การจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริดที่มีความละเอียดหลายระดับ

OVERLAPPING DATA CLUSTERING USING MULTI-RESOLUTION GRIDS

กิตติคุณ นันทา

KITTIKUN NANTA

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2554

KMITL-2011-EN-M-070-036

การจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริดที่มีความละเอียดหลายระดับ

OVERLAPPING DATA CLUSTERING USING MULTI-RESOLUTION GRIDS

กิตติกุณ นันตา

KITTIKUN NANTA

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2554

KMITL-2011-EN-M-070-086

OVERLAPPING DATA CLUSTERING USING MULTI-RESOLUTION GRIDS

KITTIKUN NANTA

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN COMPUTER ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2011

KMITL-2011-EN-M-070-086

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ การจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริดที่มีความละเอียดหลายระดับ

Thesis Title OVERLAPPING DATA CLUSTERING USING MULTI-RESOLUTION GRIDS

นักศึกษา นายกิตติคุณ นันตา

รหัสประจำตัว 49060718

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชา วิศวกรรมคอมพิวเตอร์

อาจารย์ที่ปรึกษาวิทยานิพนธ์ รศ.ดร.บุญธีร์ เครือตราชู

หมายเลขวิทยานิพนธ์ KMITL-2011-EN-M-070-086

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
ผศ.ดร.สมศักดิ์	วลัยรัชต์	
รศ.ดร.เกียรติกูล	เจียรนัยธนะกิจ	
ผศ.ดร.ทรงฤทธิ์	มณีวงศ์วัฒนา	
ดร.ปกรณ์	วัฒนจตุรพร	
รศ.ดร.บุญธีร์	เครือตราชู	

วัน / เดือน/ ปี ที่สอบ วันจันทร์ที่ 23 พฤษภาคม พ.ศ. 2554 เวลา 16.00-18.00 น.

สถานที่สอบ ณ อาคาร A ชั้น 5 ห้องประชุม 1

คณะวิศวกรรมศาสตร์ รับรองแล้ว

(รองศาสตราจารย์ ดร.สุชัชวีร์ สุวรรณสวัสดิ์)

คณบดี คณะวิศวกรรมศาสตร์

วันที่ 23 พฤษภาคม พ.ศ. 2554

หัวข้อวิทยานิพนธ์	การจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริดที่มีความละเอียดหลายระดับ
นักศึกษา	นาย กิตติคุณ นันตา
รหัสประจำตัว	49060718
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
พ.ศ.	2554
อาจารย์ที่ปรึกษาวิทยานิพนธ์	รศ.ดร.บุญธีร์ เครือตราชู

บทคัดย่อ

ในปัญหาทางด้านการรู้จำรูปแบบมักจะมีการสมมุติว่าข้อมูลที่อยู่คนละคลาสสามารถแบ่งแยกกันได้อย่างชัดเจน แต่ในความเป็นจริงเรากลับพบว่าข้อมูลมักจะมีการซ้อนทับกันเกิดขึ้น วิทยานิพนธ์นี้ได้นำเสนอการจัดกลุ่มข้อมูลแบบมีผู้สอนสำหรับข้อมูลที่มีการซ้อนทับกัน 2 วิธี วิธีแรกเป็นการจัดกลุ่มข้อมูลโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลม โดยขั้นตอนแรกสเปซจะถูกแบ่งออกเป็นกลุ่มข้อมูลหน่วยก่อนแล้วหลังจากนั้นกลุ่มข้อมูลหน่วยที่อยู่ใกล้กันและมีอัตราส่วนคลาสคล้ายกันจะถูกรวมเข้าด้วยกัน วิธีการนี้จะสร้างกลุ่มข้อมูลหน่วยขึ้นมาน้อยกว่าการใช้กริดขนาดเท่ากันหมดแต่กลับใช้เวลาในการประมวลผลมากกว่า ส่วนวิธีที่สองจะใช้กริดที่มีความละเอียดหลายระดับในการแบ่งสเปซ วิธีการนี้สร้างกริดขึ้นมาน้อยกว่าการใช้กริดขนาดเท่ากันหมดและเวลาใช้ในการประมวลผลน้อยกว่าด้วย

Thesis title	Overlapping Data Clustering using Multi-resolution Grids
Student	Mr. Kittikun Nanta
Student ID	49060718
Degree	Master of Engineering
Program	Computer Engineering
Year	2011
Thesis advisor	Assoc. Prof. Dr. Boontee Kruatrachue

ABSTRACT

Pattern recognition problems frequently assumed that data belonging to different class can be well separated. In fact, data can be overlap. This thesis proposes two supervised clustering methods for overlapping data. The first method uses circular unit clusters to divide space, and then merges connected unit clusters with similar class ratio to form clusters. The results showed that the number of unit cluster created is less than of the equal-sized grid method, but the processing time is worse. The other method uses multi-resolution grids. The results showed that the number of grid created and the processing time are less than of the equal-sized grid method.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยความกรุณาและความช่วยเหลืออย่างดียิ่งจากอาจารย์ที่ปรึกษา รศ.ดร.บุญธีร์ เครือตราชู ข้าพเจ้าขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบคุณ พี่ธนวัฒน์ ภัทรวรเมธ ผู้แต่งวิทยานิพนธ์เรื่อง วิธีการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้เทคนิคความหนาแน่น (ODBSCAN) ที่ได้ให้คำแนะนำที่มีประโยชน์ต่อการทำงานวิจัยที่นำเสนอในวิทยานิพนธ์ฉบับนี้

ขอขอบคุณ พี่พัฒนพล รัตนพงษ์พร ผู้แต่งวิทยานิพนธ์เรื่อง การจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริด ที่ได้อนุเคราะห์โคดโปรแกรมการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริด

ขอขอบคุณ พี่ณรงค์ชัย มุ่งแฝงกลาง และเพื่อนๆ พี่ๆ ภาควิชาคอมพิวเตอร์ ที่ให้ความช่วยเหลือในด้านต่างๆ

คุณค่าและประโยชน์อันพึงมีจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบให้กับ บิดา มารดา และผู้มีพระคุณทุกท่าน หากวิทยานิพนธ์ฉบับนี้มีข้อผิดพลาดประการใด ข้าพเจ้าขอน้อมรับไว้แต่เพียงผู้เดียว

กิตติคุณ นันตา

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การจัดกลุ่มข้อมูล (Clustering) เป็นวิธีการทางการคำนวณ (Computation) ที่มีความสำคัญและถูกนำไปประยุกต์ใช้ในด้านต่างๆ มากมาย เช่น การทำเหมืองข้อมูล (Data mining) การรู้จำตัวอักษร (Character recognition) เป็นต้น เดิมทีนั้นการจัดกลุ่มข้อมูลจัดได้ว่าเป็นการเรียนรู้แบบไม่มีผู้สอน (Unsupervised learning) ต่างกับการจำแนกประเภท (Classification) ที่เป็นการเรียนรู้แบบมีผู้สอน (Supervised learning) ทั้งนี้ก็เพราะว่าการแบ่งกลุ่มข้อมูลจะใช้เพียงฟีเจอร์ (Feature) ของข้อมูลเท่านั้น ไม่ได้ใช้คลาสเลเบิล (Class label) ด้วย อย่างไรก็ตามเมื่อไม่นานมานี้ได้มีการนำเสนอวิธีการจัดกลุ่มข้อมูลที่มีการนำคลาสเลเบิลมาใช้ด้วยและเรียกการจัดกลุ่มข้อมูลแบบนี้ว่าการจัดกลุ่มข้อมูลแบบมีผู้สอน (Supervised clustering) [1–3]

การจัดกลุ่มข้อมูลแบบเดิมหรือการจัดกลุ่มข้อมูลแบบไม่มีผู้สอน (Unsupervised clustering) จะทำการจัดให้จุดข้อมูลที่มีความใกล้เคียงกันให้อยู่ในกลุ่มเดียวกัน ส่วนการจัดกลุ่มข้อมูลแบบมีผู้สอนจะทำการจัดให้จุดข้อมูลที่มีความใกล้เคียงกันและมีคลาสเลเบิลเหมือนกันให้อยู่ในกลุ่มเดียวกัน ในกรณีที่จุดข้อมูลที่มีความใกล้เคียงกันมีคลาสเลเบิลเหมือนกันการจัดกลุ่มข้อมูลจะทำได้โดยง่าย แต่ในความเป็นจริงกลับไม่เป็นเช่นนั้นเสมอไป เรามักจะพบอยู่บ่อยๆ ว่าชุดข้อมูลที่นำมาจัดกลุ่มนั้นมีการซ้อนทับกันของข้อมูลเกิดขึ้น ซึ่งสาเหตุของการซ้อนทับกันของข้อมูลนั้นก็อาจมาจากการใช้ฟีเจอร์ที่ไม่เหมาะสมหรือฟีเจอร์มีอยู่จำกัด ถ้าสาเหตุของการซ้อนทับกันของคลาสมาจากการใช้ฟีเจอร์ที่ไม่เหมาะสมและเราสามารถหาฟีเจอร์ใหม่ได้การจัดกลุ่มข้อมูลชุดนั้นก็ไม่เป็นปัญหา แต่ในกรณีที่ฟีเจอร์มีอยู่จำกัดและเราไม่สามารถหาฟีเจอร์ใหม่ได้การจัดกลุ่มข้อมูลชุดนั้นก็จะเป็นปัญหาเพราะวิธีการจัดกลุ่มข้อมูลโดยทั่วไป อย่างเช่น Neural network ไม่สามารถจัดกลุ่มข้อมูลแบบนี้ได้

เพื่อแก้ปัญหาการซ้อนทับกันของคลาสที่เกิดขึ้นในการรู้จำตัวอักษรลายมือเขียนภาษาไทยที่มีฟีเจอร์จำกัดจึงได้มีการเสนอ “ต้นแบบวิธีการจัดกลุ่มข้อมูลที่มีการซ้อนทับกัน” [4] แต่จากการทดลองพบว่าผลการจัดกลุ่มข้อมูลที่ได้ยังไม่ที่น่าพอใจเพราะว่ากลุ่มข้อมูลที่ได้มีจำนวนมากและไม่สามารถบ่งบอกถึงความคลุมเครือในบริเวณที่เกิดการซ้อนทับกันของคลาสได้ ดังนั้นเพื่อแก้ปัญหาเดียวกันนี้จึงมีการเสนอวิธีการแบ่งกลุ่มข้อมูลที่ชื่อว่า ODBSCAN [5] ขึ้นมา โดย ODBSCAN นั้นมีความสามารถในการบ่งบอกถึงความคลุมเครือในบริเวณที่เกิดการซ้อนทับกันของคลาสได้ อีกทั้งยังให้จำนวนกลุ่มข้อมูลที่น้อยกว่าวิธีก่อนด้วย แต่การพัฒนาก็ไม่ได้หยุดอยู่เท่านั้น ต่อมาก็มีการนำเสนอ “การจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริด” [6] ซึ่งมีจุดประสงค์เพื่อลดเวลาที่ใช้ในการคำนวณให้น้อยลงกว่า

