được bỏ phiếu 5 lần, ta lấy nhãn được bỏ nhiều hơn, do đó q được gán nhãn là –.

Ưu điểm:

+) Phân lớp tốt với các dữ liệu đào tạo có nhiễu (đặc biệt nếu sử dụng nghịch đảo bình phương của khoảng cách trọng số.

+) Hiệu quả với dữ liệu đào tạo lớn.

Nhược điểm:

+) Cần phải xác định giá trị của tham số K (số láng giềng gần nhất)

+) Học dựa khoảng cách không nói rõ nó sử dụng loại khoảng cách nào và với những thuộc tính nào thì cho kết quả tốt nhất. Và sử dụng tất cả các thuộc tính hay chỉ một số thuộc tính mà thôi.

+) Chi phí tính toán là khá cao vì chúng ta cần phải tính khoảng các của từng mẫu đến tất cả các mẫu huấn luyện.

CHƯƠNG 2: BẢN ĐỒ TỰ TỔ CHỨC

Chương này trình bày về bản đồ tự tổ chức (Self-organizing map - SOM). Bản đồ tự tổ chức là cơ sở để thực hiện KANTS.

2.1. Giới thiệu:

Một bản đồ tự tổ chức (self-organizing map - hay self-organizing feature map (SOFM) là một loại mạng neural nhân tạo được huấn luyện sử dụng học không giám sát để sinh ra một không gian có số chiều nhỏ hơn (thông thường là hai chiều), biểu diễn rời rạc của không gian đầu và của các dữ liệu huấn luyện, được gọi là một bản đồ. Bản dồ tự tổ chức khác với những mạng neural nhân tạo khác theo nghĩa là chúng sử dụng một hàm lân cận để bảo toàn tính quan hệ hình học của không gian đầu vào.

SOM rất có lợi trong việc trực quan hóa cách nhìn của cách dữ liệu nhiều chiều, giống như việc chia các dữ liệu nhiều chiều thành các mức (hoặc các cụm). Mô hình này lần đầu tiên được miêu tả như một mạng neural nhân tạo bởi một giáo sư người Phần Lan là Teuvo Kohonen, và đôi khi còn được gọi là bản đồ Kohonen[19].

Giống như hầu hết các mạng neural nhân tạo, các SOM hoạt động ở hai chế độ huấn luyện (training) và ánh xạ (mapping) hay thực hiện phân lớp. Quá trình huấn luyện sẽ xây dựng bản đồ sử dụng các mẫu dữ liệu đầu vào. Đó là một quá trình cạnh tranh, cũng được gọi là lượng tử vector. Việc ánh xạ tự động phân lớp một vector đầu vào mới.

Một bản đồ tự tổ chức chứa các thành phần gọi là các đỉnh (node) hay neural. Kết hợp với mỗi đỉnh là một vector trọng số có số chiều bằng số chiều với các vector dữ liệu vào và một vị trí trong không gian bản đồ. Thông thường, các đỉnh được sắp xếp dạng lưới tứ giác hoặc lục giác. Bản đồ tự tổ chức sẽ mô phỏng một ánh xạ từ một không gian dữ liệu đầu vào với số chiều lớn thành một không gian có số chiều nhỏ hơn (thông thường để cho trực quan, người ta thường để 2 chiều để dữ liệu phân bố trên một mặt phẳng).

Thủ tục đặt mỗi vector từ không gian dữ liệu đầu vào vào bản đồ là việc tìm ra đỉnh với vector trọng số gần nhất với vector được lấy từ không gian dữ liệu và gán các tọa độ trong bản đồ của đỉnh này bằng vector của đầu vào của ta.

Thông thường cấu trúc của loại mạng này giống với các **mạng truyền thẳng** – ta sẽ thấy rằng trong một mạng SOM, chỉ có một tầng vào mà một tầng ra.

Những mở rộng quan trọng bao gồm việc sử dụng các lưới có hình xuyến (lục giác hoặc tứ giác), trong loại lưới này các cung đối diện được kết nối sử dụng số các đỉnh lớn hơn. Điều này chỉ ra rằng, với các bản đồ tự tổ chức với số các đỉnh nhỏ làm việc theo cách giống như K-means, các bản đồ tự tổ chức lớn hơn sắp xếp lại dữ liệu theo các đặc tính hình học của dữ liệu vào.

Thông thường, trong SOM người ta sử dụng ma trận U. Các giá trị của ma trận U là khoảng cách giữa một đỉnh và các lân cận gần nó nhất. Ví dụ, trong các lưới tứ giác, ta sẽ chọn được 4 hoặc 8 đỉnh gần nhất tùy theo cách định nghĩa hoặc trong các lưới lục giác là 6 đỉnh.

Những bản đồ tự tổ chức lớn hơn thể hiện các thuộc tính rất rõ nét. Trong những bản đồ chứa tới hàng nghìn đỉnh, ta có thể thực hiện các phép phân cụm trên chính bản đồ.

2.2. Thuật toán:

Mục đích của việc học trong bản đồ tự tổ chức là dẫn đến việc tạo ra sự phân chia thành các phần khác nhau của các neural trong mạng để thỏa mãn giống như các mẫu vào. Đây là một phần thúc đẩy bởi cách trực quan, thính giác hay các giác quan thông tin khác được điều khiển bởi các phần khác nhau, tách biệt nhau của vỏ não người.

Các trọng số của các vector được khởi tạo là những giá trị ngẫu nhiên nhỏ hay các mẫu tương tự nhau từ không gian con…

Mạng sau đó phải được huấn luyện bởi một số lớn các vector mẫu vào, gần nhau nhất có thể … Những mẫu này thường được thực hiện nhiều lần tương tự mỗi lần lặp.

Việc huấn luyện sử dụng học cạnh tranh. Khi một mẫu huấn luyện được đưa vào mạng, nó tính khoảng cách Ơclit (Euclidean distance) cho tất cả các vector trọng số. Neural với vector trọng số giống với vector trọng số của mẫu vào nhất được gọi là **best matching unit** (BMU). Các trọng số của BMU và các neural gần với nó trong lưới SOM được kéo gần về phía vector đầu vào. Biên độ của thay đổi sẽ giảm theo thời gian và theo khoảng cách với BMU. Công thức cập nhật cho một neural với vector trọng số **Wv**(t) là:

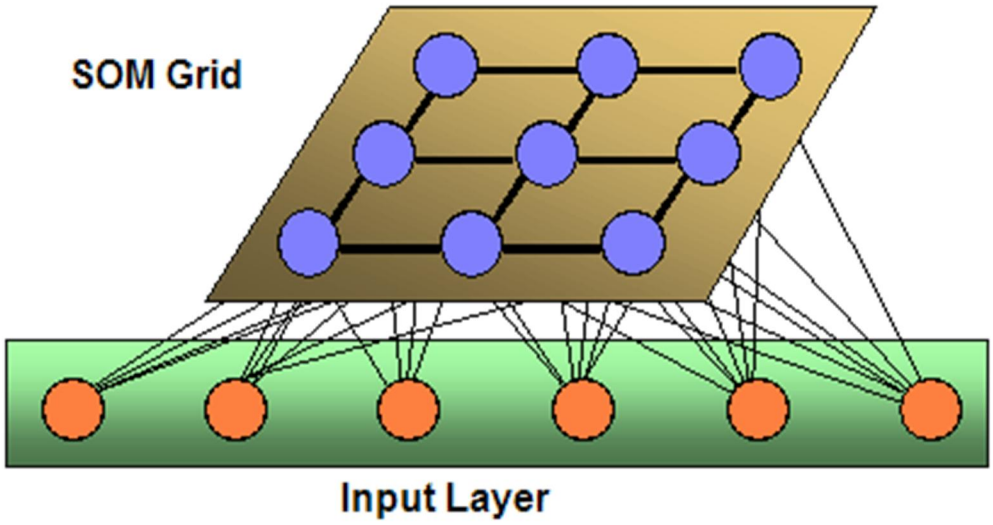
$$\mathbf{Wv}(t+1) = \mathbf{Wv}(t) + \Theta(v, t)\,\alpha(t)(\mathbf{D}(t) - \mathbf{Wv}(t))$$

Với $\alpha(t)$ là một hệ số học giảm đơn điệu và **D**(t) là vector đầu vào. Hàm lân cận Θ (v, t) phụ thuộc và khoảng cách giữa BMU và neural v. Ở dạng đơn giản nhất nó là một cho tất cả các neural đủ gần với BMU và 0 với các neural khác, nhưng thông thường ta hay chọn một hàm **gaussian.** Bất kể với dạng của hàm này, hàm lân cận giảm theo thời gian. Tại thời điểm đầu khi lân cận là rộng, bản đồ tự tổ chức sẽ cập nhật tổng thể toàn mạng, tức là coi như toàn bộ mạng thuộc vào lân cận, sau đó bán kính lân cận giảm dần. Khi lân cận co lại chỉ còn một đôi neural, các trọng số được hội tụ về ước lượng địa phương.

Quá trình này được lặp đi lặp lại cho mỗi vector đầu vào với một số (thường là lớn) cho kì λ. Mạng kết thúc việc kết hợp các đỉnh ra với các nhóm hoặc mẫu trong tập dữ liệu vào. Nếu những mẫu này được gán nhãn, các nhãn có thể được gắn để kết hợp với các đỉnh trong mạng lưới đã được huấn luyện này.

Trong quá trình ánh xạ, sẽ có một neural duy nhất gọi là neural thắng: neural có vector trọng số nằm gần nhất với vector đầu vào. Để xác định được vector này chỉ cần đơn giản tính khoảng cách Ơclit của vector đầu vào và các vector trọng số của các neural, neural nào cho khoảng cách Ơclit nhỏ nhất là neural thắng.

Việc biểu diễn dữ liệu đầu vào thành các vector được được nhấn mạnh trong khóa luận này, ta cũng chú ý rằng mọi đối tượng đều có thể biểu diễn số hóa rời rạc và có một khoảng cách thích hợp để xác định được kết hợp khi cần thực hiện huấn luyện có thể được sử dụng để tạo bản đồ tự tổ chức. bao gồm: các ma trận, các hàm liên tục hoặc thậm chí các bản đồ tự tổ chức khác.



