การแบ่งกลุ่มรูปแบบโดยใช้ครอสคอร์รีเลชันนิวรอลเน็ตเวิร์ค

PATTERN CLASSIFICATION USING CROSS-CORRELATION NEURAL NETWORK

นโรดม กล่อมเอี่ยม

NARODOM KLOMIAM

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

สาขาวิชาเทคโนโลยีสารสนเทศ

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2547

ISBN 974-0680-95-2

การแบ่งกลุ่มรูปแบบโดยใช้ครอสคอร์รีเลชันนิวรอลเน็ตเวิร์ค

PATTERN CLASSIFICATION USING CROSS-CORRELATION NEURAL NETWORK

นโรดม กล่อมเอี่ยม
NARODOM KLOMIAM

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาเทคโนโลยีสารสนเทศ
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2547

ISBN 974-9680-95-2

PATTERN CLASSIFICATION USING CROSS-CORRELATION NEURAL NETWORK

NARODOM KLOMIAM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF
MASTER OF ENGINEERING IN INFORMATION TECHNOLOGY
SCHOOL OF GRADUATE STUDIES
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2004

ISBN 974-9680-95-2

หัวข้อวิทยานิพนธ์	การแบ่งกลุ่มรูปแบบโดยใช้ครอสคอร์รีเลชันนิวรอลเน็ตเวิร์ค
นักศึกษา	นายนโรดม กล่อมเอี่ยม
รหัสประจำตัว	44067048
ปริญญา	วิทยาศาสตรมหาบัณฑิต
สาขาวิชา	เทคโนโลยีสารสนเทศ
พ.ศ.	2547
อาจารย์ผู้ควบคุมวิทยานิพนธ์	ผศ.ดร.อาริต ธรรมโน

บทคัดย่อ

การแบ่งกลุ่มรูปแบบ (Pattern Classification) มีบทบาทที่สำคัญยิ่งต่อการพัฒนาระบบเชิงอัตโนมัติในหลากหลายสาขา อัลกอริทึมที่ใช้ในการแบ่งกลุ่มที่อยู่บนฐานของการเปรียบเทียบระหว่างแพทเทิร์นอ้างอิง (reference pattern) กับแพทเทิร์นทดสอบ (test pattern) มีฟังก์ชันเฉพาะที่นิยมใช้ในการวัดความคล้ายกัน (similarity) ระหว่างแพทเทิร์น ได้แก่ ระยะทาง (Euclidean Distance) แต่เนื่องจากค่าระยะทางเป็นค่าการวัดที่ไม่เหมาะสมสำหรับชุดข้อมูลที่ให้ความสำคัญกับรูปร่างของแพทเทิร์นโดยไม่คำนึงถึงขนาด ดังนั้นจึงได้เสนอการนำฟังก์ชันครอส-คอร์รีเลชัน (Cross-correlation) ซึ่งมีความเหมาะสมกับชุดข้อมูลดังกล่าวมาใช้ในการพัฒนาโครงข่ายประสาทเทียมชนิดใหม่ นอกจากนี้ในโครงข่ายประสาทเทียมชนิดใหม่นี้ ยังได้ปรับปรุงอัลกอริทึมในการเรียนรู้ให้สามารถแก้ไขปัญหาที่เกิดจาก ผลกระทบของลำดับของอินพุตแพทเทิร์นในเทรนนิงเซต (training set) ปัญหาความเหมาะสมของแพทเทิร์นอ้างอิง และยังช่วยในการลดจำนวนของแพทเทิร์นอ้างอิงได้อีกด้วย

Thesis	Pattern Classification Using Cross-Correlation Neural Network
Student	Mr.Narodom Klomiam
Student ID	44067048
Degree	Master of Science
Programme	Information Technology
Year	2004
Thesis Advisor	Asst.Prof.Dr.Arit Thammano

ABSTRACT

Pattern Classification is a key element in many automatic systems. Algorithms based on the matching approach have their own discrimination function for similarity measure between reference and unknown pattern. Euclidean Distance, which is the most popular function, can be used to quantify the similarity between two patterns. However, the use of Euclidean distance is not suitable for some dataset that the patterns can be distinguished from its shape and are not depend on its magnitude. In this research, the new neural network architecture, which employs the Cross-correlation function, is proposed. In addition, a new training scheme used to train the proposed network can also fix the problem of the sequence of input pattern in the training set, the problem of an appropriate reference pattern and also help in reducing the number of reference pattern.

กิตติกรรมประกาศ

การวิจัยนี้เสร็จสมบูรณ์ได้จากความช่วยเหลือของหลายๆฝ่าย โดยเฉพาะอย่างยิ่งอาจารย์ที่ปรึกษา คือ ผศ. ดร. อาริต ธรรมโน ผู้ซึ่งให้คำแนะนำ และชี้แนะแนวทางที่เป็นประโยชน์มาโดยตลอดในทุกๆ เรื่อง ผู้จัดทำรู้สึกขอบพระคุณเป็นอย่างสูง

ขอขอบคุณบิดา-มารดา ที่ให้ความอนุเคราะห์ทางด้านทุนทรัพย์ตลอดการศึกษา การทำวิจัย และคอยให้กำลังใจอยู่เสมอ

ขอขอบคุณเจ้าหน้าที่ฝ่ายต่างๆ ของคณะเทคโนโลยีสารสนเทศ และสำนักหอสมุดกลาง ที่ให้ความช่วยเหลือ และอำนวยความสะดวกในหลายๆ ด้าน ระหว่างการทำวิจัย

ขอขอบคุณ คุณวิริยาภรณ์ เอกผล ที่ให้ความช่วยเหลือทางด้านเอกสารและคอยเป็นกำลังใจ

สุดท้าย ขอขอบคุณเพื่อนๆ ร่วมรุ่น IS 11 ภาคปกติ ที่ให้คำปรึกษาในหลายๆ เรื่อง ซึ่งทำให้ผู้จัดทำรู้สึกกระจ่าง และแนวคิดของทฤษฎีหลายๆ ทฤษฎี รวมทั้งช่วยให้รู้สึกมีกำลังใจทำวิจัยเมื่อยามที่ต้องการ

นโรดม กล่อมเอี่ยม

สารบัญ

หน้า

สารบัญ

หน้า

สารบัญ

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป

บทที่ 1
บทนำ

1.1 ประวัติความเป็นมา

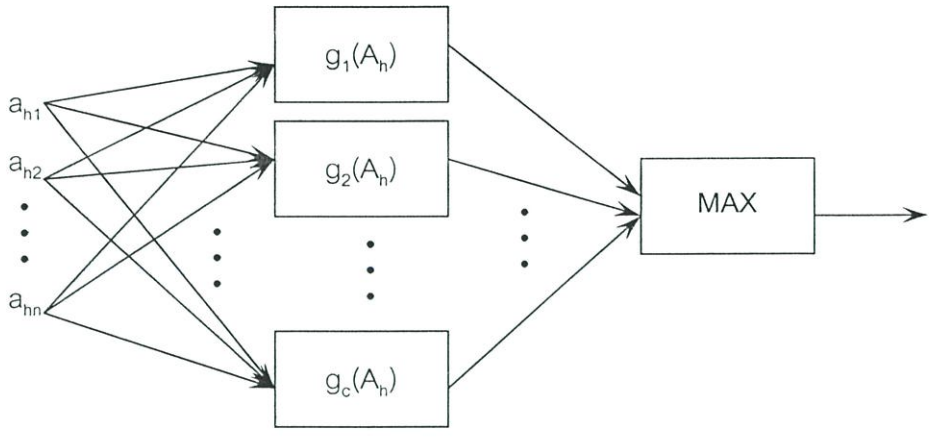
การแบ่งกลุ่มรูปแบบ (Pattern Classification) เป็นส่วนหนึ่งที่มีความสำคัญต่อการแก้ปัญหาต่างๆ โดยเฉพาะอย่างยิ่งทางวิศวกรรม ระบบโซนาร์ (sonar) เรดาร์ (radar) ระบบที่เกี่ยวกับแผ่นดินไหว (seismic) หรือการตรวจวินิจฉัยโรค ต่างก็ต้องการความสามารถในการแบ่งกลุ่มของสถานการณ์หรือเหตุการณ์ได้อย่างถูกต้อง แม่นยำ นอกจากนี้ ระบบควบคุม ระบบติดตาม หรือระบบการทำนาย ก็มักจะต้องใช้ตัวแบ่งกลุ่ม (classifier) ในการบอกถึงความสัมพันธ์ระหว่างอินพุตและเอาท์พุต จะเห็นได้ว่า pattern classification สามารถนำไปประยุกต์ใช้กับระบบอัตโนมัติต่างๆ ได้อย่างมากมาย

ความยืดหยุ่นของทฤษฎีฟัซซีเซต (Fuzzy set) และประสิทธิภาพทางด้านการคำนวณของนิวรอลเน็ตเวิร์คมีการพิสูจน์มาแล้วมากมายกับปัญหาทางด้านนี้ การนำทฤษฎีทั้งสองมารวมกันในชื่อของ Neuro-fuzzy เป็นที่สนใจอย่างมาก มีงานวิจัยจำนวนมากมายที่นำ Neuro-fuzzy มาใช้กับปัญหาของ pattern classification และ clustering

Kulkarni, Sontakke และ Randale [1] ได้เสนอ Fuzzy Hyperline Segment Neural Network สำหรับการรู้จำลายมือเขียนที่ไม่ขึ้นกับการหมุน (rotation invariant handwritten character recognition) โดยการใช้ฟัซซีเซตเป็นแพทเทิร์นคลาส และแต่ละฟัซซีเซตคือผลรวม (union) ของแต่ละ fuzzy set hyperline ค่า membership function ที่ใช้จะแปรผกผันกับระยะห่างระหว่างเส้นตรง (hyperline) กับแพทเทิร์น

Patrick Simpson [2] ได้เสนอ Fuzzy Min-Max Neural Network สำหรับ classification อัลกอริทึมนี้มีหลักการคล้ายกับ Fuzzy Hyperline Segment แต่มีการใช้ hyperbox แทน hyperline Kulkarni, Doye และ Sontakke [3] เสนอ General Fuzzy Hypersphere Neural Network ในลักษณะเดียวกันกับสองอัลกอริทึมแรก แต่ใช้ hypersphere แทนฟัซซีเซต และใช้ค่าระยะทางในการกำหนด membership function ของแต่ละ hypersphere

รูปแบบหนึ่งของ pattern classification สามารถแสดงให้เห็นได้ดังรูปที่ 1.1 [4] อินพุต-แพทเทิร์น A_h ถูกส่งผ่านเข้าไปในฟังก์ชันแบ่งแยก (discrimination function) ทั้ง m ตัว และฟังก์ชันตัวที่ให้ค่าเอาท์พุตสูงสุดออกมา จะถูกเลือกให้เป็นคำตอบ



รูปที่ 1.1 ระบบ Pattern Classification

มีวิธีการที่แตกต่างกันมากมายในการนำฟังก์ชันแบ่งแยกแบบต่างๆมาใช้งาน เช่น การใช้ค่าระยะทาง (Euclidean distance) ระหว่างจุด การเปรียบเทียบทางบูลลีน (boolean) หรือการใช้กฎ (rule) ต่างๆ เป็นต้น ขึ้นอยู่กับชนิดของแอปพลิเคชัน ฟังก์ชันแบ่งแยกแต่ละตัวก็มีความเหมาะสมกับแอปพลิเคชันที่ต่างกัน

1.2 วัตถุประสงค์

1. เพื่อศึกษาอัลกอริทึมในการแบ่งกลุ่มแบบ Neuro-fuzzy โดยเน้นไปที่โครงข่ายแบบ Fuzzy Hypersphere Neural Network
2. เพื่อศึกษาถึงโครงสร้างที่ประกอบขึ้นเป็นระบบ Pattern Classification
3. เพื่อประยุกต์ใช้ฟังก์ชันแบ่งแยก (discrimination function) อื่น ในการพัฒนาโครงข่ายชนิดใหม่ โดยใช้ฟังก์ชันครอสคอร์รีเลชัน (Cross-correlation) เป็นพื้นฐาน
4. เพื่อเป็นพื้นฐานในการพัฒนาความสามารถของ classifier ต่อไป

1.3 ข้อจำกัด และขอบเขตของการวิจัย

1. การวิจัยนี้ใช้การทดลองแบบ off-line
2. ชุดข้อมูลมี่ใช้ในการทดลองประกอบด้วยชุดข้อมูลมาตรฐานที่ใช้ทดสอบประสิทธิภาพการแบ่งกลุ่มของอัลกอริทึมในกลุ่ม Classification และชุดข้อมูลเสียงที่ผู้วิจัยทำการดึงคุณลักษณะเอง

3. ชุดข้อมูลที่เลือก จะเน้นไปที่ชุดข้อมูลที่มีจำนวนคุณลักษณะที่ค่อนข้างมาก และมีลักษณะเป็นอนุกรมทางเวลา (time series) ที่แต่ละคุณลักษณะในแพทเทิร์นมีความสัมพันธ์กัน ซึ่งไม่เหมาะสมกับการวัดความคล้ายด้วยค่าระยะทาง นอกจากนี้ยังได้ทำการทดลองเพิ่มเติมกับชุดข้อมูลอื่นๆ เพื่อดูประสิทธิภาพของครอสคอร์ลีเรชัน-นิวรอลเน็ตเวิร์ค เมื่อคุณลักษณะของชุดข้อมูลไม่มีความสัมพันธ์กันอีกด้วย

1.4 การจัดเรียงหัวข้อวิทยานิพนธ์

บทที่ 1 บทนำ กล่าวถึง ความเป็นมาและวัตถุประสงค์

บทที่ 2 Pattern Classification กล่าวถึง บทบาทความสำคัญ ลักษณะ รวมถึงส่วนประกอบต่างๆ ของระบบ classification

บทที่ 3 General Fuzzy Hypersphere Neural Network เป็นอัลกอริทึมหนึ่งที่ใช้เป็นพื้นฐานในการพัฒนา ในบทจะกล่าวถึงแนวคิดเบื้องต้น วิธีการทำงานของอัลกอริทึม รวมไปถึงตัวอย่างการใช้งาน

บทที่ 4 ครอสคอร์รีเลชันนิวรอลเน็ตเวิร์ค (Cross-Correlation Neural Network) เป็นอัลกอริทึมที่ได้พัฒนาขึ้นมาใหม่ เนื้อหาในบทจะกล่าวถึง ทฤษฎีคอร์รีเลชันฟังก์ชัน อัลกอริทึมในการทำงาน และตัวอย่างการใช้งาน

บทที่ 5 การทดลอง และผลการทดลอง

บทที่ 6 สรุปและวิจารณ์ผลการทดลอง

บทที่ 2
Pattern Classification

2.1 ความสำคัญของ Pattern Classification

Pattern Classification เป็นสาขาทางวิทยาศาสตร์ที่มีจุดมุ่งหมายที่จะทำการแบ่งกลุ่ม (Classification) ของวัตถุ (object) ออกมาเป็น category หรือคลาส (class) ทั้งนี้จะขึ้นอยู่กับชนิดของแอปพลิเคชัน วัตถุสามารถเป็นได้ทั้งรูปภาพ รูปคลื่นสัญญาณ หรือ ค่าการวัด (measurement) ใดๆ ที่ต้องการจะทำการแบ่งกลุ่ม โดยทั่วไปแล้ววัตถุที่ได้กล่าวถึงนั้นจะรู้จักกันในชื่อของแพทเทิร์น (Pattern)

Pattern Classification/Recognition มีประวัติความเป็นมาที่ยาวนาน แต่ในช่วงก่อนปี 1960 งานทางด้านนี้โดยส่วนใหญ่จะเป็นเพียงการวิจัยทางทฤษฎีในสาขาทางสถิติ การเกิดขึ้นของคอมพิวเตอร์ได้เพิ่มความต้องการที่จะใช้ระบบในเชิงปฏิบัติให้มากขึ้น และทำให้มีการพัฒนาทฤษฎีเพิ่มขึ้นด้วย

สำหรับการพัฒนาทางด้านอุตสาหกรรม ทั้งความต้องการทางด้านระบบอัตโนมัติในกระบวนการผลิตและความต้องการในการจัดการสารสนเทศ (information handling) และ information retrieval ได้เข้ามามีความสำคัญมากขึ้นเรื่อยๆ แนวโน้มอันนี้ได้ทำให้ pattern classification กลายมาเป็นหัวข้อในการวิจัยและพัฒนาแอปพลิเคชันทางวิศวกรรมที่มากขึ้นด้วย Pattern Classification มักจะเป็นส่วนหนึ่งที่ถูกรวมเข้าไปในระบบที่เรียกว่า machine intelligence system ที่ถูกสร้างขึ้นมาเพื่อใช้เป็นเครื่องมือในการช่วยตัดสินใจ

Machine vision เป็นเรื่องหนึ่งที่ pattern classification ได้เข้าไปมีส่วนสำคัญ ระบบ machine vision จะทำการจับภาพของวัตถุผ่านกล้อง และวิเคราะห์เพื่อที่จะสร้างคำอธิบายว่าภาพที่ได้เป็นภาพอะไร แอปพลิเคชันโดยทั่วๆไปของระบบ machine vision มักจะใช้ในอุตสาหกรรมการผลิต เช่น ในการตรวจสอบผลิตภัณฑ์ ผลิตภัณฑ์ที่ผ่านการตรวจสอบแล้ว อาจจะถูกลำเลียงทางสายพานและผ่านจุดตรวจซึ่งมีกล้องติดตั้งไว้ ภาพที่ได้จะถูกนำไปวิเคราะห์เพื่อที่จะบอกว่าผลิตภัณฑ์ชิ้นนั้นมีจุด บกพร่องตรงไหนบ้าง เป็นต้น

Character recognition ก็เป็นอีกเรื่องหนึ่งที่สำคัญของ pattern classification ซึ่งเกี่ยวข้องกับการจัดการสารสนเทศในแบบอัตโนมัติ ระบบในเชิงการค้าที่เป็นที่คุ้นเคยกันดีระบบหนึ่งได้แก่ การรู้จำตัวอักษร (Optical character recognition, OCR) ระบบ OCR จะมีอุปกรณ์ front end ที่ประกอบด้วย ตัวกำเนิดแสง เลนส์สำหรับการสแกน ตัวส่งข้อมูล และตัวตรวจจับ ด้วยตัวตรวจจับที่ไวต่อแสง ทำให้ความเข้มของแสง ที่จุดต่างๆถูกแปลไปเป็นปริมาณตัวเลขในลักษณะ

ของอาร์เรย์ที่ประกอบกันขึ้นเป็นรูปภาพ และใช้เทคนิคในการประมวลผลต่างๆ เพื่อทำการพรี-โปรเซสซิง (pre-processing) ข้อมูลก่อนที่จะนำไปผ่านกระบวนการ classification เพื่อที่จะแบ่งประเภทของตัวอักษร ตัวเลข หรือสัญลักษณ์ต่างๆ ออกไปเป็นชนิดของคลาสที่ถูกต้อง การจัดเก็บข้อมูลในรูปแบบของเอกสารที่ผ่านกระบวนการดังกล่าวจะมีข้อดีอยู่ 2 ข้อใหญ่ คือ อย่างแรก เอกสารสามารถที่จะนำไปประมวลผลเพิ่มเติมได้อีก ผ่านทางโปรแกรมประเภท word processor และ อย่างที่สอง คือ การเก็บข้อมูลในรูปแบบของ ASCII จะมีประสิทธิภาพมากกว่าการจัดเก็บในรูปแบบของรูปภาพ นอกจากระบบการรู้จำตัวพิมพ์แล้ว ก็ยังมีการใช้ประโยชน์จากการรู้จำตัวอักษรลายมือเขียน (handwritten) เช่น การนำระบบดังกล่าวมาใช้กับเครื่องมือตรวจสอบเช็คของธนาคาร โดยเครื่องมือนั้นต้องมีความสามารถในการรู้จำตัวหนังสือ และตัวเลขบนเช็คเพื่อให้แน่ใจว่าเป็นปริมาณเดียวกัน หรือตรวจสอบว่าเจ้าของบัญชีกับเลขที่บัญชีของผู้รับสอดคล้องกัน เป็นต้น

แอปพลิเคชันอื่นๆ ที่เกี่ยวข้องกับ character recognition ได้แก่ เครื่องมือในการเรียงจดหมายตามรหัสไปรษณีย์ที่ใช้ในที่ทำการไปรษณีย์ หรืออุปกรณ์ประเภท On-line handwriting recognition system ที่ช่วยพัฒนาส่วนติดต่อ (interface) ระหว่างมนุษย์และเครื่องจักร เป็นต้น

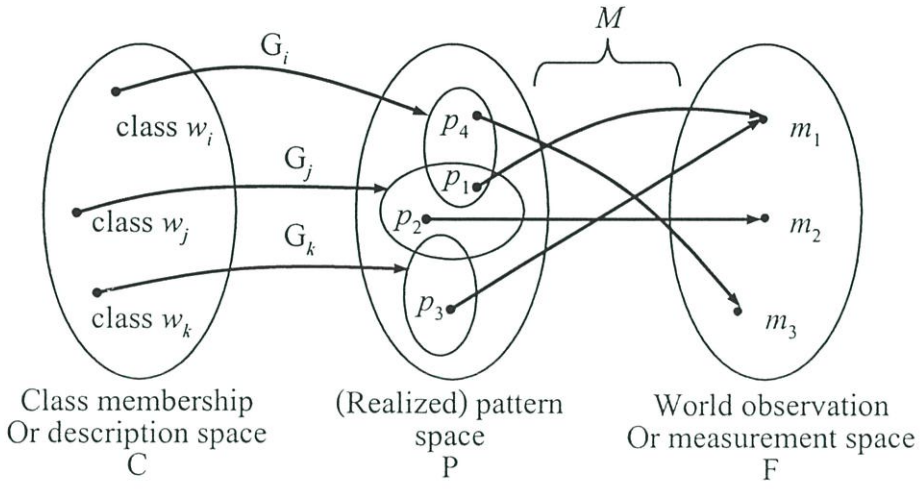
นอกจากที่ได้กล่าวไปแล้ว pattern classification ก็ยังนำไปใช้ประโยชน์อีกมากมายในหลายๆด้าน เช่น

- Computer – aided diagnosis
- Speech recognition
- Radar signal classification/ analysis
- Electrocardiographs signal analysis/ Understanding

2.2 ลักษณะของ pattern classification

พิจารณารูปที่ 2.1 ที่แสดงความสัมพันธ์ระหว่างสเปซ (space) 3 สเปซ อันประกอบด้วย Class membership space C ซึ่งเป็นสเปซที่บอกตำแหน่งของแต่ละชนิดคลาส Pattern space P เป็นสเปซที่บอกถึงตำแหน่งของแต่ละแพทเทิร์น รวมถึงขอบเขตของแพทเทิร์นที่มีชนิดคลาสเดียวกัน และ Measurement space F ซึ่งเป็นสเปซที่บอกถึงตำแหน่งของค่าการวัด (measurement) ที่สังเกต (observe) ได้จากวัตถุที่ต้องการแบ่งกลุ่ม การเชื่อมโยง (mapping) ระหว่างสเปซ C และ P จะกระทำผ่านความสัมพันธ์ G ซึ่งความสัมพันธ์นี้จะบอกได้ถึงลักษณะของแพทเทิร์นที่มีชนิดคลาสเดียวกัน ส่วนการเชื่อมโยงระหว่างสเปซ P และ F จะผ่านทางความสัมพันธ์ M

จะสังเกตได้ว่า ในบางครั้งขอบเขตของแพทเทิร์นที่ไม่ใช่ชนิดคลาสเดียวกันในสเปซ P มีโอกาสที่จะทับกัน (overlap) ได้ ทำให้เกิดความสับสนว่า บริเวณที่ทับกันคือส่วนของการเชื่อมโยงที่มาจากชนิดคลาสใด อีกทั้งในส่วนของค่าการวัดซึ่งมีโอกาสที่แพทเทิร์นมากกว่า 1 แพทเทิร์นจะมีค่าการวัดที่เหมือนกัน (เช่น p_1 และ p_3 ที่มีค่าการวัดเป็น m_1 เหมือนกัน) สิ่งเหล่านี้เป็นความซับซ้อนในการออกแบบระบบ Pattern Classification



รูปที่ 2.1 ความสัมพันธ์ในมุมมองของระบบ pattern generation/classification/interpretation

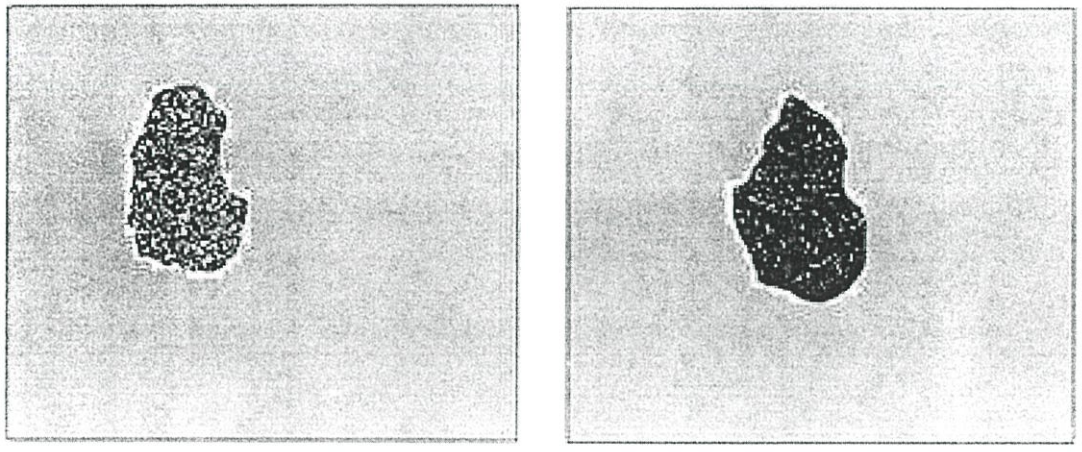
เราอาจจะมองแพทเทิร์นจริง (realized pattern) ในรูปที่ 2.1 ว่าเป็นวัตถุที่จะถูกนำไปวัดค่าพารามิเตอร์ใดๆ ที่จะทำให้แพทเทิร์นนั้นมีความแตกต่างกัน ดังนั้นลักษณะที่สำคัญอีกอย่างหนึ่งในรูปจะเกี่ยวกับความสัมพันธ์ M ซึ่งการสร้างความสัมพันธ์นี้สะท้อนให้เห็นทางเลือกของระบบการวัด ในแง่ที่ว่า การออกแบบระบบการวัดเป็นส่วนสำคัญอย่างหนึ่งสำหรับการออกแบบระบบ classification กล่าวคือ ระบบการวัดที่ผิดพลาดและไม่สมบูรณ์จะไม่สามารถนำไปสู่ระบบ classification ที่มีประสิทธิภาพดีได้

และสุดท้าย จะสังเกตได้ว่าแพทเทิร์นที่ถูกสร้างมาจากคลาสเดียวกัน (ในที่นี้คือ p_1 และ p_4 ซึ่งมาจาก w_i ทั้งคู่) และอยู่ใกล้กันใน pattern space ไม่จำเป็นที่จะให้ค่าการวัด (ในที่นี้คือ m_1 และ m_3) ที่อยู่ใกล้กันด้วย สิ่งนี้เป็นสิ่งที่สำคัญในกรณีของการวัดความคล้ายกันระหว่างแพทเทิร์น ซึ่งเน้นให้เห็นถึงความสำคัญของกระบวนการในการวัดค่า

2.3 Features, Feature Vector และ Classifiers

ในที่นี้จะจำลองตัวอย่างการวิเคราะห์ทางการแพทย์ในการแบ่งกลุ่มของรูปภาพ รูปที่ 2.2 แสดงรูปภาพ 2 รูป ซึ่งแต่ละรูปก็มีขอบเขตภายในที่แตกต่างกัน จะสังเกตได้ว่าทั้งสองรูปมีความแตกต่างกันเมื่อดูด้วยสายตา จากความรู้ทางด้านการวิเคราะห์โรค สามารถบอกได้ว่ารูปที่ 2.2(ก)

เป็นผลมาจากเนื้องอกชนิดที่ไม่เป็นอันตรายและถูกกำหนดให้เป็นคลาส A ส่วนรูปที่ 2.2(ข) เป็นผลมาจากเซลล์มะเร็งและถูกกำหนดให้เป็นคลาส B แต่อย่างไรก็ตาม รูปอื่นๆที่เป็นคลาส A หรือคลาส B ก็มีความแตกต่างกันออกไป



รูปที่ 2.2 ตัวอย่างของรูปภาพที่ใช้ในการวิเคราะห์โรค (ก) คลาส A (ข) คลาส B

ขั้นตอนแรกของการวิเคราะห์หาชนิดของโรค คือ การระบุถึงพารามิเตอร์ใดๆ ที่สามารถวัดออกมาได้ โดยพารามิเตอร์ดังกล่าวจะต้องทำให้รูปของทั้งสองคลาสมีความแตกต่างกันและแยกออกจากกันได้ รูปที่ 2.3 แสดงผลการพลอตระหว่างค่าเฉลี่ย (mean) ของความเข้มแสงเทียบกับค่าเบี่ยงเบนมาตรฐาน (standard deviation) รอบๆค่าเฉลี่ยของแต่ละรูป โดยแต่ละจุดในรูปกราฟ จะแสดงแต่ละรูป ซึ่งจะเห็นได้ว่าแพทเทิร์นที่อยู่ในคลาส A มีแนวโน้มที่จะแยกออกมาบนพื้นที่ที่แตกต่างกันจากคลาส B