ODBSCAN โดยวิธีการดังกล่าวจะทำการแบ่งพื้นในฟีเจอร์สเปซ (Feature space) ออกเป็นกริด (Grid) ก่อน แล้วจึงทำการคำนวณบนกริดเหล่านั้นแทนการทำการคำนวณบนทุกจุดข้อมูลดังที่ทำใน ODBSCAN แต่อย่างไรก็ตามการแบ่งฟีเจอร์สเปซออกเป็นกริดมักจะทำให้มีกริดที่ไม่มีจุดข้อมูลอยู่เลยเกิดขึ้น โดยเฉพาะอย่างยิ่งกับชุดข้อมูลที่มีจำนวนมิติมากๆ นอกจากนั้นการใช้ กริดที่มีขนาดเท่ากันหมดกับทุกบริเวณของฟีเจอร์สเปซยังถือเป็นข้อเสียของวิธีการนี้ นั่นคือ ถ้าใช้กริดที่มีขนาดใหญ่เกินไปก็จะเกิดความผิดพลาดขึ้นมากในขณะที่ถ้าใช้กริดที่มีขนาดเล็กเกินไปก็จะทำให้ใช้เวลาในการประมวลผลมาก ดังนั้น วิทยานิพนธ์ฉบับนี้จึงขอนำเสนอวิธีการจัดกลุ่มข้อมูลสำหรับข้อมูลที่มีการซ้อนทับกันวิธีใหม่ 2 วิธี โดยหลักการของวิธีการที่นำเสนอนี้ยังคงคล้ายกับวิธีการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริด แต่จะแตกต่างกันตรงที่วิธีการแรกจะเปลี่ยนการแบ่งพื้นที่ในฟีเจอร์สเปซโดยใช้กริดไปเป็นการใช้วงกลมซึ่งจะถูกสร้างขึ้นเฉพาะในบริเวณที่มีจุดข้อมูลอยู่เท่านั้นแทน และวิธีที่สองก็จะเปลี่ยนจากการใช้กริดขนาดเท่ากันหมดไปเป็นการใช้กริดที่มีขนาดต่างกัน

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

ความมุ่งหมายและวัตถุประสงค์ของวิทยานิพนธ์ฉบับนี้ คือ การพัฒนาวิธีการจัดกลุ่มข้อมูลสำหรับข้อมูลที่มีการซ้อนทับกัน โดยที่วิธีการแบ่งกลุ่มข้อมูลนั้นต้องมีความสามารถในการบอกถึงความคลุมเครือในบริเวณที่มีการซ้อนทับกันของข้อมูลเกิดขึ้นได้อย่างเหมาะสมและมีความรวดเร็วกว่าวิธีการที่มีอยู่

1.3 สมมติฐานของการศึกษา

เนื่องจากการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริดทำการสร้างกริดขึ้นทั่วทั้งฟีเจอร์สเปซโดยไม่คำนึงถึงว่าบริเวณที่สร้างกริดขึ้นมานั้นจะมีจุดข้อมูลอยู่หรือไม่ ดังนั้นถ้าเราสร้างกริดขึ้นเฉพาะบริเวณที่มีจุดข้อมูลอยู่หรือใช้กริดที่มีขนาดต่างกันก็น่าจะลดเวลาที่ใช้ในการประมวลผลลงได้อีก

1.4 ทฤษฎีหรือแนวคิดที่ใช้ในการวิจัย

วิทยานิพนธ์นี้เสนอแนวคิดการสร้างกลุ่มข้อมูลหน่วยหรือกริดที่ใช้ในการจัดกลุ่มข้อมูลที่มีการซ้อนทับกัน โดยแนวคิดแรกจะเป็นการสร้างกลุ่มข้อมูลหน่วยเฉพาะในบริเวณที่มีข้อมูลอยู่เท่านั้น ซึ่งวิธีนี้จะมีประโยชน์ในกรณีที่ข้อมูลมีจำนวนมิติมากๆ และข้อมูลไม่ได้มีการกระจายตัวทั่วทั้งสเปซแต่มีการกระจุกตัวอยู่เป็นกลุ่ม ๆ ส่วนแนวคิดที่สองจะเป็นการใช้กลุ่มข้อมูลหน่วย (กริด) ที่มีขนาดไม่เท่ากัน เพราะว่าโดยทั่วไปแล้วลักษณะการกระจายตัวหรือการซ้อนทับกันของข้อมูลไม่ได้เหมือนกันทั่วทั้งสเปซ ดังนั้นถ้าเราใช้กลุ่มข้อมูลหน่วยที่มีขนาดใหญ่บ้างเล็กบ้างแล้วแต่ลักษณะการกระจายตัวของ

ข้อมูล ก็จะทำให้ใช้จำนวนกลุ่มข้อมูลหน่วยน้อยกว่าการใช้กลุ่มข้อมูลหน่วยที่มีขนาดเท่ากันหมด ซึ่งจะทำให้ใช้เวลาในการรวมกลุ่มข้อมูลน้อยลงด้วย

1.5 ขอบเขตของวิทยานิพนธ์

ขอบเขตของวิทยานิพนธ์นี้คือ ศึกษาวิธีการจัดกลุ่มข้อมูลที่มีการซ้อนทับกัน และนำเสนอวิธีการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลมและโดยใช้กริดที่มีความละเอียดหลายระดับ พร้อมทั้งทำการทดลองเพื่อวัดประสิทธิภาพเทียบกับการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริดขนาดเท่ากันหมด

1.6 ขั้นตอนของการศึกษา

1. ทำการศึกษาค้นคว้างานวิจัยเกี่ยวกับการจัดการกับข้อมูลที่มีการซ้อนทับกันของคลาสเกิดขึ้น เพื่อใช้เป็นแนวทางในการทำวิจัย
2. กำหนดหัวข้อ เป้าหมาย และขอบเขตของงานวิจัย พร้อมทั้งวางแนวทางดำเนินการวิจัย
3. พัฒนาวิธีการแบ่งกลุ่มข้อมูลตามแนวทางที่วางไว้
4. ทำการทดลองและเปรียบเทียบผลกับงานวิจัยอื่น
5. สรุปผลการดำเนินงานวิจัยและจัดทำวิทยานิพนธ์

1.7 รายละเอียดในแต่ละบท

วิทยานิพนธ์ฉบับนี้แบ่งออกเป็น 6 บท ดังนี้

บทที่ 1 กล่าวถึงความเป็นมาและความสำคัญของปัญหา แนวคิดที่นำเสนอเพื่อแก้ไขปัญหา วัตถุประสงค์ ขอบเขตของวิทยานิพนธ์ ขั้นตอนการศึกษา และส่วนประกอบของวิทยานิพนธ์

บทที่ 2 กล่าวถึงการจัดกลุ่มข้อมูลโดยทั่วไป การจัดกลุ่มข้อมูลที่มีการซ้อนทับกันและงานวิจัยที่เกี่ยวกับการจัดกลุ่มข้อมูลที่มีการซ้อนทับกัน

บทที่ 3 จะเป็นรายละเอียดของการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลมและการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริดที่มีความละเอียดหลายระดับ

บทที่ 4 นำเสนอการทดลองจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลม

บทที่ 5 นำเสนอการทดลองจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริดที่มีความละเอียดหลายระดับ

บทที่ 6 จะเป็นการสรุปผลการวิจัยและเสนอแนวทางในการทำวิจัยต่อ

บทที่ 2
การจัดกลุ่มข้อมูลที่มีการซ้อนทับกันและงานวิจัยที่เกี่ยวข้อง

บทนี้จะกล่าวถึงการจัดกลุ่มข้อมูลโดยทั่วไปและการจัดกลุ่มข้อมูลแบบมีผู้สอน การนำการจัดกลุ่มข้อมูลแบบมีผู้สอนมาแก้ปัญหาการซ้อนทับกันของข้อมูล พร้อมทั้งนำเสนองานวิจัยเกี่ยวกับการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันที่วิทยานิพนธ์นี้ใช้เป็นจุดเริ่มต้นในการศึกษาและทำการวิจัยต่อ

2.1 การจัดกลุ่มข้อมูล

การจัดกลุ่มข้อมูล (หรือการแบ่งกลุ่มข้อมูล) เป็นวิธีการทางการคำนวณที่ใช้สำหรับวิเคราะห์ข้อมูลซึ่งถูกนำไปใช้ในงานด้านต่างๆ มากมาย เช่น การทำเหมืองข้อมูล การเรียนรู้ของเครื่อง เป็นต้น หน้าที่ของการจัดกลุ่มข้อมูลก็คือ การแบ่งชุดข้อมูลซึ่งมักจะอยู่ในรูปเวคเตอร์ออกเป็นกลุ่ม โดยจะนำเอาข้อมูลที่มีคุณลักษณะเหมือนกันหรือคล้ายกันมาจัดให้อยู่ในกลุ่มเดียวกัน อัลกอริธึมสำหรับการจัดกลุ่มข้อมูลจะทำการจัดกลุ่มข้อมูลโดยอาศัยความเหมือน (Similarity) หรือความใกล้ชิด (Proximity) ซึ่งหาได้จากการวัดระยะระหว่างเวคเตอร์ของข้อมูล โดยวิธีการวัดระยะที่นิยมใช้กันก็ได้แก่ การวัดระยะแบบยูคลิด (Euclidean distance) การวัดระยะแบบแมนฮัตตัน (Manhattan distance) เป็นต้น การจัดกลุ่มข้อมูลแตกต่างจากการแบ่งประเภทข้อมูลตรงที่การจัดกลุ่มข้อมูลจะทำการจัดกลุ่มข้อมูลโดยไม่มีการกำหนดประเภทของข้อมูลไว้ก่อน ดังนั้นการจัดกลุ่มข้อมูลจึงจัดเป็นการเรียนรู้แบบไม่มีผู้สอน

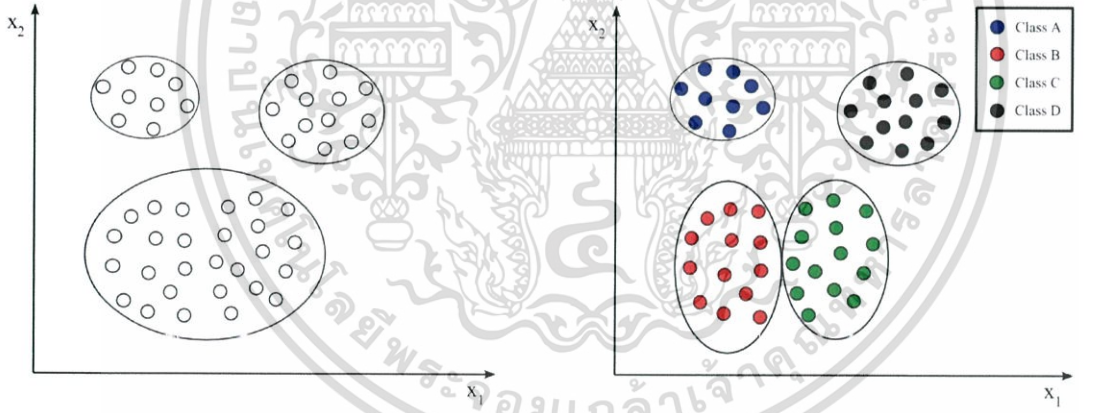
ถ้าแบ่งประเภทการแบ่งกลุ่มข้อมูลแบบไม่มีผู้สอนออกตามการกำหนดให้จุดข้อมูลเป็นสมาชิกของกลุ่มข้อมูลแล้วจะสามารถแบ่งได้เป็น 3 ประเภท คือ แบบเอ็กซคลูซีฟ (Exclusive clustering) แบบซ้อนทับกัน (Overlapping clustering) และแบบฟัซซี (Fuzzy clustering) โดย**การแบ่งกลุ่มข้อมูลแบบเอ็กซคลูซีฟ**นั้นจุดข้อมูลใดๆ จะถูกกำหนดให้เป็นสมาชิกของกลุ่มข้อมูลกลุ่มใดกลุ่มหนึ่งเพียงกลุ่มเดียวเท่านั้น ซึ่งตัวอย่างของการแบ่งกลุ่มข้อมูลแบบนี้ได้แก่ K-means และ DBSCAN เป็นต้น แต่ในบางกรณีการกำหนดให้จุดข้อมูลเป็นสมาชิกของกลุ่มข้อมูลได้มากกว่าหนึ่งกลุ่มหรือที่เรียกว่า**การแบ่งกลุ่มข้อมูลแบบซ้อนทับกัน**ก็ดูจะเป็นสิ่งเหมาะสมกว่า เช่น ในการค้นคืนข้อมูล (Information retrieval) เอกสาร 1 ชิ้นอาจจะสามารถจัดให้อยู่ในหลายหัวข้อได้ หรือ ในโรงเรียนแห่งหนึ่งครูหนึ่งคนอาจจะสอนหลายวิชาเป็นต้น ตัวอย่างของการแบ่งกลุ่มข้อมูลแบบนี้ได้แก่ OKM (Overlapping K-means) เป็นต้น ส่วน**การแบ่งกลุ่มข้อมูลแบบฟัซซี**นั้นจุดข้อมูลแต่ละจุดจะถูกกำหนดให้เป็นสมาชิกของกลุ่มข้อมูลทุกกลุ่มโดยจะมีค่าน้ำหนักความเป็นสมาชิก (membership weights) เป็นตัวบอกว่าจุดข้อมูลใดๆ มีความเป็นสมาชิกของกลุ่มข้อมูลไหนเท่าไร แต่ในทางปฏิบัติการแบ่งกลุ่มข้อมูลแบบฟัซซีมักจะถูกแปลงให้เป็น