Hình 11: Mô hình mạng SOM

Để dễ hiểu hơn, ta xét ví dụ sau:

Ví dụ:

Giả sử có một mảng nxm các đỉnh, mỗi đỉnh chưa một vector trọng số và biết vị trí của nó trong mảng. Mỗi vector trọng số có số chiều bằng với chiều của vector đầu vào. Các trọng số có thể được đặt là những giá trị ngẫu nhiên.

Bây giờ ta cần đưa các dữ liệu vào để huấn luyện mạng này. (Bản đồ sinh ra và dữ liệu vào đã cho ở trong những không gian con khác nhau). Chúng ta sẽ tạo 3 vector để biểu diễn màu. Các màu có thể được biểu diễn bởi các thành phần đỏ, xanh, và xanh da trời. Các vector đầu vào của chúng ta sẽ tuần tự có 3 thành phần này, mỗi vector tương ứng với một không gian màu. Các vector đầu vào sẽ là:

R = <255, 0, 0>

G = <0, 255, 0>

B = <0, 0, 255>

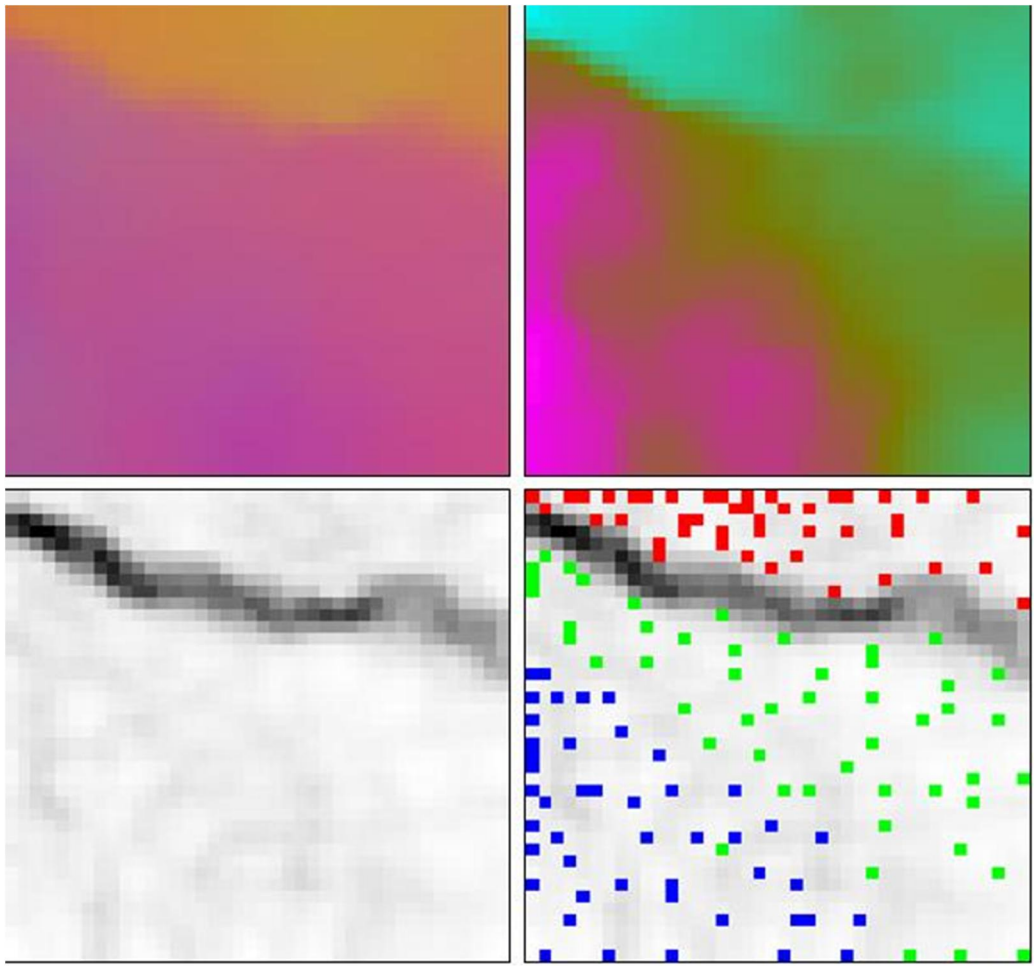
Các tập dữ liệu huấn luyện màu được sử dụng trong SOM:

threeColors = [[255, 0, 0], [0, 255, 0], [0, 0, 255]]

eightColors = [[0, 0, 0], [255, 0, 0], [0, 255, 0], [0, 0, 255], [255, 255, 0], [0, 255, 255], [255, 0, 255], [255, 255, 255]]

Các dữ liệu vào cần được chuẩn hóa (độ dài của vector là một) trước khi huấn luyện SOM.

Các neural (lưới tứ giác 40x40) được huấn luyện 250 lượt với một tốc độ học là 0.1 sử dụng tập dữ liệu hoa Iris đã được chuẩn hóa (tập dữ liệu này là các vector dữ liệu 4 chiều). Một hình màu được tạo ra bởi ba chiều đầu tiên của các vector SOM 4 chiều (hình trên trái), hình màu giả của độ lớn của các vector trọng số SOM (hình trên phải), ma trận U (khoảng các Ơ clit giữa các vector trọng số của các ô lân cận) của SOM (trái dưới) và phủ của các điểm dữ liệu (đỏ: I. setosa, xanh: I. versicolor và xanh da trời: I. verginica) trên ma trận U dựa trên khoảng cách Ơ clit nhỏ nhất giữa các vector dữ liệu và các vector trọng số SOM (phải dưới)



Hình 12: Các mạng SOM thể hiện phân bố các dữ liệu tập IRIS

Các biến:

Các vector được bôi đậm:

t = lượt hiện tại
λ = giới hạn số lượt duyệt
Wv = vector trọng số hiện tại
D = đầu vào đích
Θ(t) = hàm lân cận, tác động đến khoảng cách giữa vector hiện tại và BMU
α(t) = hàm tốc độ học phụ thuộc thời gian

Thuật toán:

1. Khởi tạo các vector trọng số cho các đỉnh của bản đồ một cách ngẫu nhiên

2. Lấy một vector đầu vào

3. Duyệt mỗi đỉnh trong bản đồ

   3.1 Sử dụng công thức tính khoảng cách khoảng cách Ơ clit để tìm sự giống nhau giữa vector đầu vào và vector trọng số của các đỉnh của bản đồ

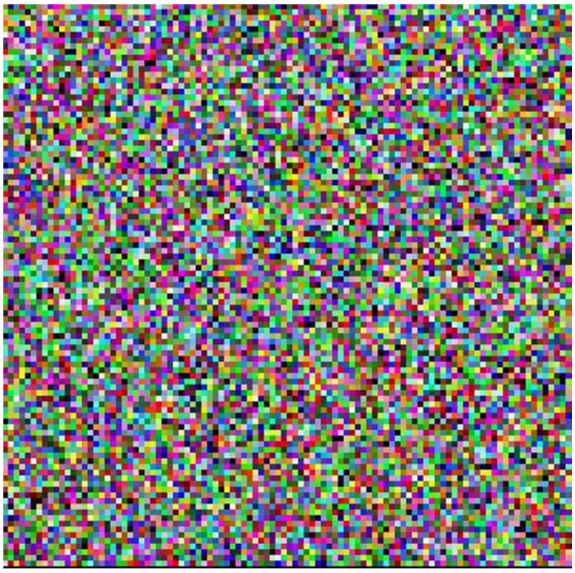
   3.2 Đỉnh cho khoảng cách nhỏ nhất là **best matching unit** (BMU)

4. Cập nhật các đỉnh trong lân cậu của BMU bằng cách kéo chúng gần hơn với vector đầu vào

   4.1 $\mathbf{Wv}(t + 1) = \mathbf{Wv}(t) + \Theta(t)\alpha(t)(\mathbf{D}(t) - \mathbf{Wv}(t))$

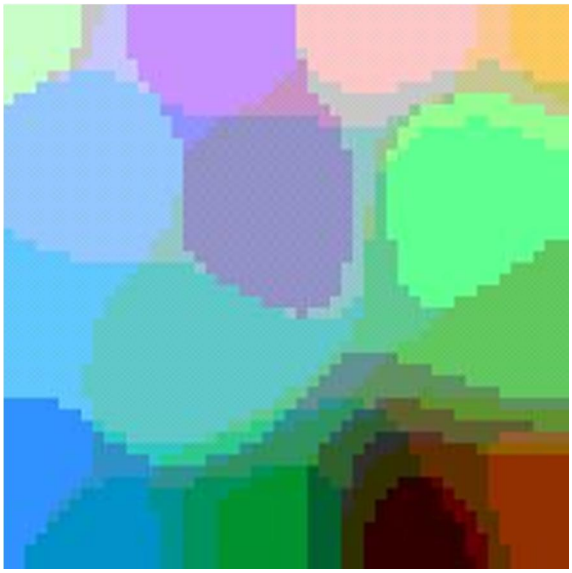
5. Tăng t và lặp từ 2 khi $t < \lambda$

Với việc khởi đầu các trọng số là ngẫu nhiên thì ta sẽ có dạng của bạn đồ một cách trực quan tương tự như sau:



Hình 13: Dạng ngẫu nhiên ban đầu của SOM

Do lấy ngẫu nhiên các trọng số nên các màu cũng là ngẫu nhiên, ta chưa thấy sự xuất hiện của một cụm nào cả. Mỗi lần đưa vào một dữ liệu huấn luyện, ta đặt nó vào vị trí thích hợp nhất (vị trí cho khoảng cách Ơ clit nhỏ nhất hay **BMU**), tại vị trí này, ta điều chỉnh các vector xung quanh để nó giống với vector dữ liệu huấn luyện (giống theo nghĩa là khoảng cách gần nhất). Mức độ điều chỉnh cũng thay đổi theo khoảng cách trên bản đồ và theo thời gian huấn luyện. Điểm càng cách xa BMU thì điều chỉnh càng ít. Thời gian huấn luyện càng lớn thì mức độ điều chỉnh càng ít để lưới ổn định và mịn hơn.