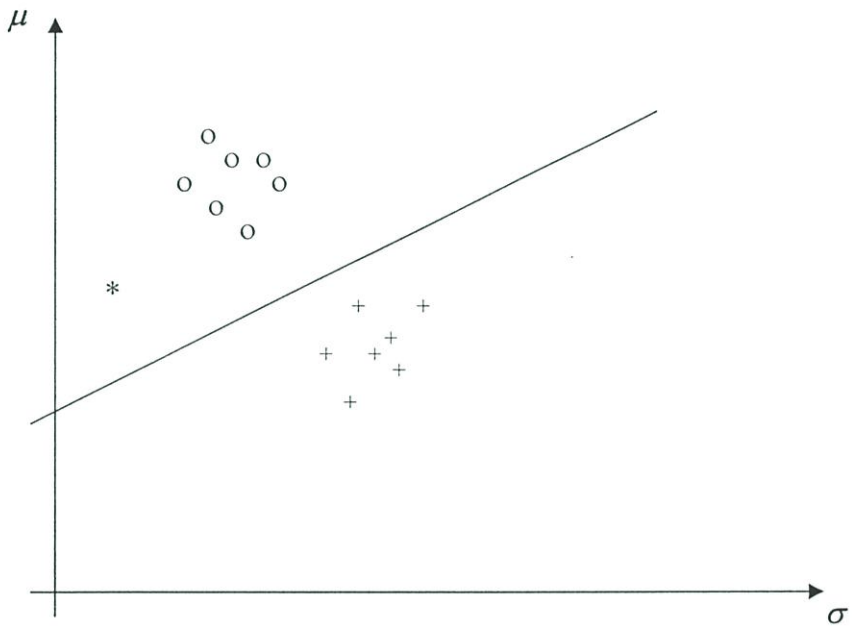
สำหรับเส้นตรงที่เห็นในรูปที่ 2.3 ก็เป็นเส้นแบ่งที่ดีอันหนึ่งที่สามารถแยกทั้งสองคลาสออกจากกันได้ สมมติว่ามีรูปภาพใหม่ที่มีลักษณะตามแบบรูปที่ 2.2 แต่ไม่ทราบว่ารูปดังกล่าวนั้นเป็นคลาสใด วิธีการที่สมเหตุสมผลก็คือ ทำการวัดค่าพารามิเตอร์ซึ่งเป็นค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐานของความเข้มแสงของรูปภาพนั้น แล้วนำไปพลอตในแกนตามรูปที่ 2.3 ซึ่งในที่นี้จุดที่ทำการพลอตใหม่แสดงไว้ด้วยเครื่องหมายดอกจัน (*) จากรูปพอที่จะสรุปได้ว่ารูปภาพใหม่ที่ไม่ทราบว่าเป็นภาพของอะไร ก็มีความเป็นไปได้ที่จะเป็นคลาส A มากกว่าคลาส B

กระบวนการ classification ที่ได้กล่าวไปนั้นมีสิ่งที่สำคัญอันหนึ่งของปัญหา pattern classification นั่นคือ ค่าการวัด (measurement) ที่ใช้ในการแบ่งคลาส ซึ่งในที่นี้คือค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐานซึ่งสิ่งเหล่านี้จะรู้จักกันในชื่อของคุณลักษณะ (feature)

สามารถกล่าวได้ว่าคุณลักษณะ คือ ค่าใดๆที่สามารถสกัดออกมาได้จากแหล่งกำเนิดตัวอย่างเช่น ค่าความเข้มของสัญญาณ หรือค่าน้ำหนักของวัตถุ เป็นต้น โดยคุณลักษณะสามารถ

เป็นได้ทั้งสัญลักษณ์ ตัวเลขหรือแบบผสมก็ได้ ตัวอย่างของคุณลักษณะที่เป็นสัญลักษณ์ได้แก่ สี ส่วนคุณลักษณะ ที่เป็นตัวเลขก็อาจจะเป็นค่าน้ำหนัก ส่วนสูง เป็นต้น

โดยทั่วไปคุณลักษณะจะได้มาจากกระบวนการสกัดลักษณะ (feature extraction algorithm) ที่กระทำกับ input data เราจะเรียกกลุ่มของคุณลักษณะที่รวมกัน n ตัวว่า feature vector หรือแพทเทิร์น (pattern) รูปที่ 2.4 แสดงตัวอย่างแพทเทิร์นแบบต่างๆ



รูปที่ 2.3 ผลการพลอตตระหว่างค่าเฉลี่ยเทียบกับค่าเบี่ยงเบนมาตรฐานของแต่ละรูป ที่เป็นคลาส A(o) และคลาส B(+)

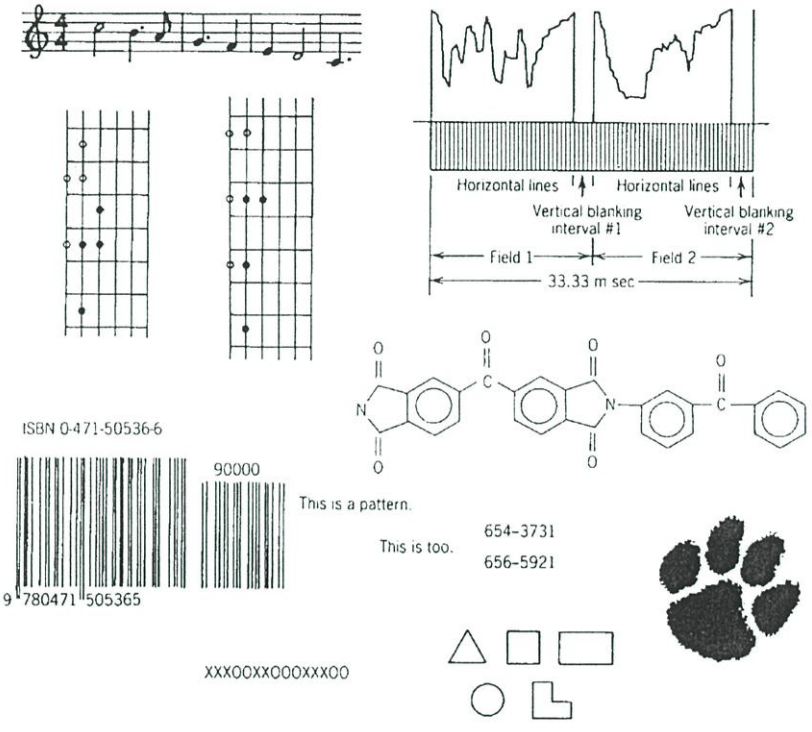
กลุ่มของคุณลักษณะจำนวน n ตัว ซึ่งได้แก่ x_i, i = 1, 2,..., n สามารถประกอบขึ้นเป็นแพทเทิร์น ได้ดังนี้

$$X = [x_1, x_2,..., x_n] \qquad ...(2.1)$$

แพทเทิร์นแต่ละตัวจะมีลักษณะเฉพาะตัวและใช้แทนวัตถุแต่ละหน่วย โดยธรรมชาติแล้วคุณลักษณะจะมีลักษณะเป็นตัวแปรสุ่ม (random variable) โดยผลการวัดค่าใดๆของแพทเทิร์นต่างๆ จะมีความผันผวนในลักษณะสุ่ม ทั้งนี้อาจจะมีส่วนมาจากสัญญาณรบกวน (noise) ที่เกิดขึ้นจากเครื่องมือวัด หรืออาจเกิดจากลักษณะเฉพาะของแพทเทิร์นเอง

เส้นตรงที่ปรากฏในรูปที่ 2.3 จะถูกเรียกว่า decision line และเป็นส่วนประกอบที่สำคัญที่ประกอบกันขึ้นเป็น classifier ซึ่งตัว classifier นี้มีหน้าที่ในการแบ่ง feature space ออกเป็น

ส่วนสำหรับแต่ละคลาส ถ้ามีเวคเตอร์ x ใดๆที่ตกอยู่ในพื้นที่ของคลาส A เวคเตอร์นั้นก็จะถูกจัดให้เป็นคลาส A แต่ก็ไม่ได้หมายความว่าการตัดสินใจนั้นจะถูกต้องเสมอไป ในการที่จะสร้างเส้นตรงดังรูป ก็จำเป็นที่จะต้องรู้ถึงชนิดของคลาสของแต่ละจุดในรูป แพทเทิร์นที่เรารู้ชนิดของคลาสและถูกใช้ในการออกแบบ classifier จะถูกเรียกว่า training pattern

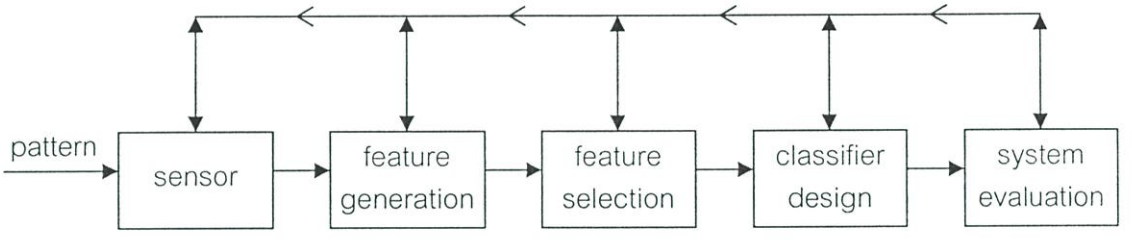


รูปที่ 2.4 ตัวอย่างของแพทเทิร์นในลักษณะต่างๆ

ในการออกแบบระบบ classification อาจจะต้องคำนึงถึงปัญหาและปัจจัยต่างๆ ดังแสดงในรูปที่ 2.5 ดังนี้

- เราจะสร้างคุณลักษณะขึ้นมาได้อย่างไร ในตัวอย่างที่ยกขึ้นมาข้างต้น เราใช้ค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐาน เนื่องจากเรารู้ว่ารูปภาพที่ใช้ถูกสร้างขึ้นมาอย่างไร แต่ในทางปฏิบัติอาจไม่ได้เป็นเช่นนั้น การสร้างคุณลักษณะขึ้นอยู่กับชนิดของปัญหาและการออกแบบระบบ classification
- คุณลักษณะควรจะมีจำนวนเท่าไรเพื่อให้ได้ผลดีที่สุด นี่ก็เป็นส่วนสำคัญที่ต้องคำนึงถึงในขั้นตอน feature selection stage ของการออกแบบระบบ classification ในทางปฏิบัติจำนวนของคุณลักษณะจะถูกสร้างขึ้นมามาก แต่จะนำบางส่วนที่ดีที่สุดไปใช้

- เมื่อสามารถเลือกคุณลักษณะที่เหมาะสมได้แล้ว จะมีวิธีการในการออกแบบ classifier ได้อย่างไร จากตัวอย่างที่ยกขึ้นมาข้างต้น เราใช้วิธีวาดเส้นตรงเพื่อแบ่งพื้นที่ออกเป็น 2 ส่วนสำหรับแต่ละคลาส แต่ในทางปฏิบัติโดยทั่วไปแล้ว พื้นผิวหรือเส้นที่ใช้แบ่ง space ออกเป็นพื้นที่ของคลาสต่างๆ มักจะไม่เป็นเชิงเส้น ดังนั้นเราจะหาวิธีการที่จะสร้างพื้นผิวหรือเส้นดังกล่าวได้อย่างไร จึงจะเป็นการเหมาะสมที่สุด กระบวนการทั้งหลายนี้ถูกกระทำในขั้นตอน classifier design stage
- และสุดท้าย เมื่อได้มีการออกแบบ classifier แล้ว จะมีวิธีการอย่างไรในการประเมินประสิทธิภาพของสิ่งที่ได้ออกแบบมา นั่นคือ ผลของการ classification มีข้อผิดพลาดมากแค่ไหน ขั้นตอนนี้เรียกว่า system evaluation stage



รูปที่ 2.5 ขั้นตอนต่างๆในการออกแบบระบบ classification

ในรูปที่ 2.5 แสดงขั้นตอนต่างๆ ที่ต่อเนื่องกันในการออกแบบระบบ classification จากรูปจะเห็นว่า มีลูกศรป้อนกลับอยู่ในทุกขั้นตอน เนื่องจากว่าแต่ละขั้นตอนไม่ได้เป็นอิสระต่อกัน ตรงกันข้ามขั้นตอนเหล่านี้มีความสัมพันธ์ระหว่างกัน ขึ้นอยู่กับผลของการ classification ขั้นตอนหนึ่งอาจจะต้องกลับไปออกแบบขั้นตอนที่อยู่ก่อนหน้าใหม่ เพื่อที่จะปรับปรุงประสิทธิภาพของระบบโดยรวมให้ดีที่สุด

บทที่ 3

General Fuzzy Hypersphere Neural Network

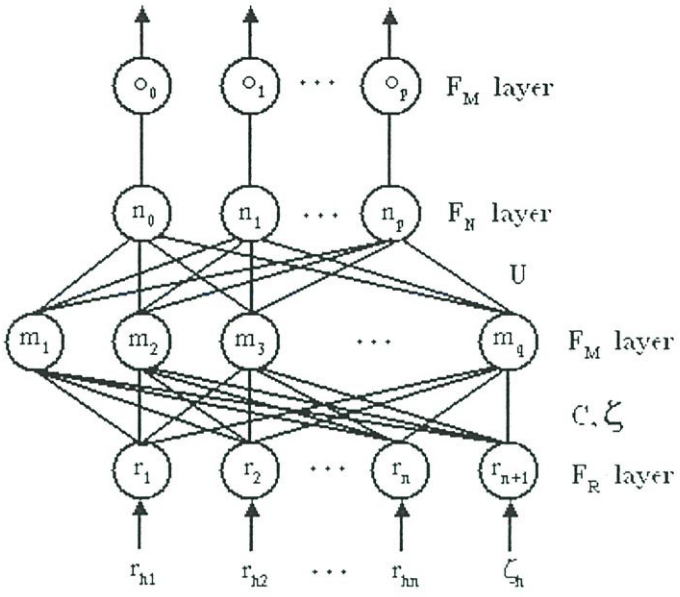
Fuzzy Neural Network (FNN) ได้กลายมาเป็นที่นิยมและมีการประยุกต์ใช้งานกันอย่างแพร่หลายในงานทางด้านการรู้จำรูปแบบ (pattern recognition) โดยพื้นฐานแล้ว จะมีวิธีการที่ใช้ในการฝึก (training) โครงข่ายอยู่ 2 วิธี ได้แก่ การเรียนรู้ในแบบมีผู้สอน (supervised learning) และแบบไม่มีผู้สอน (unsupervised learning) สำหรับการเรียนรู้แบบมีผู้สอนนั้น แต่ละอินพุตแพทเทิร์นที่ใช้ในการฝึก จะมีการกำหนดชนิดคลาส (class) ไว้ก่อนล่วงหน้า และขอบเขตในการตัดสินใจ (decision boundary) ระหว่างคลาสจะถูกสร้างขึ้น เพื่อทำให้การผิดพลาดในการจัด-กลุ่ม (misclassification) มีค่าน้อยที่สุด ปัญหานี้มักจะเป็นที่รู้จักกันในชื่อของ Classification ส่วนการเรียนรู้แบบไม่มีผู้สอนนั้น อินพุตแพทเทิร์นที่ใช้ในการฝึกจะไม่ถูกกำหนดว่าเป็นชนิดคลาสใด แต่จะมีวิธีการในการจัดกลุ่มกันของแพทเทิร์น ด้วยเครื่องมือที่เหมาะสมในการวัดความคล้าย (similarity) ระหว่างแพทเทิร์น ซึ่งปัญหาดังกล่าวจะเป็นที่รู้จักกันในชื่อ Clustering

General Fuzzy Hypersphere Neural Network (GFHSNN) เป็นอัลกอริทึมที่ใช้สำหรับการเรียนรู้ ทั้งในแบบมีผู้สอนและแบบไม่มีผู้สอน ซึ่งได้นำประโยชน์ของ fuzzy set เข้ามาใช้ในการกำหนดขอบเขตของคลาส สถาปัตยกรรมและอัลกอริทึมในการเรียนรู้ของ GFHSNN ได้รวมลักษณะเด่นไว้ ดังนี้

1. อินพุตแพทเทิร์นสามารถที่จะนำเสนอได้ทั้งในแบบ fuzzy hypersphere และ crisp-point
2. ทั้งอินพุตแพทเทิร์นที่กำหนดชนิดคลาสไว้แล้ว และแบบที่ยังไม่ได้กำหนดชนิดคลาสสามารถที่จะนำเข้าไปเรียนรู้ได้ในเวลาเดียวกัน
3. พารามิเตอร์ที่ใช้ในการกำหนดขนาดสูงสุดของ hypersphere สามารถที่จะปรับปรุงได้แบบ adaptive ถ้าต้องการ

3.1 โครงสร้างของ GFHSNN

โครงสร้างของ GFHSNN แสดงได้ดังรูปที่ 3.1 เป็นโครงสร้างแบบ feedforward neural network ขนาด 4 ชั้น ซึ่งจะค่อยๆเติบโตขึ้นแบบ adaptive เพื่อครอบคลุมความต้องการของปัญหา

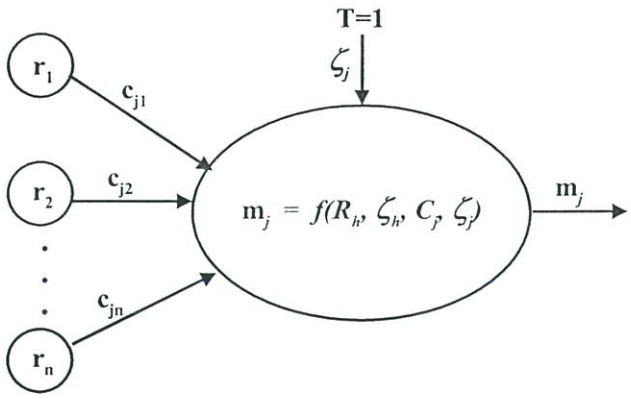


รูปที่ 3.1 โครงสร้างของ General Fuzzy Hypersphere Neural Network

ชั้น F_R จะเป็นชั้นที่รับอินพุตแพทเทิร์น ซึ่งจะมีจำนวน n โหนด สำหรับรับอินพุตแพทเทิร์นขนาด n มิติ และจะมีหน่วยประมวลผลโหนดที่ n+1 (รับอินพุต ζ_h) ซึ่งแสดงรัศมี (radius) ของ hypersphere ในที่นี้ถ้า $\zeta_h = 0$ จะหมายถึงว่าอินพุตแพทเทิร์นเป็นแพทเทิร์นในแบบ crisp-point นั่นคือเป็นเพียงจุดเดียว แต่ถ้า ζ_h เป็นค่าอื่น ก็จะหมายถึงว่าอินพุตแพทเทิร์น เป็น hypersphere หนึ่งใน hyperspace จะเห็นได้ว่าอินพุตแพทเทิร์นสามารถที่จะแทนได้ด้วยทั้ง crisp-point หรือ fuzzy hypersphere

โหนดในชั้น F_M จะค่อยๆถูกสร้างขึ้นในระหว่างกระบวนการฝึก และแต่ละโหนดจะใช้แทน hypersphere แต่ละหน่วย ซึ่งจะแสดงพฤติกรรมตาม membership function ของมัน ลักษณะการประมวลผลที่กระทำโดยแต่ละโหนดในชั้น F_M แสดงได้ดังรูปที่ 3.2

ดังแสดงในรูปที่ 3.2 ค่าน้ำหนัก (weights) ที่เชื่อมต่อระหว่างชั้น F_R และ F_M จะแสดงจุดศูนย์กลาง (center point) ของ hypersphere ในที่นี้ $C_j = (c_{j1}, c_{j2},..., c_{jn})$ จะแสดงจุดศูนย์กลางของ m_j ค่า threshold input ที่แทนด้วย T จะถูกตั้งไว้ที่ 1 และมีการให้น้ำหนักด้วย ζ_j โดยที่ ζ_j จะแสดงถึงค่ารัศมีของ hypersphere m_j และจะถูกปรับค่าไประหว่างกระบวนการฝึก ค่าจุด-ศูนย์กลางและรัศมีของ hypersphere แต่ละหน่วยจะถูกจัดเก็บไว้ในเมตริก C และเวคเตอร์ ζ ตามลำดับ ค่าขนาดสูงสุดของ hypersphere จะถูกกำหนดโดยผู้ใช้ ด้วยพารามิเตอร์ λ_1 ($0 \leq \lambda_1 \leq 1$) ซึ่งค่า λ_1 นี้ จะเป็นตัวกำหนดค่ารัศมีสูงสุดของ hypersphere ที่โครงข่ายจะยอมให้มีได้



รูปที่ 3.2 Implementation ของ fuzzy hypersphere

ค่า membership function ของ fuzzy hypersphere ถูกนิยามขึ้นให้สามารถจัดการได้กับอินพุตแพทเทิร์นที่เป็นทั้งแบบ crisp-point และแบบ hypersphere ถ้าทำการกำหนดเทรนนิง-เซต (training set) $R = \{R_h \mid h = 1, 2, \ldots, p\}$ โดยที่ $R_h = (r_{h1}, r_{h2}, \ldots, r_{hn}, \zeta_h)$ เป็นอินพุตแพทเทิร์นลำดับที่ h จะสามารถนิยาม membership function ของ fuzzy hypersphere ได้ ดังนี้

$$m_j(R_h) = 1 - f(L, \zeta_j, \zeta_h, \gamma) \qquad \ldots(3.1)$$

เมื่อ $f(L, \zeta_j, \zeta_h, \gamma)$ คือฟังก์ชันที่นิยามโดย

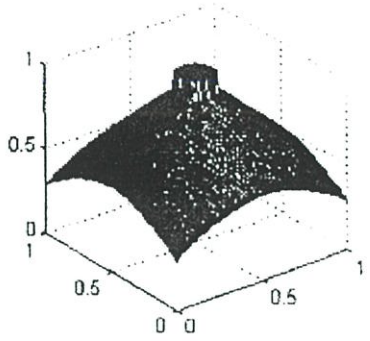
$$f(.) = \begin{cases} 0, & \text{ถ้า } 0 \le (L + \zeta_h) \le \zeta_j \\ (L + \zeta_h) \cdot \gamma, & \text{ถ้า } \zeta_j < (L + \zeta_h) \le 1 \\ 1, & \text{ถ้า } (L + \zeta_h) > 1 \end{cases} \qquad \ldots(3.2)$$

และค่า L ซึ่งเป็นระยะห่างระหว่าง R_h และจุดศูนย์กลางของ hypersphere m_j นิยามโดย

$$L = \left(\sum_{i=1}^{n} \left(c_{ji} - r_{hi} \right)^2 \right)^{1/2} \qquad \ldots(3.3)$$

Membership function จะให้ค่า m_j เป็น 1 ถ้า hypersphere ได้รวมแพทเทิร์น R_h เข้าไปด้วย (แพทเทิร์น R_h ตกอยู่ภายใน hypersphere) ค่าพารามิเตอร์ γ จะเป็นตัวกำหนดความเร็วในการลดลงของค่าความเป็นสมาชิก (membership) ภายนอก hypersphere เมื่อระยะห่างระหว่าง

R_h และ C_j เพิ่มขึ้น รูปที่ 3.3 แสดงตัวอย่างผลการพลอต fuzzy membership function ของ hypersphere ที่มีจุดศูนย์กลาง อยู่ที่ [0.5 0.5] และมีรัศมีเป็น 0.3 ด้วยพารามิเตอร์ $\gamma = 1$



รูปที่ 3.3 ตัวอย่างค่า fuzzy membership function ด้วยพารามิเตอร์ $\gamma = 1$

ค่าน้ำหนักที่เชื่อมต่อระหว่างโหนดในชั้นที่สองและชั้นที่สามจะเป็นแบบไบนารี (เฉพาะค่า 0 และ 1) ซึ่งจะถูกจัดเก็บไว้ในเมตริก U และจะถูกปรับในระหว่างกระบวนการฝึก โดย

$$u_{jk} = \begin{cases} 1, & \text{ถ้า } m_j \text{ เป็น hypersphere ที่อยู่ในคลาส } n_k \\ 0, & \text{ถ้า } m_j \text{ ไม่ใช่ hypersphere ที่อยู่ในคลาส } n_k \end{cases} \qquad \text{...(3.4)}$$

$$k = 0,1,...,p \text{ และ } j = 1,2,...,q$$

เมื่อ m_j เป็นโหนดที่ j ในชั้น F_M และ n_k เป็นโหนดที่ k ในชั้น F_N

แต่ละโหนดในชั้น F_N จะทำการรวม (union) ผลที่เป็น fuzzy จาก hypersphere ที่มีชนิดคลาสเดียวกัน และให้ค่าเอาท์พุตออกมาดังนี้

$$n_k = \max_{j=1}^{q} m_j \bullet u_{jk}, \qquad k = 0,1,...,p \qquad \text{...(3.5)}$$

แต่ละโหนดในชั้น F_N จะทำการส่งต่อค่าเอาท์พุตที่เป็นแบบ non-fuzzy ออกมา ดังนี้

$$o_k = \begin{cases} 0, & \text{ถ้า } n_k < T \\ 1, & \text{ถ้า } n_k = T \end{cases} \qquad \text{...(3.6)}$$

เมื่อ $T = \max(n_k)$

$k = 0,1,...,p$

โครงสร้างของ GFHSNN ที่ได้นำเสนอไป จะเป็นโครงสร้างในแบบทั่วไป (general) อย่างไรก็ตาม ในหลายๆแอปพลิเคชันโดยส่วนใหญ่แล้ว มักจะมีอินพุตแพทเทิร์นของระบบเป็นแบบ crisp-point ซึ่งหมายความว่า แต่ละอินพุตแพทเทิร์นสามารถแสดงได้ด้วยจุด 1 จุด ใน pattern space ดังนั้น เพื่อที่จะให้โครงสร้างของโครงข่ายเหมาะกับแอปพลิเคชันดังกล่าว ก็ทำได้โดยการกำหนดให้ค่า $\zeta_h = 0$ (ให้ค่า $\zeta_h = 0$ ในสมการที่ 3.1 และ 3.2)

3.2 อัลกอริทึมในการเรียนรู้ของ GFHSNN

กำหนดให้เทรนนิงเซต R ประกอบด้วยเซตของคู่อันดับ $\{R_h, d_h\}$ เมื่อ $R_h = (r_{h1}, r_{h2},..., r_{hn}, \zeta_h) \in I^{n+1}$ เป็นอินพุตแพทเทิร์นลำดับที่ h และ $d_h \in \{0, 1, 2,...,p\}$ เป็นอินเด็กซ์ที่ชี้ไปยังชนิดคลาส โดยเมื่อ $d_h = 0$ จะหมายความว่า R_h หรืออินพุตแพทเทิร์นลำดับที่ h ไม่ได้มีการกำหนดคลาส (เรียกอินพุตแพทเทิร์นชนิดนี้ว่า unlabeled)

อัลกอริทึมในการเรียนรู้ของ GFHSNN จะเป็นการจัดวางตำแหน่งและปรับแต่งขนาดของแต่ละ hypersphere ใน pattern space ให้เหมาะสม ซึ่งประกอบด้วย 3 ขั้นตอน คือ 1. การสร้างและขยาย hypersphere 2. การทดสอบการทับกัน และ 3. การกำจัดการทับกัน โดยแต่ละขั้นตอนมีรายละเอียด ดังนี้

3.2.1 การสร้างและขยาย hypersphere (Creation/Expansion of Hyperspheres)

กระบวนการในการฝึกจะเริ่มจากการนำเสนอคู่อันดับ $\{R_h, d_h\}$ จากฐานข้อมูล เมื่ออินพุต-แพทเทิร์นลำดับที่ h หรือ R_h ถูกนำเข้าไปในโครงข่าย โครงข่ายจะทำการหา hypersphere m_j ที่ให้ค่า membership function สูงสุดและมีโอกาสที่จะทำการขยายได้ (ถ้าต้องการ) กระบวนการในการหา hypersphere ที่สนใจ (m_j) นี้ ขึ้นอยู่กับชนิดคลาสของอินพุตแพทเทิร์น กล่าวคือ ถ้าอินพุตแพทเทิร์นเป็นแบบ unlabeled ($d_h = 0$) แล้ว hypersphere ทั้งหมดที่มีในโครงข่ายจะถูกนำเข้ามาพิจารณาในการหา m_j แต่ถ้าอินพุตแพทเทิร์นเป็นแบบ labeled ($d_h \neq 0$) แล้ว เฉพาะ hypersphere ที่เป็นสมาชิกของคลาสนั้นและ hypersphere ที่เป็นชนิด unlabeled (hypersphere ที่เป็นสมาชิกของคลาส 0) เท่านั้นที่จะนำมาพิจารณา หลังจากนั้นก็จะทำตามขั้นตอนดังนี้

ขั้นตอนที่ 1

พิจารณาว่า R_h ถูกบรรจุอยู่ภายใน m_j หรือไม่ โดยสามารถพิสูจน์ได้ด้วยสมการที่ 3.1 ถ้าพบว่า $m_j(R_h, \zeta_h, C_j, \zeta_j) = 1$ ซึ่งแสดงว่า R_h ถูกบรรจุอยู่ภายใน m_j ก็ให้ข้ามขั้นตอนที่เหลือทั้งหมด และเริ่มกระบวนการฝึกกับคู่อันดับถัดไป

ขั้นตอนที่ 2

ถ้า R_h ตกอยู่ภายนอก m_j ก็จะทำการขยายขนาดของ m_j ให้ไปครอบคลุม R_h แต่ในการขยาย m_j นั้น จะต้องเป็นไปตามเงื่อนไขที่กำหนด โดยข้อจำกัดในการขยายขนาดของ hypersphere คือ

$$\left(\sum_{i=1}^{n}\left(c_{ji}-r_{hi}\right)^2\right)^{1/2}+\zeta_h \le \lambda_1 \qquad ...(3.7)$$

เมื่อเงื่อนไขตามสมการที่ 3.7 เป็นจริงแล้ว ก็จะทำการขยายขนาดของ m_j เพื่อที่จะได้ครอบคลุม R_h ไว้ภายใน โดยให้รัศมีใหม่ของ m_j ถูกปรับเป็น

$$\zeta_j^{new}=\left(\sum_{i=1}^{n}\left(c_{ji}-r_{hi}\right)^2\right)^{1/2}+\zeta_h \qquad ...(3.8)$$

ถ้า hypersphere ถูกขยายไปครอบคลุมอินพุตแพทเทิร์นชนิด unlabeled แล้ว คลาสของ hypersphere นั้นจะยังคงเหมือนเดิม แต่ถ้า hypersphere ดังกล่าวเป็นชนิด unlabeled และได้ขยายไปครอบคลุมอินพุตแพทเทิร์นชนิด labeled ในกรณีนี้จะต้องทำการเปลี่ยนชนิดคลาสของ hypersphere นั้นให้เป็นชนิดคลาสเดียวกันกับชนิดคลาสของอินพุตแพทเทิร์น ดังนี้

$$\begin{aligned} &If\ d_h \neq 0\ and\ \text{class}(m_j)=0 \\ &then\ \text{class}(m_j)=d_h \end{aligned} \qquad ...(3.9)$$