แบบเอ็คซคลูซีฟโดยการกำหนดให้จุดข้อมูลเป็นสมาชิกของกลุ่มข้อมูลที่มันมีค่าความเป็นสมาชิกสูงที่สุด ตัวอย่างของการแบ่งกลุ่มข้อมูลแบบฟัซซีได้แก่ Fuzzy c-mean เป็นต้น

การจัดกลุ่มข้อมูลอาจนำมาใช้เป็นขั้นตอนเบื้องต้นของการวิเคราะห์ข้อมูลได้ คือ นำมาใช้ในการลดขนาดของสเปซ เช่น นำข้อมูลมาแยกเป็นกลุ่มๆ ก่อนแล้วจึงนำข้อมูลแต่ละกลุ่มไปแยกกันวิเคราะห์โดยวิธีการอื่นต่อไป

2.2 การจัดกลุ่มข้อมูลแบบมีผู้สอน

ในปัจจุบันพบว่ามีการนำเสนอวิธีการจัดกลุ่มข้อมูลที่ได้ทำการกำหนดประเภท (class) ของข้อมูลไว้ก่อนแล้ว โดยได้เรียกวิธีการใหม่นี้ว่าเป็นการจัดกลุ่มข้อมูลแบบมีผู้สอน ส่วนการจัดกลุ่มข้อมูลแบบเดิมก็จัดเป็นการจัดกลุ่มข้อมูลแบบไม่มีผู้สอน ซึ่งการที่รู้ประเภทของข้อมูลอยู่ก่อนแล้วทำให้เราสามารถจัดกลุ่มข้อมูลได้เหมาะสมมากขึ้น ดังเช่นรูปที่ 2.1 ถ้าเราใช้คลาสเลเบิลด้วยเราจะได้กลุ่มข้อมูลที่แตกต่างจากกลุ่มข้อมูลที่ได้จากการจัดกลุ่มข้อมูลแบบเดิม โดยข้อมูลที่อยู่ใกล้กันและมีคลาสเลเบิลเหมือนกันจะถูกจัดให้อยู่ในกลุ่มเดียวกัน



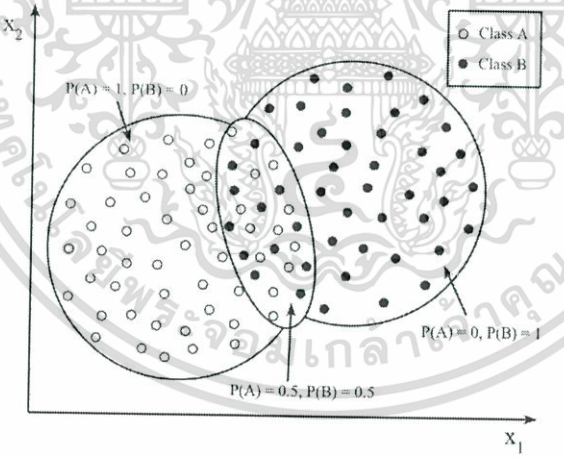
(a) การจัดกลุ่มข้อมูลแบบเดิม (b) การจัดกลุ่มข้อมูลแบบมีผู้สอน

รูปที่ 2.1 แสดงการจัดกลุ่มข้อมูลแบบเดิมเทียบกับการจัดกลุ่มข้อมูลแบบมีผู้สอน

เนื่องจากเรามีการแบ่งประเภทข้อมูลซึ่งเป็นการเรียนรู้แบบมีผู้สอนอยู่แล้ว ดังนั้นจึงอาจเกิดคำถามว่า *แล้วการจัดกลุ่มข้อมูลแบบมีผู้สอนต่างจากการแบ่งประเภทข้อมูลอย่างไร* คำตอบก็คือ การแบ่งประเภทข้อมูลนั้นเป็นการเรียนรู้บนแต่ละคุณลักษณะ (Feature) ของข้อมูลในขณะที่การจัดกลุ่มข้อมูลแบบมีผู้สอนเป็นการเรียนรู้บนความเหมือนหรือความใกล้ชิดดังได้กล่าวไปแล้ว

2.3 การจัดกลุ่มข้อมูลที่มีการซ้อนทับกัน

ดังได้กล่าวไปแล้วว่าวิธีการจัดกลุ่มข้อมูลแบบมีผู้สอนโดยทั่วไปจะทำการกำหนดให้ข้อมูลที่อยู่ใกล้กันและมีคลาสเลเบิลเหมือนกันอยู่ในกลุ่มเดียวกัน แต่ในบางครั้งกลับพบว่าข้อมูลที่อยู่ใกล้กันเป็นข้อมูลคนละประเภทและข้อมูลเหล่านั้นก็มีจำนวนมาก ซึ่งถ้าเราพยายามจัดกลุ่มข้อมูลข้อมูลเหล่านั้นเราจะได้กลุ่มข้อมูลขนาดเล็กเป็นจำนวนมาก เราจะเรียกข้อมูลที่มีลักษณะเช่นนี้ว่า *ข้อมูลที่มีการซ้อนทับกัน (Overlapping data)* โดยสาเหตุของการซ้อนทับกันของข้อมูลนั้นอาจมาจากการมีฟีเจอร์จำกัดหรือเพราะโดยธรรมชาติของมูลแล้วเป็นเช่นนั้นเอง ยกตัวอย่างเช่น ในการรู้จำลายมือเขียนภาษาไทยที่ตัวอักษร บ ของคนหนึ่งอาจจะเป็นตัวอักษร ข ของอีกคนหนึ่ง ซึ่งในกรณีนี้แทบจะเป็นไปไม่ได้เลยที่จะหาฟีเจอร์ที่สามารถแบ่งประเภทตัวอักษรที่อยู่ในกลุ่มนี้ได้อย่างมีประสิทธิภาพ หรืออาจกล่าวได้ว่าการพยายามแบ่งแยกสิ่งที่มีคุณลักษณะเหมือนกันดูจะเป็นการกระทำที่ไม่เหมาะสมเท่าใดนัก วิธีที่เหมาะสมคือทำการจัดกลุ่มให้ข้อมูลที่อยู่ใกล้ชิดกันและมีลักษณะการผสมปนกันของข้อมูลโดยรอบคล้ายคลึงกันอยู่ในกลุ่มเดียวกันแล้วบอกว่าข้อมูลในแต่ละกลุ่มมีโอกาสเป็นข้อมูลของแต่ละคลาสด้วยความน่าจะเป็นเท่าไร ดังตัวอย่างในรูปที่ 2.2 ข้อมูลที่มีการซ้อนทับกันจะถูกจัดให้เป็นกลุ่มข้อมูลหนึ่ง ส่วนข้อมูลที่ไม่ได้มีการซ้อนทับกันก็จัดให้เป็นอีกกลุ่มหนึ่ง



รูปที่ 2.2 แสดงวิธีการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันที่เหมาะสม

2.4 งานวิจัยที่เกี่ยวกับการจัดกลุ่มข้อมูลที่มีการซ้อนทับกัน

งานวิจัยที่จะนำเสนอต่อไปนี้เป็นงานวิจัยเกี่ยวกับการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันที่มีหลักการเหมือนกับที่ได้กล่าวถึงในหัวข้อที่ 2.3 คือสามารถจัดกลุ่มข้อมูลที่มีการซ้อนทับกันได้อย่างเหมาะสมและสามารถบอกได้ว่ากลุ่มข้อมูลแต่ละกลุ่มมีอัตราการผสมกันของข้อมูลในแต่ละคลาสเป็นเท่าไร

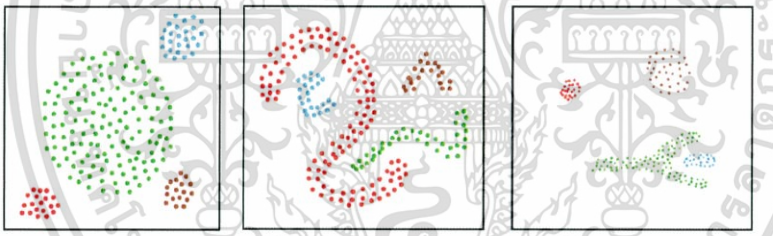
2.4.1 ODBSCAN

ODBSCAN หรือ Overlapping DBSCAN เป็นวิธีการจัดกลุ่มข้อมูลแบบมีผู้สอนที่พัฒนามาจาก DBSCAN ซึ่งเป็นวิธีการจัดกลุ่มข้อมูลแบบไม่มีผู้สอนให้มีความสามารถจัดการกับข้อมูลที่มีการซ้อนทับกันได้ โดย ODBSCAN สามารถระบุได้ว่าบริเวณไหน (กลุ่มข้อมูลไหน) มีการซ้อนทับกันของคลาสเกิดขึ้นและอัตราการผสมปนกันของข้อมูลของแต่ละคลาสเป็นอย่างไร

เนื่องจากหลักการของ ODBSCAN ยังคงเหมือนกับ DBSCAN เสียเป็นส่วนมาก ดังนั้นจึงจะยกหลักการของ DBSCAN มาอธิบายก่อนจากนั้นจึงค่อยนำเสนอ ODBSCAN ซึ่งเป็นการนำเอา DBSCAN มาประยุกต์ใช้ในการจัดกลุ่มข้อมูลที่มีการซ้อนทับกัน