Hình 14: Tráng thái lưới SOM sau một số bước huấn luyện

Nhìn vào hình ta có thể thấy rằng việc phân cụm các màu đã rõ ràng hơn. Ta có thể nhận ra ngay các cụm một cách trực quan.

Mạng SOM sau khi huấn luyện là một công cụ để phân lớp. Các dữ liệu huấn luyện đã biết nhãn, do đó ta có thể xác định được nhãn các neural trong lưới dựa vào các nhãn này và khoảng các Ơ clit vector giữa dữ liệu và các neural. Bây giờ giả sử ta có đối tượng I cần phân lớp. Đối tượng I cũng có vector có số chiều bằng số chiều của các đỉnh trong SOM, ta muốn biết I thuộc lớp nào, trước hết tìm k đỉnh gần I nhất (tức là có khoảng cách Ơclit vector nhỏ nhất), từ k đỉnh này ta xác định tập các nhãn lớp và lấy nhãn được bỏ phiếu nhiều nhất. Phương pháp này giống như phương pháp KNN đã trình bày ở trên.

2.3. Phân tích

Có 2 cách để giải thích một SOM. Bởi trong quá trình học các vector trọng số của toàn bộ lân cận được di chuyển theo những hướng giống nhau, các mẫu giống nhau có xu hướng kích thích các neural gần kề. Vì vậy, SOM hình thành lên một bản đồ các mẫu giống nhau được ánh xạ để gần nhau và các mẫu khác nhau được kéo xa nhau ra. Điều này có thể được trực quan hóa bằng một ma trận U (Khoảng cách Ơ clit giữa các vector trọng số của các ô lân lận) của SOM. Tại những đỉnh mà ma trận U cho khoảng cách nhỏ sẽ là nơi tập trung các vector gần nhau

Cách khác để giải thích là: coi các trọng số neural như là những con trỏ trỏ vào không gian đầu vào. Chúng hình thành nên một xấp xỉ rời rạc phân bố của các mẫu ví dụ huấn luyện. Càng nhiều neural trỏ vào các vùng với sự tập trung cao các mẫu huấn luyện và thì các neural càng bị tác động để tạo khoảng cách với nhau. Quá trình này tạo lên một dạng ổn định của lưới mà nhờ đó, các dữ liệu giống nhau sẽ được cụm lại gần nhau.

CHƯƠNG 3: KANTS – HỆ KIẾN NHÂN TẠO CHO PHÂN LỚP

Chương này sẽ giới thiệu về mô hình nhận thức bầy đàn của Chialvo và Minonas. Sau đó sẽ trình bày sự kết hợp của mô hình này với bản đồ tự tổ chức và chi tiết về KANTS.

3.1. Giới thiệu:

Như đã nói ở trên, KANTS là một sự kết hợp ưu điểm giữa SOM và ACO dùng trong bài toán nhận dạng và phân lớp mẫu. SOM là một bài toán học mạng neural không giám sát, sau quá trình huấn luyện mạng, các trọng số và vệt mùi trên mỗi đỉnh được giữ lại để dùng trong bài toán phân lớp giống như các hệ số trong mạng neural.

Trong cách tiếp cận này, mỗi đối tượng dữ liệu được gán vào một con kiến. Sau đó, khi các con kiến di chuyển trên lưới KANTS, nó sẽ thay đổi môi trường nó đi qua bằng cách nhả mùi ra trên đường đi, tại những đường đi được các con kiến đi nhiều, vệt mùi sẽ nặng hơn, còn những đường đi còn lại, vệt mùi sẽ bị bay hơi do đó giảm nồng độ.

Thuật toán này đã được áp dụng thành công trong những bài toán và những vấn đề kinh điển và cho kết quả tốt hơn những thuật toán kinh điển khác như K-nearest neibouhoods và Mạng neural.

Việc phân cụm được thực hiện một cách tự nhiên bằng một số loại kiến theo hai con đường khác nhau. Thứ nhất, hệ kiến nhận dạng bởi mùi của các thành viên khác trong hệ của chúng dẫn đến một sự phân cụm tự nhiên theo các tổ giống nhau;thứ 2, các con kiến làm việc phân cụm một cách thủ công các ấu trùng của chúng và các con đã chết, đặt những thứ đó vào từng đống, vị trí và kích cỡ của các đống này là hoàn toàn độc lập. Thuật toán kiến được dùng trong các mô hình như được đưa ra trong [9] đã được áp dụng vào việc phân cụm và phân lớp. Thông thường, các phương pháp này đi theo hành động phân cụm thứ 2: dữ liệu cho việc huấn luyện các cụm được biểu diễn là các con chết, những con này đã được những con còn sống thu lại theo một số quy tắc nào đó và sắp xếp lại thành đống. Trong khi cùng với việc di chuyển để mang các con chết lại thành đống, các con kiến này đã nhả ra mùi trên đường đi của chúng và đồng thời, việc xác định đường đi của nó cũng được quyết định dựa vào lượng mùi mà nó ngửi được trên đường đi, nhưng cách tiếp cận khác có là hoàn toàn có thể bởi xem mỗi dữ liệu như là một con kiến.

Sau đây khóa luận này trình bày **KohonAnts (KANTS)**, một thuật toán kiến kết hợp với ưu điểm của một thuật toán phân cụm khác của Kohonen: Self-Organizing Map với hệ kiến (Ant System) của Chialvo và Millonas và giới thiệu một vài ý kiến để giải quyết việc phân loại, sau đó sẽ đưa ra các cải tiến bằng cách thay KNN bằng thuật toán tương tự hiệu quả hơn. Đầu tiên, mọi con kiến biểu diễn một đối tượng dữ liệu đầu vào để huấn luyện mạng như trường hợp với mạng SOM đã nói ở trên. Ban đầu, ta cho những con kiến đi trong lưới và nhả mùi lên các đường đi mà nó đi qua. Lưới này khởi tạo với tất cả các vector là những giá trị ngẫu nhiên có cùng số chiều với số chiều của dữ liệu vào, và mỗi lần một con kiến đi qua đỉnh trên lưới, nó thay đổi mùi theo một phương thức giống như phương thức đã được sử dụng bởi SOM, bằng việc "đẩy" mùi gần với dữ liệu đã được lưu trong bản thân con kiến đó. Khi những con kiến đi khắp lưới, vị trí kiến và mùi kiến đồng thời thay đổi, vì vậy các con kiến với dữ liệu giống như nhau sẽ được co lại gần nhau hơn trong lưới đó, và bản thân lưới đó sẽ chứa các vector giống như là các vector đã được lưu trong con kiến...Lưới có thể được sử dụng để phân lớp theo cách giống như SOM ở trên, trong khi các con kiến có thể được sử dụng để xác định một cách trực quan các vị trí các cụm. Phần hay của phương pháp này là nó sự tự tổ chức của đàn kiến qua các vệt mùi: những con kiến thay đổi môi trường (bằng cách thay đổi mùi trong lưới trong lưới KANTS), và điều đó ảnh hưởng tới hoạt động của toàn bộ đàn kiến.

3.2. Các khái niệm mở đầu:

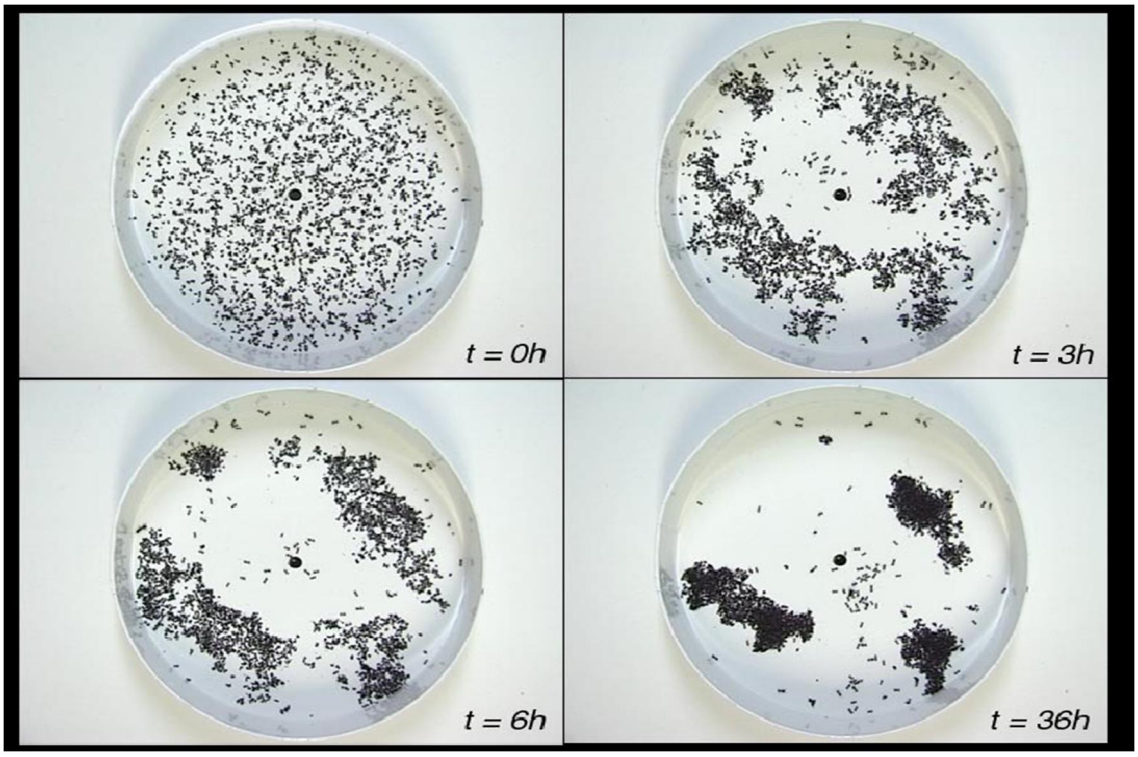
3.2.1 Mô hình nhận thức bầy đàn và hệ kiến nhân tạo[9]:

Các loại côn trùng trong tự nhiên như ong, kiến, mối…thực tế không có bộ não tốt như con người và các động vật có vú khác song lại có thể ghi nhớ được đường đi rất dài, hoặc tìm được đường đi từ tổ tới nguồn thức ăn ngắn nhất dựa vào đặc điểm tự nhiên của chúng: sống theo bầy đàn và đánh dấu môi trường bằng mùi. Trong khi các động vật có vú giao tiếp thường giao tiếp trực tiếp với nhau và ghi nhớ những thứ chúng học được và đường đi vào vỏ não thì công trùng giao tiếp gián tiếp thông qua môi trường và ghi nhớ cũng dựa vào mùi có trong môi trường. Cụ thể là mỗi khi đi qua một vị trí nào đó, các con kiến này lại nhả ra một hóa chất đặc biệt, có khả năng ứ đọng và bay hơi. Dựa vào đặc điểm về độ nặng mùi của môi trường, những con kiến có thể biết được các con kiến khác đã đi qua môi trường ấy hay chưa và mức độ đi qua nhiều hay ít. Ngoài ra ở một số loài

kiến, mùi mà chúng nhả ra còn đặc trưng cho hệ gien mà dựa vào đó, các con kiến có thể xác định mùi này thuộc con kiến của loài nào (hoặc tổ nào). Đây chính là các truyền thông gián tiếp mà các con kiến tạo ra, chúng dựa vào môi trường để biết trạng thái của các con kiến khác. Qua đó chúng tự tổ chức để tìm được những đường đi đủ tốt (tối ưu hoặc gần tối ưu). Thí nghiệm sau sẽ giúp hiểu rõ hơn về sự tự tổ chức này.

Thí nghiệm:

Cho một số lượng kiến và ấu trùng đủ lớn vào trong một chiếc đĩa, đĩa có mép nghiêng đủ để các con kiến di chuyển tự do trong đĩa nhưng không đi ra ngoài. Các con kiến sẽ có xu hướng thu thập các ấu trùng lại thành từng đống, quan sát trạng thái của đĩa theo thời gian, người ta thấy như sau:



Hình 15: Thí nghiệm cho thấy sự phân cụm các ấu trùng của kiến

+ Tại t = 0h: bắt đầu quan sát, các con kiến và ấu trùng được phân bố ngẫu nhiên trên đĩa, chưa có gì đặc biệt.

+ Tại t = 3h: một số vị trí trên đĩa có tập trung nhiều ấu trùng hơn các vùng con lại do các con kiến gom lại, nhưng sự phân cụm chưa rõ ràng.

+ Tại t = 6h: sự phân cụm các ấu trùng đã rõ ràng hơn, nhưng các cụm vẫn chưa "gọn"

+ Tại t = 36h: sự phân cụm đã thấy rất rõ ràng, các con kiến xếp các ấu trùng lại thành 3 đống riêng biệt, các cụm được tạo ra là gần tối ưu, tức là khoảng cách giữa các tâm cụm với các điểm còn lại là nhỏ theo thuật toán k-means[*xem thêm k-means*].

3.2.2 Nhắc lại SOM – bản đồ tự tổ chức:

Như đã nói ở trên, bản đồ tự tổ chức được giới thiệu bởi Teuvo Kohonen là một mạng neural không giám sát, cố gắng làm giả sự tự tổ chức của giác quan vỏ não của não người, có thể được sử dụng như một công cụ phân cụm/phân lớp hay như một phương thức để tìm ra mỗi quan hệ chưa rõ trong một tập các biến mô tả vấn đề. SOM thực hiện một phép chiếu không tuyến tính từ một không gian dữ liệu chiều cao(mỗi chiều 1 biến) trên một lên một lưới neural thông thường, thấp chiều hơn (thường là 2). Từ đây, với kiểu mạng này thì các đối tượng dữ liệu sẽ được phân bố trong một mặt phẳng (nếu là 2 chiều), nhìn vào mặt phẳng này ta có thể kết luận rằng phép chiếu bảo toàn quan hệ hình học trong khi đồng thời tạo ra một biểu diễn không gian vector có số chiều giảm đi. SOM xử lí một tập các vector vào, chúng được đưa vào trong các biến (các đặc trưng) là điển hình của mỗi mẫu, và tạo ra một mạng hình học đầu ra với mỗi neural cũng được kết hợp với một vector các biến (vector mô hình) nó là sự biểu diễn cho nhóm các vector đầu vào. Sự lặp đi lặp lại liên tiếp của phương thức(hành động) có hiệu ứng làm "di chuyển" các vector mô hình từ neural thắng tới vector đầu vào: vectors sẽ có xu hướng theo phân bố của các vector đầu vào. Do đó, thuật toán dẫn tới một sự sắp xếp lại bản đồ hình học các đặc tính/tính chất của không gian vào, theo nghĩa rằng neural gần kề trong mạng đó có xu hướng gần giống với vector ảnh hưởng nhất (vector trọng số). Từ đó, nhìn vào kết quả hiện ra của SOM, ta có thể nhận ra một số cụm như là mối qua hệ hình học của các mục dữ liệu và các biến đó. Những đỉnh gần nhau có vector trọng số gần giống nhau sẽ cùng một lớp…

3.2.3 Ant System

Ant System[9] là một thuật toán kiến mà các vệt mùi và các mạng đường đi của kiến hiện lên không cần phải cố một điều kiện nào về biên, mạng hình học, hay thêm quy tắc hoạt động nào. Trạng thái của một con kiến có thể được biểu diễn bởi vị trí r của nó và hướng θ. Các quy tắc di chuyển được kế thừa từ các ***hàm đáp ứng nhiễu***, chúng được tạo

ra để tái sinh một số kết quả thực nghiệm với những con kiến thật. Hàm trả về có thể được chuyển một cách hiệu quả thành quy tắc di chuyển hai tham số giữa các ô bằng sử dụng hàm độ nặng mùi:

$$W(\sigma) = \left(1 + \frac{\delta}{1 + \sigma \cdot \delta}\right)^{\beta}$$

hàm này xác định mối quan hệ của các xác suất của việc di chuyển từ 1 ô r với mật độ mùi σ(r). Tham số β (kết hợp với mật độ tính theo áp suất thẩm thấu) điều khiển độ ngẫu nhiên với mỗi con kiến theo bởi một hướng mùi. Khả năng của giác quan 1/ δ cho biết khả năng của mỗi con kiến phát hiện ra mùi giảm tại nơi có sự tập trung cao. (Những phương trình trước đây của Chialvo mà Millonas, có một nhân tử trọng số $w(\Delta\theta)$, $\Delta\theta$ là sự thanh đổi hướng trực tiếp tại mỗi bước đi. Nhân tử này đảm bảo rằng, việc đi thẳng so với hướng đi hiện tại của con kiến sẽ có xác suất cao hơn đi lệch, và xác suất đi lệnh lại cao hơn so với xác suất đi ngược trở lại. Khi kiểm tra một số giả thiết, cần một mô hình riêng biệt và vì đó cần một lưới tứ giác để các con kiến có thể đi vòng quanh, đi mỗi bước tại mỗi lần lặp.

Quyết định bước tiếp theo là đỉnh nào được tạo ra theo nồng độ mùi tại các ô láng giềng (lân cận Von Neumanm). Như một diều kiện tất yếu, mỗi con riêng để lại một lượng cố định η mùi tại ô mà nó đi qua. Mùi này bị bay hơi tại mỗi bước của thuật toán với tốc độ k. Chú ý tằng không có một sự giao tiếp trực tiếp nào giữa các con kiến nhưng có một kiểu truyền thông gián tiếp thông qua mùi, các con kiến căn cứ vào nồng độ mùi trên một đường đi để xác định xem trước đó các con kiến khác đã đi nhiều hay ít qua đường đi đó.

Algorithm 1. KANTS Algorithm

```
initialize_randomly_grid_vectors
place_randomly_ants_in_grid
for N_iterations do
  for each ant a at cell (x, y) do
    j = decide_where_to_go(a,(x, y))
  end for
  update_grid    //
  evaporate_grid    //
  update_neighbourhood_radio // decreases radio
end for
```

Hình 16: Mã giả thuật toán KA NTS

3.3. Mô hình kiến tự tổ chức

Mô hình đã đưa ra ở trên có một số đặc điểm giống với hệ kiến của Chialvo và Millonas và bản đồ tự tổ chức của Kohonen. Do đó nó được đặt tên là KohonAnts (hoặc gọi tắt là KANTS). KANTS được thiết kế để chạy là một thuật toán phân cụm giống như SOM, để nhóm một tập các mẫu đầu vào (các mẫu huấn luyện) thành các cụm với những đặc điểm giống nhau. Sau đó, nó hoạt động giống như một thuật toán phân cụm, hoạt động theo cách học không giám sát, không đề cập đến nhãn lớp của các dữ liệu đầu vào trong suốt quá trình học đó.

Ý tưởng chính của thuật toán là gán mỗi mẫu đầu vào (một vector) vào một con kiến, và sau đó đặt nó vào một lưới đã được tạo trước (giống như lưới SOM ở trên). Sau đó, khi các con kiến mang theo các vector di chuyển trên lưới và thay đổi môi trường. Tại mỗi ô trên lưới cũng chứa một vector có cùng số chiều và cùng miền với vector huấn luyện mà con kiến đó mang theo.

Khi mỗi con kiến di chuyển đến một miền nào đó trong lưới, nó kéo các vector dữ liệu trên các đỉnh mà nó đi qua lại gần với vector mà nó mang theo, tức là điều chỉnh lại hệ số của các vector này bằng một hàm học tăng cường tương tự như trong SOM, đồng thời mùi và vị trí cũng được thay đổi theo. Điều này khiến cho những con kiến với những

dữ liệu giống nhau sẽ có xu hướng tiến lại gần nhau trên lưới, và những vùng đó sẽ chứa các vector giống với những vector được các còn kiến mang theo kia.