ขั้นตอนที่ 3

ถ้า R_h ไม่เป็นไปตามเงื่อนไขในขั้นตอนที่ 1 หรือขั้นตอนที่ 2 ก็ให้สร้าง hypersphere ที่เป็นสมาชิกของคลาสนั้นขึ้นมาใหม่ โดยมี

$$C_{new}=R_h \quad \text{และ} \quad \zeta_{new}=0 \qquad ...(3.10)$$

3.2.2 **การทดสอบการทับกัน** (Overlap test)

เนื่องจากมีการยอมให้ใช้อินพุตแพทเทิร์นทั้งชนิด labeled และชนิด unlabeled ในการฝึกโครงข่าย ดังนั้นจึงต้องมีการพิจารณาถึงการทับกันของ hypersphere ทั้ง 2 ประเภท ถ้าให้ m_j

เป็น hypersphere ที่ถูกขยายขนาดตามหัวข้อ 3.2.1 การทดสอบการทับกันกับ hypersphere m_k ให้กระทำดังนี้

$$\text{ถ้า } \text{class}(m_j) = \begin{cases} 0, & \text{ให้ทดสอบการทับกันกับ hypersphere อื่นทั้งหมด} \\ \text{ค่าอื่น}, & \text{ให้ทดสอบการทับกันกับ hypersphere ที่ } \text{class}(m_j) \neq \text{class}(m_k) \end{cases}$$

การทดสอบการทับกันนี้จะกระทำหลังจาก hypersphere มีการขยายหรือมีการสร้าง hypersphere ขึ้นใหม่ (ขั้นตอนที่ 2 และขั้นตอนที่ 3 ในหัวข้อ 3.2.1 ตามลำดับ)

(a) **ทดสอบการทับกันสำหรับขั้นตอนที่** 2

กำหนดให้ hypersphere m_u เป็น hypersphere ที่ถูกขยายเพื่อให้ครอบคลุม R_h และการขยาย ดังกล่าวทำให้เกิดการทับกันกับ hypersphere m_v สมมติให้ $C_u = [c_{u1}, c_{u2}, \ldots, c_{un}]$ และ ζ_u เป็นจุดศูนย์กลางและรัศมีของ m_u และ $C_v = [c_{v1}, c_{v2}, \ldots, c_{vn}]$ และ ζ_v คือจุดศูนย์กลางและรัศมีของ m_vตามลำดับแล้ว ถ้าปรากฏว่า

$$\left(\sum_{i=1}^{n} \left(c_{ji} - r_{hi} \right)^2 \right)^{1/2} + \zeta_h \leq \lambda_1 \qquad \ldots(3.7)$$

ก็แสดงว่า m_u และ m_v ทับกัน

(b) **ทดสอบการทับกันสำหรับขั้นตอนที่** 3

ถ้า hypersphere ที่สร้างขึ้นใหม่ไปตกอยู่ภายใน hypersphere อื่นแล้ว แสดงว่ามีความเป็นไปได้ที่จะเกิดการทับกัน และถ้า hypersphere ที่สร้างขึ้นใหม่นั้น เป็นชนิด unlabeled แล้วก็ไม่จำเป็นต้องมีการทดสอบและกำจัด (remove) การทับกันนั้น

สมมติให้ m_p เป็น hypersphere ที่ถูกสร้างขึ้นใหม่เพื่อที่จะครอบคลุม R_h ไว้ และ m_q เป็น hypersphere ของคลาสอื่น การทับกันของทั้งสอง hypersphere ทำได้โดยการใช้สมการที่ (3.1) นั่นคือ เมื่อ $m_p(R_h, \zeta_h, C_p, \zeta_p) = m_q(R_h, \zeta_h, C_q, \zeta_q) = 1$ แล้ว แสดงว่าทั้งสอง hypersphere ที่อยู่ต่างคลาสกันเกิดการทับกัน

3.2.3 การกำจัดการทับกัน

ถ้าการทับกันของ hypersphere เกิดขึ้นจากขั้นตอนที่ 2 ก็ให้กำจัดการทับกันนั้นโดยการใช้ค่ารัศมีเดิมของ hypersphere ก่อนมีการขยาย นั่นคือ

$$\zeta_u^{new} = \zeta_u^{old} \tag{3.12}$$

ถ้าการทับกันเกิดขึ้นจากขั้นตอนที่ 3 การกำจัดก็จะทำได้ดังนี้ ให้ C_p และ ζ_p เป็นจุดศูนย์กลางของ hypersphere ชนิด labeled ที่ถูกสร้างขึ้นมาใหม่ และ C_q และ ζ_q เป็นจุดศูนย์กลางและรัศมีของ hypersphere ที่เป็นสมาชิกของคลาสอื่นตามลำดับแล้ว การกำจัดการทับกันจะทำได้โดยให้

$$\zeta_q^{new} = \left(\sum_{i=1}^{n} \left(c_{pi} - c_{qi} \right)^2 \right)^{1/2} - \zeta_p - \delta \qquad ...(3.13)$$

เมื่อ δ คือค่าจริงขนาดเล็กๆ ที่ถูกใช้เพียงเพื่อให้สามารถกำจัดการทับกันได้ จากการทดลอง ค่า δ จะถูกเลือกไว้ที่ประมาณ 0.0001

ในอีกกรณีหนึ่ง ถ้า hypersphere ที่ถูกสร้างขึ้นใหม่เป็นชนิด labeled และเกิดการทับกันกับ hypersphere ที่เป็นชนิด unlabeled ก็ให้เปลี่ยนคลาสของ hypersphere ชนิด unlabeled นั้นให้เป็นคลาสของ hypersphere ชนิด labeled ที่ถูกสร้างขึ้นใหม่

$$\begin{aligned} &If \ \text{class}(m_p) \neq 0 \ and \ \text{class}(m_q) = 0 \\ &then \ \text{class}(m_q) = \text{class}(m_p) \end{aligned} \qquad ...(3.14)$$

จากอัลกอริทึมในการเรียนรู้นี้ ค่าของ λ_1 ซึ่งเป็นพารามิเตอร์ในการกำหนดขนาดสูงสุดของ hypersphere สามารถที่จะทำการเปลี่ยนแปลงได้แบบ adaptive โดยกำหนดให้

$$\lambda_1^{new} = \Psi \cdot \lambda_1^{old} \qquad ...(3.15)$$

เมื่อ Ψ ($0 < \Psi < 1$) เป็นค่าคงที่

3.3 ตัวอย่างการเรียนรู้ใน 2 มิติ

หัวข้อนี้จะใช้ชุดข้อมูลใน 2 มิติจำนวน 16 ข้อมูลในการบรรยายการทำงานของ GFHSNN โดยแพทเทิร์นที่ถูกใช้ในกระบวนการฝึกแสดงในตารางที่ 3.1 ลำดับของข้อมูลจะมีความแน่นอนดังที่ได้แสดงไว้ในตาราง จุดของข้อมูลได้ถูกเลือกอย่างเหมาะสมเพื่อที่จะครอบคลุมและอธิบายถึงเงื่อนไขที่เป็นไปได้ทั้งหมดของอัลกอริทึมในการเรียนรู้ ในที่นี้พารามิเตอร์ γ จะถูกตั้งไว้ที่ 1 และ λ_1 มีค่าเป็น 0.2

ในกระบวนการฝึก อินพุตแพทเทิร์นชนิด labeled สามอันดับแรกจะถูกรวมเข้าไปอยู่ใน hypersphere ที่สร้างขึ้นใหม่ H_1, H_2 และ H_3 ตามลำดับ โดยข้อมูลของ hypersphere แต่ละหน่วยจะถูกเก็บไว้ในเมตริก C และเวกเตอร์ ζ ดังที่ได้กล่าวไปแล้ว ซึ่งข้อมูลของ hypersphere ทั้งสามที่สร้างขึ้นใหม่ ขณะนี้ เป็น $C_1 = [0.1\ 0.1] \in d_1$, $\zeta_1 = 0$, $C_2 = [0.4\ 0.6] \in d_2$, $\zeta_2 = 0$, $C_3 = [0.8\ 0.9] \in d_1$ และ $\zeta_3 = 0$

อินพุตแพทเทิร์นชนิด unlabeled ลำดับที่ 4 จะถูกรวมเข้าไปอยู่ใน hypersphere H_1 โดยการขยายขนาด ζ_1 ให้เท่ากับ 0.0447 ส่วนแพทเทิร์นลำดับที่ 5 สามารถบรรจุได้ใน hypersphere H_1 โดยไม่จำเป็นต้องขยาย และลำดับที่ 6 ถูกรวมเข้าไปใน hypersphere โดยการขยาย H_2 ให้มี ζ_2 เป็น 0.0361

อินพุตแพทเทิร์นลำดับที่ 7 ก็สามารถครอบคลุมได้ด้วย H_3 โดยการขยายรัศมี ζ_3 เป็น 0.0539 ลำดับที่ 8 เป็นอินพุตแพทเทิร์นชนิด unlabeled ซึ่งถูกบรรจุโดยการขยาย H_2 ให้มี $\zeta_2 = 0.05$ ในกรณีนี้คลาสของ H_2 ไม่จำเป็นต้องเปลี่ยนแปลง ส่วนอินพุตแพทเทิร์นชนิด unlabeled ลำดับที่ 9 ได้สร้าง hypersphere ชนิด unlabeled ขึ้นมาใหม่ เนื่องจากไม่สามารถรวมแพทเทิร์นนี้ให้เข้ากับการขยายของ hypersphere ใดเลยที่มีอยู่ในขณะนี้ hypersphere ที่สร้างขึ้นใหม่มี $C_4 = [0.7\ 0.4] \in d_0$ และ $\zeta_4 = 0$

เมื่ออินพุตแพทเทิร์นลำดับที่ 10 ซึ่งเป็นคลาส 1 ถูกใส่เข้าไปในเครือข่าย จะถูกครอบคลุมโดยการขยาย hypersphere H_4 ที่เป็นชนิด unlabeled ให้มี $\zeta_4 = 0.0707$ และในกรณีนี้ ชนิด-คลาสของ H_4 จะต้องถูกเปลี่ยนให้เป็นคลาสเดียวกันกับแพทเทิร์นลำดับที่ 10 นั่นคือ $H_4 \in d_1$

แพทเทิร์นลำดับที่ 11 ได้สร้าง hypersphere ใหม่ H_5 ที่มี $C_5 = [0.2\ 0.8] \in d_0$ และ $\zeta_5 = 0$ เนื่องจากไม่สามารถรวมแพทเทิร์นนี้ด้วยการขยาย hypersphere อื่นได้ แพทเทิร์นถัดไปเป็นชนิด labeled ซึ่งเป็นสมาชิกของคลาส 1 ถูกรวมเข้าไปโดยการขยาย H_5 ให้มีขนาด ζ_5 เป็น 0.1118 และจะต้องเปลี่ยน H_5 ให้เป็นสมาชิกของ d_1

มาถึงลำดับที่ 13 ซึ่งอาจทำได้โดยการขยาย H_2 (เนื่องจากเงื่อนไขในการขยายเป็นจริงตามสมการที่ 3.7) แต่เนื่องจากถ้าทำการขยายขนาด H_2 ให้ครอบคลุมแพทเทิร์นนี้ จะทำให้ H_2 ไปทับกับ H_5 ดังนั้นจึงไม่ทำการขยาย โดยจะคงที่ค่ารัศมีเดิม ζ_2 ไว้ และทำการสร้าง hypersphere

ใหม่ H_6 ขึ้น โดยมี $C_6 = [0.26\ 0.74] \in d_2$ และ $\zeta_6 = 0$ แต่อย่างไรก็ตาม H_6 ที่สร้างขึ้นใหม่นี้ก็เกิดการทับกันกับ H_5 ซึ่งเป็นสมาชิกของต่างคลาสกัน ดังนั้นจึงจำเป็นต้องลดรัศมี ζ_5 ให้มีขนาดเหลือ 0.0848

ตารางที่ 3.1 ชุดข้อมูลที่ใช้ในการบรรยายการทำงานของ GFHSNN

ลำดับที่	อินพุตแพทเทิร์น R_h	คลาส
1	[0.1 0.1]	1
2	[0.4 0.6]	2
3	[0.8 0.9]	3
4	[0.14 0.12]	0
5	[0.12 0.14]	1
6	[0.42 0.63]	2
7	[0.82 0.95]	3
8	[0.4 0.65]	0
9	[0.7 0.4]	0
10	[0.65 0.35]	1
11	[0.2 0.8]	0
12	[0.3 0.85]	1
13	[0.26 0.74]	2
14	[0.49 0.4]	0
15	[0.55 0.25]	0
16	[0.6 0.35]	2

แพทเทิร์นลำดับที่ 14 ได้สร้าง H_7 ขึ้นมาใหม่ โดยมี $C_7 = [0.49\ 0.4] \in d_0$ และ $\zeta_7 = 0$ เมื่อพยายามที่จะขยาย H_7 ให้ครอบคลุมแพทเทิร์นที่ 15 ปรากฏว่า เกิดการทับกันกับ H_4 ดังนั้นจึงคงขนาดเดิมของ H_7 ไว้ และสร้าง H_8 ขึ้นมาใหม่ โดยมี $C_8 = [0.55\ 0.25] \in d_0$ และ $\zeta_8 = 0$ และแพทเทิร์นลำดับสุดท้ายเป็นชนิด labeled ที่เป็นสมาชิกของคลาส 2 ถูกรวมเข้าไปใน H_8 โดยการขยาย ζ_8 เป็น 0.1118 และเนื่องจาก H_8 เป็น hypersphere ชนิด unlabeled ดังนั้นจึงต้องเปลี่ยนชนิดคลาส ของมันด้วย นั่นคือให้ $H_8 \in d_2$

สถานะสุดท้ายของ hypersphere ทั้งหมดเมื่อเสร็จสิ้นการฝึก แสดงได้ดังตารางที่ 3.2

ตารางที่ 3.2 สถานะสุดท้ายของ hypersphere

Hypersphere	จุดศูนย์กลาง (C)	คลาส	รัศมี (ζ)
H_1	[0.1 0.1]	1	0.0447
H_2	[0.4 0.6]	2	0.05
H_3	[0.8 0.9]	3	0.0539
H_4	[0.7 0.4]	1	0.0707
H_5	[0.2 0.8]	1	0.0848
H_6	[0.26 0.74]	2	0
H_7	[0.49 0.4]	0	0
H_8	[0.55 0.25]	2	0.1118

3.4 Modified Fuzzy Hypersphere Neural Network (MFHSNN)

สำหรับ General Fuzzy Hypersphere Neural Network (GFHSNN) ที่ได้กล่าวถึงไปนั้น จะเห็นได้ว่า การสร้าง hypersphere ขึ้นมาใหม่แต่ละตัว จะใช้อินพุตแพทเทิร์นที่ทำให้เกิดการสร้าง hypersphere นั้น มาเป็นจุดศูนย์กลาง หรือกล่าวได้ว่า อินพุตแพทเทิร์นแรกที่ถูกบรรจุอยู่ใน hypersphere ก็คือ จุดศูนย์กลางของ hypersphere นั่นเอง และอินพุตแพทเทิร์นอื่นๆ ที่จะเข้ามาเป็นสมาชิกของ hypersphere ในภายหลัง จะทำให้ขนาดของ hypersphere ใหญ่ขึ้น แต่จุดศูนย์กลางยังคงเป็นจุดเดิม

Modified Fuzzy Hypersphere Neural Network (MFHSNN) ได้ดัดแปลงกระบวนการเรียนรู้ของ GFHSNN โดยการทำให้จุดศูนย์ของ hypersphere เปลี่ยนแปลงไปตามสมาชิกของ hypersphere ด้วย วิธีการ คือ MFHSNN จะเพิ่มจำนวนรอบในการเรียนรู้เป็น 2 รอบ (GFHSNN ใช้การเรียนรู้เพียงรอบเดียว) โดยในรอบที่หนึ่งจะเป็นการเรียนรู้แบบเดียวกันกับ GFHSNN หลังจากเสร็จสิ้นการเรียนรู้ในรอบแรก ก็จะทำการเปลี่ยนจุดศูนย์กลางของ hypersphere ทุกตัวที่ถูกสร้างขึ้นในรอบแรกให้เป็นค่าเฉลี่ยของทุกอินพุตแพทเทิร์นที่เป็นสมาชิกของ hypersphere เดียวกันและเรียกจุดนี้ว่าจุดเซ็นทรอยด์ (centroid) หลังจากนั้นเปลี่ยนค่ารัศมี (รอบจุดเซ็นทรอยด์) ของ hypersphere ให้เป็นศูนย์ แล้วเริ่มกระบวนการเรียนรู้แบบเดิมอีกครั้งในรอบที่ 2

บทที่ 4
ครอสคอร์รีเลชันนิวรอลเน็ตเวิร์ค

4.1 คอร์รีเลชันและครอสคอร์รีเลชัน

โดยทั่วไปแล้วในการแบ่งกลุ่มหรือจัดกลุ่มของวัตถุ (object) ไม่ว่าวัตถุนั้นจะอยู่ในรูปของเวคเตอร์แบบเรขาคณิต (geometrical vector) รูปคลื่นแบบต่อเนื่องทางเวลา (continuous-time waveform) หรือจะเป็นลำดับแบบไม่ต่อเนื่อง (discrete-time sequence) มักจะกระทำโดยการวัดความคล้าย (similarity) หรือไม่คล้าย (dissimilarity) ระหว่างวัตถุ ยกตัวอย่างเช่น ถ้ามีเวคเตอร์ 2 ตัวที่มีทั้งขนาดและทิศทางเดียวกัน ก็สามารถจะบอกได้ว่าเวคเตอร์ทั้งสองนั้นคล้ายกันหรือเหมือนกัน (equivalent) แต่ถ้าเวคเตอร์ 2 ตัวมีเฉพาะขนาดที่เท่ากันแต่ทิศทางแตกต่างกัน เวคเตอร์ทั้งสองก็แค่คล้ายกันแต่ไม่เหมือนกัน ในการจะบอกถึงปริมาณของความคล้ายกันระหว่างเวคเตอร์คู่หนึ่ง เรามักจะใช้เครื่องมืออย่าง inner product , scalar product หรือ dot product

Dot product ระหว่างเวคเตอร์ X และ Y ใดๆ นิยามดังนี้

$$\mathbf{X} \cdot \mathbf{Y} = |\mathbf{X}||\mathbf{Y}| \cos\theta \qquad ...(4.1)$$

ในกรณีที่เวคเตอร์ทั้งคู่มีขนาดเท่ากันและชี้ไปในทิศทางเดียวกัน ค่าของ dot product จะมีค่าสูงสุด และกล่าวได้ว่าเวคเตอร์ทั้งสองเหมือนกันทุกประการ แต่ในกรณีที่เวคเตอร์ทั้งสองมีเพียงขนาดที่เท่ากันแต่มีทิศทางที่ตั้งฉากกัน จะได้ว่า X•Y = 0 ในกรณีนี้ ค่าของ dot product จะมีค่าเป็นศูนย์และกล่าวได้ว่าเวคเตอร์ทั้งคู่ไม่คล้ายกัน จะเห็นได้ว่า dot product เป็นเครื่องมืออันหนึ่งซึ่งสามารถบอกได้ถึงปริมาณของความคล้ายกันระหว่างเวคเตอร์

พิจารณารูปคลื่นค่าจริงแบบต่อเนื่องทางเวลา x(t) และ y(t) ถ้ารูปคลื่นทั้งสองมีการกระจายทั้งทางด้านแอมพลิจูด (amplitude distribution) และทางด้านเวลา (time distribution) แบบเดียวกันแล้ว แสดงว่ารูปคลื่นทั้งสองเหมือนกัน โดยทั่วไปแล้วในปัญหาของการประเมินการหน่วงทางเวลา (time delay estimation) นั้นรูปคลื่นสองรูปจะมีการกระจายทางแอมพลิจูดเหมือนกันแต่จะแตกต่างกันที่การกระจายทางเวลา นั่นคือ สองรูปคลื่นดังกล่าวสามารถเขียนได้เป็น x(t) และ y(t) = x (t-τ) ในการบอกถึงความคล้ายกันของรูปคลื่นแบบต่อเนื่องทางเวลานั้นจะใช้ inner product ที่นิยาม ดังนี้

$$< x(t), y(t) > = \int_a^b x(t)y(t)\,dt \qquad ...(4.2)$$

เมื่อ $a \le t \le b$ คือช่วงเวลาที่สนใจ และถ้า $y(t) = x(t-\tau)$ แล้ว จะได้ว่า

$$< x(t), y(t) > = \int_a^b x(t)x(t-\tau)\,dt \qquad ...(4.3)$$

ในกรณีที่ $\tau = 0$ รูปคลื่น x(t) และ $y(t) = x(t-\tau)$ จะเป็นรูปคลื่นเดียวกันและทำให้ inner product ตามสมการที่ 4.3 มีค่าสูงสุด ซึ่งกล่าวได้ว่าทั้งสองรูปคลื่นมีความคล้ายกันและเหมือนกันทุกประการ แต่ในกรณีที่ไม่มีส่วนหนึ่งส่วนใดของรูปคลื่น x(t) และ y(t) อยู่ในช่วงเวลา [a, b] เลย (นั่นคือรูปคลื่นทั้งสองไม่ต่อกันทางเวลา) สมการที่ 4.3 ก็จะให้ค่าเป็นศูนย์ ในกรณีนี้สามารถกล่าวได้ว่ารูปคลื่นทั้งสองไม่มีความคล้ายกัน

สำหรับการวัดความคล้ายกันของลำดับแบบไม่ต่อเนื่อง (discrete-time sequence) ก็ได้มีการให้นิยาม inner product ไว้ ดังนี้

$$< x(n), y(n) > = \sum_{n=1}^{N} x(n)y(n) \qquad ...(4.4)$$

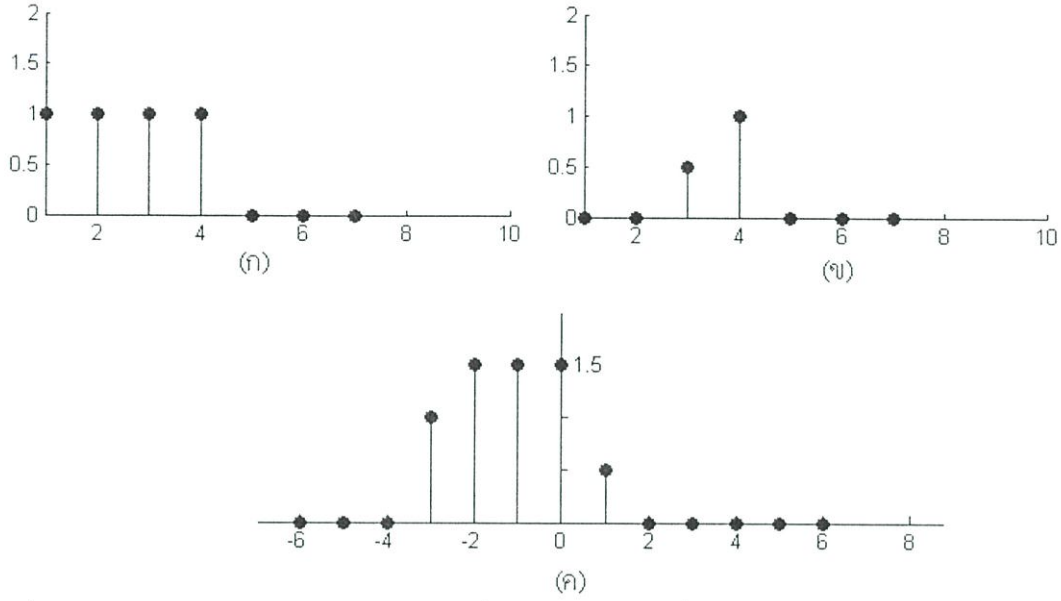
เมื่อ $1 \le n \le N$ คือช่วงที่สนใจ เช่นเดียวกับกรณีของเวคเตอร์และรูปคลื่นแบบต่อเนื่องทางเวลา inner product ตามสมการที่ 4.4 ก็สามารถใช้ในการวัดความคล้ายกันของสองลำดับแบบไม่ต่อเนื่องได้

ในปัญหาการประเมินการหน่วงทางเวลา จะใช้ครอสคอร์รีเลชันฟังก์ชัน (Cross-correlation function) ซึ่งครอสคอร์รีเลชันฟังก์ชัน ระหว่าง x(n) และ y(n) ใดๆ มีนิยามดังนี้

$$r_{xy}(m) = \sum_{n=-\infty}^{\infty} x(n)y^{*}(n-m) \qquad ...(4.5)$$

เมื่อ x(n) และ y(n) อาจจะอยู่ในรูปของจำนวนเชิงซ้อน และ y* คือ conjugate ของ y ตัวแปร m ในสมการที่ 4.5 จะถูกเรียกว่า lag variable ซึ่งทำให้ค่า r_{xy} เป็นฟังก์ชันของ m หน้าที่ของฟังก์ชันนี้ก็คือ ทำการเลื่อน (shift) รูปคลื่นอันใดอันหนึ่งด้วยระยะ m และหาค่าของฟังก์ชันที่ระยะการเลื่อน m นั้นออกมา เช่นเดียวกับ inner product ในแบบอื่นๆ ครอสคอร์รีเลชันฟังก์ชันจะให้ค่าความคล้ายกันระหว่างสองรูปคลื่น ณ ตำแหน่งการเลื่อน m ใดๆ

รูปที่ 4.1 แสดงรูปคลื่นของ x(n) และ y(n) และค่าครอสคอร์รีเลชันฟังก์ชันของทั้งสองรูปคลื่น ซึ่งโดยทั่วไปแล้ว ค่าครอสคอร์รีเลชันจะมีลักษณะที่ไม่สมมาตร (symmetric) ถ้า x(n) และ y(n) ไม่ได้มีการกระจายทางแอมพลิจูดเหมือนกัน ดังแสดงในรูปที่ 4.1(ค)



รูปที่ 4.1 ครอสคอร์รีเลชันฟังก์ชัน (ก) รูปคลื่น x(n), (ข) รูปคลื่น y(n), (ค) $r_{xy}(m)$ หรือครอสคอร์รีเลชันฟังก์ชันระหว่าง x(n) และ y(n)

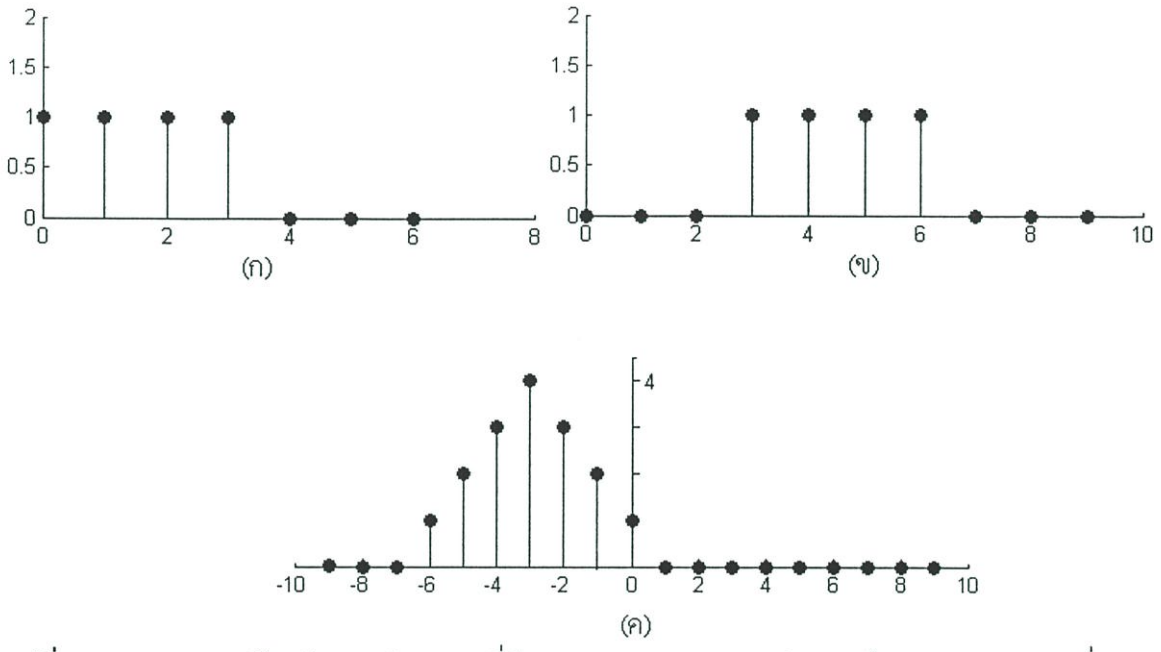
รูปที่ 4.2 แสดงค่าครอสคอร์รีเลชันของรูปคลื่น x(n) และ x(n-D) ซึ่งเป็นรูปคลื่นที่ได้จากการหน่วงรูปคลื่น x(n) ไปเป็นระยะ D ในกรณีที่สัญญาณทั้งสองเหมือนกันแต่มีการเลื่อนไปทางเวลา จะทำให้ค่าครอสคอร์รีเลชันมีลักษณะที่สมมาตรเสมอ และตำแหน่งที่ค่าครอสคอร์รีเลชันมีค่าสูงสุด (peak) นี้ ก็จะเป็นตำแหน่งที่บอกถึงระยะของการเลื่อนทางเวลานั่นเอง