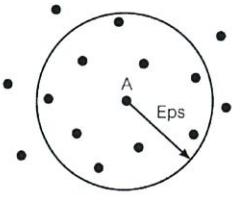
2.4.1.1 DBSCAN

DBSCAN เป็นวิธีการจัดกลุ่มข้อมูลโดยใช้ความหนาแน่น (Density-based clustering) ที่ง่ายและมีประสิทธิภาพ กลุ่มข้อมูลที่ได้จาก DBSCAN นั้นสามารถมีรูปร่างอย่างไรก็ได้ และเช่นเดียวกับวิธีการจัดกลุ่มข้อมูลโดยใช้ความหนาแน่นอื่นๆ การระบุกลุ่มข้อมูลจะระบุเป็นพื้นที่ที่มีความหนาแน่นสูงที่ถูกคั่นแบ่งจากกันด้วยพื้นที่ที่มีความหนาแน่นต่ำ รูปที่ 2.3 แสดงตัวอย่างกลุ่มข้อมูลที่ได้จากการจัดกลุ่มโดยใช้ DBSCAN



รูปที่ 2.3 แสดงตัวอย่างผลการจัดกลุ่มข้อมูลโดย DBSCAN

การนิยามความหนาแน่นของ DBSCAN เป็นแบบ center-based ซึ่งจุดข้อมูลแต่ละจุดจะมีการประมาณความหนาแน่นโดยการนับจำนวนจุดข้อมูลที่อยู่ภายในรัศมี Eps ของจุดข้อมูลนั้น รวมจุดที่เป็นศูนย์กลางด้วย และเรียกพื้นที่ที่อยู่ภายในรัศมี Eps ของจุดข้อมูลใดๆ ว่าเป็น Eps-neighborhood ของจุดข้อมูลนั้น ดังเช่นตัวอย่างในรูปที่ 2.4 ความหนาแน่นที่จุด A เท่ากับจำนวนจุดข้อมูลที่อยู่ภายใน Eps-neighborhood ของจุด A ซึ่งก็คือ 10



รูปที่ 2.4 ความหนาแน่นแบบ center-based

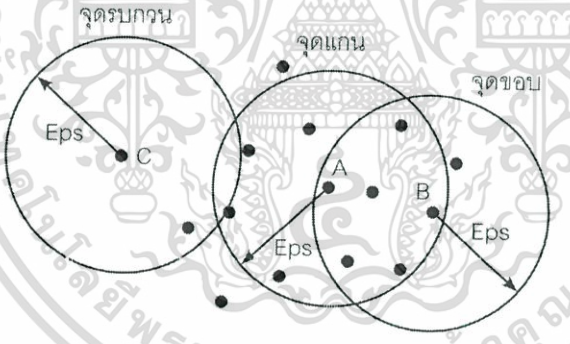
นิยาม 2.1 (Eps-neighborhood) Eps-neighborhood ของจุด p เขียนแทนโดย $N_{Eps}(p)$ นิยามโดย $N_{Eps}(p)=\{q \in D \mid \text{dist}(p,q) \le Eps\}$ เมื่อ D คือเซ็ตของจุดข้อมูลทั้งหมด

วิธีการหาความหนาแน่นแบบ center-based ทำให้เราสามารถแบ่งประเภทของจุดข้อมูลออกตามความหนาแน่นที่จุดข้อมูลนั้นได้เป็น 3 ชนิด คือ 1. จุดแกน (Core point) 2. จุดขอบ (Border point) และ 3. จุดรบกวน (Noise) รูปที่ 2.5 แสดงแนวคิดของจุดแกน จุดขอบ และจุดรบกวน รายละเอียดของจุดข้อมูลทั้ง 3 ชนิดมีดังต่อไปนี้

จุดแกน: จุดแกน คือ จุดที่มีจำนวนจุดที่อยู่ใน Eps-neighborhood เกินกว่าค่าที่กำหนด (MinPts) ในรูปที่ 2.5 จุด A คือจุดแกนถ้ากำหนดให้ $MinPts \le 10$

จุดขอบ: จุดขอบ คือ จุดที่ไม่ใช่จุดแกนแต่อยู่ใน Eps-neighborhood ของจุดแกน ในรูปที่ 2.5 จุด B คือจุดขอบ จุดขอบจะอยู่ในบริเวณขอบของกลุ่มข้อมูลและสามารถอยู่ใน Eps-neighborhood ของจุดแกนได้หลายจุด

จุดรบกวน: จุดรบกวน คือ จุดที่ไม่ได้เป็นจุดแกนหรือจุดขอบ ในรูปที่ 2.5 จุด C คือจุดรบกวน



รูปที่ 2.5 จุดแกน จุดขอบ และจุดรบกวน

จากคำจำกัดความของจุดแกน จุดขอบและจุดรบกวนดังกล่าวไว้ในข้างต้น สามารถอธิบายอัลกอริธึมของ DBSCAN อย่างย่อได้ดังนี้ คือ ทำการรวมจุดแกนสองจุดใดๆ ที่อยู่ภายในรัศมี Eps ของกันและกันเข้าเป็นกลุ่มข้อมูลเดียวกัน และเลือกรวมจุดขอบที่อยู่ภายในรัศมี Eps ของจุดแกนซึ่งอาจมีหลายจุดเข้าเป็นกลุ่มเดียวกันกับจุดแกนจุดใดจุดหนึ่งเพียงจุดเดียว ส่วนจุดรบกวนไม่ต้องนำไปรวมกับกลุ่มข้อมูลใด อัลกอริธึมที่ 2.1 แสดงอัลกอริธึมของ DBSCAN

อัลกอริธึมที่ 2.1 อัลกอริธึม DBSCAN

1. Label all points as core, border, or noise points.
2. Eliminate noise points.
3. Put an edge between all core points that are within *Eps* of each other.
4. Make each group of connected core point into a separate cluster.
5. Assign each border point to one of the clusters of its assosizated core points.

2.4.1.2 การประยุกต์ใช้ DBSCAN ในการจัดกลุ่มข้อมูลที่มีการซ้อนทับกัน

เพื่อที่จะทำให้สามารถบอกได้ว่าพื้นที่ในฟีเจอร์สเปซหรือกลุ่มข้อมูลใดมีการปนกันของข้อมูลของแต่ละคลาสในสัดส่วนเท่าไร ใน ODBSCAN จะมีการหาค่าอัตราส่วนคลาส (Class ratio) สำหรับจุดข้อมูลแต่ละจุด ซึ่งอัตราส่วนคลาสนี้จะเป็นค่าที่บอกว่าพื้นที่ใน Eps-neighborhood ของจุดข้อมูลใดๆ มีการปนกันของข้อมูลของแต่ละคลาสในอัตราส่วนเท่าไร และในการสร้างกลุ่มข้อมูล จุดแกนที่อยู่ภายในรัศมี Eps ของกันและกันและมีอัตราส่วนคลาสคล้ายกันจะถูกกำหนดให้อยู่ในกลุ่มเดียวกัน โดยที่การพิจารณาความคล้ายกันของอัตราส่วนคลาสจะดูจากค่าความต่างของอัตราส่วนคลาส

นิยาม 2.2 (อัตราส่วนคลาสของจุดข้อมูล) อัตราส่วนคลาสของจุด p คือ เวกเตอร์ของอัตราส่วนระหว่างจำนวนจุดข้อมูลของแต่ละคลาสในพื้นที่ Eps-neighborhood ของจุด p กับจำนวนจุดข้อมูลในพื้นที่ Eps-neighborhood ของจุด p ทั้งหมด อัตราส่วนคลาสของจุด p เขียนแทนด้วย $CR(p)$ โดยที่

$$CR(p) = [r_1, r_2, \ldots, r_m]^T \tag{2.1}$$

และ

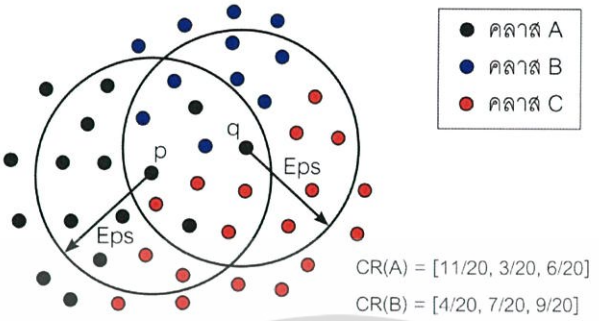
$$r_i = \frac{n_i}{n_1 + n_2 + \cdots + n_m} \tag{2.2}$$

เมื่อ n_i คือ จำนวนจุดข้อมูลของคลาส i ใน Eps-neighborhood ของจุด p

m คือ จำนวนคลาส

จากตัวอย่างในรูปที่ 2.6 จะพบว่าจุดข้อมูลของคลาส A B และ C ที่อยู่ภายในพื้นที่ Eps-neighborhood ของจุด p มีจำนวนเท่ากับ 11 3 และ 6 จุดตามลำดับ ดังนั้นอัตราการปนกันของข้อมูลแต่ละคลาสภายในพื้นที่ Eps-neighborhood ของจุด p จึงเท่ากับ 11/20 3/20 และ 6/20 ตามลำดับ และเขียนเป็นอัตราส่วนคลาสได้ดังนี้ $CR(p) = [11/20,\ 3/20,\ 6/20]^T$ ในทำนองเดียวกัน อัตราส่วนคลาสของจุด q คือ $CR(q) = [4/20,\ 7/20,\ 9/20]^T$

นิยาม 2.3 (ความต่างของอัตราส่วนคลาสของจุดข้อมูล) ความต่างของอัตราส่วนคลาสของจุด p และจุด q เขียนแทนด้วย $DCR(p,q)$ โดยที่ $DCR(p,q)=|CR(p)-CR(q)|$



รูปที่ 2.6 อัตราส่วนคลาส

ความต่างของอัตราส่วนคลาส คือ ค่าที่แสดงถึงความแตกต่างระหว่างอัตราส่วนคลาสของจุดสองจุดซึ่งในงานวิจัย [5] ได้มีการนำเสนอวิธีการหาค่าความต่างของอัตราส่วนคลาสไว้ 3 วิธี ดังนี้คือ

1. การวัดค่าความต่างของอัตราส่วนคลาสโดยดูจากผลรวมของผลต่างระหว่างอัตราส่วนคลาสในแต่ละคลาส

$$|CR(p)-CR(q)|=\sum_{i=1}^{m}|r_i^p-r_i^q| \tag{2.3}$$

เมื่อ

r_i^p, r_i^q คือ ค่าอัตราส่วนคลาสที่ i ของจุด p และ q ตามลำดับ

m คือ จำนวนคลาสทั้งหมด

จากตัวอย่างในรูปที่ 2.6 สามารถหาค่าความแตกต่างระหว่างจุด p กับจุด q ได้ดังนี้

$$\begin{aligned}DCR(p,q)&=\left|r_a^p-r_a^q\right|+\left|r_b^p-r_b^q\right|+\left|r_c^p-r_c^q\right|\\&=\left|\frac{11}{20}-\frac{4}{20}\right|+\left|\frac{3}{20}-\frac{7}{20}\right|+\left|\frac{6}{20}-\frac{9}{20}\right|\\&=\frac{7}{20}+\frac{4}{20}+\frac{3}{20}\\&=0.35+0.2+0.15\\&=0.7\end{aligned}$$

2. การวัดค่าความต่างของอัตราส่วนคลาสโดยดูจากผลรวมของผลต่างระหว่างอัตราส่วนคลาสที่มากกว่าค่าที่กำหนด

$$|CR(p)-CR(q)|=\sum_{i=1}^{m} d\left(\left|r_i^p-r_i^q\right|\right) \tag{2.4}$$

โดยที่

$$d\left(\left|r_i^p-r_i^q\right|\right)=\begin{cases}0 & \text{where} \quad \left|r_i^p-r_i^q\right|\le\alpha \\ \left|r_i^p-r_i^q\right|-\alpha & \text{otherwise}\end{cases} \tag{2.5}$$