Vậy, sau quá trình huấn luyện đủ lớn, lưới được hình thành với những miền xác định, nó có thể được sử dụng như một công cụ phân lớp như một lưới SOM sau quá trình huấn luyện vậy. Tuy nhiên ở đây, các con kiến được kéo lại gần nhau, nhóm thành những cụm riêng với những vector của chúng tương tự nhau. Mã giả của thuật toán được chỉ ra trong Thuật toán 1.

Ở đây, ta sẽ xem xét kĩ hàm DecideWheretoGo, hàm này dùng để xác định xem bước tiếp theo một con kiến sẽ đi đâu:

Algorithm 2. Decide_Where_To_Go $(a,(i,j))$

```
for all cells (x, y) in neighbourhood of (i, j) do
  σ_{ij,xy} = E_D((i, j),centroid((x, y)))
  compute W(σ_{ij,xy})    //
end for
q = random(0,1)
if q ≤ q0 then
  (k, l) = MAX(P_{ij,xy}) // selected cell = the one with maximum probability
else
  (k, l) = roulette_wheel(P_{ij,xy}) // selected cell = roulette_wheel
end if
```

Hình 17: Mã giả hàm quyết định bước đi tiếp theo

Xác suất di chuyển đến một đỉnh j trong lưới của một con kiến tại đỉnh i là P_{ij} được tính theo công thức:

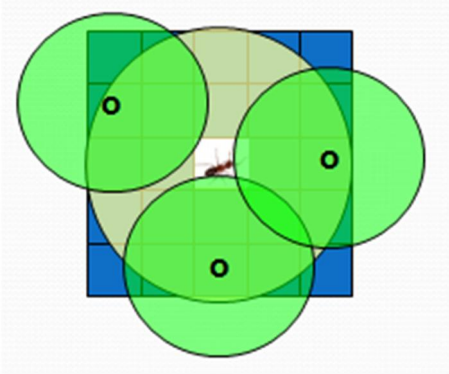
$$P_{ij} = \begin{cases} \dfrac{W(\sigma_{ij})}{\sum_{u \in N_i^t} W(\sigma_{iu})} & if\ j \in N_i^t \\ 0 & otherwise \end{cases}$$

Hình 18: Công thức xác suất di chuyển

Với N_i là lân cận của đỉnh I. Ngoài ra còn có bán kính của lân cận (*neighbourhood radius - nr*) bán kính này giảm trong quá trình chạy thuật toán, nghĩa là bán kính thay đổi theo thời gian thực hiện t. Các giá trị σ được định nghĩa là:

$$\sigma_{ij} = \sqrt{V_i(v)^2 - CTR_j(v)^2} \qquad \forall v = 1..nvars$$

Với V_i là vector kết hợp với đỉnh I và CTR_j là trọng tâm của một vùng mà trong tâm là đỉnh j. Công thức này bằng với công thức tính khoảng các Oclit giữa vector của đỉnh i và vector trọng tâm với trung tâm là đỉnh j, cả 2 vector này có một số các chiều (biến) là *nvars*.



Hình 19: Lân cận khả dĩ

Cuối cùng, trong luật *DecideWheretoGo*, $W(\sigma)$ là hàm độ nặng mùi trong Ant System. Luật này làm việc như sau: khi một con kiến đang xây dựng lời giải và được đặt vào một đỉnh I, một số ngẫu nhiên q trong đoạn [0, 1] được sinh ra; nếu $q < q_0$ thì lân cận tốt nhất j được chọn là đỉnh đi tiếp theo trên đường đi. Nếu không, con kiến sẽ chọn trong lân cận bằng cách sử dụng một vòng quay sổ xố để xác định P_{ij} là xác suất đi cho mỗi lân cận j.

Hàm UpdateGrid giống như các hàm tương tự trong các thuật toán kiến, nó tăng cường mùi lên đường đi mà con kiến đi qua. Tại mỗi bước đi, mỗi con kiến k cập nhật đỉnh I của nó sử dụng hàm học của SOM [*xem lại chương 2: bản đồ tự tổ chức*]. Và với mỗi mẫu dữ liệu vào của mỗi vector kiến mang theo ta có *nvars* biến, do vậy công thức sẽ là:

$$V_i^t(v) = V_i^{t-1}(v) + R \cdot [a_k(v) - V_i^{t-1}(v)] \quad \forall v = 1..nvars$$

Với V_i là vector của đỉnh i, t là bước lặp hiện tại, và a_k là vector kết hợp với con kiến k. R là một loại tốc độ học tăng cường:

$$R = \alpha \cdot (1 - \overline{D}(a_k, CTR_i))$$

Với α là nhân tử tốc độ học thường thấy trong SOM (là hằng số trong thuật toán này), CTRi lại là trọng tâm của một vùng mà có tâm là i. Cuối cùng D là khoảng các Ơ clit trung bình giữa vector của kiến và vector tọng tâm:

$$\overline{D} = \sum_{v=1}^{nvars} \frac{\sqrt{a_k(v)^2 - C_i(v)^2}}{nvars}$$

Như tất cả các thuật toán kiến khác, một điều rất quan trọng đó là việc môi trường quay lại trạng thái trước đó (trạng thái khởi đầu). Việc bay hơi trong KANTS được thực hiện tại mỗi đỉnh cho tất cả các con kiến đã di chuyển vào cập nhật môi trường.

$$V_i(v) = V_i(v) - \rho \cdot V_{i0}(v) \qquad \forall v = 1..nvars$$

Với ρ là tham số bay hơi thông thường và Vi0 là vector khởi tạo tương ứng với đỉnh i. Hàm này thay đổi vector đó để nó gần hơn với các giá trị khởi tạo của nó. Hàm này có thể được hiểu như một sự mô phỏng sự bay hơi của các vệt mùi trong môi trường.

Sau khi lưới đã được tạo, thông thường ta sử dụng các phương pháp tìm kiến địa phương để phân lớp. Trong khóa luận này sẽ dùng thuật toán k láng giềng gần nhất để tìm nhãn cho các ô trên lưới. Sau đó, sẽ tiến hành test cũng bằng thuật toán k láng giềng gần nhất. Cụ thể như sau:

Pha 1: Với mỗi ô trên lưới, ta tìm k con kiến huấn luyện có khoảng cách Ơ clit với nó nhỏ nhất, sau đó, dựa vào nhãn của các con kiến huấn luyện ta tiến hành bỏ phiếu để lấy nhãn lớp được bỏ nhiều nhất, gán nhãn này cho ô hiện tại. Làm như vậy cho tất cả các ô trên lưới.

Pha 2: Với mỗi dữ liệu muốn test, ta đưa nó vào một con kiến, lại tính khoảng cách Ơ clit của con kiến này với tất cả các ô trong lưới. Tìm k ô có khoảng cách Ơ clit gần nhất

với vector của con kiến này. Dựa vào nhãn của các ô tìm được, tìm nhãn xuất hiện nhiều nhất và so sánh nhãn đó với nhãn thực của con kiến. Nếu hai nhãn này giống nhau tức ta đã phân lớp đúng.

Chương 4: KẾT QUẢ VÀ THỰC NGHIỆM

Chương này trình bày các xây dựng phần mềm và kiểm tra kết quả của KANTS, so sánh với KNN, đồng thời chỉ ra sự phụ thuộc kết quả vào các tham số.

Cuối chương sẽ trình bày thuật toán mới để cải tiến KNN

4.1. Xây dựng chương trình kiểm thử:

Trong khóa luận này, chúng tôi viết một chương trình để tính toán và kiểm tra độ chính xác của thuật toán phân loại KANTS, đồng thời cũng viết chương trình cho thuật toán k láng giềng gần nhất để tiện so sánh. Chương trình được viết bằng ngôn ngữ C++ trên nền Microsoft Windows bằng bộ công cụ Visual Studio.

Phần mềm gồm 3 class chính: Cell, Ant và Kants. Mỗi đối tượng Cell là biểu diễn một ô trên lưới. Mỗi đối tượng Ant biểu diễn một con kiến. Kants là một đối tượng gồm một mảng 2 chiều các ô (Cell) và một mảng các con kiến (Ant).

Mỗi ô được xác định bằng một tọa độ (x, y). Mỗi ô được đặc trưng bằng 1 vector trọng số. Số chiều của ô được xác định bằng số chiều của dữ liệu đầu vào. Ngoài ra trong ô còn có biến để xác định class tương ứng và một cờ để xác định đã có con kiến nào trong ô chưa (trường hợp chỉ cho một con kiến trong một ô).

Mỗi con kiến được đặc trưng bởi vector trọng số mà nó mang theo để huấn luyện mạng, vị trí (x, y) chỉ ra tọa độ của ô mà nó đang đứng, class tương ứng với vector trọng số mà nó mang theo.

Ma trận trọng tâm được xác định là ma trận có kích thước bằng với kích thước lưới. Vị trí (x, y) trong ma trận là trọng tâm của vùng có tâm là ô (x, y) trên lưới, được tính bằng trung bình cộng của các vector trọng số. Tham số bán kính tâm được tùy chọn trong chương trình, thông thường bán kính tâm xấp xỉ bán kính một cụm là tối ưu.

Hàm quyết định Decide_where_to_go: Hàm này xác định xem tại mỗi bước lặp, mỗi con kiến sẽ đi đâu. Theo KANTS đã nói ở trên: chương trình sẽ sinh ra một số ngẫu nhiên q, nếu q < q0. Chương trình sẽ chọn một điểm (x, y) trên lưới sao cho khoảng cách Ơ clit vector giữa vector của con kiến với vector trên ma trận trọng tâm (x, y) cho hàm xác xuất nhỏ nhất.

Hàm updateVector: cập nhật các vector xung quanh con kiến theo vector của nó.

Hàm centroid_calculate: tính lại ma trận trọng tâm sau mỗi bước lặp.

Hàm vote_cell: gán nhãn cho mỗi ô trên lưới dựa vào khoảng cách Ơ clit

Hàm read_patterns: đọc các mẫu vào

Hàm main: trước hết chương trình đọc các tham số vào, đọc file mẫu vào, khởi tạo lưới với các trọng số ngẫu nhiên. Đặt các con kiến ngẫu nhiên trên lưới. Sau đó, tại mỗi bước lặp, chương trình tính ma trận trọng tâm, xác định các bước đi tiếp theo cho mỗi con kiến, cập nhật các môi trường xung quanh, bay hơi mùi đến khi thuật toán đạt điều kiện dừng.