ครอสคอร์รีเลชันฟังก์ชันที่ได้กล่าวถึงไปนั้นเป็นชนิดที่ไม่ได้นอร์มอไลซ์ (unnormalized) ซึ่งมีจุดบกพร่องหลายอย่างในการบอกถึงความคล้ายกันของสองรูปคลื่น เช่น ในกรณีที่ x(n) และ y(n) เป็นเวคเตอร์ที่มีขนาดใดๆ ค่าของครอสคอร์รีเลชันระหว่าง x(n) และ y(n) จะมีค่าขึ้นอยู่กับขนาดของเวคเตอร์ด้วย นั่นคือ ถ้า x(n) และ/หรือ y(n) ประกอบด้วยสมาชิก (element) ที่มีค่าสูงก็จะทำให้ค่าครอสคอร์รีเลชันสูงตามไปด้วย การนอร์มอไลซ์ค่าของครอสคอร์รีเลชันฟังก์ชันจะทำให้ฟังก์ชันมีค่าสูงสุดอยู่ที่ 1 ซึ่งหมายถึงว่ารูปคลื่นสองรูปมีความเหมือนกัน และทำให้สมาชิกที่มีค่าเป็นศูนย์ของทั้ง x(n) และ y(n) ได้เข้ามามีผลกระทบต่อความคล้ายกันโดยรวมของรูปคลื่น

ครอสคอร์รีเลชันฟังก์ชันในแบบที่ผ่านการนอร์มอไลซ์แล้ว เขียนได้ ดังนี้

$$r_{xy}^{n}(m) = \frac{1}{E}\sum_{n} x(n)y^{*}(n-m) \qquad ...(4.6)$$

โดยที่ $$E = \left(\sum_{n} x^{2}(n) \sum_{n} y^{2}(n) \right)^{1/2} \qquad ...(4.7)$$

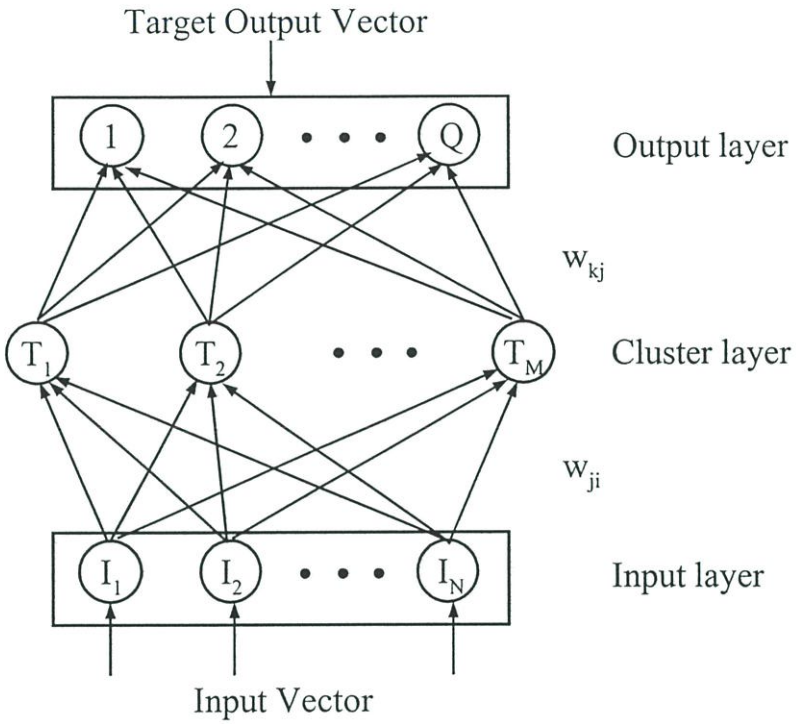


รูปที่ 4.2 ครอสคอร์รีเลชันของสัญญาณที่มีการกระจายทางแอมพลิจูดเหมือนกัน (ก) รูปคลื่น x(n), (ข) รูปคลื่น x(n) ที่ผ่านการหน่วงด้วยระยะ d = 3 หน่วย, (ค) ครอสคอร์รีเลชัน-ฟังก์ชันหรือออโตคอร์รีเลชันของรูปคลื่น

4.2 โครงสร้างของครอสคอร์รีเลชันนิวรอลเน็ตเวิร์ค

ครอสคอร์รีเลชันนิวรอลเน็ตเวิร์ค (Cross-correlation Neural Network, CCNN) ได้นำแนวคิดในการสร้างโครงข่ายมาจาก General Fuzzy Hypersphere Neural Network (GFHSNN) ที่ได้กล่าวถึงไปในบทที่ 3 ซึ่งโครงข่ายทั้งสองแบบนั้น อยู่บนรากฐานของการวัดความคล้ายกันระหว่างแพทเทิร์น โดย GFHSNN ใช้ระยะทาง (Euclidean distance) ระหว่างแพทเทิร์นเป็นตัวบอกความคล้าย นั่นคือ ถ้าระยะทางระหว่างแพทเทิร์นยิ่งน้อย ก็แสดงว่าแพทเทิร์นทั้งสองมีความคล้ายคลึงกันมาก ในขณะที่ CCNN จะใช้ค่าครอสคอร์รีเลชันเป็นตัวบอกความคล้ายกัน ซึ่งค่าครอสคอร์รีเลชันนั้นจะมีลักษณะที่ตรงกันข้ามกับระยะทาง กล่าวคือ ถ้าแพทเทิร์นสองแพทเทิร์นใดๆ ที่ให้ค่าครอสคอร์รีเลชันสูง จะแสดงว่า แพทเทิร์นคู่นั้นมีความสัมพันธ์และมีความคล้ายกันสูงด้วย

สถาปัตยกรรมของ CCNN แสดงไว้ในรูปที่ 4.3 ซึ่งเป็นโครงสร้างแบบ feed forward neural network ขนาด 3 ชั้น โดยโครงข่ายจะขยายตัวขึ้นแบบ adaptive เพื่อครอบคลุมขอบเขตของปัญหาที่นำเข้ามาฝึก



รูปที่ 4.3 สถาปัตยกรรมของครอสคอร์รีเลชันนิวรอลเน็ตเวิร์ค

ชั้นที่ 1 เป็น Input layer ขนาด N โหนด ซึ่งทำหน้าที่ในการรับอินพุตแพทเทิร์นขนาด N มิติ ในขณะที่ชั้นที่ 2 จะถูกเรียกว่า Cluster layer ซึ่งเป็นชั้นที่จะทำการสร้าง reference หรือ Template ขึ้นเรื่อยๆ ตามกระบวนการฝึก โดยแต่ละโหนดในชั้นนี้ จะแทน 1 reference pattern

ค่าน้ำหนัก w_j ที่เชื่อมต่อระหว่างชั้น Input layer และโหนด T_j ในชั้น Cluster layer จะแสดง reference pattern ของ T_j ซึ่งเวคเตอร์ w_j นี้จะนำไปใช้ในการคำนวณหาค่า membership function ระหว่าง reference pattern ตัวที่ j กับอินพุตแพทเทิร์นที่ถูกใส่เข้ามาในโครงข่าย

ถ้ากำหนดให้เทรนนิงเซต คือ $I = \{I_h \mid h = 1, 2,..., p\}$ โดยที่ $I_h = (i_{h1}, i_{h2},..., i_{hn})$ เป็นอินพุตแพทเทิร์นลำดับที่ h แล้ว จะสามารถนิยาม membership function ของ T_j ได้ดังนี้

$$T_j(I_h) = \begin{cases} 1, & \text{ถ้า } f(w_j, I_h) \geq mC_j \\ f(w_j, I_h), & \text{ถ้า } f(w_j, I_h) < mC_j \end{cases} \qquad ...(4.8)$$

และค่าฟังก์ชัน f(x, y) มีนิยามดังนี้

$$f(x,y) = \frac{\left(\sum_i x_i y_i\right)^2}{\left(\sum_i x_i^2\right)\left(\sum_i y_i^2\right)} \quad , \qquad i=1,2,...,N \qquad ...(4.9)$$

ฟังก์ชัน f(x, y) ในสมการที่ 4.9 เป็นค่านอร์มอไลซ์ครอสคอร์รีเลชันที่การเลื่อนเป็นศูนย์ยกกำลังสอง (square of normalized cross-correlation at zero lag)

ค่า mC_j ในสมการที่ 4.8 เป็นค่าครอสคอร์รีเลชันต่ำสุด (minimum Cross-correlation) ซึ่งเป็นตัวกำหนดขอบเขตของ reference pattern ตัวที่ j โดยค่า mC_j นี้จะมีลักษณะทางการทำงานคล้ายๆกับค่ารัศมีของ hypersphere ใน GFHSNN ในระหว่างกระบวนการฝึก ค่า mC_j จะถูกปรับขึ้นเพื่อป้องกันการทับกัน (overlap) กับ reference pattern อื่นที่อยู่ต่างคลาสกัน

ชั้นสุดท้าย คือ ชั้นที่ 3 เป็นชั้น Output layer ซึ่งแต่จะโหนดในชั้นนี้จะแทนแต่ละคลาส ดังนั้นจำนวนโหนดทั้งหมดที่มีในชั้นนี้ จะเท่ากับจำนวนของคลาสที่มีในระบบ ค่าน้ำหนัก w_{kj} ที่เชื่อมต่อระหว่างชั้นที่ 2 และชั้นที่ 3 จะเป็นเพียงค่า 0 หรือ 1 เท่านั้น โดยค่า w_{kj} จะเป็น 1 เมื่อโหนด j ในชั้นที่ 2 เป็นสมาชิกของคลาส k และจะเป็น 0 เมื่อโหนด j เป็นสมาชิกของคลาสอื่น

ค่าเอาท์พุตของแต่ละโหนดในชั้น Output layer จะเป็นดังนี้

$$D_k = \max_{j=1}^{M} T_j \bullet w_{kj} \, , \qquad k=0,1,...,Q \qquad ...(4.10)$$

สำหรับขั้นตอนในการทดสอบ (test) แต่ละอินพุตแพทเทิร์นจะถูกนำเข้ามาเพื่อทำนายชนิดของคลาส โดยโหนดของคลาสที่ให้ค่าเอาท์พุต (D_k) สูงสุดก็จะเป็นผลของการทำนาย

4.3 อัลกอริทึมในการเรียนรู้ของ CCNN

กำหนดให้เทรนนิงเซต I เป็นเซตของคู่อันดับ $\{I_h, O_h\}$ โดย $I_h = (i_{h1}, i_{h2}, \ldots, i_{hn}) \in R^n$ เป็นอินพุตแพทเทิร์นลำดับที่ h และ $O_h \in \{1, 2, \ldots, Q\}$ เป็นอินเด็กซ์ที่ชี้ไปยังชนิดของคลาส

กระบวนการฝึกจะกระทำกับอินพุตแพทเทิร์นที่อยู่ในคลาสเดียวกันไปทีละคลาส โดยจะสมมุติให้อินพุตแพทเทิร์นทุกตัวที่เป็นสมาชิกของคลาสนั้นเป็น reference pattern ก่อน แล้วจึงทำการพิจารณาเลือกแพทเทิร์นที่เหมาะสมเข้ามาเป็น reference pattern จริงของโครงข่าย ด้วยกระ-

บวนการฝึกในรูปแบบนี้ จะไม่ทำให้ลำดับของอินพุตแพทเทิร์นในเทรนนิงเซตมีผลกระทบต่อลักษณะของโครงข่าย

ขั้นตอนในการเรียนรู้ของ CCNN ประกอบด้วย 6 ขั้นตอน ดังนี้

1. **สร้างเซตของอินพุตแพทเทิร์น** P

กำหนดให้ P เป็นเซตของอินพุตแพทเทิร์นทั้งหมดที่เป็นสมาชิกของคลาสที่กำลังสนใจ ดังนี้

$$P = \{P_g\} = \{ I_h \mid I_h \in v\} \qquad ...(4.11)$$

เมื่อ g = 1, 2, ..., m คือ อินเด็กซ์ของอินพุตแพทเทิร์นในเซต P

v คือ อินเด็กซ์ที่ชี้ไปยังชนิดของคลาสที่กำลังสนใจ

m คือ จำนวนอินพุตแพทเทิร์นที่เป็นสมาชิกของคลาส v

2. **สร้าง reference pattern สมมุติ**

ทำการสร้าง reference pattern จากอินพุตแพทเทิร์นทั้งหมดในเซต P โดย reference pattern ที่สร้างขึ้นในขั้นตอนนี้เป็นเพียง reference pattern สมมุติ ไม่ใช่ reference pattern จริงของโครงข่าย

$$R\_TEMP_g = P_g, \qquad g = 1, 2, ..., m \qquad ...(4.12)$$

หลังจากนั้นก็จะกำหนดค่าครอสคอร์รีเลชันต่ำสุด (mC) ให้กับ reference pattern แต่ละตัวที่สร้างขึ้น ซึ่งค่าครอสคอร์รีเลชันต่ำสุดนี้เป็นค่าที่กำหนดขึ้นเพื่อบอกขอบเขตของ reference pattern และป้องกันการทับกันระหว่าง reference pattern ที่อยู่ต่างคลาสกัน ดังนั้นในการกำหนดค่า mC นั้น จะต้องทำให้

$$\forall s : T_g(I_s) < 1, \qquad g = 1, 2, ..., m \qquad ...(4.13)$$

เมื่อ s คือ อินเด็กซ์ของอินพุตแพทเทิร์นในเทรนนิงเซตที่ไม่เป็นสมาชิกของคลาส v

$T_g(I_s)$ คือ ฟังก์ชันที่นิยามตามสมการที่ 4.8 โดยการกำหนดให้ R_TEMP_g เป็น reference pattern

mC_TEMP_g คือ ค่าครอสคอร์รีเลชันต่ำสุดของ R_TEMP_g

v คือ อินเด็กซ์ที่ชี้ไปยังชนิดคลาสที่กำลังสนใจ

m คือ จำนวนอินพุตแพทเทิร์นที่เป็นสมาชิกของคลาส v

กล่าวคือ จะต้องไม่มีอินพุตแพทเทิร์นใดที่ไม่ใช่ชนิดคลาส v และมีค่าเอาท์พุต T_g เป็น 1 เพื่อให้เป็นไปตามเงื่อนไขดังกล่าว ค่า mC_TEMP_g จะต้องเป็นไปตามสมการที่ 4.14

$$mC\_TEMP_g > \max C_g, \qquad g = 1, 2, ..., m \qquad ...(4.14)$$

$$\max C_g = \max_s \left\{ f(R\_TEMP_g, I_s) \right\}, \qquad g = 1, 2, ..., m \qquad ...(4.15)$$

ให้นำค่า $\max C_g$ ที่ได้จากสมการที่ 4.15 มากำหนด mC_TEMP_g ดังนี้

$$mC\_TEMP_g = \begin{cases} \lambda, & \text{ถ้า } \max C_g < \lambda \\ \max C_g + \epsilon, & \text{ถ้า } \lambda \le \max C_g < 1 \\ 1, & \text{อื่นๆ} \end{cases} \qquad ...(4.16)$$

เมื่อ f(x, y) คือ ฟังก์ชันที่นิยามตามสมการที่ 4.9

v คือ อินเด็กซ์ที่ชี้ไปยังชนิดคลาสที่กำลังสนใจ

m คือ จำนวนอินพุตแพทเทิร์นที่เป็นสมาชิกของคลาส v

λ คือ ค่าเริ่มต้นของ mC_TEMP_g ซึ่งมีค่าอยู่ระหว่าง 0 ถึง 1

ε คือ ค่าจริงขนาดเล็กๆ ที่ถูกใช้เพียงเพื่อให้สามารถกำจัดการทับกันได้ จากการทดลองค่า ε จะถูกเลือกไว้ที่ประมาณ 0.0001

3. **สร้างเวคเตอร์แสดงการเป็นสมาชิก M**

สำหรับ reference pattern แต่ละตัวที่ถูกสมมุติสร้างขึ้นในขั้นตอนที่ 2 ให้ตรวจสอบดูว่ามีอินพุตแพทเทิร์นใดบ้างในเซต P ที่เป็นสมาชิกอยู่ (อินพุตแพทเทิร์นที่เป็นสมาชิกคืออินพุต-แพทเทิร์นที่ทำให้ค่าเอาท์พุตของ reference pattern มีค่าเป็น 1) แล้วเก็บค่าการเป็นสมาชิกไว้ในเวคเตอร์ M_g ที่มีขนาดเท่ากับจำนวนของอินพุตแพทเทิร์นในเซต P โดยเวคเตอร์ M_g นี้จะเป็นไบนารีเวคเตอร์ที่มีเฉพาะค่า 0 และ 1 เท่านั้น โดย 0 จะหมายถึงการไม่เป็นสมาชิก และ 1 หมายถึงการเป็นสมาชิก ดังนี้

$$M_g(x) = \begin{cases} 1, & \text{เมื่อ } P_x \in R\_TEMP_g \\ 0, & \text{เมื่อ } P_x \notin R\_TEMP_g \end{cases}, \qquad \begin{matrix} g = 1, 2, ..., m \\ x = 1, 2, ..., m \end{matrix} \qquad ...(4.17)$$

ตัวอย่างเช่น ถ้า reference pattern R_TEMP_1 มีสมาชิกเป็น P_1, P_3, P_5, P_6, และ P_8 แล้ว ดังนั้น

$$M_1 = [1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0 \ldots 0]_m \qquad \ldots(4.18)$$

เมื่อ M_g คือ เวคเตอร์แสดงการเป็นสมาชิกสำหรับ reference pattern R_TEMP_g

m คือ จำนวนอินพุตแพทเทิร์นในเซต P

4. **พิจารณาเลือก reference pattern ที่เหมาะสม**

ขั้นตอนนี้จะเป็นการเลือก reference pattern จริงของโครงข่าย จาก reference pattern ที่สมมุติสร้างขึ้น โดยมีข้อจำกัดในการเลือก คือ เลือกจำนวน reference pattern ให้ได้น้อยที่สุด ที่สามารถครอบคลุมได้ทุกอินพุตแพทเทิร์นในเซต P

Reference pattern จะถูกเลือกทีละตัวไปเรื่อยๆ โดยมีลำดับในการเลือกที่พิจารณาจากเงื่อนไข ดังต่อไปนี้

- การเลือก reference pattern นั้น ทำให้จำนวนสมาชิกที่ครอบคลุมโดยรวมมีค่ามากที่สุด
- จำนวนอินพุตแพทเทิร์นที่เป็นสมาชิกของ reference pattern นั้น มีค่ามากที่สุด
- reference pattern นั้นมีค่า mC น้อยที่สุด

โดยจะพิจารณาจากเงื่อนไขแรกก่อน ถ้าพบว่ามี reference pattern มากกว่า 1 ตัวที่เป็นไปตามเงื่อนไขแรก ก็จะพิจารณาเงื่อนไขที่ 2 และ 3 ต่อไป ตามลำดับ และการเลือกจะสิ้นสุดลงเมื่อสามารถครอบคลุมสมาชิกทั้งหมดในเซต P

อัลกอริทึมที่ใช้ในการเลือก สามารถเขียนได้ ดังนี้

```
Covered_Member = [0 0 ... 0]_m ;
Selected_Index = [];
WHILE (Covered_Member ≠ [1 1 ... 1]_m)
        Choice = [];
        FOR g = 1:m
                Temp = Covered_Member OR M_g;
                Choice(g,1) = g;
                Choice(g,2) = COUNT(Temp, 1);
                Choice(g,3) = COUNT(M_g, 1);
```

Choice(g,4) = 1 - mC_TEMP_g;

END;

Choice = SORTROWS(Choice, [2 3 4]);

Index = Choice(m, 1);

Covered_Member = Covered_Member OR M_{Index};

Selected_Index = Selected_Index ∪ Index;

END;

อินเด็กซ์ของ reference pattern ที่ถูกเลือกจะถูกเก็บไว้ในเวคเตอร์ Selected_Index ดังนั้น

$$R\_SELECTED = \{ R\_TEMP_x \mid x \in Selected\_Index \} \quad \ldots(4.19)$$

$$mC\_SELECTED = \{ mC\_TEMP_x \mid x \in Selected\_Index \} \quad \ldots(4.20)$$

เมื่อ R_SELECTED คือ เซตของ reference pattern ที่ถูกเลือก

mC_SELECTED คือ เซตของค่า mC ที่ถูกเลือก

m คือ จำนวนอินพุตแพทเทิร์นในเซต P

Covered_Member คือ เวคเตอร์แสดงการเป็นสมาชิกโดยรวม

Selected_Index คือ เวคเตอร์ที่เก็บอินเด็กซ์ของ reference pattern ที่ถูกเลือก

M_g คือ เวคเตอร์แสดงการเป็นสมาชิกของ reference pattern R_TEMP_g

COUNT(X, a) คือ ฟังก์ชันที่คืนค่าจำนวนสมาชิกของ X ที่มีค่าเท่ากับ a

SORTROWS(X, Y) คือ ฟังก์ชันที่คืนค่าเมตริกซ์ X ที่ผ่านการเรียงแล้ว โดยการเรียงจะกระทำกับทั้งแถวของ X และเรียงจากค่าในคอลัมน์ที่ระบุในเวคเตอร์ Yตามลำดับ จากน้อยไปมาก

5. สร้างโหนดใหม่ให้กับโครงข่าย

จำนวนโหนดที่จะสร้างขึ้นใหม่ในชั้น Cluster layer จะเท่ากับจำนวน reference pattern ที่ถูกเลือกเก็บไว้ในเซต R_SELECTED ที่ได้จากขั้นตอนที่ 4 โดยในการสร้างโหนดจะต้องทำการกำหนดค่าต่างๆ ดังนี้

5.1 ค่าน้ำหนัก w_j ที่เชื่อมต่อระหว่าง Input layer และโหนดที่สร้างขึ้นใหม่

$$w_{j+q} = R\_SELECTED_q, \qquad q = 1, 2, \ldots, r \qquad \ldots(4.21)$$

เมื่อ r คือ จำนวน reference pattern ที่ถูกเลือก

j คือ จำนวนโหนดในชั้น Cluster layer ที่มีอยู่ ก่อนหน้าที่จะทำการเพิ่มโหนดใหม่

5.2 ค่าน้ำหนัก w_{kj} ที่เชื่อมต่อระหว่างโหนดที่สร้างใหม่กับ Output layer

$$w_{k(j+q)} = \begin{cases} 1, & \text{ถ้า } k = v \\ 0, & \text{ถ้า } k \neq v \end{cases}, \qquad q = 1, 2, \ldots, r \qquad \ldots(4.22)$$

เมื่อ r คือ จำนวน reference pattern ที่ถูกเลือก

j คือ จำนวนโหนดในชั้น Cluster layer ที่มีอยู่ ก่อนหน้าที่จะทำการเพิ่มโหนดใหม่

v คือ อินเด็กซ์ที่ชี้ไปยังชนิดคลาสที่กำลังสนใจ

5.3 ค่าครอสคอร์รีเลชันต่ำสุด (mC_j)

$$mC_{j+q} = mC\_SELECTED_q, \qquad q = 1, 2, \ldots, r \qquad \ldots(4.23)$$

เมื่อ r คือ จำนวน reference pattern ที่ถูกเลือก

j คือ จำนวนโหนดในชั้น Cluster layer ที่มีอยู่ ก่อนหน้าที่จะทำการเพิ่มโหนดใหม่

6. **ทำซ้ำ**

ทำซ้ำขั้นตอนที่ 1 ถึง 6 กับชนิดคลาสอื่น จนกระทั่งครบทุกชนิดของคลาส

4.4 ตัวอย่างการทำงานของอัลกอริทึม

ในหัวข้อนี้จะใช้ชุดข้อมูลขนาด 4 มิติ จำนวน 12 ข้อมูล ในการบรรยายการเรียนรู้ของ CCNN ตารางที่ 4.1 แสดงข้อมูลทั้งหมดที่ใช้ในการฝึก ในที่นี้กำหนดให้ $\lambda = 0.95$ และ $\varepsilon = 0.0001$

ตารางที่ 4.1 ชุดข้อมูลที่ใช้ในการบรรยายการทำงานของ CCNN

ลำดับที่	อินพุตแพทเทิร์น I_h	ชนิดคลาส
1	[2.8 3.9 5.4 6.0]	1
2	[5.8 4.2 6.1 6.0]	2
3	[7.7 5.0 4.8 3.8]	3
4	[3.3 3.1 5.2 5.7]	1
5	[7.3 6.5 4.5 3.2]	3
6	[5.5 5.9 4.0 6.0]	2
7	[2.1 3.5 4.5 7.1]	1
8	[4.1 4.3 6.0 5.5]	2
9	[3.9 3.3 4.1 5.9]	1
10	[5.1 4.9 4.0 3.9]	2
11	[3.7 4.7 4.4 5.2]	1
12	[5.9 6.4 4.5 4.4]	3

กระบวนการฝึกจะเริ่มกับอินพุตแพทเทิร์นที่เป็นชนิดคลาส 1 ก่อน ดังนี้

1. **สร้างเซตของอินพุตแพทเทิร์น** P

ในที่นี้ P จะเป็นเซตของอินพุทแพทเทิร์นที่มีชนิดคลาสเป็นชนิดที่กำลังสนใจ นั่นคือ ชนิดที่ 1

$$P = \{ I_h \mid O_h = 1 \} = \left\{ \begin{matrix} [2.8 \;\; 3.9 \;\; 5.4 \;\; 6.0] \\ [3.3 \;\; 3.1 \;\; 5.2 \;\; 5.7] \\ [2.1 \;\; 3.5 \;\; 4.5 \;\; 7.1] \\ [3.9 \;\; 3.3 \;\; 4.1 \;\; 5.9] \\ [3.7 \;\; 4.7 \;\; 4.4 \;\; 5.2] \end{matrix} \right\}$$

2. **สร้าง** reference pattern **สมมุติ**

สร้าง reference pattern จากอินพุตแพทเทิร์นทั้งหมดในเซต P

$$R\_TEMP = P = \begin{Bmatrix} [2.8\ 3.9\ 5.4\ 6.0] \\ [3.3\ 3.1\ 5.2\ 5.7] \\ [2.1\ 3.5\ 4.5\ 7.1] \\ [3.9\ 3.3\ 4.1\ 5.9] \\ [3.7\ 4.7\ 4.4\ 5.2] \end{Bmatrix}$$

คำนวณหาค่าครอสคอร์รีเลชันสูงสุด (maxC) ระหว่าง R_TEMP แต่ละตัวกับอินพุต-แพทเทิร์นอื่นทั้งหมดที่ไม่ได้เป็นชนิดคลาส 1 ตารางที่ 4.2 แสดงผลการคำนวณค่าครอสคอร์รีเลชัน

ตารางที่ 4.2 ค่าครอสคอร์รีเลชันที่ได้จากการคำนวณ

R_TEMP	f(R_TEMP, I_h)							maxC
	2	3	5	6	8	10	12	
1	0.9393	0.7504	0.7172	0.8927	0.9791	0.8576	0.8349	0.9791
2	0.9656	0.7862	0.7312	0.8925	0.9837	0.8664	0.8347	0.9837
3	0.8643	0.6362	0.5969	0.8425	0.9076	0.7627	0.7389	0.9076
4	0.9703	0.8300	0.7772	0.9449	0.9641	0.9022	0.8742	0.9703
5	0.9627	0.8619	0.8531	0.9757	0.9788	0.9535	0.9436	0.9788

เมื่อ f(x, y) คือ ฟังก์ชันที่นิยามตามสมการที่ 4.9

จากตารางที่ 4.2 ทำให้ได้ค่า maxC ดังนี้

$$maxC = \begin{Bmatrix} 0.9791 \\ 0.9837 \\ 0.9076 \\ 0.9703 \\ 0.9788 \end{Bmatrix}$$

นำค่า maxC ที่ได้มากำหนดค่า mC_TEMP ตามเงื่อนไขที่ระบุไว้ในสมการที่ 4.16

$$mC\_TEMP = \begin{Bmatrix} maxC_1 + \epsilon \\ maxC_2 + \epsilon \\ \lambda \\ maxC_4 + \epsilon \\ maxC_5 + \epsilon \end{Bmatrix} = \begin{Bmatrix} 0.9791 + 0.0001 \\ 0.9837 + 0.0001 \\ 0.95 \\ 0.9703 + 0.0001 \\ 0.9788 + 0.0001 \end{Bmatrix} = \begin{Bmatrix} 0.9792 \\ 0.9838 \\ 0.95 \\ 0.9704 \\ 0.9789 \end{Bmatrix}$$

3. **สร้างเวคเตอร์แสดงการเป็นสมาชิก M**

สำหรับ reference pattern แต่ละตัวที่ถูกสมมุติสร้างขึ้นให้ตรวจสอบดูว่า มีอินพุต-แพทเทิร์นใดบ้างในเซต P ที่เป็นสมาชิกอยู่ โดยตรวจสอบจากค่าเอาท์พุตของ reference pattern นั้น ตารางที่ 4.3 แสดงค่าเอาท์พุตของ reference pattern แต่ละตัว

ตารางที่ 4.3 ค่าเอาท์พุตของ reference pattern แต่ละตัว

R_TEMP	$T(P_g)$				
	1	2	3	4	5
1	1	1	0.9698	0.9648	0.9653
2	1	1	0.9540	0.9794	0.9562
3	1	1	1	0.9449	0.9130
4	0.9648	1	0.9449	1	0.9687
5	0.9653	0.9562	0.9130	0.9687	1

เมื่อ T(x) คือ ฟังก์ชันที่นิยามตามสมการที่ 4.8

จากตารางที่ 4.3 สามารถสร้างเวคเตอร์แสดงการเป็นสมากชิก M_gได้ดังนี้

$$\begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \\ M_5 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

4. **พิจารณาเลือก** reference pattern **ที่เหมาะสม**

Reference pattern จะถูกเลือกทีละตัวไปเรื่อยๆ โดยมีลำดับในการเลือกที่พิจารณาจากเงื่อนไข ดังต่อไปนี้

- การเลือก reference pattern นั้น ทำให้จำนวนสมาชิกที่ครอบคลุมโดยรวมมีค่ามากที่สุด
- จำนวนอินพุตแพทเทิร์นที่เป็นสมาชิกของ reference pattern นั้น มีค่ามากที่สุด
- reference pattern นั้นมีค่า mC น้อยที่สุด

เมื่อพิจารณาจากเวคเตอร์แสดงการเป็นสมาชิก M_g ที่ได้ในขั้นตอนที่ 3 แล้ว จะพบว่า R_TEMP_3 จะถูกพิจารณาเลือกเป็นตัวแรก เนื่องจากเมื่อเลือก R_TEMP_3 เข้ามาเป็น reference pattern จะทำให้จำนวนสมาชิกที่ครอบคลุมโดยรวมมีค่ามากที่สุด นั่นคือ สามารถครอบคลุมสมาชิกได้ 3 ตัว (P_1, P_2 และ P_3)