จากตัวอย่างในรูปที่ 2.6 ถ้ากำหนดให้ $\alpha=0.2$ จะสามารถหาค่าความแตกต่างระหว่างจุด p กับจุด q ได้ดังนี้

$$\begin{aligned}DCR(p,q) &= d\left(\left|r_a^p-r_a^q\right|\right)+d\left(\left|r_b^p-r_b^q\right|\right)+d\left(\left|r_c^p-r_c^q\right|\right) \\ &= d\left(\left|\frac{11}{20}-\frac{4}{20}\right|\right)+d\left(\left|\frac{3}{20}-\frac{7}{20}\right|\right)+d\left(\left|\frac{6}{20}-\frac{9}{20}\right|\right) \\ &= 0.15+0+0 \\ &= 0.15\end{aligned}$$

3. การวัดค่าความต่างของอัตราส่วนคลาสโดยดูจากผลต่างสูงสุดของอัตราส่วนคลาส

$$|CR(p)-CR(q)|=\max_i\left|r_i^p-r_i^q\right|; \quad i=1,2,\ldots,m \tag{2.6}$$

จากตัวอย่างในรูปที่ 2.6 สามารถหาค่าความแตกต่างระหว่างจุด p กับจุด q ได้ดังนี้

$$\begin{aligned}DCR(p,q) &= \max\left(\left|r_a^p-r_a^q\right|,\left|r_b^p-r_b^q\right|,\left|r_c^p-r_c^q\right|\right) \\ &= \max\left(\left|\frac{11}{20}-\frac{4}{20}\right|,\left|\frac{3}{20}-\frac{7}{20}\right|,\left|\frac{6}{20}-\frac{9}{20}\right|\right) \\ &= \max\left(0.35,0.2,0.15\right) \\ &= 0.35\end{aligned}$$

นิยาม 2.4 (Eps-neighborhood with class ratio) Eps-neighborhood with class ratio หรือ เซ็ตของจุดข้อมูลที่อยู่ในรัศมี Eps ของจุด p และมีค่าความต่างของอัตราส่วนคลาสไม่เกินค่าที่กำหนด เขียนแทนโดย $NCR_{Eps}(p)$ นิยามโดย

$$NCR_{Eps}(p)=\{q\in D \mid \text{dist}(p,q)\le Eps \text{ and } DCR(p,q)\le MaxDiff\}$$

เมื่อ D คือเซ็ตของจุดข้อมูลทั้งหมด และ $MaxDiff$ คือค่าความต่างของอัตราส่วนคลาสที่มากที่สุดที่ยังถือว่าอัตราส่วนคลาสของจุด p และ q มีความคล้ายคลึงกัน

จากนิยามต่างๆ ดังกล่าวไว้ข้างบนสามารถอธิบายอัลกอริธึมของ ODBSCAN ได้ดังนี้ คือ 1) ทำการเลือกจุดแกน p มาหนึ่งจุด และสร้างกลุ่มข้อมูลขึ้นมาใหม่โดยกำหนดให้จุดข้อมูลที่เป็นสมาชิกของ

Eps-neighborhood with class ratio ของจุด p เป็นสมาชิกของกลุ่มข้อมูลที่สร้างขึ้นใหม่นี้และกำหนดให้จุดขอบที่เป็นสมาชิกของ Eps-neighborhood ของจุด p แต่ไม่ได้เป็นสมาชิกของ Eps-neighborhood with class ratio ของจุด p เป็นข้อมูลรบกวน จากนั้นนำสมาชิกของ Eps-neighborhood with class ratio ของจุด p ที่เป็นจุดแกนมาใช้เป็นจุดเริ่มในการแผ่ขยายกลุ่มข้อมูลต่อไปโดยทำเช่นเดียวกันกับวิธีการในข้างต้น ทำการแผ่ขยายกลุ่มข้อมูลไปเรื่อยๆ จนกว่าจะไม่สามารถขยายได้อีกต่อไป 2) เมื่อไม่สามารถขยายกลุ่มข้อมูลเดิมต่อไปได้แล้วก็จะทำการเลือกจุดแกนที่ยังไม่ได้เป็นสมาชิกของกลุ่มข้อมูลใดขึ้นมาและสร้างกลุ่มข้อมูลขึ้นใหม่แล้วทำซ้ำกระบวนการตามข้อ 1 จนกว่าจะไม่มีจุดแกนที่ยังไม่ได้เป็นสมาชิกของกลุ่มข้อมูลใดเหลืออยู่อีก 3) กำหนดให้จุดข้อมูลที่ไม่ใช่จุดแกนหรือจุดขอบเป็นข้อมูลรบกวน อัลกอริธึมที่ 2.2 แสดงอัลกอริธึมของ ODBSCAN

อัลกอริธึมที่ 2.2 อัลกอริธึม ODBSCAN

```
1.  clusterID ← 0
2.  Noise ← {}
3.  FOR EACH point p in Point DO
4.      IF p is not labeled THEN
5.          IF p is a core point THEN
6.              clusterID ← clusterID + 1
7.              N ← a set of points in the Eps-neighborhood (or Eps-neighborhood with
                    class ratio) of p
8.              Cluster[clusterID] ← {p} ∪ {n | n∈N}
9.              Queue ← points in the Eps-neighbor with class ratio of p
10.             prc ← RepresentativePoint(Cluster[clusterID])
11.             REPEAT
12.                 q ← Dequeue(Queue)
13.                 IF q is a core point compare with prc THEN
14.                     N ← a set of points in the Eps-neighborhood (or Eps-neighborhood
                            with class ratio) of q
15.                     Cluster[clusterID] ← Cluster[clusterID] ∪ {n | n∈N}
16.                     Add points in the Eps-neighborhood with class ratio of q compare
                            with prc to Queue
17.                 END IF
18.             UNTIL Queue is empty
19.         ELSE
20.             Noise ← Noise ∪ {p}
21.         END IF
22.     END IF
23. END FOR
```

2.4.2 การจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริด

การจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริดเป็นวิธีการ จัดกลุ่มข้อมูลที่พัฒนาขึ้นหลัง ODBSCAN โดยมีวัตถุประสงค์เพื่อลดเวลาที่ใช้ในการประมวลผลให้น้อยกว่า ODBSCAN ซึ่งแต่เดิมใน ODBSCAN เราต้องทำการคำนวณกับทุกจุดข้อมูล เลยทำให้ใช้เวลาในการประมวลผลนาน ไม่เหมาะที่จะนำไปใช้กับฐานข้อมูลที่มีขนาดใหญ่ แต่ในการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริด พื้นที่ในฟีเจอร์สเปซจะถูกแบ่งออกเป็นช่องกริดก่อนแล้วจึงค่อยทำการรวมกริดที่อยู่ติดกันและมีลักษณะการปนกันของข้อมูลคล้ายกันให้เป็นกลุ่มข้อมูลเดียวกันแทนการรวมจุดข้อมูลทีละจุด ซึ่งถ้าเรากำหนดให้กริดมีขนาดใหญ่มากพอก็จะทำให้จำนวนของกริดมีน้อยกว่าจำนวนของจุดข้อมูลทั้งหมดมาก และส่งผลให้เวลาที่ใช้ในการรวมกลุ่มข้อมูลลดลงมากตามไปด้วย

รูปที่ 2.7 แสดงพื้นที่ภายในรัศมี Eps (a) เทียบกับกริดที่มีด้านแต่ละด้านยาวเท่ากับ Eps (b) ซึ่งใน ODBSCAN เราต้องทำการคำนวณหาความหนาแน่นกับอัตราส่วนคลาสภายในรัศมี Eps ของจุดข้อมูลทุกจุด แต่ถ้าทำการแบ่งฟีเจอร์สเปซออกเป็นช่องกริด การคำนวณหาความหนาแน่นกับอัตราส่วนคลาสจะทำสำหรับแต่ละช่องกริดเท่านั้น

รูปที่ 2.7 แสดงพื้นที่ภายในรัศมี Eps เทียบกับพื้นที่ของกริดที่มีด้านยาวเท่ากับ Eps

นิยาม 2.5 (อัตราส่วนคลาสของกริด) อัตราส่วนคลาสของกริด A คือ เวกเตอร์ของอัตราส่วนระหว่างจำนวนจุดข้อมูลของแต่ละคลาสที่อยู่ในกริด A กับจำนวนจุดข้อมูลที่อยู่ในกริด A ทั้งหมด อัตราส่วนคลาสของกริด A เขียนแทนด้วย $CR(A)$ โดยที่

$$CR(A) = [r_1, r_2, \ldots, r_m] \tag{2.7}$$

และ

$$r_i = \frac{n_i}{n_1 + n_2 + \cdots + n_m} \tag{2.8}$$

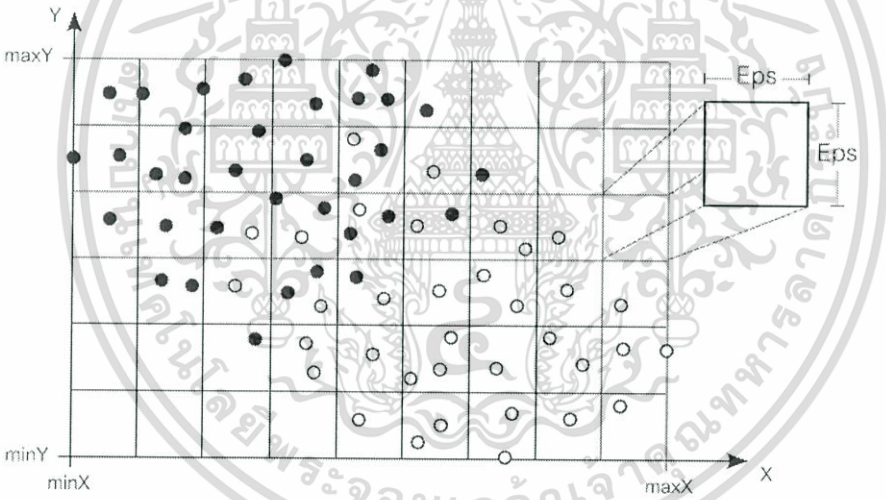
เมื่อ n_i คือ จำนวนจุดข้อมูลของคลาส i ที่อยู่ในกริด A

m คือ จำนวนคลาส

2.4.2.1 การสร้างกริด

ในการแบ่งพื้นที่ในฟีเจอร์สเปซออกเป็นกริด ขอบเขตของการสร้างกริดในแต่ละมิติจะอยู่ระหว่างค่าพิกัดต่ำสุดของมิตินั้นๆ (จากพิกัดของจุดข้อมูลทั้งหมด) และค่าพิกัดสูงสุดของมิตินั้นๆ (จากพิกัดของจุดข้อมูลทั้งหมด) และแต่ละมิติจะถูกแบ่งออกเป็นช่วงๆ โดยแต่ละช่วงจะมีความยาวเท่ากับค่า Eps (กำหนดโดยผู้ใช้งาน) ทั้งนี้ช่วงสุดท้ายของแต่ละมิติอาจมีความยาวน้อยกว่า Eps ก็ได้ ตัวอย่างในรูปที่ 2.8 แสดงการแบ่งพื้นที่ในฟีเจอร์สเปซ 2 มิติออกเป็นกริด จากรูปจะเห็นว่าขอบเขตของการสร้างกริดในแกน X จะอยู่ระหว่างค่า minX และmaxX ซึ่งเป็นค่าพิกัดบนแกน X ที่มีค่าน้อยที่สุดและมากที่สุดตามลำดับ และเช่นเดียวกัน ขอบเขตของการสร้างกริดในแกน Y อยู่ระหว่างค่า minY และ maxY