Chương trình cho thuật toán k láng giềng gần nhất đơn giản hơn vì chỉ có hàm đọc dữ liệu vào và hàm vote để tính toán độ chính xác phân lớp.

4.2. Chuẩn bị dữ liệu kiểm tra

Các tập dữ liệu được sử dụng để kiểm tra và kiểm chứng mô hình là những cơ sở dữ liệu thế giới thực quen thuộc được lấy từ UCI Machine Learning repository (http://archive.ics.uci.edu/ml/). IRIS chứa các dữ liệu gồm 3 loài hoa iris(Iris Setosa, Versicolo và Virginica), 50 mẫu mỗi loại và 4 thuộc tính số học (độ dài và độ rộng của lá và cánh được đo bằng cm). GLASS chưa các dữ liệu từ các loại ống nhòm khác nhau trong nghành tội phạm học. Có 6 lớp với 214 mẫu (được phân bố không đều nhau trong các lớp) và 9 đặc tính số học liền quan đến thành phần hóa học của thủy tinh. PIMA (cơ sở dữ liệu bệnh đái đường của Ấn độ) chứa các dữ liệu liên quan đến một số bệnh nhân và một nhãn lớp biểu diễn các chuẩn đoán bênh đái đường theo tiêu chuẩn của tổ chức y tế thế giới. Có 768 mẫu với 8 thuộc tính số học (dữ liệu thành phần hóa học).

Với mỗi cơ sở dữ liệu, 3 tập được dựng lên bằng việc chuyển dữ liệu gốc thành 3 tập rời nhau có cùng kích cỡ. Phân bố của lớp gốc vẫn được bảo toàn trong mỗi tập hợp. Vậy 3 cặp các tập dữ liệu traning-test được tạo ra bằng cách chia mỗi tập thành 2; chúng được đặt tên là 50tran-50tst (nghĩa là một nửa để huấn luyện và một nửa để kiểm tra). Và, 3 cặp khác được tạo ra nhưng phân bố gồm 90% mẫu cho huấn luyện và 10 % để kiểm tra. Những tập này được đặt tên là 90tra-10tst. Để phân lớp với KANTS, một than số nữa cần là: số các lân cận cần so sánh với mẫu kiểm tra. Theo cách này, thuật toán tìm kiếm K

vector gần nhất trong lưới (sử dụng khoảng cách Ơclit) tới vector tương ứng với mẫu muốn phân lớp. Nó gán lớp cho mẫu này là lớp của phần lớp các vector kia tìm được.

Nói cách khác ta đang sử dụng phương pháp K-Nearest Neihbours (KNN – hay K láng giềng gần nhất), nhưng trong trường hợp này ta sử dụng đồng thời cho cả việc gán nhãn neural và tìm nhãn lớp của dữ liệu kiểm tra và nhiều lần thuật toán làm việc tốt thậm chí với K = 1. Với K = 10, ta có bảng so sánh giữa KANTS và KNN với các tập dữ liệu khác nhau như sau:

Tập dữ liệu	KANTS	KNN
IRIS (9-1)	86.6666%	86.6666%
PIMA(9-1)	72.7272%	71.4286%
GLASS(9-1)	54.5454%	50.00%
IRIS(5-5)	89.3333%	94.6667%
PIMA(5-5)	70.833332%	73.4375%
GLASS(5-5)	59.090908%	51.4019%

Sử dụng cách tiếp cận thống kê, chạy 10 lần với các cặp các tập dữ liệu (huấn luyện và test). Thu được các kết quả phân loại tốt nhất và làm phép thống kê. Khi so sánh với phương pháp kinh điển ta thấy KANTS nổi trội hơn hẳn nếu chọn các hệ số tốt.

4.3. Sự phụ thuộc chất lượng thuật toán vào các tham số:

Các tham số có ảnh hưởng rất lớn đến chất lượng của thuật toán, việc chọn tham số sao cho đúng thường rất khó, phụ thuộc vào đặc điểm của mẫu dữ liệu huấn luyện: số mẫu, số lớp…

Sau đây ta sẽ xét tiến hành thí nghiệm để xem các tham số ảnh hướng đến kết quả như thế nào

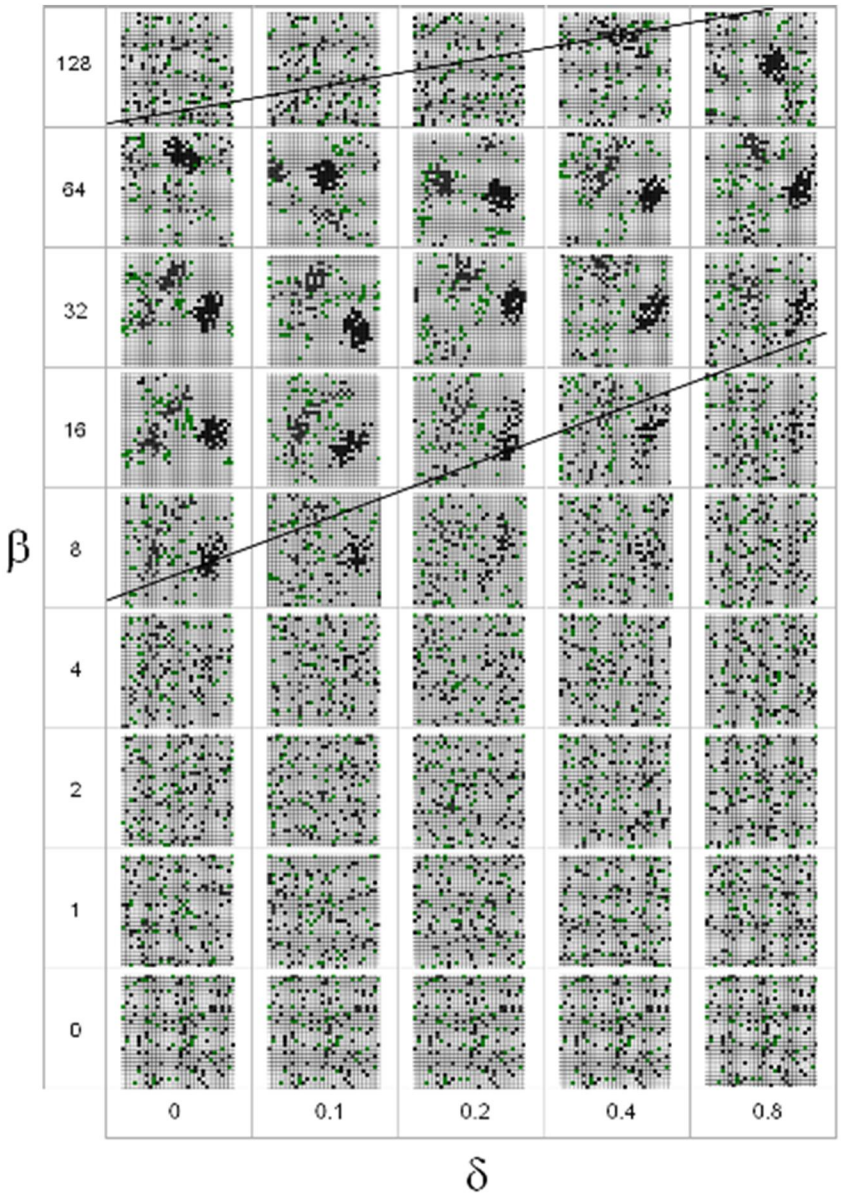
4.3.1. β-δ – Độ ngẫu nhiên theo mùi:

Trong [9] các tác giả đã thực hiện so sánh phân bố của các con kiến trong AS, với các cặp β-δ trong không gian tham số khác nhau. Ba loại hành vi đã quan được quan sát khi nhìn vào bức ảnh chụp của hệ thống sau khi lặp lại 1000 lần: rối hoạn, vá lỗ hổng và tạo đường mòn. Rối loạn là trạng thái mà các cụm chưa được phân, đây là trạng thái của hệ thống khi bắt đầu học, ta không thể nhìn được ra cụm trong trạng thái này. Vá lỗ hổng là giai đoạn các cụm được hình thành chưa rõ ràng (chưa tròn), còn có những "lỗ hổng" trong cụm, đây là trạng thái của hệ thống khi nó đang học sau một số bước nào đó. Tạo đường mòn là giai đoạn mà các vệt mùi đã được hình thành rõ nét, các cụm đã được phân bố tương đối rõ, các con kiến đi theo "đường mòn" này để cụm lại với nhau.

Dưới đây là biểu đồ thể hiện sự phân bố phụ thuộc vào hai tham số β-δ . Các tham số α *(nr) và (cr)* được lấy lần lượt là: 1, 1 và 3.

Nhìn vào biểu đồ ta có thể thấy được là: tham số lý tưởng để việc phân cụm diễn ra nhanh là: $\beta \sim$ 32-64 và δ gần như phụ thuộc tuyến tính vào β với $\delta \sim 0 - 0.4$

Quá trình làm thí nghiệm để rút ra điều kiện để các tham số tối ưu, chúng tôi thu được bảng thể hiện sự phân bố của kiến như sau ():



Hình 20: Sự phân cụm của kiến theo tham sô

Nhìn vào biểu đồ trên ta có thể thấy được 3 quá trình như đã miêu tả ở trên.

Dựa vào kết quả này, KANTS đã chỉ ra một công cụ phân cụm hiệu quả đầy hứa hẹn. Với các tham số *β-δ* được khởi tạo hợp lí, các dữ liệu được biểu diễn bằng các con kiến tạo nên các cụm, các cụm này có thể dễ dàng phân biệt trong lưới. Trong thực tế ta cần sử dụng một số loại tìm kiếm địa phương để xử lí gán nhãn cho một ô trong lưới dựa vào khoảng cách Ơ clit và gán tìm k ô trên lưới gần với một mẫu dữ liệu test nhất.