ตารางที่ 4.4 แสดงสมาชิกที่ครอบคลุมเมื่อมีการเลือก reference pattern ตัวที่สอง

ตารางที่ 4.4 สมาชิกที่ครอบคลุมเมื่อมีการเลือก reference pattern ตัวที่สอง

Reference pattern ตัวที่สอง	สมาชิกที่ครอบคลุมโดยรวม เมื่อมีการเลือก	สมาชิกของ reference pattern
R_TEMP_1	3 (P_1, P_2 และ P_3)	2 (P_1 และ P_2)
R_TEMP_2	3 (P_1, P_2 และ P_3)	2 (P_1 และ P_2)
R_TEMP_4	4 (P_1, P_2, P_3 และ P_4)	2 (P_2 และ P_4)
R_TEMP_5	4 (P_1, P_2, P_3 และ P_5)	1 (P_5)

จากตารางจะเห็นว่า การเลือก R_TEMP_4 หรือ R_TEMP_5 จะทำให้สมาชิกที่สามารถครอบคลุมได้มีจำนวนเท่ากัน แต่เนื่องจาก R_TEMP_4 มีจำนวนสมาชิกมากกว่า ดังนั้น reference pattern ตัวที่สองที่ถูกเลือกคือ R_TEMP_4

หลังจากนั้น R_TEMP_5 จะถูกเลือกเป็นตัวสุดท้าย ซึ่งสามารถครอบคลุมสมาชิกได้ครบทั้ง 5 ตัว และทำให้ได้

$$R\_SELECTED = \begin{Bmatrix} [2.1\ 3.5\ 4.5\ 7.1] \\ [3.9\ 3.3\ 4.1\ 5.9] \\ [3.7\ 4.7\ 4.4\ 5.2] \end{Bmatrix} \quad \text{และ} \quad mC\_SELECTED = \begin{Bmatrix} 0.9500 \\ 0.9704 \\ 0.9789 \end{Bmatrix}$$

5. สร้างโหนดใหม่ให้กับโครงข่าย

สร้างโหนดในชั้น Cluster layer ที่มีพารามิเตอร์ต่างๆ ดังนี้

5.1 ค่าน้ำหนัก w_j ที่เชื่อมต่อระหว่าง Input layer และโหนดที่สร้างขึ้นใหม่

$$w_j = \text{R\_SELECTED} = \begin{Bmatrix} [2.1 \;\; 3.5 \;\; 4.5 \;\; 7.1] \\ [3.9 \;\; 3.3 \;\; 4.1 \;\; 5.9] \\ [3.7 \;\; 4.7 \;\; 4.4 \;\; 5.2] \end{Bmatrix}$$

5.2 ค่าน้ำหนัก w_{kj} ที่เชื่อมต่อระหว่างโหนดที่สร้างใหม่กับ Output layer

$$w_{kj} = \begin{Bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{Bmatrix}$$

5.3 ค่าครอสคอร์รีเลชันต่ำสุด (mC_j)

$$mC_j = \text{mC\_SELECTED} = \begin{Bmatrix} 0.9500 \\ 0.9704 \\ 0.9789 \end{Bmatrix}$$

6. ทำซ้ำ

ทำการฝึกกับชนิดคลาสที่ 2 และ 3 ต่อไป (ไม่ได้แสดงให้ดู) เมื่อครบทุกชนิดของคลาสแล้ว โครงข่ายจะมีพารามิเตอร์ต่างๆ ดังนี้

$$w_j = \begin{Bmatrix} [2.1\ 3.5\ 4.5\ 7.1] \\ [3.9\ 3.3\ 4.1\ 5.9] \\ [3.7\ 4.7\ 4.4\ 5.2] \\ [5.8\ 4.2\ 6.1\ 6.0] \\ [5.5\ 5.9\ 4.0\ 6.0] \\ [7.3\ 6.5\ 4.5\ 3.2] \end{Bmatrix} \quad , \quad mC_j = \begin{Bmatrix} 0.9500 \\ 0.9704 \\ 0.9789 \\ 0.9704 \\ 0.9758 \\ 0.9683 \end{Bmatrix}$$

$$w_{kj} = \begin{Bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{Bmatrix}$$

บทที่ 5
การดำเนินการทดลองและผลการทดลอง

การทดลองเพื่อทดสอบประสิทธิภาพ ในเชิงความสามารถในการแบ่งกลุ่มของ Cross-correlation Neural Network (CCNN) จะกระทำเปรียบเทียบกับอีก 2 อัลกอริทึม นั่นคือ Modified Fuzzy Hypersphere Neural Network (MFHSNN) และ Fuzzy ARTMAP (FAM) ในหัวข้อที่ 5.1 จะใช้ชุดข้อมูลมาตรฐานจำนวน 8 ชุดข้อมูลในการทดสอบประสิทธิภาพของ CCNN ที่ใช้อัลกอริทึมในการเรียนรู้แบบเดิม นั่นคือ อัลกอริทึมในการเรียนรู้ที่ใช้กับ MFHSNN เพื่อเปรียบเทียบการทำงานของฟังก์ชันครอสคอร์รีเลชันที่ใช้ใน CCNN และฟังก์ชันระยะทางที่ใช้ใน MFHSNN ส่วนในหัวข้อที่ 5.2 จะเป็นการทดสอบ CCNN ที่ใช้อัลกอริทึมในการเรียนรู้แบบใหม่ที่ได้กล่าวไว้ในหัวข้อที่ 4.3 และในหัวข้อที่ 5.3 จะเป็นการทดสอบเพิ่มเติมกับชุดข้อมูลเสียงพูดตัวเลขภาษาไทย

ในการทดสอบกับชุดข้อมูลแต่ละชุด จะทำการปรับเปลี่ยนค่าพารามิเตอร์ต่างๆของแต่ละอัลกอริทึม และเลือกเฉพาะชุดของพารามิเตอร์ ที่ให้อัตราถูกต้องสูงสุดมาเป็นผลการทดลอง โดยพารามิเตอร์ที่จะต้องทำการปรับ ของแต่ละอัลกอริทึม ได้แก่

- Fuzzy ARTMAP คือ vigilance ($\overline{\rho}_a$), learning rate (β_a) และจำนวนรอบในการให้ระบบเรียนรู้ เนื่องจากการทดลองนี้เน้นไปที่การแบ่งกลุ่มรูปแบบ ดังนั้น ART_b ของ Fuzzy ARTMAP จึงกำหนดให้ไม่มีการเรียนรู้ และ ρ_{ab} มีค่าเท่ากับ 1
- Modified Fuzzy Hypersphere พารามิเตอร์ที่ต้องปรับ คือ λ
- Cross-correlation Neural Network พารามิเตอร์ที่ต้องปรับ คือ λ

การทดลองทั้งหมด ได้เลือกใช้โปรแกรม MATLAB เวอร์ชัน 6.1 ในการเขียนฟังก์ชันการทำงานของอัลกอริทึม

5.1 การทดสอบประสิทธิภาพของ CCNN ที่ใช้อัลกอริทึมในการเรียนรู้แบบเดิม

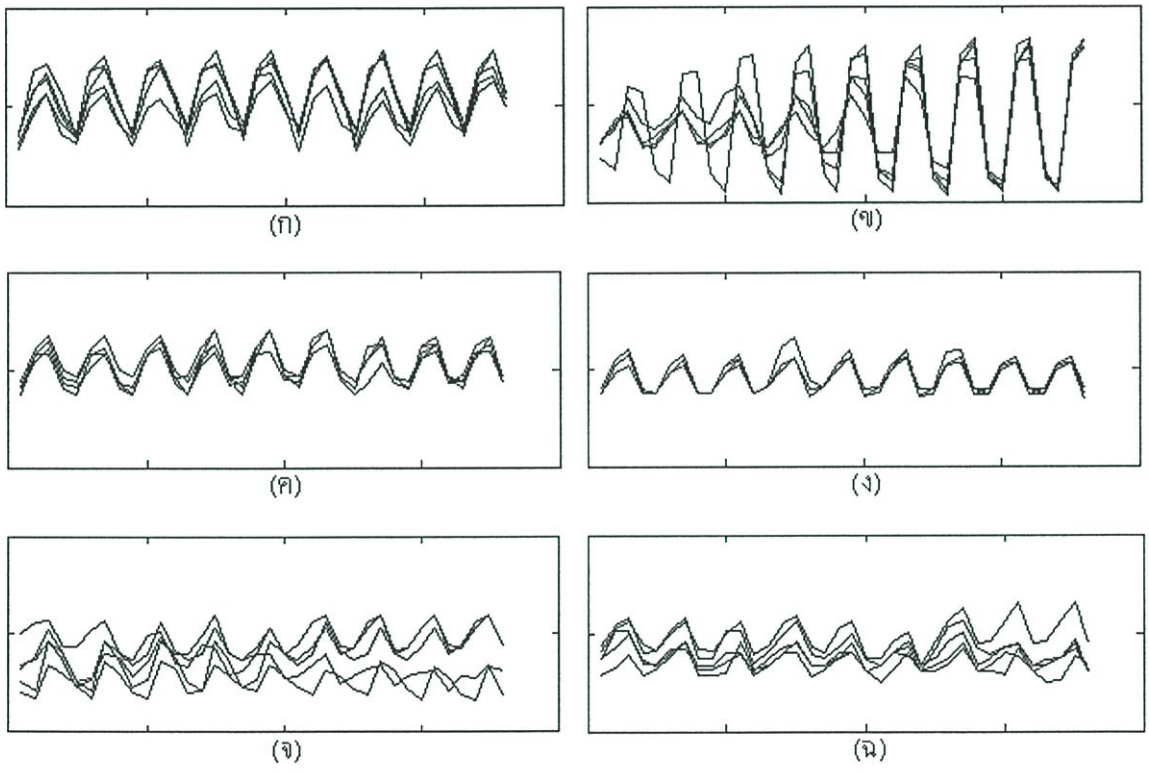
สถาปัตยกรรมของ CCNN มีพื้นฐานมาจาก MFHSNN ความแตกต่างระหว่างโครงข่ายทั้งสอง คือ CCNN จะใช้ฟังก์ชันครอสคอร์รีเลชันในการวัดความคล้ายกันระหว่างแพทเทิร์น ในขณะที่ MFHSNN ใช้ค่าระยะทาง ในหัวข้อนี้จะทำการทดลองกับชุดข้อมูลมาตรฐาน เพื่อเปรียบเทียบความแตกต่างระหว่างการใช้งานฟังก์ชันทั้งสอง โดยจะกำหนดให้ CCNN มีอัลกอริทึมในการเรียนรู้แบบเดียวกันกับ MFHSNN

ชุดข้อมูลมาตรฐานที่ใช้ทดสอบอัลกอริทึมแบบ Classification มาจากแหล่งข้อมูลหลายประเภท เช่น ข้อมูลทางด้านชีวภาพ ข้อมูลเกี่ยวกับการประมวลผลภาพ ข้อมูลการประมวลผลสัญญาณ ข้อมูลที่จำลองสร้างขึ้น และข้อมูลการประมวลผลเสียง เป็นต้น ชุดข้อมูลเหล่านี้มีการเผยแพร่โดยทั่วไป สำหรับชุดข้อมูลที่นำมาใช้ในการทดลองนี้มี 8 ชุดข้อมูล โดยแต่ละชุดมีรายละเอียด ดังนี้

5.1.1 รายละเอียดของชุดข้อมูลมาตรฐาน

5.1.1.1 ชุดข้อมูล Landsat [14]

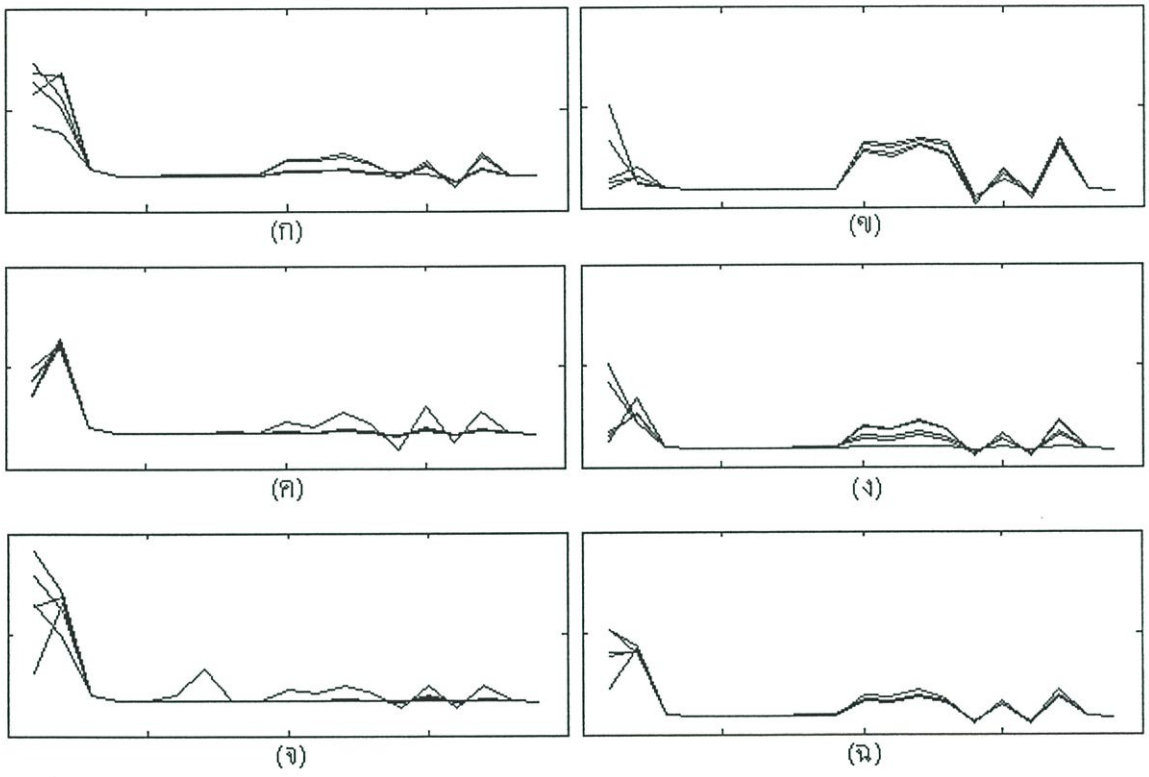
Landsat เป็นข้อมูลภาพถ่ายจากดาวเทียมของพื้นที่ 6 แบบ ชุดข้อมูลประกอบด้วยคุณลักษณะที่เป็นความเข้มของแสงที่ช่วงความถี่ต่างๆ 4 ช่วงความถี่จำนวน 36 คุณลักษณะ และแบ่งเป็นรูปแบบสำหรับเรียนรู้จำนวน 4435 รูปแบบ สำหรับทดสอบจำนวน 2000 รูปแบบ



รูปที่ 5.1 ตัวอย่างชุดข้อมูล Landsat (ก) Red soil (ข) Cotton crop (ค) Grey soil (ง) Damp grey soil (จ) Soil with vegetation stubble (ฉ) Very damp grey soil

5.1.1.2 **ชุดข้อมูล** Outdoor Image [15]

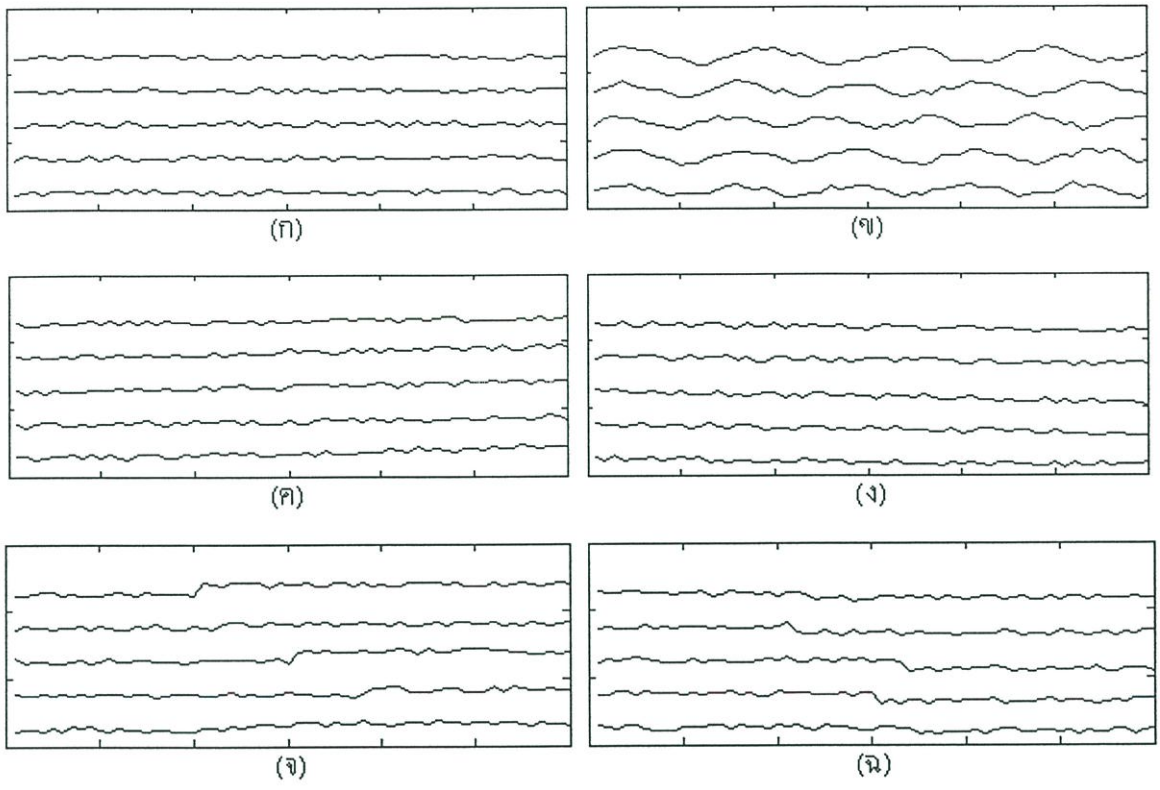
Outdoor Image เป็นชุดข้อมูลภาพถ่ายกลางแจ้งของวัตถุ 7 ประเภท ได้แก่ brick face, sky, foliage, cement, window, path และ grass อย่างละ 330 รูปแบบ รวม 2310 ซึ่งจะสุ่มแบ่งเป็นรูปแบบที่ใช้เรียนรู้จำนวน 1310 รูปแบบ และทดสอบจำนวน 1000 รูปแบบ



รูปที่ 5.2 ตัวอย่างชุดข้อมูล Outdoor Image บางชนิดคลาส (ก) brick face (ข) sky (ค) foliage (ง) cement (จ) window (ฉ) path

5.1.1.3 **ชุดข้อมูล** Synthetic Control Chart Time Series [16]

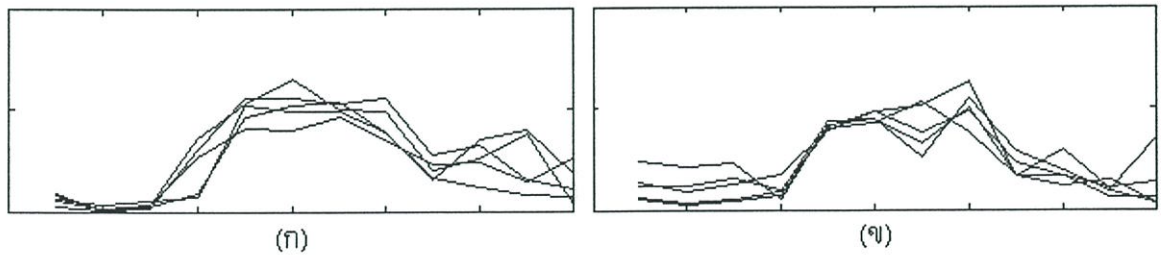
Synthetic Control Chart Time Series เป็นอนุกรมทางเวลาที่ถูกสังเคราะห์ขึ้นโดย Alcock และ Manolopoulos (1999) ชุดข้อมูลเป็นอนุกรมขนาด 60 ตัว แบ่งเป็น 6 ชนิดคลาส คือ Normal, Cyclic, Increasing trend, Decreasing trend, Upward shift และ Downward shift ชนิดละ 100 รูป-แบบ รวม 600 รูปแบบ ในจำนวนนี้ จะแบ่งข้อมูลไว้สำหรับเรียนรู้ 480 รูปแบบ (ชนิดคลาสละ 80 รูปแบบ) และทดสอบ 120 รูปแบบ (ชนิดคลาสละ 20 รูปแบบ) รูปที่ 5.3 แสดงตัวอย่างข้อมูลในแต่ละชนิดคลาส



รูปที่ 5.3 ตัวอย่างชุดข้อมูล Synthetic Control Chart Time Series (ก) Normal (ข) Cyclic (ค) Increasing trend (ง) Decreasing trend (จ) Upward (ฉ) Downward

5.1.1.4 **ชุดข้อมูล** Neuron Image [17]

Neuron Image เป็นชุดข้อมูลที่ใช้ในการแข่งขันอัลกอริทึมแบบ Classification ของ NIPS ในปี 2001 ชุดข้อมูลประกอบด้วยคุณลักษณะ 12 อย่าง ที่สกัดมาจากภาพของเซลล์ประสาท 2 ชนิด แบ่งออกเป็นข้อมูลสำหรับให้ระบบเรียนรู้ 530 รูปแบบ และทดสอบ 2710 รูปแบบ ตัวอย่างของข้อมูลแสดงฝนรูปที่ 5.4

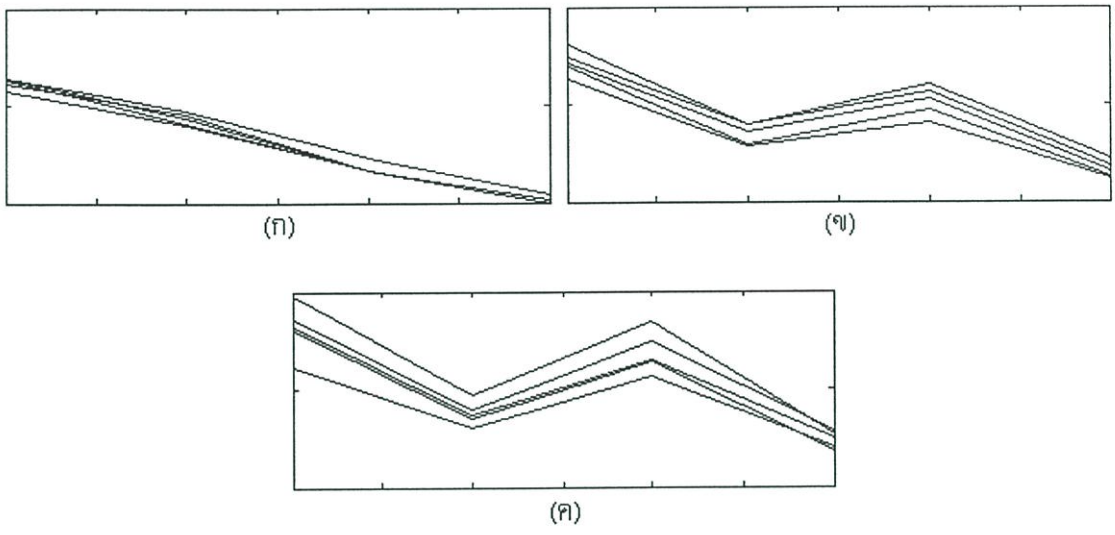


รูปที่ 5.4 ตัวอย่างชุดข้อมูล Neuron Image (ก) ชนิดคลาส 0 (ข) ชนิดคลาส 1

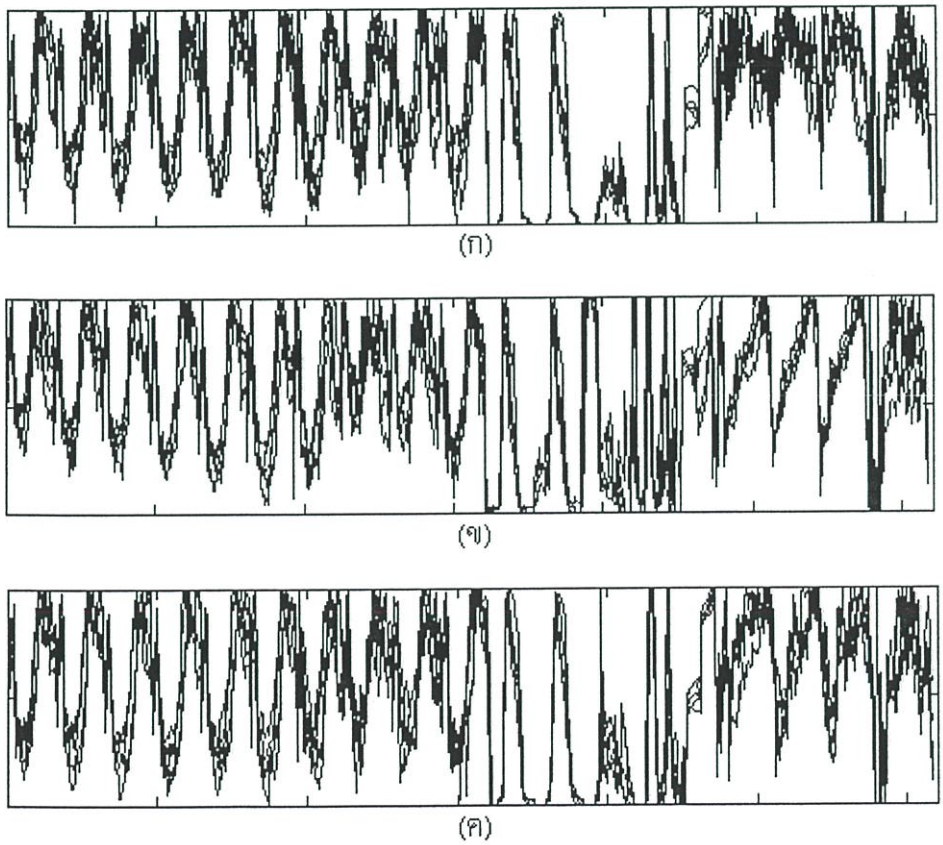
5.1.1.5 **ชุดข้อมูล** Iris [18]

Iris เป็นชุดข้อมูลที่ใช้ในการทดสอบอัลกอริทึมแบบ Classification ค่อนข้างบ่อย อินพุต-แพทเทิร์นในชุดข้อมูลประกอบด้วยคุณลักษณะ 4 อย่าง คือ ความยาวของก้านดอก ความกว้าง

ของก้านดอก ความยาว ของกลีบดอก และความกว้างของกลีบดอก ในหน่วยเซนติเมตร ของดอกไอริส 3 สายพันธุ์ คือ Setosa, Versicolour และ Virginica สายพันธุ์ละ 50 รูปแบบ จากนั้นทำการสุ่ม 40 รูปแบบของแต่ละสายพันธุ์มาให้ระบบเรียนรู้ และที่เหลืออีก 10 รูปแบบของแต่ละสายพันธุ์ใช้ในการทดสอบ



รูปที่ 5.5 ตัวอย่างชุดข้อมูล Iris (ก) Setosa (ข) Versicolour (ค) Virginica



รูปที่ 5.6 ตัวอย่างชุดข้อมูล E-set บางชนิดคลาส (ก) อักษร B (ข) อักษร C (ค) อักษร D

5.1.1.6 **ชุดข้อมูล** E-set [19]

E-set เป็นส่วนหนึ่งของฐานข้อมูล ISOLET (Isolated Letter Speech Recognition) ซึ่งเป็นฐานข้อมูลเสียงพูดตัวอักษรภาษาอังกฤษทั้ง 26 ตัว เสียงพูดจะออกโดยผู้พูด 150 คน รวมจำนวนเสียงทั้งสิ้น 7800 เสียง โดยแต่ละเสียงจะถูกสกัดคุณลักษณะจากสัมประสิทธิ์ของสเปคตรัมจำนวน 617 คุณลักษณะ เนื่องจากฐานข้อมูล ISOLET มีขนาดใหญ่มาก และไม่สามารถทดสอบกับคอมพิวเตอร์ที่ผู้วิจัยใช้งานอยู่ได้เนื่องจากข้อกำจัดทางด้านหน่วยความจำของคอมพิวเตอร์ ดังนั้นจึงได้เลือกทำการทดลองกับชุดข้อมูล E-set ซึ่งมีเฉพาะตัวอักษรที่ลงท้ายด้วยเสียงอี ได้แก่ B, C, D, E, G, P, T, V และ Z เท่านั้น โดยรูปแบบที่ใช้ในการเรียนรู้จะมาจากอักษรแต่ละตัว (จาก 120 คน) ตัวละ 240 รูป-แบบ รวม 2160 รูปแบบ และสำหรับทดสอบ (จาก 30 คน) จำนวน 540 รูปแบบ

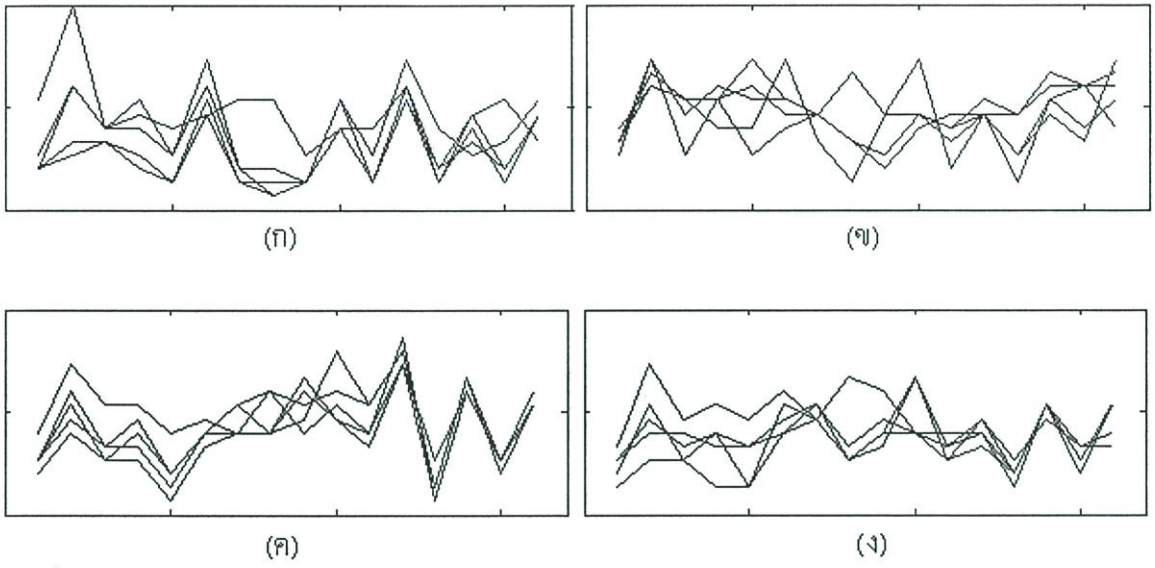
5.1.1.7 **ชุดข้อมูล** Letter [20]