การเลือกค่า Eps ที่เหมาะสมก็ถือว่ามีความสำคัญมาก เพราะว่าถ้าเลือกค่าน้อยเกินไป ขนาดของกริดที่ได้ก็จะมีขนาดเล็กมากตามไปด้วย และทำให้กริดที่อยู่ใกล้เคียงกันมีอัตราการปนกันของข้อมูลแตกต่างกันมาก แต่ถ้าเลือกค่ามากเกินไป ขนาดของกริดก็จะใหญ่มาก และทำให้บริเวณภายในกริดมีอัตราการปนกันของข้อมูลแตกต่างกันมาก



รูปที่ 2.8 แสดงกริดในฟีเจอร์สเปซ 2 มิติ

2.4.2.2 การรวมกริดให้เป็นกลุ่มข้อมูล

หลังจากที่ทำการแบ่งฟีเจอร์สเปซออกเป็นช่องกริดแล้วก็จะต้องทำการรวมกริดที่อยู่ใกล้เคียงกันและมีลักษณะการปนกันของข้อมูลคล้ายกันหรือมีอัตราส่วนคลาสคล้ายกันให้เป็นกลุ่มข้อมูลเดียวกัน โดยที่การวัดความคล้ายกันของอัตราส่วนคลาสจะดูจากค่าความต่างของอัตราส่วนคลาสของกริด ถ้ากริดที่อยู่ติดกันมีค่าความต่างของอัตราส่วนคลาสน้อยโอกาสที่กริดเหล่านั้นจะถูกจัดเข้าเป็นกลุ่มเดียวกันก็จะมีมาก

นิยาม 2.6 (ความต่างของอัตราส่วนคลาสของกริด) ความต่างของอัตราส่วนคลาสของกริด A และ กริด B เขียนแทนด้วย $DCR(A,B)$ โดยที่ $DCR(A,B) = |CR(A) - CR(B)|$ และ

$$|CR(p) - CR(q)| = \sum_{i=1}^{m} |r_i^A - r_i^B| \tag{2.9}$$

เมื่อ

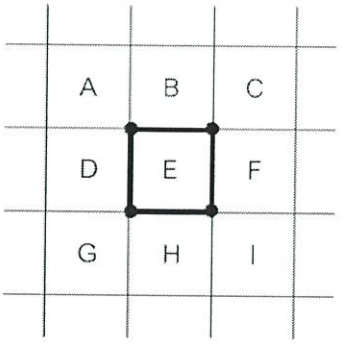
r_i^A, r_i^B คือ ค่าอัตราส่วนคลาสที่ i ของกริด A และกริด B ตามลำดับ

m คือ จำนวนคลาสทั้งหมด

ในการรวมกริดให้เป็นกลุ่มข้อมูล ลำดับของการรวมกริดก็นับว่ามีความสำคัญเพราะว่าถ้าลำดับของการรวมกริดเปลี่ยนไปกลุ่มข้อมูลที่ได้ก็อาจจะไม่เหมือนเดิมก็ได้ ดังนั้นเพื่อศึกษาดูผลของการรวมกริดที่แตกต่างกัน ในงานวิจัย [6] จึงได้มีการนำเสนอวิธีการรวมกริดให้เป็นกลุ่มข้อมูลไว้หลายวิธีซึ่งสามารถแบ่งออกได้เป็น 2 วิธีใหญ่ๆ คือ 1) การรวมกริดแบบเนเบอร์ฮูดกริด และ 2) การรวมกริดที่อยู่ใกล้กันมากที่สุดและมีค่าความต่างของอัตราส่วนคลาสน้อยที่สุดก่อน

นิยาม 2.7 (เนเบอร์ฮูดกริด – Neighborhood grid) เนเบอร์ฮูดกริดของกริด g คือ กริดที่มีจุดพิกัดของมุมเหมือนกับจุดพิกัดของมุมของกริด g อย่างน้อยหนึ่งมุมและจุดข้อมูลที่อยู่ภายในกริดนั้นมีจำนวนมากกว่าหรือเท่ากับค่าที่กำหนด (MinPts)

สำหรับกริดใด ๆ ที่มีจำนวนจุดข้อมูลน้อยกว่าค่า MinPts จะถือว่ากริดนั้นเป็นกริดของข้อมูลรบกวนและจะไม่นำกริดนั้นมาพิจารณาในการรวมกริดให้เป็นกลุ่มข้อมูล ตัวอย่างในรูปที่ 2.9 กริดที่มีโอกาสเป็นเนเบอร์ฮูดกริดของกริด E ถ้าจำนวนจุดข้อมูลที่อยู่ในกริดนั้นมีมากกว่าหรือเท่ากับค่า MinPts มีจำนวนเท่ากับ 8 กริด ได้แก่ กริด A B C D F G H และ I โดยที่ กริด A C G และ I มีมุมเหมือนกับกริด E หนึ่งมุม และกริด B D F และ H มีมุมเหมือนกับกริด E สองมุม สำหรับข้อมูล n มิติ กริดที่มีโอกาสเป็นเนเบอร์ฮูดกริดของกริดใด ๆ จะมีจำนวนเท่ากับ $3^n - 1$ กริด



รูปที่ 2.9 เนเบอร์ฮูดกริดในฟีเจอร์สเปซ 2 มิติ

นิยาม 2.8 (กริดที่เชื่อมต่อกัน) กริดที่เชื่อมต่อกัน หมายถึง กริดที่เป็นเนเบอร์ฮูดกริดของกันและกันและมีความต่างของอัตราส่วนคลาสน้อยกว่าหรือเท่ากับค่าที่กำหนด (MaxDiff)

2.4.2.3 อัลกอริธึมของการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริด

เมื่อกำหนดให้ข้อมูลอินพุตมีจำนวนเท่ากับ n จุดและทุกจุดข้อมูลมีการระบุคลาส 1 คลาสจากจำนวนคลาสทั้งหมด m คลาส การจัดกลุ่มข้อมูลโดยใช้กริดจะมีขั้นตอนดังต่อไปนี้

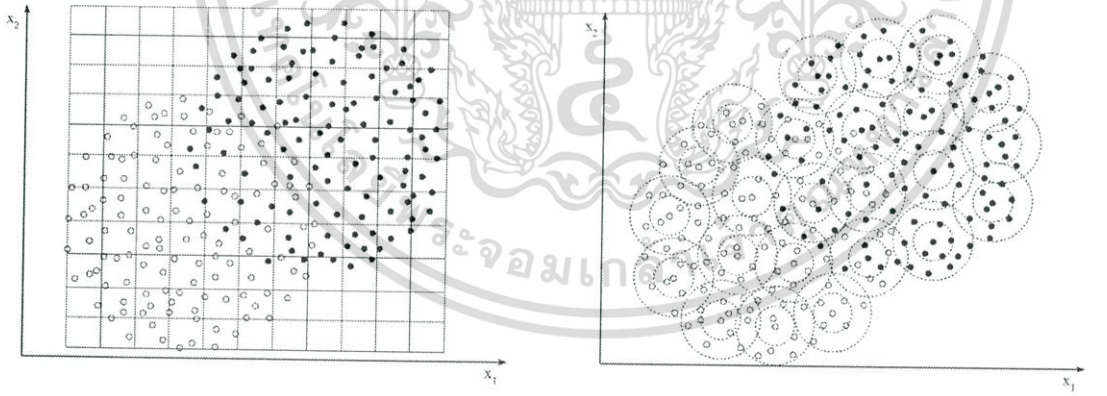
1. อ่านจุดข้อมูลเข้ามา n จุด เพื่อหาขอบเขตในการสร้างกริด (ฟีเจอร์สเปซ)
2. ทำการสร้างกริดที่มีขนาดความกว้างของแต่ละด้านเท่ากับ Eps บนฟีเจอร์สเปซ
3. อ่านจุดข้อมูลเข้ามา n จุด อีกครั้งหนึ่งเพื่อหาว่าจุดข้อมูลแต่ละจุดอยู่ในกริดใด
4. ทำการหาอัตราส่วนคลาสของแต่ละกริด และกำหนดให้กริดที่มีจำนวนจุดข้อมูลน้อยกว่า MinPts เป็นกริดของข้อมูลรบกวนซึ่งจะไม่นำมาพิจารณาในการจัดกลุ่ม
5. ทำการสุ่มเลือกกริดที่ยังไม่ได้กำหนดให้เป็นสมาชิกของกลุ่มข้อมูลใดขึ้นมา 1 ช่องเพื่อเป็นตัวแทนในการรวมกริด
6. ทำการหาเนเบอร์ฮูดกริดของกริดที่เลือกมาในข้อ 5. แล้วทำการหาค่าความต่างของอัตราส่วนคลาสของกริดที่เลือกมาในข้อ 5. กับเนเบอร์ฮูดกริดเหล่านั้น
7. ทำการรวมกริดที่เลือกมากับเนเบอร์ฮูดกริดที่เป็นกริดที่เชื่อมต่อกันกับกริดที่เลือกมานั้นให้เป็นกลุ่มข้อมูลเดียวกัน จากนั้นทำการคำนวนหาอัตราส่วนคลาสของกลุ่มข้อมูลนั้นใหม่
8. ทำการเลือกกริดที่เชื่อมต่อกันในข้อ 7. ขึ้นมา แล้วทำการหาเนเบอร์ฮูดกริดและค่าความต่างของอัตราส่วนคลาสของกริดนั้นกับเนเบอร์ฮูดกริด จากนั้นทำข้อ 7. ซ้ำจนกระทั่งไม่สามารถรวมกริดกับเนเบอร์ฮูดกริดของมันได้
9. ทำซ้ำข้อที่ 5. จนไม่เหลือกริดให้รวมอีก

บทที่ 3
การจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลมและโดยใช้กริดที่มีความละเอียดหลายระดับ

เนื้อหาของบทนี้เป็นรายละเอียดของการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันของคลาสเกิดขึ้นโดยการใช้กลุ่มข้อมูลหน่วยแบบวงกลมและโดยการใช้กริดที่มีความละเอียดหลายระดับ โดยตอนแรกจะนำเสนอการจัดกลุ่มข้อมูลโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลมและกล่าวถึงปัญหาของวิธีการนี้ก่อน จากนั้นจึงนำเสนอการจัดกลุ่มข้อมูลโดยใช้กริดหลายระดับ

3.1 การจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลม

หัวข้อนี้จะเป็นการนำเสนอวิธีการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยการใช้กลุ่มข้อมูลหน่วยแบบวงกลม ซึ่งหลักการโดยรวมของวิธีการนี้ก็ยังคงเหมือนกับวิธีที่ใช้กริดอยู่ นั่นคือ จะสร้างกลุ่มข้อมูลหน่วยขึ้นมาก่อนและจากนั้นก็จะทำการรวมกลุ่มข้อมูลหน่วยที่อยู่ใกล้กันและมีอัตราส่วนคลาสใกล้เคียงกันเข้าด้วยกัน แต่สิ่งที่แตกต่างของวิธีการนี้กับวิธีที่ใช้กริดนี้ก็คือการสร้างกลุ่มข้อมูลหน่วยจะทำเฉพาะในบริเวณที่มีจุดข้อมูลอยู่เท่านั้นดังเช่นตัวอย่างในรูปที่ 3.1