Phương pháp k láng giềng gần nhất đã đủ tốt do quá trình huấn luyện đã "làm mịn" các dữ liệu vào, xong chưa hoàn toàn "mịn" hẳn, ở cuối chương này chúng tôi có giới thiệu phương pháp học tập hợp để cải thiện hiệu quả của thuật toán khi các quá trình huấn luyện không thể làm mịn và tăng thêm độ chính xác

4.3.2. Tham số k trong thuật toán k láng giềng gần nhất:

K là hằng số xác định số các lân cận được dùng trong thuật toán KNN, thực tế sau khi lưới KANTS đã được huấn luyện, k = 1 vẫn cho kết quả đủ tốt. Nguyên nhân là do sự phân cụm của các neural đã làm giảm đáng kể nhiễu. Tuy nhiên, nếu số neural trên lưới nhỏ mà số cụm (số nhãn lớp của dữ liệu huấn luyện) lại lớn thì bán kính các cụm lại nhỏ, do đó nếu chọn k lớn sẽ có nhiễu, do đó sai sốlớn đáng kể.

Ta có bảng thống kê sau:

k	1	2	3	4	5
Iris(9-1)	86.6666 %	86.666664%	93.333336%	93.333336%	93.333336%
Pima	65.62500%	64.843750%	69.791672%	69.010422%	70.833328%
Glass	36.363636%	63.636364%	59.090908%	59.090908%	54.545456%

4.3.3. Kích thước lưới:

Bảng thống kê khảo sáo sự thay đổi theo kích thước lưới:

Kích thước	30x50	35x50	50x50	80x50	100x100
Iris	80.000000%	80.000000%	86.666664%	86.666664%	80.000000%
Pima	67.968750%	67.447922%	63.541668%	68.489578%	66.666672%
Glass	40.909092%	54.545456%	59.090908%	63.636364%	59.090908%

4.3.4. Bán kính lân cận:

Bảng thống kê với bán kính lân cận thay đổi:

nr	1	2	3	4
Iris(9-1)	86.666664%	80.000000%	86.666664%	86.666664%
Pima(9-1)	68.831169%	68.831169%	66.233765%	67.532471%
Glass(9-1)	31.818182%	45.454548%	50.000000%	59.090908%

4.3.5. Tham số q0:

Tham số này điều khiển sự cân bằng giữa khai thác và khám phá. Nghĩa là khả năng một con kiến sẽ chọn đường đi mới để tìm ra cụm mới hay tiếp tục đi đường đi có nồng độ mùi cao hơn. Nhìn chung tham số q0 không ảnh hưởng nhiều lắm đến kết quả phân loại với các tập dữ liệu nhỏ như trong chương trình

4.3.6. Tham số bán kính trọng tâm cr:

Bán kính trọng tâm, ảnh hưởng rất nhiều đến thời gian chạy của thuật toán, nếu cr nhỏ, thời gian chạy thuật toán sẽ nhỏ nhưng các cụm cũng sẽ nhỏ, khả năng con kiến đi xa sẽ thấp, điều này làm lưới xuất hiện nhiều cụm bé và cho kết quả phân lớp kém chính xác. Tuy nhiên cr không được quá lớn, nếu cr lớn, thời gian chạy thuật toán sẽ lớn mà các cụm vừa được hình thành đã bị xé ra…

Cr	1	2	3	4	5
Iris(9-1)	86.666664%	86.666664%	86.666664%	86.666664%	86.666664%
Pima(9-1)	63.636364%	63.636364%	63.636364%	63.636364%	63.636364%
Glass(9-1)	59.090908%	59.090908%	59.090908%	59.090908%	59.090908%

4.3.7. Tham số bay hơi

Tham số thể hiện tốc độ bay hơi mùi, nếu tốc độ bay hơi lớn, vector của các ô trên lưới sẽ dễ tiến về (0,..0), tức là gần với class có vector trọng số nhỏ mà các con kiến chưa kịp cập nhật. Nếu tốc độ bay hơi nhỏ, các vệt mùi khó được hình thành, không có nhiều thông tin học tăng cường.

4.3.8. Số lần lặp tối thiểu và cách xác định điều kiện dừng của thuật toán:

Điều kiện dừng thuật toán tại bước lặp t được xác định là khi hình dạng của lưới không thay đổi sau bước lặp t + 1. Nghĩa là có lặp thêm nữa cũng không thay đổi dạng của lưới, thực tế điều này rất khó xảy ra vì đồng thời trên lưới xảy ra hai hành động trái ngược nhau: bay hơi mùi và cập nhật mùi. Hai hành động này bù trừ nhau và khiến lưới luôn không ổn định. Tuy nhiên khi sự thay đổi là đủ nhỏ, ta có thể xem như lưới đủ ổn định, xác định sự ổn định này bằng cách tính khoảng cách Ơ clit vector giữa vector của con kiến và vector mà con kiến đang ở.

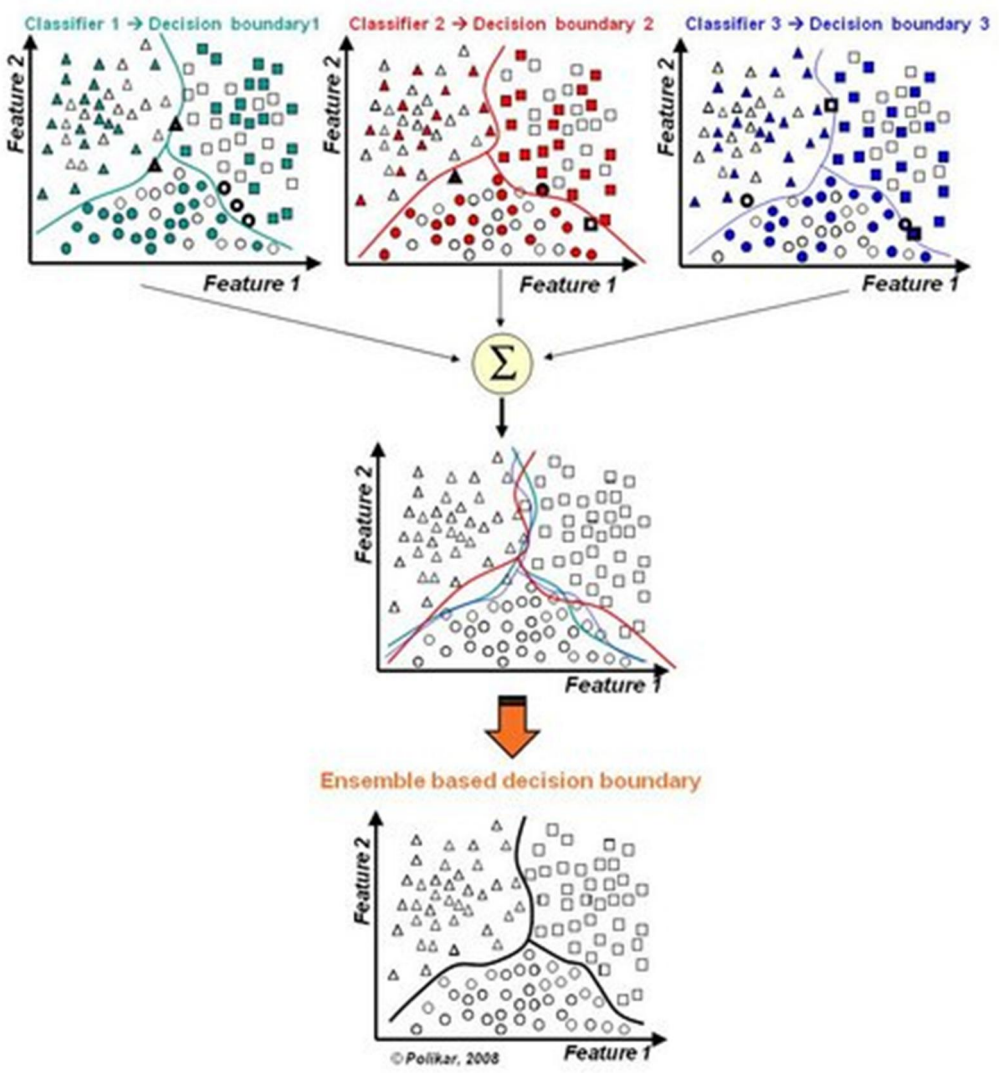
4.4. Mở rộng của KANTS:

Trong thực tế khi thực hiện gán nhãn cho các ô và cho các dữ liệu test, ta đã thực hiện thuật toán k láng giềng gần nhất (KNN), tuy nhiên k láng giềng gần nhất có nhược điểm là trong một số trường hợp các dữ liệu nhiễu làm sai kết quả. Để làm giảm sự ảnh hưởng của nhiễu, ta sử dụng Emsembler learning cho KNN, tức là tiến hành bỏ phiếu với các k thay đổi và dựa trên kết quả này, tìm nhãn lớp được bỏ nhiều nhất sau mỗi giá trị của k, rồi gán cho nhãn lớp này.

4.4.1. Giới thiệu Ensembler learning:

Ensembler learning là quá trình học tập hợp mà nhiều mô hình hoặc nhiều bộ dữ liệu huấn luyện được sử dụng trong phân loại, là chiến lược kết hợp để sinh ra một bộ các kết quả, kết hợp các kết quả này để sinh ra kết quả cuối cùng. Ensembler learning chủ yếu được sử dụng để cải thiện (phân loại, dự báo, xấp xỉ…) hiệu suất của một mô hình, hoặc làm giảm khả năng lựa chọn không may của một mô hình kém chính xác.