Letter เป็นข้อมูลรูปภาพขาวดำของตัวอักษรพิมพ์ใหญ่ในภาษาอังกฤษ 26 ตัว แต่ละตัวจะมีแบบอักษร (font) 20 แบบ ภาพเหล่านี้จะถูกนำไปดึงคุณลักษณะอย่างง่ายจำนวน 16 คุณลักษณะ รวมรูปแบบที่ได้ทั้งสิ้น 20000 รูปแบบ ซึ่งแบ่งไว้สำหรับเรียนรู้ 15000 รูปแบบและทดสอบ 5000 รูปแบบ

คุณลักษณะที่สกัดมาจากรูปภาพ ได้แก่

- x-box : horizontal position of box
- y-box : vertical position of box
- width : width of box
- high : height of box
- onpix : total # on pixels
- x-bar : mean x of on pixels in box
- y-bar : mean y of on pixels in box
- 2bar : mean x variance
- y2bar : mean y variance
- xybar : mean x y correlation
- x2ybr : mean of x * x * y
- xy2br : mean of x * y * y
- x-ege : mean edge count left to right
- xegvy : correlation of x-edge with y

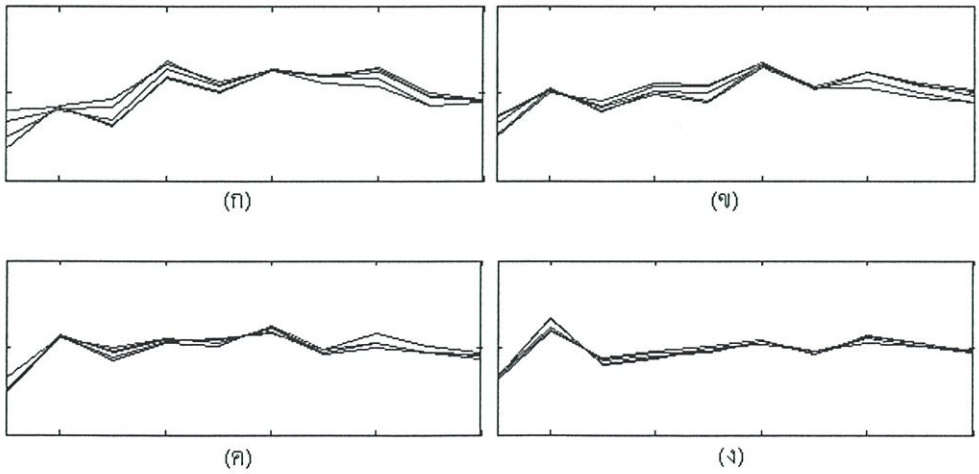
- y-ege : mean edge count bottom to top
- yegvx : correlation of y-edge with x



รูปที่ 5.7 ตัวอย่างชุดข้อมูล Letter บางชนิดคลาส สำหรับอักษร (ก) A (ข) B (ค) C และ (ง) D

5.1.1.8 **ชุดข้อมูล** Vowel [21]

Vowel เป็นชุดข้อมูลเสียงในแบบไม่ขึ้นกับผู้พูดของเสียงสระในภาษาอังกฤษจำนวน 11 เสียง แต่ละเสียงจะใช้ LPC Spectrum จำนวน 10 ค่ามาเป็นคุณลักษณะ รูปแบบที่ใช้ในการเรียนรู้จะมาจากผู้พูดจำนวน 8 คน รวม 528 รูปแบบ และรูปแบบจากผู้พูดอีก 7 คนจำนวน 468 รูปแบบ จะใช้ในการทดสอบ คำ (word) ที่เลือกใช้กับเสียงสระแต่ละประเภทประกอบด้วย heed (เสียง i), hid (เสียง I), head (เสียง E), had (เสียง A), hard (เสียง a:), hud (เสียง Y), hod (เสียง O), hoard (เสียง C:), hood (เสียง U), who'd (เสียง U:) และ heard (เสียง 3:)



รูปที่ 5.8 ตัวอย่างชุดข้อมูล Vowel บางชนิดคลาส สำหรับเสียง (ก) i (ข) I (ค) E และ (ง) A

5.1.2 ผลการทดสอบ

ผลการทดสอบกับชุดข้อมูลมาตรฐานเป็นไปตามตารางที่ 5.1, 5.2 และ 5.3

ตารางที่ 5.1 อัตราความถูกต้องในการแบ่งกลุ่มรูปแบบของชุดข้อมูลมาตรฐาน

ชุดข้อมูล (จำนวนข้อมูลให้ระบบเรียนรู้ / ทดสอบ)	อัตราความถูกต้อง (%)		
	FAM	MFHSNN	CCNN
Landsat (4435/2000)	85.50	89.50	89.35
Outdoor Image (1310/1000)	89.20	95.20	95.30
Synthetic Control Chart (480/120)	100.00	97.50	100.00
Neuron Image (530/2710)	77.53	76.53	77.38
Iris (120/30)	100.00	100.00	100.00
E-set (2160/540)	81.67	81.11	81.11
Letter (15000/5000)	90.42	95.56	94.32
Vowel (528/462)	61.69	56.93	61.69

ตารางที่ 5.2 จำนวน reference pattern ที่สร้างในการแบ่งกลุ่มรูปแบบของชุดข้อมูลมาตรฐาน

ชุดข้อมูล (จำนวนข้อมูลให้ระบบเรียนรู้ / ทดสอบ)	จำนวน reference pattern ที่สร้าง		
	FAM	MFHSNN	CCNN
Landsat (4435/2000)	84	2544	975
Outdoor Image (1310/1000)	28	980	661
Synthetic Control Chart (480/120)	28	370	151
Neuron Image (530/2710)	6	481	278
Iris (120/30)	9	28	18
E-set (2160/540)	1403	2160	1477
Letter (15000/5000)	1414	8196	4576
Vowel (528/462)	231	227	121

ตารางที่ 5.3 ค่าพารามิเตอร์ที่ตั้งให้กับอัลกอริทึมในการทดสอบกับชุดข้อมูลมาตรฐาน

ชุดข้อมูล	FAM			MFHSNN	CCNN
	$\overline{\rho}_a$	β_a	จำนวนรอบในการเรียนรู้	λ	λ
Landsat	0.95	0.10	1	0.20	0.97
Outdoor Image	0.95	0.20	8	0.0075	0.50
Synthetic Control Chart	0.75	0.70	2	0.5525	0.50
Neuron Image	0.75	0.10	1	0.11	0.988
Iris	0.85	0.90	2	0.15	0.9988
E-set	0.90	0.80	2	30	0.95
Letter	0.90	0.50	1	0.40	0.99
Vowel	0.95	0.20	22	0.07	0.995

จากผลการทดลองพบว่า CCNN ที่ใช้อัลกอริทึมในการเรียนรู้แบบเดิมจะให้อัตราความถูกต้องในการแบ่งกลุ่มสูงกว่า MFHSNN ใน 4 ชุดข้อมูล คือ Outdoor Image, Synthetic Control Chart, Neuron Image และ Vowel สำหรับชุดข้อมูล Landsat และ Letter นั้น MFHSNN จะให้ผลที่ดีกว่า ส่วนในชุดข้อมูล Iris และ E-set ทั้งสองอัลกอริทึมจะให้อัตราความถูกต้องเท่ากัน ทั้งนี้ชุดข้อมูลที่ให้ความสำคัญระหว่างรูปร่างของแพทเทิร์นโดยไม่ขึ้นกับขนาด เช่น ชุดข้อมูล Synthetic Control Chart ชุดข้อมูล Neuron Image และชุดข้อมูล Vowel นั้น CCNN จะให้อัตราความถูกต้องที่สูงกว่า MFHSNN

พิจารณาอินพุตแพทเทิร์นในชุดข้อมูล Synthetic Control Chart Time Series ที่ลักษณะของอนุกรมแสดงถึงแนวโน้มในแบบต่างๆ ที่ขึ้นอยู่กับรูปร่างของอนุกรมเป็นสำคัญ ทำให้ CCNN ให้ผลดีกว่า MFHSNN

พิจารณาอินพุตแพทเทิร์นในชุดข้อมูล Vowel รูปที่ 5.8 จะเห็นได้ว่าการวัดค่าระยะทางระหว่างอินพุตแพทเทิร์นในชนิดคลาสเดียวกัน และต่างชนิดคลาสจะให้ผลไม่แตกต่างกันมากนัก แต่การวัดด้วยครอสคอร์รีเลชันจะให้ผลที่ค่อนข้างแตกต่างกัน ทำให้ CCNN ให้อัตราความถูกต้องสูงกว่า MFHSSNN อย่างเห็นได้ชัด สำหรับตัวอย่างของชุดข้อมูล Letter ในรูปที่ 5.7 จะสังเกตได้ว่าอินพุตแพทเทิร์นในชนิดคลาสเดียวกัน ไม่ค่อยมีความสัมพันธ์กันในเชิงของทิศทาง ในกรณีนี้การวัดระยะทางจึงให้ผลที่ดีกว่า

5.2 การทดสอบประสิทธิภาพของ CCNN ที่ใช้อัลกอริทึมในการเรียนรู้แบบใหม่

กระบวนการฝึก CCNN ในหัวข้อนี้จะใช้อัลกอริทึมในการเรียนรู้ที่ได้พัฒนาขึ้นใหม่และได้กล่าวไว้ในหัวข้อที่ 4.3 แล้วทำการทดสอบเปรียบเทียบกับอัลกอริทึม MFHSNN และ FAM โดยใช้ชุดข้อมูลมาตรฐานจำนวน 8 ชุดข้อมูลเช่นเดียวกับหัวข้อที่ 5.1

ผลการทดลองกับชุดข้อมูลมาตรฐานเป็นไปตามตารางที่ 5.4, 5.5 และ 5.6

ตารางที่ 5.4 อัตราความถูกต้องในการแบ่งกลุ่มรูปแบบของชุดข้อมูลมาตรฐาน

ชุดข้อมูล (จำนวนข้อมูลให้ระบบเรียนรู้ / ทดสอบ)	อัตราความถูกต้อง (%)			
	FAM	MFHSNN	CCNN*	CCNN**
Landsat (4435/2000)	85.50	89.50	89.35	90.50
Outdoor Image (1310/1000)	89.20	95.20	95.30	95.10
Synthetic Control Chart (480/120)	100.00	97.50	100.00	100.00
Neuron Image (530/2710)	77.53	76.53	77.38	78.60
Iris (120/30)	100.00	100.00	100.00	100.00
E-set (2160/540)	81.67	81.11	81.11	82.59
Letter (15000/5000)	90.42	95.56	94.32	92.56
Vowel (528/462)	61.69	56.93	61.69	63.64

ตารางที่ 5.5 จำนวน reference pattern ที่สร้างในการแบ่งกลุ่มรูปแบบของชุดข้อมูลมาตรฐาน

ชุดข้อมูล (จำนวนข้อมูลให้ระบบเรียนรู้ / ทดสอบ)	จำนวน reference pattern ที่สร้าง			
	FAM	MFHSNN	CCNN*	CCNN**
Landsat (4435/2000)	84	2544	975	820
Outdoor Image (1310/1000)	28	980	661	594
Synthetic Control Chart (480/120)	28	370	151	134
Neuron Image (530/2710)	6	481	278	141
Iris (120/30)	9	28	18	11
E-set (2160/540)	1403	2160	1477	1466
Letter (15000/5000)	1414	8196	4576	2174
Vowel (528/462)	231	227	121	97

* CCNN ที่ใช้อัลกอริทึมในการเรียนรู้แบบเดิม

** CCNN ที่ใช้อัลกอริทึมในการเรียนรู้แบบใหม่

ตารางที่ 5.6 ค่าพารามิเตอร์ที่ตั้งให้กับอัลกอริทึมในการทดสอบกับชุดข้อมูลมาตรฐาน

ชุดข้อมูล	FAM			MFHSNN	CCNN*	CCNN**
	$\overline{\rho}_a$	β_a	จำนวนรอบในการเรียนรู้	λ	λ	λ
Landsat	0.95	0.10	1	0.20	0.97	0.99
Outdoor Image	0.95	0.20	8	0.0075	0.50	0.50
Synthetic Control Chart	0.75	0.70	2	0.5525	0.50	0.981
Neuron Image	0.75	0.10	1	0.11	0.988	0.92
Iris	0.85	0.90	2	0.15	0.9988	0.50
E-set	0.90	0.80	2	30	0.95	0.95
Letter	0.90	0.50	1	0.40	0.99	0.98
Vowel	0.95	0.20	22	0.07	0.995	0.9905

* CCNN ที่ใช้อัลกอริทึมในการเรียนรู้แบบเดิม

** CCNN ที่ใช้อัลกอริทึมในการเรียนรู้แบบใหม่

จากผลการทดลองพบว่า CCNN ที่ใช้อัลกอริทึมในการเรียนรู้แบบใหม่จะให้อัตราความถูกต้องในการแบ่งกลุ่มค่อนข้างสูง โดยส่วนใหญ่แล้วจะเห็นผลดีที่สุดเมื่อเทียบกับอีก 2 อัลกอริทึม ส่วนจำนวน reference pattern ที่สร้างโดยส่วนใหญ่จะน้อยกว่า MFHSNN แต่มากกว่า FAM

เมื่อเปรียบเทียบอัตราความถูกต้องระหว่าง CCNN ที่ใช้อัลกอริทึมในการเรียนรู้แบบใหม่กับ CCNN ที่ใช้อัลกอริทึมในการเรียนรู้แบบเดิมแล้ว พบว่า อัลกอริทึมในการเรียนรู้แบบใหม่สามารถทำให้ประสิทธิภาพของ CCNN เพิ่มขึ้นอย่างเห็นได้ชัดในหลายๆชุดข้อมูล ทั้งนี้อาจจะเป็นผลมาจาก อัลกอริทึมในการเรียนรู้แบบใหม่สามารถที่จะพิจารณานำอินพุตแพทเทิร์นมาสร้างเป็น reference pattern ได้อย่างเหมาะสมมากขึ้น นอกจากนี้ยังพบว่าจำนวน reference pattern ที่สร้างโดยอัลกอริทึมแบบใหม่จะมีจำนวนน้อยกว่ามาก ซึ่งจะมีข้อดีในแง่ของเวลาที่ใช้ในช่วงการทดสอบ

5.3 การทดสอบกับชุดข้อมูลเสียงพูดตัวเลขภาษาไทย

การเก็บตัวอย่างเสียงพูดจะใช้เครื่องคอมพิวเตอร์ส่วนบุคคลและการ์ดเสียง โดยทำการเก็บจากผู้พูดที่เป็นผู้ชายจำนวน 4 คนและผู้หญิงจำนวน 8 คน รวม 12 คน ซึ่งมีอายุอยู่ในช่วง 22-35 ปี โดยแต่ละคนจะออกเสียง ศูนย์ หนึ่ง สอง สาม สี่ ห้า หก เจ็ด แปด และเก้า อย่างละ 10 ครั้ง ดังนั้นจำนวนเสียงทั้งหมดที่บันทึกได้จะเท่ากับ 1200 เสียง ข้อมูลเสียงจะถูกจัดเก็บอยู่ในรูป

ไฟล์ข้อมูลแบบ '.wav' ผ่านทางโปรแกรม Sound Recorder ของ Microsoft Windows XP ข้อมูล 1 ตัวอย่างเสียงจะแทนด้วยข้อมูลขนาด 8 บิต ซึ่งใช้ความถี่ในการชักตัวอย่าง (sampling frequency) เท่ากับ 11.025 kHz หลังจากนั้นก็จะนำไฟล์ข้อมูลเสียงเหล่านี้มาทำพรีโปรเซสซิง (pre-processing) ด้วยกระบวนการต่างๆ ในหัวข้อที่ 5.3.1

5.3.1 กระบวนการพรีโปรเซสซิงสัญญาณเสียง

ขั้นตอนในการประมวลผลสัญญาณเสียงที่ได้จากการเก็บบันทึกจะมีลำดับ ดังนี้

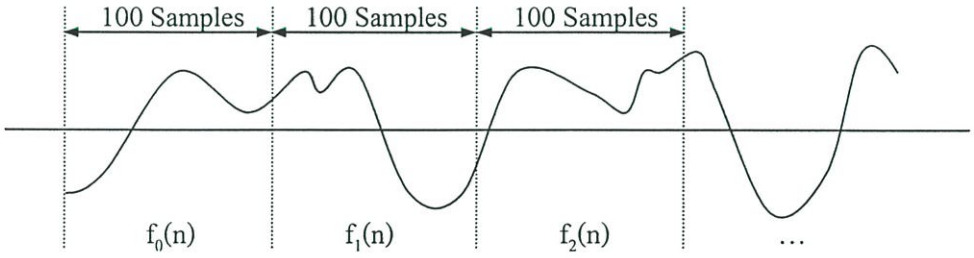
5.3.1.1 การหาจุดเริ่มและจุดท้ายของเสียงพูด (End Point Detection)

เนื่องจากกระบวนการในการเก็บเสียง จะกระทำไปทีละคำ โดยผู้เก็บเสียงจะเป็นผู้กำหนดจุดเริ่มและจุดหยุดพูดเอง ดังนั้นเสียงที่ถูกบันทึกไว้ในไฟล์ อาจจะไม่ใช่ส่วนของเสียงพูดทั้งหมด จึงจำเป็นที่จะต้องตัดสัญญาณบางส่วนที่ไม่ใช่เสียงพูดออกไป วิธีการ คือ การแบ่งสัญญาณเสียงทั้งหมดในไฟล์ออกเป็นเฟรมเล็กๆ ในวิทยานิพนธ์นี้ได้ใช้ขนาดเฟรมเท่ากับ 100 ตัวอย่าง ดังแสดงในรูปที่ 5.9

กำหนดให้ s(n) คือสัญญาณเสียงที่บันทึกไว้ในไฟล์ และ $f_k(n)$ คือสัญญาณของเฟรมที่ k ดังนั้น

$$f_k(n) = s(100k + n), \qquad n = 0, 1, \ldots, 99, \quad k = 0, 1, \ldots, K\text{-}1 \qquad \ldots(5.1)$$

เมื่อ K คือ จำนวนเฟรมทั้งหมดตลอดช่วงสัญญาณเสียง



รูปที่ 5.9 การแบ่งเฟรมเพื่อหาจุดเริ่มต้นและสิ้นสุดของเสียงพูด

หลังจากนั้นก็จะคำนวณหาค่าพลังงานของแต่ละเฟรมตามสมการ

$$E(f_k) = \sum_{n=0}^{99} s^2(100k + n), \qquad \ldots(5.2)$$

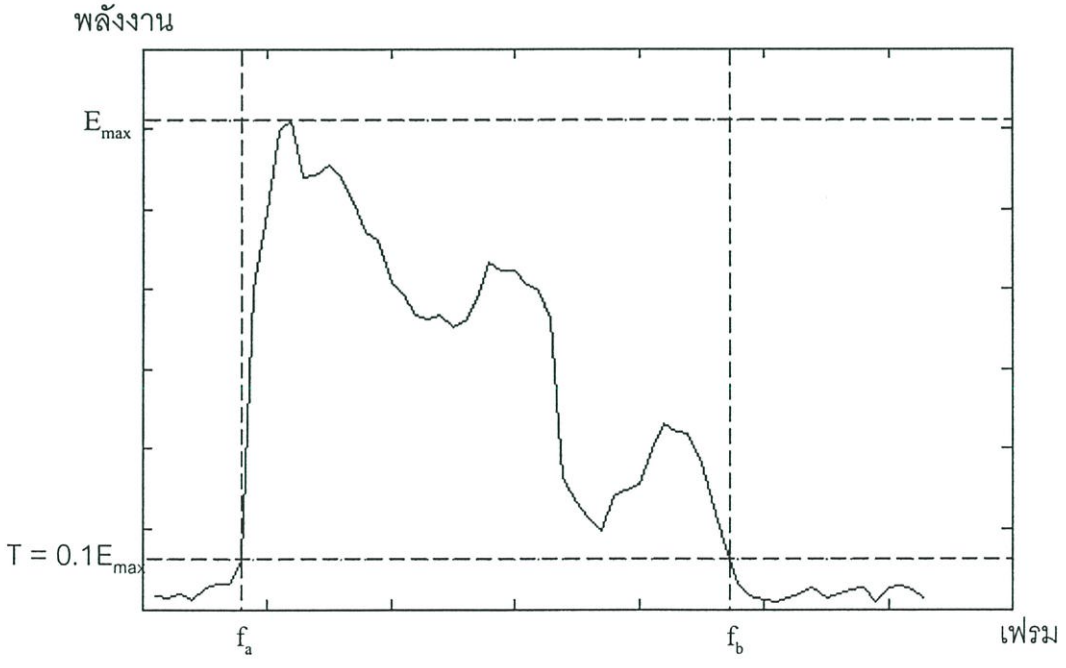
เพื่อความสะดวกในการคำนวณ จะลดรูปสมการที่ 5.2 ลงเป็น

$$E(f_k) = \sum_{n=0}^{99} |s(100k + n)|, \qquad ...(5.3)$$

เมื่อ $E(f_k)$ คือ ค่าพลังงานของเฟรมที่ k

แล้วจึงกำหนดระดับพลังงาน (threshold) T เพื่อใช้หาจุดเริ่มและจุดท้ายของเสียงพูด ดังแสดงในรูปที่ 5.10 โดยในที่นี้กำหนดให้ T มีค่าเท่ากับ 10 เปอร์เซ็นต์ของพลังงานของเฟรมที่มีค่าพลังงานสูงสุด (E_{max}) ซึ่งสัญญาณที่จะถูกตัดออกไป คือ สัญญาณจากเฟรมที่มีพลังงานน้อยกว่า 10 เปอร์เซ็นต์ของพลังงานสูงสุด

$$T = 0.1Emax \qquad ...(5.4)$$



รูปที่ 5.10 เฟรมเริ่มต้น (f_a) และเฟรมสุดท้าย (f_b) ของสัญญาณเสียงพูด

ถ้ากำหนดให้ a คือเฟรมเริ่มต้นและ b คือเฟรมสุดท้ายของเสียงพูด จะสามารถเขียนสัญญาณเสียงพูดที่ผ่านการตัดสัญญาณแล้ว ($\hat{s}$) ได้ดังนี้

$$\hat{s} = s(100a + n), \qquad n = 0, 1, ..., (b - a + 1) * 100 - 1 \qquad ...(5.5)$$

เมื่อ s(n) คือ สัญญาณเสียงก่อนการตัดสัญญาณ

5.3.1.2 **การเติมศูนย์** (Zero Padding)

เนื่องจากสัญญาณเสียงพูดแต่ละสัญญาณที่ผ่านการตัดสัญญาณเสียงอื่นออกแล้ว จะมีขนาดความยาวไม่เท่ากัน และเนื่องจากลักษณะการทำงานของอัลกอริทึมที่ต้องการอินพุต-แพทเทิร์นที่มีขนาดเท่ากันทุกตัว ดังนั้นจึงจำเป็นที่จะต้องทำให้ทุกสัญญาณเสียงมีจำนวนตัวอย่าง (sample) เท่ากันด้วย สำหรับการทำให้สัญญาณมีขนาดเท่ากันสามารถทำได้หลายวิธี เช่น วิธีการเติมศูนย์ต่อท้าย หรือ วิธีการ re-sampling สัญญาณใหม่ เป็นต้น วิธีการที่เลือกใช้ในวิทยานิพนธ์นี้ คือ วิธีการเติมศูนย์ ซึ่งมีการอ้างอิงว่าให้อัตราความถูกต้องสูงกว่า [11]

สัญญาณเสียงพูดที่ได้จากหัวข้อที่ 5.3.1.1 จะมีขนาดความยาวมากที่สุดเท่ากับ 8400 ตัวอย่าง ดังนั้นสัญญาณอื่นที่มีความยาวน้อยกว่านี้ จะถูกเติมศูนย์ต่อท้ายสัญญาณจนมีขนาดเท่ากันหมดทุกตัว นั่นคือขนาด 8400 ตัวอย่าง

5.3.1.3 **พรีเอมฟาซิส** (Pre-emphasis)

หลังจากนั้นสัญญาณที่ได้จะนำมาผ่านกระบวนการพรีเอมฟาซิส ซึ่งเป็นกระบวนการในการกรองสัญญาณ (filter) ชนิดหนึ่งด้วยระบบดิจิตอลอันดับต่ำ (low-order digital system) จุดประสงค์ในการทำพรีเอมฟาซิส ก็เพื่อที่จะทำให้อัตราส่วนสัญญาณเสียงต่อสัญญาณรบกวน (Signal to Noise Ratio : SNR) มีค่าค่อนข้างคงที่ตลอดช่วงความถี่ปฏิบัติงาน เนื่องจากองค์ประกอบทางด้านความถี่สูงของเสียงพูดมีขนาดค่อนข้างต่ำ การทำพรีเอมฟาซิสจะเน้นองค์ประกอบเหล่านี้ให้มีขนาดสูงขึ้น

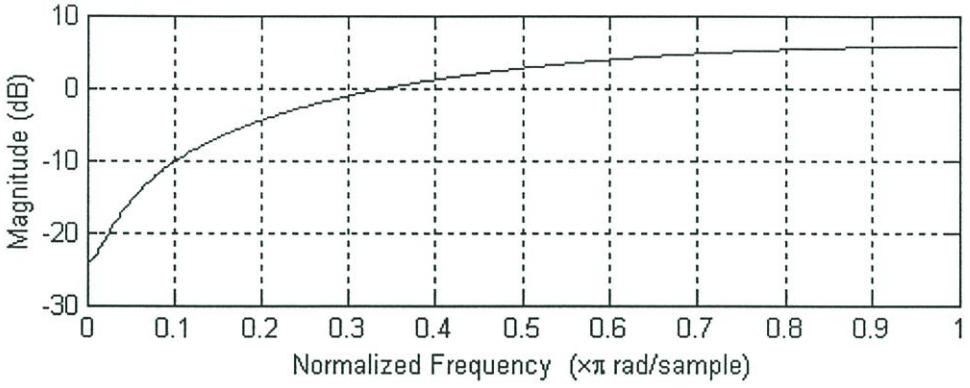
ระบบที่ใช้ในการทำพรีเอมฟาซิสจะมีทั้งแบบคงที่ (fixed) และแบบปรับเปลี่ยนอย่างช้าๆ (slowly adaptive) ระบบหนึ่งที่มีการใช้งานกันอย่างแพร่หลาย คือ ระบบอันดับหนึ่งแบบคงที่ (fixed first-order system) [9, หน้า 112] ซึ่งมีฟังก์ชันถ่ายโอน (transfer function) ดังนี้

$$H(z)= 1-az^{-1}, \qquad 0.9 \leq a \leq 1 \qquad ...(5.6)$$

ในกรณีนี้ สัญญาณขาออกของการทำพรีเอมฟาซิส ($s'(n)$) จะมีความสัมพันธ์กับสัญญาณขาเข้า (s(n)) ด้วยสมการผลต่างสืบเนื่อง (difference equation) ดังนี้

$$s'(n) = s(n) - as(n-1) \qquad ...(5.7)$$

โดยทั่วไป ค่าของ a ในสมการที่ 5.5 และ 5.6 จะอยู่ที่ประมาณ 0.95 และสำหรับระบบในแบบ fixed-point ค่าที่นิยมใช้กันบ่อย คือ a = 15/16 = 0.9375 [9, หน้า 113] ซึ่งเป็นค่าที่ถูกใช้ในวิทยานิพนธ์นี้ด้วย



รูปที่ 5.11 ขนาดสเปคตรัมของระบบพรีเอมฟาซิสแบบ fixed-point ที่มี a = 0.9375

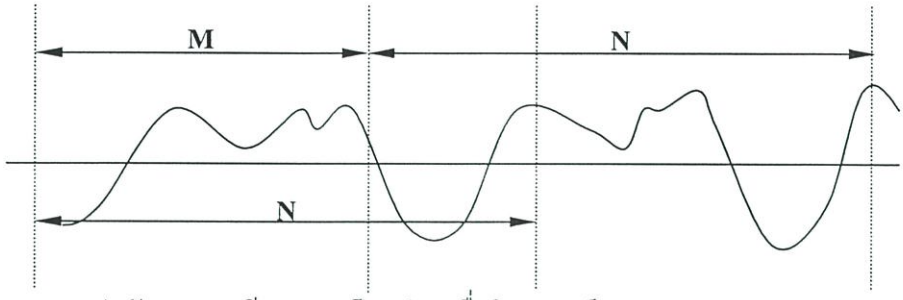
5.3.1.4 การแบ่งสัญญาณออกเป็นเฟรม (Frame Blocking)