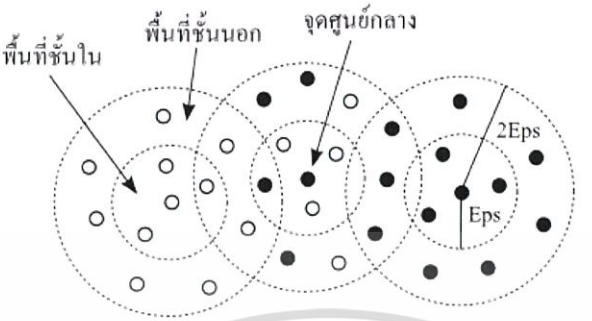
(a) (b)

รูปที่ 3.1 แสดงการเปรียบเทียบการแบ่งสเปซโดยใช้กริดกับกลุ่มข้อมูลหน่วยแบบวงกลม

3.1.1 กลุ่มข้อมูลหน่วยแบบวงกลม (Circular Unit Clusters)

กลุ่มข้อมูลหน่วยที่ใช้ในการจัดกลุ่มข้อมูลที่นำเสนอนี้เทียบได้กับกริดแต่ละช่องในการจัดกลุ่มข้อมูลโดยใช้กริดนั่นเอง ซึ่งในที่นี้กลุ่มข้อมูลหน่วยจะไม่ได้มีลักษณะเป็นสี่เหลี่ยมที่วางเรียงชิดติดกัน แต่จะมีลักษณะเป็นรูปวงกลมที่วางซ้อนทับกัน พื้นที่ของกลุ่มข้อมูลหน่วยจะประกอบไปด้วยพื้นที่ 2 ส่วน คือ พื้นที่ชั้นในกับพื้นที่ชั้นนอก พื้นที่ชั้นในคือพื้นที่ที่อยู่ในรัศมี Eps จากจุดศูนย์กลาง ส่วนพื้นที่

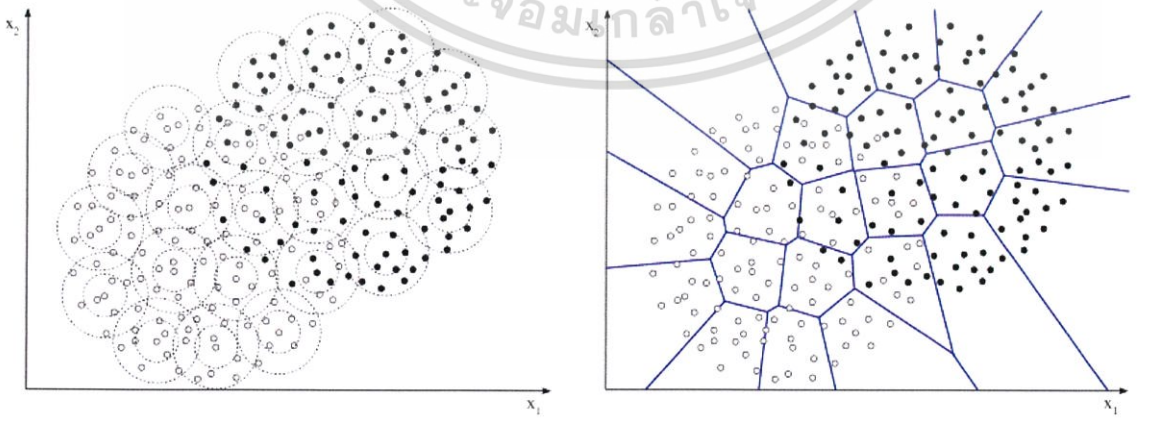
ชั้นนอกคือพื้นที่ที่อยู่ระหว่างรัศมี Eps กับ 2Eps ดังแสดงในรูปที่ 3.2 พื้นที่ชั้นนอกเป็นพื้นที่ที่สามารถซ้อนทับกันได้แต่พื้นที่ชั้นในของกลุ่มข้อมูลหน่วย 2 กลุ่มใด ๆ จะเป็นพื้นที่ที่ไม่สามารถซ้อนทับกันได้



รูปที่ 3.2 แสดงกลุ่มข้อมูลหน่วยในฟีเจอร์สเปซ 2 มิติ

เหตุผลที่กลุ่มข้อมูลหน่วยต้องมีพื้นที่สองชั้นก็คือ เพื่อลดจำนวนการคำนวนหาว่าจุดข้อมูลเป็นสมาชิกของกลุ่มข้อมูลหน่วยใดลง ทั้งนี้ก็เพราะว่าเราจะอนุญาตให้จุดข้อมูลใดๆ เป็นสมาชิกของกลุ่มข้อมูลหน่วยได้เพียงกลุ่มเดียวเท่านั้น ซึ่งถ้าใช้พื้นที่เพียงชั้นเดียวเราจะต้องทำการคำนวนหาระยะห่างระหว่างจุดข้อมูลกับจุดศูนย์กลางของกลุ่มข้อมูลหน่วยที่มีอยู่ทั้งหมด แล้วกำหนดให้จุดข้อมูลนั้นเป็นสมาชิกของกลุ่มข้อมูลหน่วยที่อยู่ใกล้ที่สุด แต่ถ้าใช้พื้นที่ 2 ชั้น เมื่อพบว่าจุดข้อมูลนั้นอยู่ในพื้นที่ชั้นในของกลุ่มข้อมูลหน่วยวงใดวงหนึ่งก็สามารถกำหนดให้เป็นสมาชิกของกลุ่มข้อมูลหน่วยนั้นได้เลย มีเพียงจุดข้อมูลที่อยู่ในพื้นที่ชั้นนอกเท่านั้นที่ต้องคำนวนเทียบกับกลุ่มข้อมูลหน่วยที่มีอยู่ทั้งหมด

การที่กำหนดให้จุดข้อมูลใดๆ เป็นสมาชิกของกลุ่มข้อมูลหน่วยได้เพียงกลุ่มเดียวนั้นทำให้ขอบเขตที่แท้จริงของกลุ่มข้อมูลหน่วยไม่ได้เป็นวงกลมแต่จะเป็นรูปทรงหลายด้าน ดังเช่นตัวอย่างในรูปที่ 3.3 รูปที่ 3.3 (a) แสดงการจัดเรียงตัวของกลุ่มข้อมูลหน่วยแบบวงกลม ในขณะที่รูป 3.3 (b) แสดงขอบเขตที่แท้จริงของกลุ่มข้อมูลหน่วยแบบวงกลม

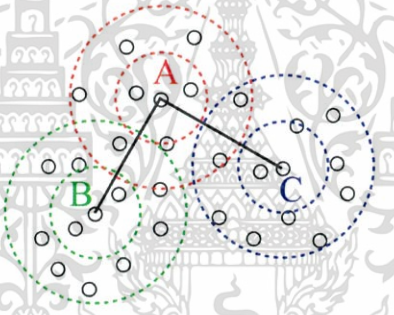


(a) (b)

รูปที่ 3.3 แสดงขอบเขตที่แท้จริงของกลุ่มข้อมูลหน่วยแบบวงกลม

3.1.2 กลุ่มข้อมูลหน่วยข้างเคียง

ในการจัดกลุ่มข้อมูลโดยการใช้กริด เนื่องจากช่องกริดมีการเรียงชิดติดกันอย่างเป็นระเบียบ ดังนั้นเราจึงสามารถทราบได้ทันทีว่าช่องกริดช่องไหนบ้างเป็นช่องกริดข้างเคียงกัน แต่ในการจัดกลุ่มข้อมูลโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลม เนื่องจากกลุ่มข้อมูลหน่วยถูกสร้างขึ้นมาอย่างไม่เป็นระเบียบ ดังนั้นจึงเป็นการยากที่จะระบุว่ากลุ่มข้อมูลหน่วยไหนบ้างเป็นกลุ่มข้อมูลหน่วยข้างเคียงกัน หากจะพิจารณาจากขอบเขตที่แท้จริงของกลุ่มข้อมูลซึ่งมีลักษณะเป็นรูปทรงหลายด้านแล้วกำหนดให้กลุ่มข้อมูลหน่วยที่อยู่ชิดติดกันเป็นกลุ่มข้อมูลหน่วยข้างเคียงกันก็จะต้องใช้เวลาในการประมวลผลเพิ่มขึ้นอีก ดังนั้นในวิทยานิพนธ์นี้เราจึงได้กำหนดให้กลุ่มข้อมูลหน่วยที่มีพื้นที่วงกลมซ้อนทับกันเป็นกลุ่มข้อมูลหน่วยข้างเคียงกัน ซึ่งนั่นก็หมายความว่าถ้ากลุ่มข้อมูลหน่วยคู่ใดมีระยะห่างจากจุดศูนย์กลางของกันน้อยกว่า 4 เท่าของ Eps กลุ่มข้อมูลหน่วยคู่นั้นก็จะเป็นกลุ่มข้อมูลหน่วยข้างเคียงของกันและกัน ดังตัวอย่างในรูปที่ 3.4 กลุ่มข้อมูลหน่วย A เป็นกลุ่มข้อมูลหน่วยข้างเคียงของ B และ C แต่ B กับ C ไม่ได้เป็นกลุ่มข้อมูลหน่วยข้างเคียงของกันและกัน



รูปที่ 3.4 แสดงกลุ่มข้อมูลหน่วยข้างเคียง

3.1.3 อัลกอริธึมการสร้างกลุ่มข้อมูลหน่วย

สำหรับการสร้างกลุ่มข้อมูลหน่วยแบบวงกลมจะทำโดยการอ่านข้อมูล p เข้ามาทีละจุดแล้วทำการตรวจสอบว่าจุดข้อมูล p อยู่ใกล้วงกลมไหนมากที่สุด จากนั้นจึงตรวจสอบต่อว่าจุดข้อมูล p อยู่ในพื้นที่ชั้นในหรือชั้นนอกของวงกลมที่อยู่ใกล้ที่สุด ถ้าอยู่ในพื้นที่ชั้นในก็จะกำหนดให้จุด p เป็นสมาชิกของวงกลมนั้นเลย หรือถ้าอยู่ในพื้นที่ชั้นนอกก็จะเก็บจุด p ไว้ก่อน แต่ถ้าจุด p ไม่ได้อยู่ทั้งในพื้นที่ชั้นในและชั้นนอกของวงกลมที่อยู่ใกล้ที่สุดก็จะสร้างวงกลมขึ้นมาใหม่โดยใช้จุด p เป็นจุดศูนย์กลาง จนกระทั่งเมื่ออ่านข้อมูลเข้ามาครบทุกจุดแล้วจึงนำข้อมูลที่ได้เก็บไว้ (ข้อมูลที่ยังไม่ได้กำหนดให้เป็นสมาชิกของวงกลมวงไหนเลย) มาตรวจสอบอีกครั้งหนึ่งว่าอยู่ใกล้วงกลมวงไหนมากที่สุดแล้วกำหนดให้เป็นสมาชิกของวงกลมที่อยู่ใกล้ที่สุด