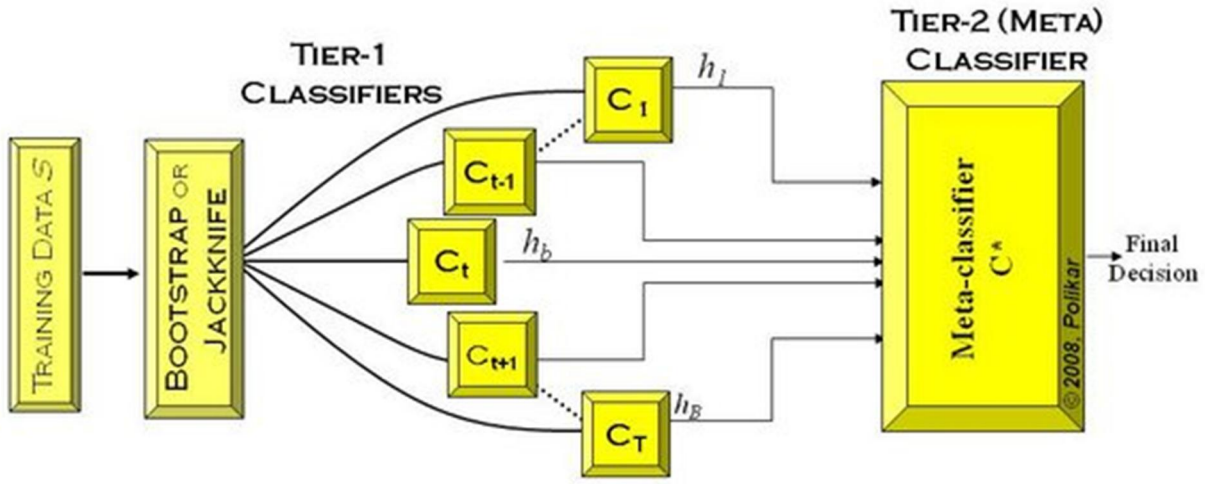
Mô hình trực quan như sau:



Hình 21: Mô hình trực quan giải thích học tập hợp

Giải thích sơ đồ: với mỗi mô hình (phương pháp) cho ta một lời giải (đường biên phân lớp) khác nhau, tất cả đều có chung một nhược điểm là có sai số, ta cần giảm thiểu tối đa các sai số này, lẽ di nhiên các phương pháp trên đều không thể cải thiện thêm nữa, tuy nhiên nếu kết hợp các kết quả của các phương pháp trên thì theo tư tưởng thống kê, lời giải kết hợp cho kết quả đáng tin cậy hơn. Tức là, trong sơ đồ trên, đường biên là gộp chung các đường biên cho kết quả tin cậy hơn cả.

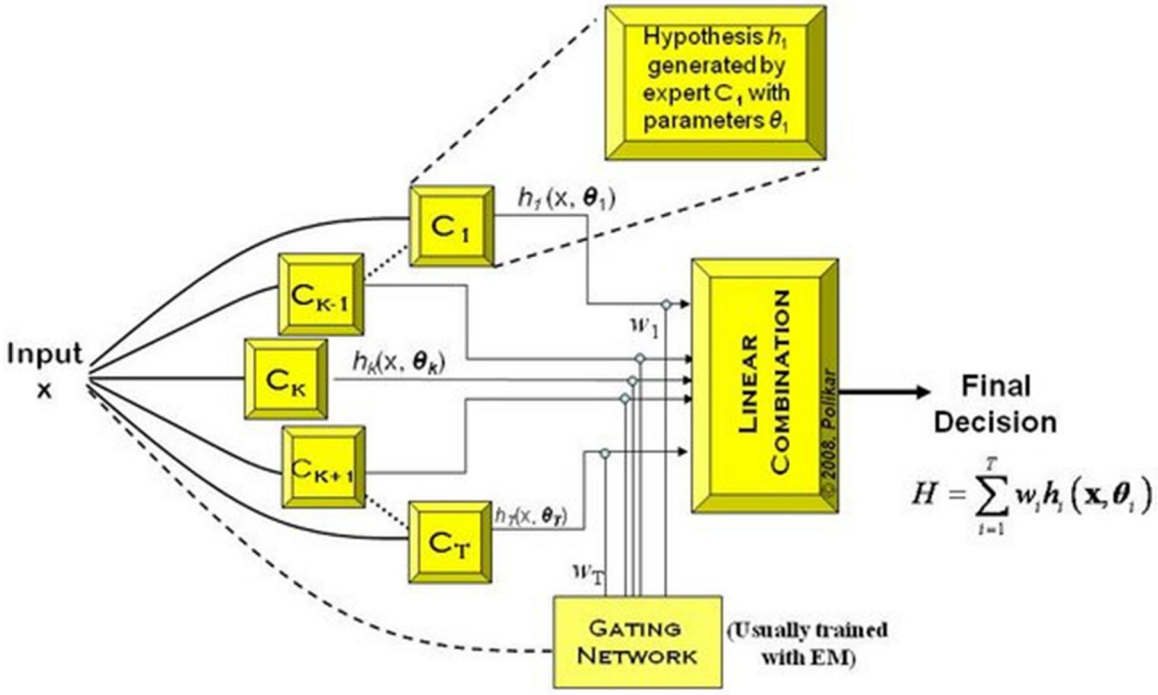
Sơ đồ thuật toán:



Hình 22: Mô hình nguyên lý học tập hợp

Việc kết hợp các bộ học Ci cho ta kết quả cuối cùng

Ngoài ra còn có học tập hợp kết hợp mô hình chuyên gia, nghĩa là với mỗi mô hình được kết hợp với một trọng số thể hiện độ chính xác để tăng cường các bộ tốt. Do tính phức tạp nên khoa luận này chỉ đưa ra mô hình. Mô hình như sau:



Hình 23: Ensembler learning với hỗ trợ mô hình chuyên gia

4.4.2. Áp dụng ensembler learning vào bài toán phân lớp với KANTS:

Có hai gian đoạn mà ta có thể áp dụng học tập hợp ensembler learning vào bài toán này.

Thứ nhất: giai đoạn gán nhãn cho ô: việc gán nhãn cho một ô i trong lưới là việc áp dụng phương pháp k láng giềng gần nhất để tìm ra nhãn lớp được bỏ phiếu nhiều nhất, kết quả của nhãn lớp này sẽ được gán cho ô đó. Áp dụng học tập hợp, thay vì gán luôn cho ô đó, ta chọn N bộ kết quả, tức là chọn cho k = 1,N. Áp dụng phương pháp k láng giềng gần nhất với mỗi k để tìm ra K nhãn được bỏ phiếu, chọn nhãn được bảo phiếu nhiều nhất trong N bộ này và gán nhãn này cho ô đó. Vậy việc gán nhãn này là hai lần bỏ phiếu, nhãn được gán chính là nhãn đã qua vòng hai.

Thứ hai: giai đoạn tìm nhãn cho một mẫu dữ liệu (phân lớp): Việc gán nhãn cũng được tiến hành tương tự như giai đoạn một nhưng thay vì gán nhãn cho ô, ta gán nhãn cho mẫu dữ liệu và thay vì tính khoảng cách với các con kiến, ta tính khoảng cách với các ô. Độ chính xác của thuật toán cũng được tính tương tự.

Kết quả so sánh giữa thuật toán cũ và mới:

Kiểu học thuật toán	KANTS Với KNN	KANTS với Ensembler learning
Iris(9-1)	86.666664%	93.333336%
Pima(9-1)	72.727272%	74.025978%
Glass(9-1)	45.454548%	54.545456%

Nhận xét:

Nhìn chung **ensembler learning** có cải thiện thuật toán và cho kết quả tốt hơn KANTS thông thường, việc cải thiện nhiều hay ít phụ thuộc vào việc chọn tham số và bộ dữ liệu huấn luyện. Tuy nhiên trong trường hợp lưới KANTS đủ "mịn" thì việc N quá lớn sẽ làm sai số tăng lên. Nếu N = 1 thì thuật toán trở về dạng ban đầu với k = 1.

CHƯƠNG 5: KẾT LUẬN

Khóa luận này đã trình bày thuật toán KohonAnts (hay còn gọi là KANTS), một phương pháp mới cho việc phân lớp dữ liệu, dựa trên sự kết hợp giữa các thuật toán kiến và bản đồ tự tổ chức của Kohonen. Mô hình này đưa các mẫu dữ liệu n-biến vào trong các con kiến nhân tạo trong lưới xuyến 2D với các vector n-chiều. Dữ liệu/kiến được di chuyển trên lưới để tạo ra sự khác biệt về mặt dữ liệu, từ đó các cụm được hình thành. Quá trình di chuyển của các con kiến dần dần sẽ tạo ra độ mịn của lưới. Khi lưới đủ ổn định, các con kiến có thể dừng và ta tiến hành gán nhãn cho các ô trên lưới.

Lưới sau khi đã gán nhãn giống như lưới SOM đã huấn luyện, là một công cụ để phân lớp tốt hơn rất nhiều các công cụ thông thường khác.

Khóa luận cũng đồng thời chỉ ra việc kết hợp KANTS với phương pháp học tập hợp cho kết quả rất khả quan.

Tuy nhiên hiệu quả của KANTS khi phân lớp các dữ liệu phức tạp, nhiều biến, nhiều lớp mặc dù tốt hơn KNN xong vẫn còn nhiều hạn chế. Việc chọn các hệ số thích hợp là khá khó khăn nhưng chắc chắn cho kết quả tốt hơn KNN.

Tham khảo:

1. KohonAnts: A Self-Organizing Ant Algorithm for Clustering and Pattern Classification: C. Fernandes1,2, A.M. Mora2, J.J. Merelo2, V. Ramos1,J.L.J. Laredo
2. KANTS: Artificial Ant System for classification: C. Fernandes1,2, A.M. Mora2, J.J. Merelo2, V. Ramos1,J.L.J. Laredo
3. Self-organizing maps: http://en.wikipedia.org/wiki/Self-organizing_map
4. Ensemble learning: http://en.wikipedia.org/wiki/Ensemble_learning
5. K-nearest neibourds algorithm: http://www.scholarpedia.org/article/K-nearest_neighbor
6. Ant Colony Optimization: http://en.wikipedia.org/wiki/Ant_colony_optimization
7. Theodoridis S., Koutroumbas K. Pattern Recognition.3rd.ed.(AP, 2006)
8. Artificial neural network: http://en.wikipedia.org/wiki/Artificial_neural_network
9. Swarn Chialvo, D.R., Millonas, M.M., “How Swarms build Cognitive Maps”
10. http://www.scholarpedia.org/article/Ensemble_learning