ในขั้นตอนนี้ สัญญาณที่ผ่านการทำพรีเอมฟาซิสแล้ว ($s'(n)$) จะถูกแบ่งออกเป็นเฟรม ก่อนที่แต่ละเฟรมจะถูกนำไปวิเคราะห์เพื่อดึงค่าสัมประสิทธิ์ LPC ซึ่งจะกล่าวถึงในภายหลัง สัญญาณแต่ละเฟรมจะประกอบด้วยตัวอย่างเสียงจำนวน N ตัวอย่าง และเฟรมที่อยู่ถัดไปจะอยู่ห่างออกไป M ตัวอย่าง รูปที่ 5.12 แสดงตำแหน่งของแต่ละเฟรมในกรณีที่ M = 2N/3 โดยที่เฟรมแรกจะประกอบด้วยตัวอย่างเสียง N ตัวอย่างแรก และเฟรมที่สองจะเริ่มจากระยะ M ตัวอย่างหลังจากเฟรมแรก ซึ่งทำให้เกิดการทับกันระหว่างเฟรมอยู่ N-M ตัวอย่าง การแบ่งสัญญาณเสียงออกเป็นเฟรมจะทำอย่างนี้ไปเรื่อย ๆ จนกว่าจะครอบคลุมสัญญาณเสียงทั้งหมด จะสังเกตเห็นว่าถ้า M>N จะทำให้บางส่วนของเสียงหายไป และไม่ได้ถูกวิเคราะห์ ทำให้ LPC Spectral อาจจะประกอบด้วยสัญญาณรบกวนบางอย่างที่มีค่าขึ้นอยู่กับขนาดของ M

สำหรับวิทยานิพนธ์นี้ได้เลือกใช้ค่า N เท่ากับ 300 ตัวอย่าง และ M เท่ากับ 200 ตัวอย่าง ด้วยอัตราการชักตัวอย่างในการบันทึกเสียงที่ 11.025 kHz ทำให้ขนาดของเฟรมมีความยาวประมาณ 27.21 มิลลิวินาที ซึ่งเป็นช่วงเวลาที่สัญญาณเสียงพูดถูกอนุโลมได้ว่าเป็นสัญญาณที่มีคุณลักษณะคงที่ (Stationary signal)

ถ้ากำหนดให้สัญญาณเสียงในเฟรมที่ k เป็น $x_k(n)$ และใช้ K เฟรมเพื่อครอบคลุมสัญญาณเสียงทั้งหมด จะได้ว่า

$$x_k(n) = s(M \cdot k + n), \quad n = 0, 1, \ldots, N-1, \quad k = 0, 1, \ldots, K-1 \qquad \ldots(5.8)$$



รูปที่ 5.12 การแบ่งสัญญาณเสียงออกเป็นเฟรมเพื่อวิเคราะห์

5.3.1.5 **การวินโดว์** (Windowing)

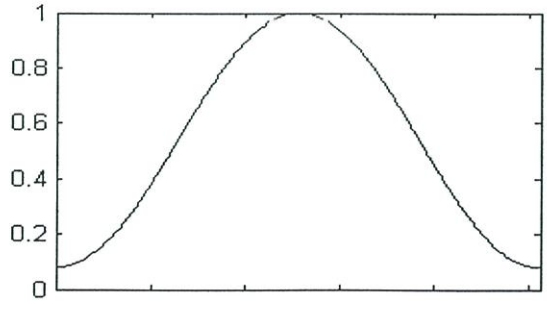
จะสังเกตเห็นได้ว่าสัญญาณของแต่ละเฟรมที่ถูกแบ่งออกมา จะไม่มีความต่อเนื่องบริเวณขอบของเฟรมเนื่องจากมีการเปลี่ยนแปลงของข้อมูลอย่างกะทันหัน ซึ่งจะให้องค์ประกอบที่ไม่ต้องการทางด้านความถี่สูงออกมา ดังนั้นเพื่อที่จะลดปรากฏการณ์นี้ จึงต้องทำการลดความไม่ต่อเนื่องด้วยการคูณสัญญาณ ด้วยฟังก์ชันวินโดว์ที่ไม่มีผลกระทบต่อองค์ประกอบของสัญญาณทางด้านความถี่ต่ำมากนักฟังก์ชันวินโดว์จะทำให้สัญญาณบริเวณขอบค่อยๆลดลงเป็นศูนย์

ถ้าให้ w(n), 0 < n < N-1 เป็นฟังก์ชันวินโดว์ จะได้ว่าสัญญาณที่ผ่านการวินโดว์แล้ว คือ

$$x'_k(n) = x_k(n) \cdot w(n), \qquad 0 \le n \le N-1 \qquad ...(5.9)$$

รูปแบบของวินโดว์ที่เป็นที่นิยมใช้ในงานทางด้านการรู้จำ คือ แฮมมิงวินโดว์ (Hamming window) [9, หน้า 114] และเป็นฟังก์ชันวินโดว์ที่เลือกใช้ในวิทยานิพนธ์นี้ นิยามของแฮมมิงวินโดว์เป็นไปตามสมการที่ 5.10

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \qquad 0 \le n \le N-1 \qquad ...(5.10)$$



รูปที่ 5.13 ฟังก์ชันแฮมมิงวินโดว์

5.3.1.6 **การวิเคราะห์การประมาณเชิงเส้น** (Linear Predictive Coding Analysis)

การประมาณเชิงเส้น (Linear Predictive Coding : LPC) เป็นวิธีการที่ถูกพัฒนาขึ้นเพื่อลดอัตราการส่งสัญญาณดิจิตอลลงอย่างมากๆ โดยทำการวิเคราะห์สัญญาณเสียงที่ผ่านเข้ามาเพื่อหาชุดของสัมประสิทธิ์ตัวทำนาย (predictor coefficient) หลักการพื้นฐานของแบบจำลอง LPC คือ ตัวอย่างเสียงที่เวลา n, s(n) สามารถที่จะประมาณได้โดยการรวมแบบเชิงเส้นของตัวอย่างเสียง ก่อนหน้าจำนวน p ตัว ดังนี้

$$s(n) = a_1 s(n-1) + a_2 s(n-2) + \ldots + a_p s(n-p) \qquad \ldots(5.11)$$

เมื่อ $a_1, a_2, \ldots, a_p$ คือ สัมประสิทธิ์ตัวทำนายที่สมมุติให้เป็นค่าคงที่ตลอดช่วงของเฟรมที่วิเคราะห์ และเมื่อเพิ่ม excitation term,Gu(n) จะทำให้สมการที่ 5.11 เปลี่ยนเป็น

$$s(n) = \sum_{i=1}^{p} a_i s(n-i) + Gu(n) \qquad \ldots(5.12)$$

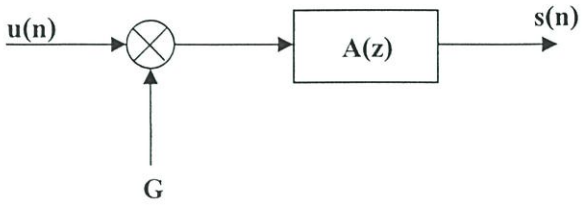
เมื่อ u(n) คือ normalized excitation และ G คือ อัตราการขยาย ด้วยการเปลี่ยนสมการที่ 5.12 ไปสู่โดเมน Z จะได้ว่า

$$S(z) = \sum_{i=1}^{p} a_i z^{-1} S(z) + GU(z) \qquad \ldots(5.13)$$

ซึ่งจะทำให้ได้ฟังก์ชันถ่ายโอน ดังนี้

$$H(z) = \frac{S(z)}{GU(z)} = \frac{1}{1 - \sum_{i=1}^{p} a_i z^{-1}} = \frac{1}{A(z)} \qquad \ldots(5.14)$$

รูปที่ 5.14 แสดงถึงความหมายตามสมการที่ 5.14 นั่นคือ แหล่งกำเนิด normalized excitation u(n) ถูกขยายด้วยอัตราการขยาย G และส่งเป็นอินพุทให้กับระบบที่มีฟังก์ชันถ่ายโอน H(z) = 1/A(z) เพื่อที่จะสร้างสัญญาณเสียง s(n) และจากการทดลองพบว่า ฟังก์ชัน excitation สำหรับสัญญาณเสียง จำเป็นที่จะต้องเป็นขบวนพัลส์ที่เป็นกึ่งรายคาบ (สำหรับ voiced region) และเป็นฟังก์ชันสัญญาณรบกวนแบบสุ่ม (สำหรับ unvoiced region) [9, หน้า 100]



รูปที่ 5.14 แบบจำลองการทำนายเชิงเส้นของเสียงพูด

รูปที่ 5.15 แสดงแบบจำลองที่เหมาะสมในการสังเคราะห์สัญญาณเสียง โดยหลักการวิเคราะห์แบบ LPC จากรูปจะเห็นว่าแหล่งกำเนิด normalized excitation จะถูกเลือกโดยสวิตช์ที่ควบคุมโดยลักษณะ voiced/unvoiced ของเสียง ขนาดของอัตราการขยาย G จะถูกประเมินจากสัญญาณเสียง และเมื่อผ่านการขยายแล้วจะส่งเป็นอินพุทให้กับ digital filter (H(z)) ที่ถูกควบคุมโดยลักษณะของสัญญาณเสียง (vocal tract parameters) ดังนั้น ค่าพารามิเตอร์ของแบบจำลอง ดังกล่าวจะประกอบด้วย การแยกส่วนของ voiced/unvoiced, ระยะพิตช์ (pitch period) ของส่วน voiced, ค่าอัตราการขยาย และสัมประสิทธิ์ของ digital filter (a_k) ซึ่งค่าพารามิเตอร์เหล่านี้จะเปลี่ยน-แปลงอย่างช้า ๆ ตามเวลา

พิจารณาแบบจำลองตามรูปที่ 5.14 ซึ่งให้ความสัมพันธ์ระหว่าง s(n) และ u(n)

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + Gu(n) \qquad ...(5.15)$$

โดยการพิจารณาให้ $\tilde{s}(n)$ คือ สัญญาณประมาณซึ่งเป็นผลการรวมเชิงเส้นของตัวอย่างสัญญาณเสียงที่อยู่ก่อนหน้า ดังนี้

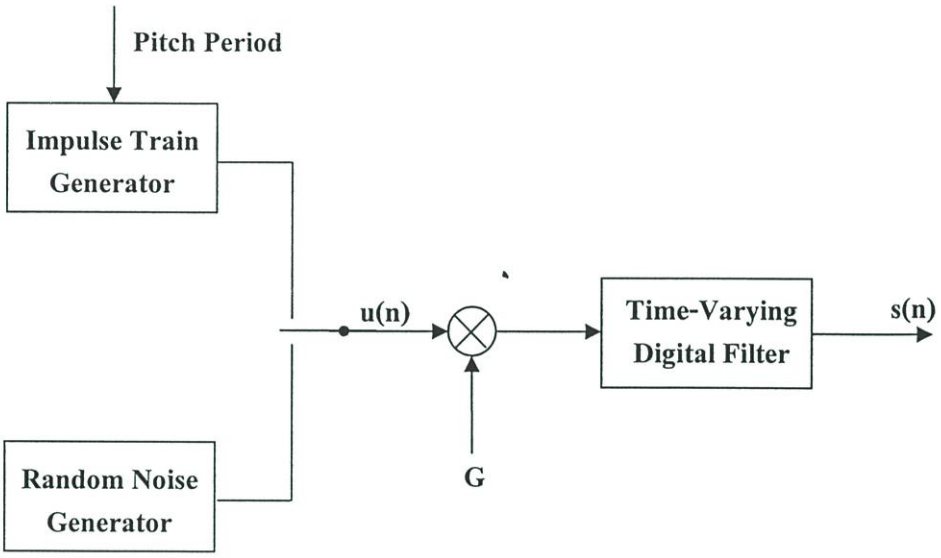
$$\tilde{s}(n) = \sum_{k=1}^{p} a_k s(n-k) \qquad ...(5.16)$$

และนิยามค่าความผิดพลาดของการทำนาย (prediction error, e(n)) คือ

$$e(n) = s(n) - \tilde{s}(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k) \qquad ...(5.17)$$

ปัญหาของการวิเคราะห์ LPC คือ การพิจารณาหาเซตของสัมประสิทธิ์ตัวทำนาย (a_k) โดยตรงจากสัญญาณเสียง เพื่อไปกำหนดคุณสมบัติทางความถี่ของ digital filter ในรูปที่ 5.15

ในการสังเคราะห์สัญญาณเสียง เนื่องจากคุณสมบัติทางความถี่ของเสียงเปลี่ยนแปลงตามเวลา ดังนั้น ค่าสัมประสิทธิ์ที่เวลา n ใด ๆ จะต้องประเมินมาจากช่วงสั้น ๆ ของสัญญาณเสียงรอบ ๆ เวลา n หลักการพื้นฐานในการหาเซตของสัมประสิทธิ์ คือ การหาเซตของสัมประสิทธิ์ที่ทำให้ผลรวมของค่าความผิดพลาดของการทำนายยกกำลังสอง (mean-squared prediction error) มีค่าต่ำที่สุด



รูปที่ 5.15 แบบจำลองการสังเคราะห์เสียงพูดตามรูปแบบของ LPC

เพื่อที่จะสร้างสมการที่สามารถแก้หาค่าสัมประสิทธิ์ดังกล่าวได้ จะนิยามสัญญาณเสียงและค่าความผิดพลาดในช่วงสั้น ๆ ที่เวลา n ดังนี้

$$s_n(m) = s(n+m) \qquad ...(5.18)$$

$$e_n(m) = e(n+m) \qquad ...(5.19)$$

และนิยามผลรวมของค่าความผิดพลาดของการทำนายยกกำลังสอง

$$E_n = \sum_m e_n^2(m) \qquad ...(5.20)$$

$$= \sum_m \left[s_n(m) - \sum_{k=1}^{p} a_k s_n(m-k) \right]^2 \qquad ...(5.21)$$

ในการแก้สมการที่ 5.21 เพื่อหาสัมประสิทธิ์ตัวทำนาย จะทำได้โดยการหาค่าอนุพันธ์ของ E_n เทียบกับสัมประสิทธิ์ a_k และให้ค่าเป็นศูนย์

$$\frac{\partial E_n}{\partial ak} = 0, \qquad k = 1, 2, ..., p \qquad ...(5.22)$$

และให้ชุดของสมการ

$$\sum_m s_n(m-i)s_n(m) = \sum_{k=1}^{p} \hat{a}_k \sum_m s_n(m-i)s_n(m-k) \qquad ...(5.23)$$

เมื่อ $\hat{a}_k$ คือ ชุดของสัมประสิทธิ์ตัวทำนาย ที่ทำให้ผลรวมของค่าความผิดพลาดของการทำนายยกกำลังสองมีค่าต่ำที่สุด

และเมื่อกำหนดให้ $\sum s_n(m-i)s_n(m-k)$ เป็น short-term covariance ของ $s_n(m)$ นั่นคือ

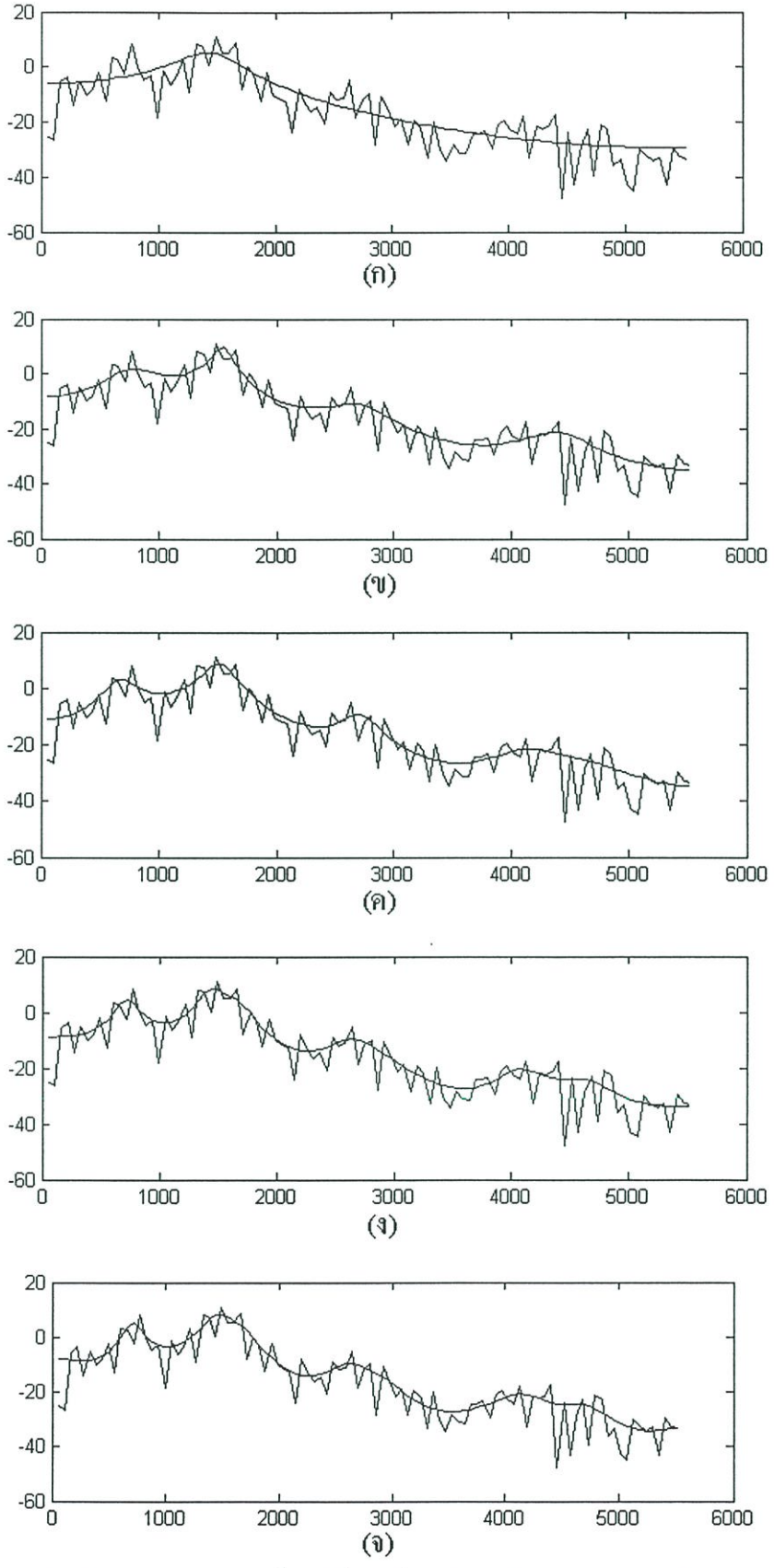
$$\phi_n(i,k) = \sum_m s_n(m-i)s_n(m-k) \qquad ...(5.24)$$

เราสามารถลดรูปสมการที่ 5.23 ลงได้ เป็น

$$\phi_n(i,0) = \sum_{k=1}^{p} \hat{a}_k \phi_n(i,k) \qquad ...(5.25)$$

สมการที่ 5.25 จะให้เซตของสมการจำนวน p สมการ โดยที่มีตัวแปรไม่ทราบค่า p ตัว ($\hat{a}_1$, $\hat{a}_2$, ..., $\hat{a}_p$) ซึ่งเมื่อคำนวณหาค่า $\phi_n(i, k)$ สำหรับ $1 \leq i \leq p$ และ $0 \leq k \leq p$ ก็สามารถนำไปแก้สมการหาค่าสัมประสิทธิ์ตัวทำนายที่เหมาะสม ($\hat{a}_k$) ได้

ที่ได้กล่าวไป เป็นการวิเคราะห์หาสัมประสิทธิ์ตัวทำนายจำนวน p ตัว (p คือ อับดับในการทำนาย) ของสัญญาณแต่ละเฟรม ในการสร้างอินพุตแพทเทิร์นจะนำค่าสัมประสิทธิ์ที่ได้จากทุกเฟรมมาต่อกัน (concatenate) สำหรับจำนวนเฟรมที่ได้ในหัวข้อที่ 5.3.1.4 มีค่าเท่ากับ 42 เฟรม และในวิทยานิพนธ์นี้ได้เลือกใช้การวิเคราะห์การประมาณเชิงเส้นที่อันดับ 14 ดังนั้น จำนวนคุณลักษณะในแต่ละอินพุตแพทเทิร์นจะเป็น 588 คุณลักษณะ



รูปที่ 5.16 สเปคตรัมของสัญญาณเสียงเปรียบเทียบกับสเปคตรัมของการประมาณเชิงเส้นอันดับ (order) (ก) p = 4 (ข) p = 8 (ค) p = 12 (ง) p = 16 (จ) p = 20

5.3.2 ผลการทดลองกับชุดข้อมูลเสียงพูดตัวเลขภาษาไทย

จำนวนอินพุตแพทเทิร์นที่ใช้ในการทดสอบประสิทธิภาพการแบ่งกลุ่มกับชุดข้อมูลเสียงพูดตัวเลขภาษาไทยจะมีทั้งหมด 1200 ตัว จากสัญญาณเสียงที่เก็บบันทึกไว้ 1200 เสียง ดังที่ได้กล่าวไปในหัวข้อที่ 5.3.1 โดยการทดลองจะแบ่งออกเป็น 2 โหมด คือ โหมดที่ขึ้นกับผู้พูด (Speaker Dependent Mode) และโหมดที่ไม่ขึ้นกับผู้พูด (Speaker Independent Mode) ซึ่งมีรายละเอียดดังนี้

5.3.2.1 โหมดที่ขึ้นกับผู้พูด (Speaker Dependent Mode)

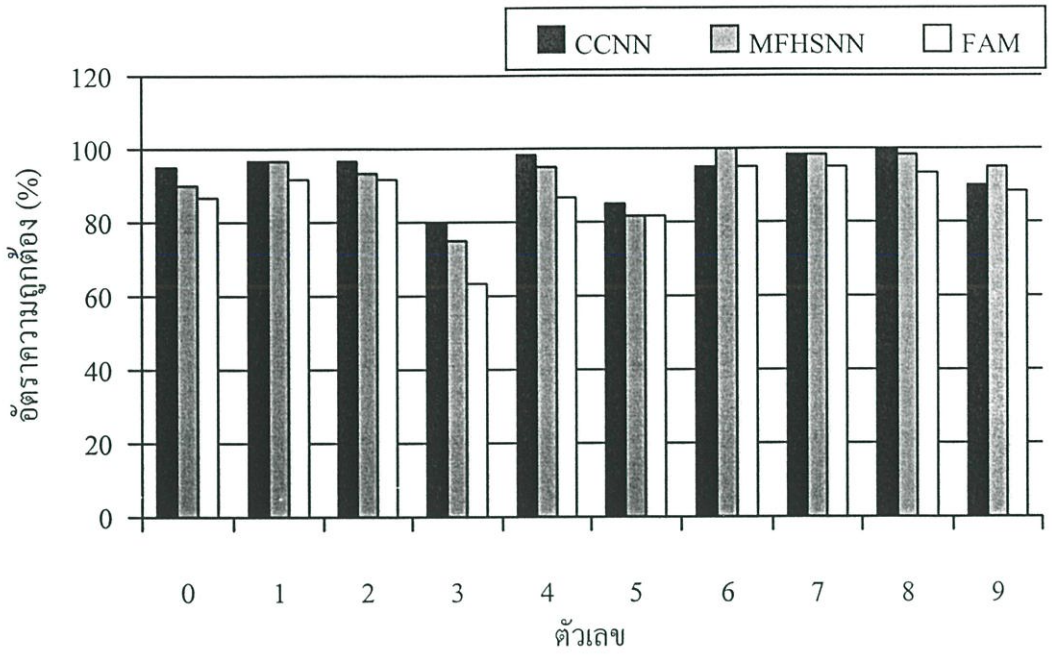
ในโหมดนี้ อินพุตแพทเทิร์นที่ใช้สำหรับฝึก (train) และอินพุตแพทเทิร์นที่ใช้สำหรับทดสอบ (test) จะประกอบด้วยอินพุตแพทเทิร์นที่มาจากเสียงพูดทุกตัวเลขของทุกคน กล่าวคือ เสียงตัวเลขศูนย์ถึงเก้าจำนวนตัวเลขละ 5 ครั้ง (จาก 10 ครั้ง) จากผู้พูด 12 คนรวมทั้งหมด 600 เสียงจะใช้ในการฝึกโครงข่าย ส่วนอีก 600 เสียงที่เหลือจะใช้สำหรับการทดสอบ ผลของการทดสอบแสดงดังตารางที่ 5.7 และ 5.8 และรูปที่ 5.17

ตารางที่ 5.7 อัตราความถูกต้องในการแบ่งกลุ่มรูปแบบของชุดข้อมูลเสียงพูดตัวเลขภาษาไทยในโหมดที่ขึ้นกับผู้พูด

อัลกอริทึม	จำนวน Reference pattern ที่สร้าง	อัตราความถูกต้อง (%)
CCNN	289	93.50
MFHSNN	557	92.33
FAM	189	87.33

ตารางที่ 5.8 ค่าพารามิเตอร์ที่ตั้งให้กับอัลกอริทึมในโหมดที่ขึ้นกับผู้พูด

FAM			MCCNN	CCNN
$\overline{\rho}_a$	β_a	จำนวนรอบในการเรียนรู้	λ	λ
0.9	0.5	10	1	0.9929



รูปที่ 5.17 อัตราความถูกต้องสำหรับแต่ละตัวเลขในโหมดที่ขึ้นกับผู้พูด

5.3.2.2 **โหมดที่ไม่ขึ้นกับผู้พูด** (Speaker Independent Mode)

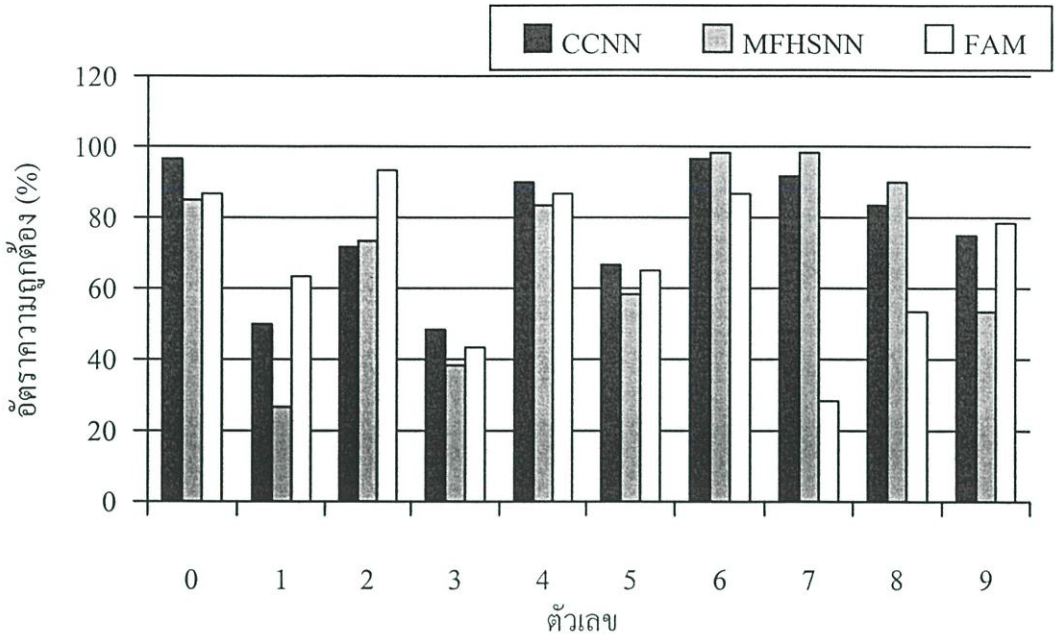
สำหรับโหมดนี้ อินพุตแพทเทิร์นที่ใช้สำหรับฝึก จะมาจากเสียงพูดของทุกตัวเลข ตัวเลขละ 10 ครั้งของผู้พูดที่เลือกมาอย่างสุ่มจำนวน 6 คน รวม 600 เสียง จะใช้ในการฝึก ส่วนเสียงจาก 6 คนที่เหลือจะใช้ในการทดสอบ ผลของการทดสอบแสดงดังตารางที่ 5.9 และ 5.10 และรูปที่ 5.18

ตารางที่ 5.9 อัตราความถูกต้องในการแบ่งกลุ่มรูปแบบของชุดข้อมูลเสียงพูดตัวเลขภาษาไทยในโหมดที่ไม่ขึ้นกับผู้พูด

อัลกอริทึม	จำนวน Reference pattern ที่สร้าง	อัตราความถูกต้อง (%)
CCNN	158	77.00
MFHSNN	583	70.50
FAM	260	68.50

ตารางที่ 5.10 ค่าพารามิเตอร์ที่ตั้งให้กับอัลกอริทึมในโหมดที่ไม่ขึ้นกับผู้พูด

FAM			MCCNN	CCNN
$\overline{\rho}_a$	β_a	จำนวนรอบในการเรียนรู้	λ	λ
0.8	0.3	10	1	0.99



รูปที่ 5.18 อัตราความถูกต้องสำหรับแต่ละตัวเลขในโหมดที่ไม่ขึ้นกับผู้พูด

จากผลการทดลอง การแบ่งกลุ่มรูปแบบของชุดข้อมูลเสียงพูดตัวเลขภาษาไทยของทั้งสามอัลกอริทึม จะเห็นได้ว่า CCNN ให้อัตราความถูกต้องสูงสุด และมีจำนวน reference pattern ที่สร้างค่อนข้างน้อยเมื่อเทียบกับอีก 2 อัลกอริทึม ทั้งนี้อาจจะเนื่องจากฟังก์ชันครอสคอร์รีเลชันที่ใช้ใน CCNN ซึ่งเป็นฟังก์ชันที่วัดความคล้ายกันโดยเน้นจากรูปร่าง (หรือทิศทาง) มากกว่าขนาดของสัญญาณ ค่อนข้างเหมาะกับคุณสมบัติของสัญญาณเสียง

บทที่ 6
สรุปผลการทดลองและข้อเสนอแนะ

วิทยานิพนธ์ฉบับนี้ได้นำเสนอโครงข่ายประสาทเทียมชนิดใหม่ ที่อาศัยทฤษฎีของ Fuzzy Set มาประยุกต์สร้างฟังก์ชันในการเลือก (choice function) ซึ่งอยู่บนพื้นฐานของการวัดความคล้ายกันระหว่างอินพุตแพทเทิร์นกับ reference pattern โดย Cross-correlation Neural Network (CCNN) นั้น ได้ใช้ฟังก์ชันครอสคอร์รีเลชันในการวัดความคล้ายกันดังกล่าว สำหรับการทดลอง ได้ใช้ชุดข้อมูลจำนวนหลายชุดทั้งที่เป็นข้อมูลมาตรฐานและข้อมูลที่จัดทำขึ้นเองมาทดสอบอัลกอริทึม และเปรียบเทียบประสิทธิภาพกับโครงข่ายอื่นอีก 2 ชนิด คือ Fuzzy ARTMAP (FAM) และ Modified Fuzzy Hypersphere Neural Network (MFHSNN)

จากผลการทดลอง สามารถสรุปถึง ข้อดี ข้อเสีย รวมถึงข้อเสนอแนะ ได้ดังนี้

1. **ความเหมาะสมกับประเภทของชุดข้อมูล**

จากลักษณะเฉพาะของฟังก์ชันครอสคอร์รีเลชัน ซึ่งเป็นฟังก์ชันที่บอกถึงความคล้ายกันระหว่างแพทเทิร์นโดยอาศัยเฉพาะทิศทางของแพทเทิร์นเท่านั้น กล่าวคือ ถ้ากำหนดแพทเทิร์น x และ y ใดๆ โดย $x = [x_1, x_2, \ldots, x_n]$ และ $y = ax$ เมื่อ a เป็นค่าคงที่ใดๆแล้ว จะได้ว่า ค่าฟังก์ชันครอสคอร์รีเลชัน (ตามสมการที่ 4.9) ระหว่าง x และ y จะให้ค่าสูงสุด คือ มีค่าเท่ากับ 1

ด้วยคุณสมบัตินี้ ทำให้ CCNN เหมาะกับชุดข้อมูลที่ให้ความสำคัญกับรูปร่างมากกว่าขนาดของแพทเทิร์น เช่น ชุดข้อมูลเสียง หรือชุดข้อมูลภาพที่ใช้ความเข้มแสงเป็นคุณลักษณะเป็นต้น แต่จะไม่เหมาะกับชุดข้อมูลที่แต่ละคุณลักษณะในแพทเทิร์นมีความหมายที่แน่นอนซึ่งแต่ละคุณลักษณะเป็นอิสระต่อกัน เช่น ชุดข้อมูลที่มีคุณลักษณะประกอบด้วย ความกว้าง ความยาว น้ำหนัก ค่าเฉลี่ยสี เป็นต้น

อย่างไรก็ตาม เพื่อทำให้ฟังก์ชันครอสคอร์รีเลชัน ให้ความสำคัญกับขนาดของแพทเทิร์นด้วย สามารถทำได้โดยการเพิ่มส่วนที่เป็น Complement Coding ให้กับแพทเทิร์นในลักษณะเดียวกันกับที่ใช้ใน FAM นั่นคือ ถ้ากำหนด $x = [x_1, x_2, \ldots, x_n]$ เป็นแพทเทิร์นใดๆ โดยที่ $0 \le x_1, x_2, \ldots, x_n \le 1$ แล้ว จะได้ว่า

$$x_c = [x_1, x_2, \ldots, x_n, 1-x_1, 1-x_2, \ldots, 1-x_n]$$

เมื่อ x_c คือแพทเทิร์นในแบบ Complement Coding ของแพทเทิร์น x

ข้อเสียของการใช้แพทเทิร์นในแบบนี้ คือ ขนาดของแพทเทิร์นจะเพิ่มขึ้นเป็น 2 เท่า ซึ่งมีผลเสียกับเวลาและทรัพยากรที่ใช้ แต่ก็มีส่วนช่วยให้อัตราความถูกต้องในการแบ่งกลุ่มของ CCNN สูงขึ้นในหลายชุดข้อมูล

2. **ลักษณะในการเรียนรู้**

ลักษณะในการเรียนรู้ของ MFHSNN จะเป็นการเรียนรู้แบบเป็นลำดับ กล่าวคือ การเรียนรู้จะกระทำกับแพทเทิร์นที่อยู่ในเทรนนิงเซตไปทีละตัวตามลำดับ โดยจะพิจารณาว่าอินพุตแพทเทิร์นที่นำเข้ามาในโครงข่ายมีความคล้ายกันกับ reference pattern ที่มีอยู่ในโครงข่ายมากพอหรือไม่ ถ้าคล้ายกันเพียงพอ ก็ให้ข้ามไปฝึกแพทเทิร์นถัดไป โดยไม่ได้คำนึงถึงข้อมูลที่มีอยู่ในแพทเทิร์นนั้น การเรียนรู้ในลักษณะนี้จะมีข้อเสียหลักๆอยู่ 3 ข้อ คือ

- ลำดับของอินพุตแพทเทิร์นในเทรนนิงเซตจะมีผลกระทบต่อลักษณะและค่าพารา-มิเตอร์ต่างๆของโครงข่ายหลังจากเสร็จสิ้นกระบวนการฝึกแล้ว
- ไม่ได้มีการพิจารณาถึงความเหมาะสมในการเลือกอินพุตแพทเทิร์นที่จะนำมาใช้เป็น reference pattern ของโครงข่าย
- จำนวนของ reference pattern ที่สร้างจะค่อนข้างสูง

สำหรับอัลกอริทึมในการเรียนรู้ของ CCNN ที่ได้นำเสนอไป จะสามารถแก้ข้อเสียดังกล่าวได้

3. **ค่าพารามิเตอร์ ε**

ε เป็นค่าเล็กๆ ที่กำหนดขึ้นเพื่อป้องกันการทับกันระหว่าง reference pattern ที่อยู่ต่างคลาสกัน ซึ่งโดยส่วนใหญ่จะถูกตั้งไว้ที่ 0.0001 จากการทดลองพบว่าการเปลี่ยนค่า ε นี้ จะมีผลต่อประสิทธิภาพของ CCNN ในระดับหนึ่ง วิทยานิพนธ์นี้ไม่ได้ศึกษาถึงผลกระทบที่แน่นอนและวิธีการในการกำหนดค่า ε ที่เหมาะสม อย่างไรก็ตาม การทำให้ ε เป็นพารามิเตอร์ที่สามารถปรับปรุงได้ (adaptive) โดยกำหนดให้ ε เป็นตัวแปรที่ขึ้นกับค่าอื่นๆ เช่น ค่า maxC หรือค่าจำนวนของอินพุตแพทเทิร์นที่เป็นสมาชิกของ reference pattern นั้น เป็นต้น หรือด้วยวิธีการอื่นๆ ที่สามารถกำหนดค่า ε ได้อย่างถูกต้องและเหมาะสม ก็อาจจะมีผลทำให้ประสิทธิภาพในการแบ่งกลุ่มของ CCNN เพิ่มขึ้นได้

เอกสารอ้างอิง

[1] Kulkarni, U.V. Sontakke, T.R. and Randale, G.D. "Fuzzy Hyperline Segment Neural Network for Rotation Invariant Handwritten Character Recognition." IJCNN '01. Proc. of International Joint Conference on Neural Networks. vol. 4, July 2001. pp. 2918-2923.

[2] Simpson, P.K. "**Fuzzy Min-Max Neural Networks-Part I: Classification.**" IEEE Trans. On Neural Networks. vol. 3, issue 5, Sept. 1992. pp. 776-786.

[3] Kulkarni, U.V. Doye, D.D. and Sontakke, T.R. "General Fuzzy Hypersphere Neural Network." In proceedings of the 2002 International Joint Conference in Neural Networks. vol. 3, May 2002. pp. 2369-2374.

[4] Duda, R.O. and Hart, P.E. **Pattern Classification and Scene Analysis.** New York : Wiley. 1993.

[5] Kulkarni, U.V. Doye, D.D. and Sontakke, T.R. "Speech Recognition Using Modified Fuzzy Hypersphere Neural Network." IJCNN '02. Proc. of the 2002 International Joint Conference on Neural Networks. vol. 1, May 2002. pp. 65-68.

[6] Sergios Theodoridis. Konstantinos Koutroumbas. **Pattern Recognition.** San Diego : Academic Press. 1998.

[7] Robert J. Schalkoff. **Pattern Recognition : Statistical, Structural and Neural Approaches.** New York : John Wiley & Sons, Inc. 1992.

[8] Douglas F. Elliott. **Handbook of Digital Signal Processing : Engineering Applications.** San Diego : Academic Press, Inc. 1987.

[9] Rabiner, L. R. and Juang, B. H. **Fundamentals of Speech Recognition.** New Jersey : Prentice Hall. 1993.

[10] Elvira, J. M. and Carrasco, R. A. "Neural network architectures for speech recognition." IEE Colloquium on Telecommunications, Consumer and Industrial Applications of Speech Technology. May 1992. pp. 4/1-4/5.

[10] Arriola, Y. and Carrasco, R. A. "Integration of multi-layer perceptron and Markov models for automatic speech recognition." IEE Conference. March 1990. pp. 413-420.

[11] El-Ramly, S. H. Abdel-Kader, N. S. and El-Adawi, R. "Neural networks used for speech recognition." Proceedings of the Nineteenth National Radio Science Conference. March 2002. pp. 200-207.

[12] บัณฑิต โรจน์อารยานนท์. **หลักการไฟฟ้าสื่อสาร**. สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย. 2541.

[13] ทัศเวท วีระวัฒน์. การรู้จำเสียงคำไทยเฉพาะบุคคล. วิทยานิพนธ์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. พ.ศ. 2541.

[14] UCI Machine Learning Repository Content Summary, Retrieved March 3, 2003, from ftp://ftp.ics.uci.edu/pub/machine-learning-databases/statlog/satimage/

[15] Statlog Project Datasets, Retrieved March 28, 2003, from http://www.liacc.up.pt/ML/statlog/datasets/segment/segment.doc.html

[16] UCI Dataset, Retrieved March 20, 2003, from http://kdd.ics.uci.edu/databases/-synthetic_control/synthetic_control.data.html

[17] The NIPS*2001 Unlabeled Data Competition and Workshop, Retrieved March 14, 2003, from http://q.cis.uoguelph.ca/~skremer/Unlabeled2/

[18] UCI Machine Learning Repository Content Summary, Retrieved February 19, 2003, from ftp://ftp.ics.uci.edu/pub/machine-learning-databases/iris/

[19] UCI Machine Learning Repository, Retrieved May 25, 2003, from ftp://ftp.ics.uci.edu/pub/machine-learning-databases/isolet/

[20] UCI Machine Learning Repository, Retrieved February 25, 2003, from ftp://ftp.ics.uci.edu/pub/machine-learning-databases/letter-recognition/

[21] Repository of Machine learning datasets on speech, Retrieved December 2, 2002, from http://www.laps.ufpa.br/aldebaro/repository/

ภาคผนวก

SPEECH RECOGNITION OF THAI DIGITS USING MODIFIED CROSS-CORRELATION NEURAL NETWORK

Arit Thammano* and Narodom Klomiam**
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang,
Bangkok, 10520 Thailand
E-mail: arit@it.kmitl.ac.th* and S4067048@kmitl.ac.th**

Abstract

In this paper, Modified Cross-Correlation Neural Network (MCCNN), which is an extension of Cross-Correlation Neural Network (CCNN) [1], is proposed. Unlike the CCNN, which utilizes the normalized cross-correlation at zero lag as a choice function to determine the winning cluster node, MCCNN uses the maximum of the normalized cross-correlation instead. In this work, spoken Thai digits (0-9) are used as the experimental data. The performance of MCCNN, CCNN and other two well-known algorithms, Back-propagation and Fuzzy ARTMAP, are compared. The results show that MCCNN has the best performance with respect to the recognition rate.

Key Words

Neural Network, Cross-correlation, Classification, Data Mining, Speech Recognition

1. Introduction

Speech recognition has its long history. Many automatic systems have been embedded with speech recognition devices because talking is the easiest way for a person to communicate with a machine. The goal of speech recognition researches is to create a machine that can receive spoken information and act appropriately upon receiving the information. Several approaches have been proposed to solve the problem. Neural Networks (NNs), which are parallel distributed processing models, are one of the most popular approaches in the area of speech recognition or language processing. Elvira and Carrasco [2] investigated and compared the performance of six different NN architectures -- Adaline, Monolayer perceptron, Back-propagation with the sigmoid function, Back-propagation with the hyperbolic tangent function, Radial basis function, and Volterra connectionist model -- in the speech recognition of Spanish words. Two different data sets were used in their experiment: the Spanish vowels (/a/, /e/, /i/, /o/, /u/) and the Spanish digits (/uno/, /dos/, /tres/, /cuatro/, /cinco/, /seis/, /siete/, /ocho/, /nueve/, /cero/). The reported recognition rates varied from 62.2 – 87.5% for the vowel recognition and from 56.24 – 69.21% for the digit recognition. The best recognition rates of both data were obtained from the Back-propagation with the hyperbolic tangent function. Arriola and Carrasco [3] have presented the implementation of a speech recognition system based on the integration of the Multi-layer Perceptron and the Hidden Markov model on 10 Spanish digits spoken by seven male speakers. The best recognition rate was found to be 88%. El-Ramly et al. [4] applied Time Delay neural network (TDNN) to the recognition of two Arabic phoneme categories, nasals and voiced stops. TDNN was chosen for the problem because of its ability to represent relationships between acoustic events. The maximum recognition rate obtained for nasals was 91% and for voiced stops was 82%. Several researchers -- Kulkarni et al., Doye et al., and Simpson [5, 6, 7, 8] -- have focused on another type of neural network known as the fuzzy neural network, which utilizes fuzzy sets as pattern classes. Doye et al. [6] proposed the modified fuzzy hypersphere neural network (MFHSNN), which is an extension of the fuzzy hypersphere neural network (FHSNN) [9]. The experiment compared this modified version of fuzzy hypersphere neural network with its original version for the speech recognition of Marathi (language spoken in the state of Maharashtra, India) digits and found that the former gives better performance with average recognition rate in speaker dependent mode at 88.7% and 61.6% in speaker independent mode.

Modified Cross-Correlation Neural Network (MCCNN) described in this paper modifies the choice function used in the original CCNN. The choice function measures the likeness or similarity between the incoming input pattern and the reference patterns of each class to determine the winning cluster node. Unlike the CCNN, which utilizes the normalized cross-correlation at zero lag as a choice function, MCCNN uses the maximum of the normalized cross-correlation at any shifted period instead. This modification to the choice function can alleviate the problem of slight misalignments (shifting) of the input pattern that might occur to the speech signals during the preprocessing of speech signals.

Following this introduction, section 2 describes the architecture of the proposed model and its learning algorithm. The preprocessing and the feature extraction of speech signals are described in section 3. In section 4, the experimental results are demonstrated and discussed. Finally, section 5 is a conclusion.

2. Architecture of MCCNN

The architecture of the proposed model is a three-layer feedforward neural network as shown in Figure 1. The first layer is the input layer, which consists of N nodes. Each node represents a feature component of the input data. The second layer is the cluster layer. The nodes in this second layer are constructed during the training phase and each node represents a reference pattern. The third layer is the output layer. Each node in the output layer represents a class. During supervised learning, the input vector is presented to the model, together with its respective target output vector. The input vector is denoted by $I_h = (I_{h1}, \ldots, I_{hN})$, where h is the h^{th} input pattern, and N is the number of features in I. Each node in the second layer is fully connected to the nodes in the input layer via the connections w_{ji}. The weight vector w_j of dimension N represents the reference pattern of the j^{th} node in the cluster layer. Once the model receives the input and its associated target output (I_h, O_h), the maximum of the normalized cross-correlation between the input vector I_h and each weight vector w_j is computed and the outputs of the j^{th} node, $T_j(I_h)$, in the cluster layer are then determined.

$$T_j(I_h) = \begin{cases} 1, & \text{if } f(w_j, I_h) \geq mC_j \\ f(w_j, I_h), & \text{if } f(w_j, I_h) < mC_j \end{cases} \quad (1)$$

$$f(x, y) = \max_p \left\{ \frac{\sum_i x_i y_{i-p}}{\sqrt{\left(\sum_i x_i^2\right)\left(\sum_i y_i^2\right)}} \right\} \quad (2)$$

where $i = 1, \ldots, N$

$j = 1, \ldots, M$

N is the number of features in input vector I.

M is the number of nodes in cluster layer.

f(x, y) is the maximum of the normalized cross-correlation at any shifted period p.

p is the lag variable, which is between -(N-1) to (N-1).

mC_j is the minimum normalized cross-correlation value between 0 and 1 initially predefined to be equal to λ. However, the value of mC_j might be increased during the training phase. The network uses mC_j to prevent the overlap between cluster nodes of different classes. However, the overlap of cluster nodes from the same class is allowed.

Next, the system finds all cluster nodes which belong to the class defined by the target output vector, and have the output value equal to 1, $T_j(I_h) = 1$. If there is at least one node meeting the above criterion, the remaining steps in the training process are skipped and the training is continued with the next input pattern. However, if such node does not exist, a new cluster node is recruited to code the input pattern; the connection between a new cluster node and the target output is created. The connections between the cluster layer and the output layer, w_{kj}, are either 1 or 0. It is 1 when the j^{th} node belongs to class k, and 0 otherwise.

$$w_J = I_h \quad (3)$$

$$w_{kJ} = \begin{cases} 1, & \text{if } k = K \\ 0, & \text{if } k \neq K \end{cases} \quad (4)$$

$$\max C = \max_{g \notin K} \{f(w_J, I_g)\} \quad (5)$$

$$mC_J = \begin{cases} \lambda, & \text{if } \max C < \lambda \\ \max C + \varepsilon, & \text{if } \lambda \leq \max C < 1 \\ 1, & \text{otherwise} \end{cases} \quad (6)$$

where J is the newly recruited cluster node.

K is the class to which J node belongs.

ϵ is a very small number close to zero.

maxC is the maximum of the normalized cross-correlation between the reference pattern of node J and the rest of the input patterns which do not belong to class K.

During testing, each test vector is applied in turn and its class is predicted. The class whose cluster node returns the maximum output value is the result of the prediction.

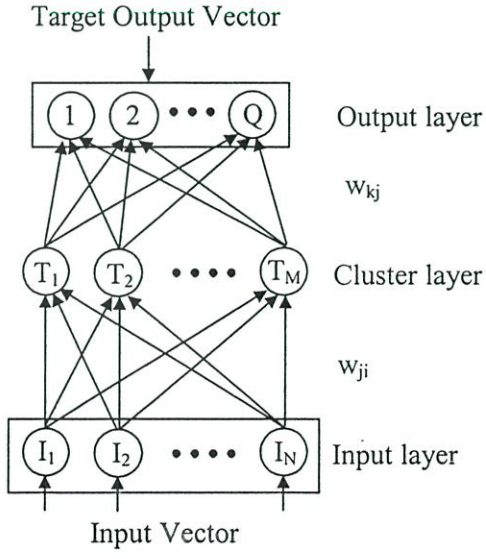


Figure 1 Architecture of MCCNN

3. Preprocessing

A list of ten Thai digits (0: soon, 1: neung, 2: sorng, 3: sahm, 4: see, 5: haa, 6: hok, 7: jet, 8: bpairt, 9: gao) was presented to 12 speakers. Each speaker was asked to pronounce the list ten times; each time is referred later in this paper as a sample. Therefore, the total number of recorded words was 1200 (120 samples). The sampling frequency used in the recording processes was 11025 Hz. The speech signals have then been filtered with band-pass filter, having cut-off frequencies of 100 and 5000 Hz. End-point detection is obtained by calculating the energy of each 100 sampled frame; the frames whose energy is less than 10% of the maximum energy will be truncated at both sides. Pre-emphasis with first order system having transfer function $H(z) = 1 - az^{-1}$, where $a = 0.95$, have been used. For windowing, Hamming window with 700 samples has been used with 30% overlap. The 22^{th} order Linear Predictive Coefficients (LPC) [9] are extracted from each frame. Then the extracted 22 features of each frame are concatenated to produce an input pattern. By using the above criteria, the longest-pronounced word yields 32 frames; therefore, the number of features in input vector I are 704. Zero padding is performed on the shorter-pronounced words to make all patterns the same size. For MCCNN and CCNN, the normalization of features is not necessary due to the ability of the cross-correlation function that can deal with arbitrary vectors of real numbers and with any magnitude. However, the features are normalized in the range of zero to one in case of Back-propagation and Fuzzy ARTMAP.

4. Experimental Results

The experiments are performed in 2 modes: speaker dependent and speaker independent. In speaker dependent mode, five samples of each speaker are used for training and the remaining five samples are used for testing. The five samples used for training are sample 1, 3, 5, 7, and 9. Since each sample consists of 10 pronounced words (0-9), the total of 600 input patterns are used to train the network and the remaining 600 input patterns are used as the testing data. The comparative performance of MCCNN, CCNN, Fuzzy ARTMAP (FAM) and Back-propagation (BPN) to the speech recognition of Thai digits is shown in Table 1 and Figure 2-4. In this work, the best performances of MCCNN and CCNN are obtained when the parameter λ of both models is equal to 0.5. For FAM, the best recognition rate of 86.33% is achieved when the vigilance parameter (ρ_a) is set at 0.97; the learning rate (β) is 0.7 and twenty epochs of training are used. For BPN, a network of single 400-node hidden layer and 10-node output layer (one output node for each class) is used, while learning rate (η) and number of epochs used in this training are 0.5 and 5000 respectively.

Table 1 The Comparative Performance of MCCNN, CCNN, Fuzzy ARTMAP and Back-propagation to the Speech Recognition of Thai Digits in Speaker Dependent Mode

Algorithm	Average Number of Reference Pattern per Speaker	Recognition Rate (%)
MCCNN	32.9	96.17
CCNN	37	92.83
FAM	47.5	86.33
BPN	-	76.50

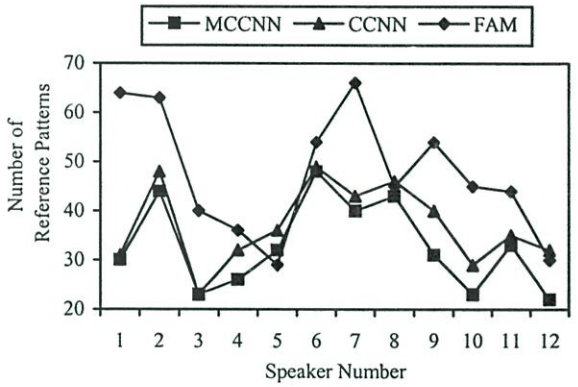


Figure 2 The Comparative Performance of MCCNN, CCNN and Fuzzy ARTMAP with Respect to Number of Created Reference Patterns in Speaker Dependent Mode

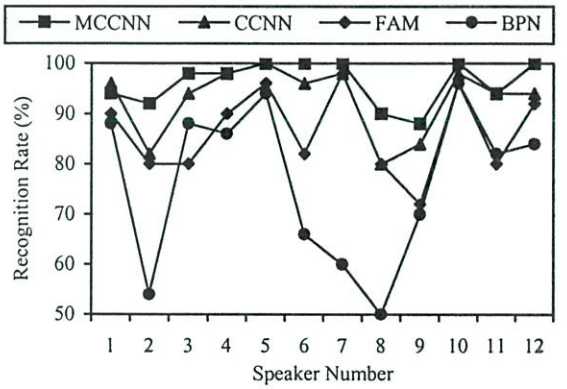


Figure 3 The Comparative Performance of MCCNN, CCNN, Fuzzy ARTMAP and Back-propagation with Respect to Recognition Rate in Speaker Dependent Mode.

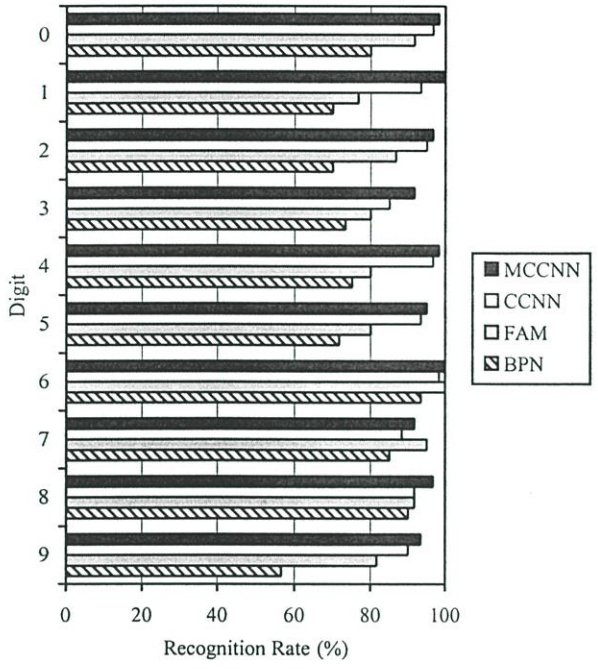


Figure 4 The Recognition Rate of Each Thai Digit in Speaker Dependent Mode

In speaker independent mode, all samples from randomly chosen 6 speakers are used for training and from remaining 6 speakers are used for testing. The comparative performance of MCCNN, CCNN, Fuzzy ARTMAP and Back-propagation is shown in Table 2 and Figure 5. The recognition rates stated in Table 2 are obtained from the MCCNN and CCNN using $\lambda = 0.4$; from the Fuzzy ARTMAP using $\rho_a = 0.97$, $\beta = 0.4$, and epoch = 20; and from Back-propagation with one hidden layer of 400 neurons using $\eta = 0.3$ and epoch = 5000.

5. Conclusion

A modification of CCNN is proposed and its performances are compared with the original CCNN and other two well-known algorithms, Back-propagation and Fuzzy ARTMAP. In comparison to CCNN, MCCNN has to deal with the shifted versions of input signal; the training and recall time spent by the algorithm are therefore larger. However, its recognition rate compared with CCNN is found to be superior. The maximum recognition rate obtained in speaker dependent mode is 96.17%, which is about 3.3% higher. In speaker independent mode, the result is about 4.2% better. In addition, the number of reference patterns of MCCNN is also less.

Moreover, the experimental results also show that MCCNN outperforms both Back-propagation and Fuzzy ARTMAP by 19.67% and 9.84% in speaker dependent mode, and by 18% and 7.67% in speaker independent mode.

Table 2 The Comparative Performance of MCCNN, CCNN, Fuzzy ARTMAP and Back-propagation to the Speech Recognition of Thai Digits in Speaker Independent Mode

Algorithm	Number of Reference Pattern	Recognition Rate (%)
MCCNN	249	77.67
CCNN	318	73.50
FAM	374	70
BPN	-	59.67

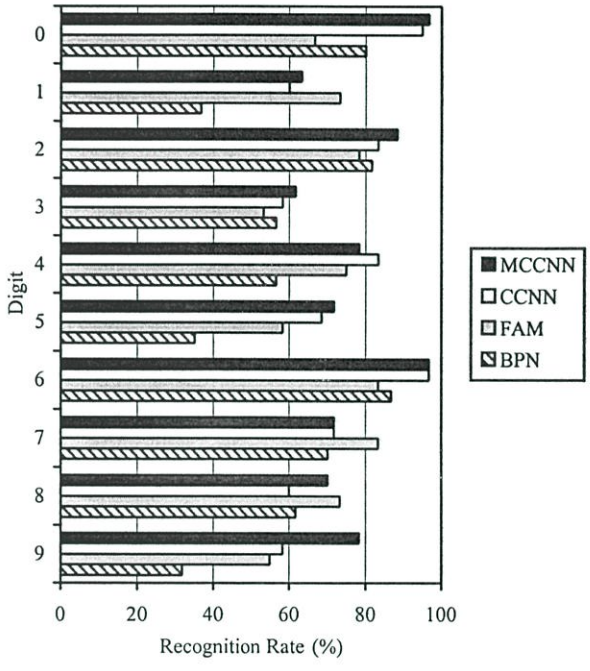


Figure 5 The Recognition Rate of Each Thai Digit in Speaker Independent Mode

References

[1] A. Thammano and N. Klomiam, Cross-correlation neural network: A new neural network classifier, *WSEAS Transactions on Systems*, issue 1, vol. 3, January 2004, 194-199.

[2] J. M. Elvira and R. A. Carrasco, Neural network architectures for speech recognition, *IEE Colloquium on Telecommunications, Consumer and Industrial Applications of Speech Technology*, May 1992, 4/1-4/5.

[3] Y. Arriola and R. A. Carrasco, Integration of multi-layer perceptron and Markov models for automatic speech recognition, *IEE Conference*, March 1990, 413-420.

[4] S. H. El-Ramly, N. S. Abdel-Kader and R. El-Adawi, Neural networks used for speech recognition, *Proceedings of the Nineteenth National Radio Science Conference*, March 2002, 200-207.

[5] U. V. Kulkarni, T. R. Sontakke and G. D. Randale, Fuzzy Hyperline Segment Neural Network for rotation invariant handwritten character recognition, *Proceedings of the International Joint Conference on Neural Networks*, vol. 4, July 2001, 2918-2923.

[6] D. D. Doye, U. V. Kulkarni and T. R. Sontakke, Speech recognition using Modified Fuzzy Hypersphere Neural Network, *Proceedings of the 2002 International Joint Conference on Neural Networks*, vol. 1, May 2002, 65-68.

[7] U. V. Kulkarni, D. D. Doye and T. R. Sontakke, General Fuzzy Hypersphere Neural Network, *Proceedings of the 2002 International Joint Conference on Neural Networks*, vol. 3, May 2002, 2369-2374.

[8] P. K. Simpson, Fuzzy Min-Max Neural Networks-Part I: Classification, *IEEE Transactions on Neural Networks*, vol. 3, issue 5, September 1992, 776-786

[9] U. V. Kulkarni and T. R. Sontakke, Fuzzy Hypersphere Neural Network Classifier, *Proceedings of the 10th IEEE Conference on Fuzzy Systems*, December 2001.

[10] L. Rabiner and B. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[11] D. F. Elliott, *Handbook of Digital Signal Processing: Engineering Applications*, Academic Press, San Diego, 1987.

[12] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1993.

[13] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, San Diego, 1999.

[14] R. J. Schalkoff, *Pattern Recognition: Statistical, Structural and Neural Approaches*, John Wiley & Sons, New York, 1992.

ประวัติผู้เขียน

นาย นโรดม กล่อมเอี่ยม เกิดเมื่อวันที่ 25 ตุลาคม พ.ศ. 2520 และจบการศึกษาระดับปริญญาตรีจาก คณะวิศวกรรมศาสตร์ สาขาวิศวกรรมไฟฟ้า วิชาเอก วิศวกรรมไฟฟ้าสื่อสาร จุฬาลงกรณ์มหาวิทยาลัย