อัลกอริธึมของการสร้างกลุ่มข้อมูลหน่วยแบบวงกลมเป็นดังอัลกอริธึมที่ 3.1

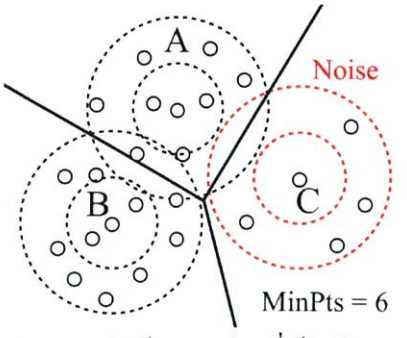
อัลกอริธึมที่ 3.1 อัลกอริธึมการสร้างกลุ่มข้อมูลหน่วยแบบวงกลม

```
1.  v_2ePoint = an empty list
2.  v_cluster = an empty list
3.  FOR EACH point p in Point DO
4.     IF v_cluster is empty THEN
5.        Create a circle with p is a center and push it to
             v_cluster's back
6.        CONTINUE
7.     END IF
8.     minDist = a distance from p to the nearest cluster
9.     IF minDist is less than EPS THEN
10.       Set p to be a member of the nearest cluster
11.    ELSE IF minDist is less than or equal to 2*EPS THEN
12.       Push p to v_2ePoint
13.    ELSE
14.       Create a circle with p is a center and push it to
             v_cluster's back
15.    END IF
16. END FOR
17. FOR EACH point p in v_2ePoint DO
18.    Find the nearest cluster from the nearest cluster's neighbors
19.    IF the new nearest cluster is nearer than the old one THEN
20.       Set p to be a member of the new nearest cluster
21.    ELSE
22.       Set p to be a member of the old nearest cluster
23.    END IF
24. END FOR
```

จากอัลกอริธึมเราจะพบว่า การสร้างกลุ่มข้อมูลหน่วยในฟีเจอร์สเปซจะขึ้นอยู่กับลำดับของการอ่านจุดข้อมูล ดังนั้นถ้าลำดับของจุดข้อมูลที่อ่านเข้ามาเปลี่ยนไป การจัดเรียงตัวและจำนวนของกลุ่มข้อมูลหน่วยที่ได้ก็จะเปลี่ยนตามไปด้วย

3.1.4 ข้อมูลรบกวน

ข้อมูลรบกวน (Noise) หมายถึงจุดข้อมูลที่เป็นสมาชิกของกลุ่มข้อมูลหน่วยที่มีจำนวนสมาชิกน้อยกว่าค่า MinPts นั่นก็คือ เราจะถือว่ากลุ่มข้อมูลหน่วยที่มีจำนวนสมาชิกน้อยกว่าค่า MinPts เป็นกลุ่มข้อมูลหน่วยของข้อมูลรบกวน ซึ่งถ้ากลุ่มข้อมูลหน่วยไหนเป็นกลุ่มข้อมูลหน่วยของข้อมูลรบกวน กลุ่มข้อมูลหน่วยนั้นก็จะไม่ถูกนำไปใช้ในการรวมกลุ่มข้อมูล ตัวอย่างในรูปที่ 3.5 กลุ่มข้อมูลหน่วย C ถือว่าเป็นกลุ่มข้อมูลหน่วยของข้อมูลรบกวนเพราะมีจำนวนจุดข้อมูลที่เป็นสมาชิกน้อยกว่า MinPts



รูปที่ 3.5 แสดงกลุ่มข้อมูลหน่วยที่เป็นข้อมูลรบกวน

3.1.5 อัตราส่วนคลาส

อัตราส่วนคลาส คือ เวคเตอร์ของอัตราส่วนระหว่างจำนวนจุดข้อมูลของแต่ละคลาสต่อจำนวนจุดข้อมูลที่เป็นสมาชิกของกลุ่มข้อมูลนั้นทั้งหมด อัตราส่วนคลาสของกลุ่มข้อมูล c เขียนแทนโดย

$$CR(c) = [r_1, r_2, \ldots, r_m], \quad r_i = \frac{n_i}{\sum_{j=1}^{m} n_j} \tag{3.1}$$

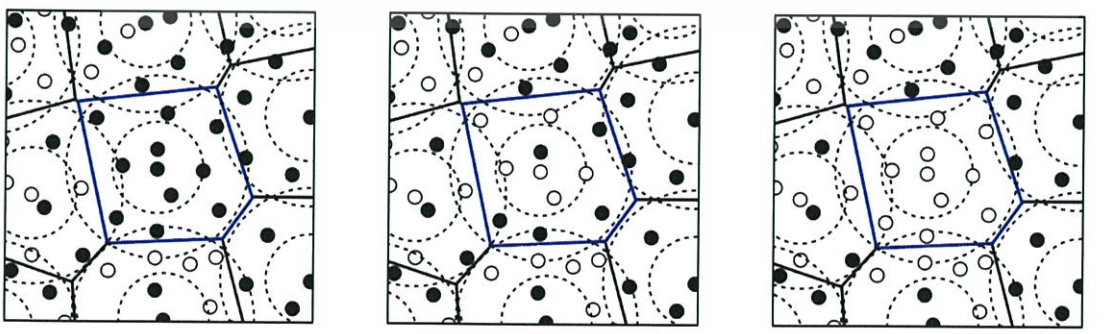
โดยที่

r_i คือ อัตราส่วนคลาสของคลาส i

n_i คือ จำนวนจุดข้อมูลที่เป็นคลาส i

m คือ จำนวนคลาสทั้งหมด

อัตราส่วนคลาสของกลุ่มข้อมูลหน่วยเป็นค่าที่บอกว่าสัดส่วนของการปนกันของข้อมูลของแต่ละคลาสในกลุ่มข้อมูลหน่วยนั้นเป็นเท่าไร ในการรวมกลุ่มข้อมูลเราจะนำอัตราส่วนคลาสของกลุ่มข้อมูลหน่วยที่อยู่ใกล้กันมาเปรียบเทียบกันว่ามีความเหมือนหรือความแตกต่างกันเท่าไร การที่กลุ่มข้อมูลหน่วยที่อยู่ใกล้กัน 2 กลุ่มมีอัตราส่วนคลาสเหมือนกันหมายความว่ากลุ่มข้อมูล 2 กลุ่มนั้นเป็นกลุ่มเดียวกันดังนั้นจึงสามารถรวมเข้าด้วยกันได้ รูปที่ 3.6 แสดงอัตราส่วนคลาสของกลุ่มข้อมูลหน่วยเมื่อลักษณะการปนกันของข้อมูลแตกต่างกัน



(a) CR = [1, 0] (b) CR = [5/11, 6/11] (c) CR = [0, 1]

รูปที่ 3.6 แสดงอัตราส่วนคลาสของกลุ่มข้อมูลหน่วย

3.1.6 ความต่างของอัตราส่วนคลาส

ความต่างของอัตราส่วนคลาส คือค่าที่ใช้บอกถึงความต่างระหว่างอัตราส่วนคลาสของกลุ่มข้อมูล 2 กลุ่ม โดยค่าความต่างระหว่างอัตราส่วนคลาสของกลุ่มข้อมูล c และ d เขียนแทนโดย

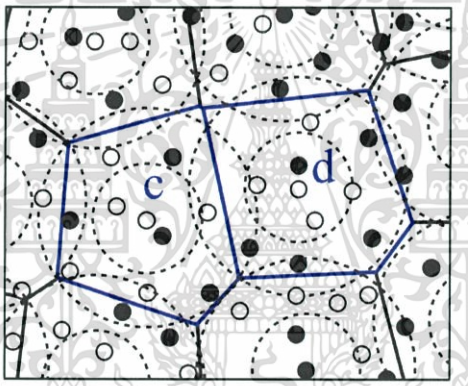
$$DCR(c,d) = \left|CR(c) - CR(d)\right| = \sum_{i=1}^{m} \left|r_i^c - r_i^d\right| \tag{3.2}$$

โดยที่

r_i^c , r_i^d คือ อัตราส่วนคลาสของคลาส i ของกลุ่มข้อมูล c และ d ตามลำดับ

m คือ จำนวนคลาสทั้งหมด

ถ้ากลุ่มข้อมูลหน่วย 2 กลุ่มใดมีค่าความต่างของอัตราส่วนคลาสต่ำก็แสดงว่าสัดส่วนการปนกันของข้อมูลภายในกลุ่มข้อมูล 2 กลุ่มนั้นมีความคล้ายคลึงกันมาก แต่ถ้ามีค่าความต่างของอัตราส่วนคลาสสูงก็แสดงว่าสัดส่วนการปนกันของข้อมูลภายในกลุ่มข้อมูล 2 กลุ่มนั้นมีความแตกต่างกันมาก



รูปที่ 3.7 แสดงตัวอย่างกลุ่มข้อมูลหน่วยที่ใช้ในการหาความต่างของอัตราส่วนคลาส

ต่อไปนี้จะขอยกตัวอย่างการหาความต่างของอัตราส่วนคลาสโดยใช้รูปที่ 3.7 ซึ่งจากรูปเราจะได้อัตราส่วนคลาสของกลุ่มข้อมูลหน่วย c และ d เป็น CR(c) = [4/10, 6/10] และ CR(d) = [5/10, 5/10] ตามลำดับ ดังนั้นเราสามารถหาความต่างของอัตราส่วนคลาสของ c กับ d ได้ดังนี้

$$\begin{aligned} DCR(c,d) &= |4/10 - 5/10| + |6/10 - 5/10| \\ &= 0.1 + 0.1 \\ &= 0.2 \end{aligned}$$

เมื่อพิจารณาความต่างของอัตราส่วนคลาสของกลุ่มข้อมูลหน่วย c กับ d จะพบว่าค่าความต่างมีค่าต่ำ ซึ่งนั่นก็หมายความว่าอัตราการปนกันของข้อมูลในกลุ่มข้อมูลหน่วยทั้งสองมีความคล้ายคลึงกัน

3.1.7 การรวมกลุ่มข้อมูลหน่วย

ในวิทยานิพนธ์ของการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริด [6] ได้มีการนำเสนอวิธีการรวมกริดไว้หลายวิธีอยู่แล้วดังนั้นวิทยานิพนธ์ฉบับนี้จึงไม่เสนอวิธีการใหม่ขึ้นมาอีกแต่จะขอยืมวิธีการรวมกริดนั้นมาใช้เลย วิธีการรวมกริดที่ยืมมานั้นก็คือการรวมกริดแบบ NGM Type 1 ซึ่งเป็นการ

รวมกริดใด ๆ กับกริดข้างเคียงที่มีความต่างของอัตราส่วนคลาสน้อยที่สุดเพียงกริดเดียว โดยเหตุผลที่เลือกใช้การรวมกริดวิธีนี้ก็คือ การรวมกริดแบบ NGM Type 1 ให้ผลการรวมกลุ่มที่ดีกว่าวิธีการอื่นอีกทั้งยังใช้เวลาประมวลผลไม่มากจนเกินไปด้วย

อัลกอริธึมสำหรับการรวมกลุ่มข้อมูลหน่วยแบบ NGM Type 1 เป็นดังอัลกอริธึมที่ 3.2

อัลกอริธึมที่ 3.2 อัลกอริธึมการรวมกลุ่มข้อมูลหน่วย

```
1.  ClusterList = a set of unit clusters
2.  canMergeFlag = TRUE
3.  REPEAT
4.     IF ClusterList[0].canMergeFlag is equal to TRUE THEN
5.        Find the neighbor with smallest DCR from ClusterList and
             assign bestNeighbor = ClusterList[0]'s neighbor with
             smallest DCR, dcr = smallest DCR
6.        IF bestNeighbor is equal to NULL THEN
7.           ClusterList[0].canMergeFlag = FALSE
8.        ELSE
9.           IF dcr is not greater than MAXDIFF THEN
10.             Merge ClusterList[0] with bestNeighbor
11.             Move ClusterList[0] to ClusterList's back
12.          ELSE
13.             ClusterList[0].canMergeFlag = FALSE
14.          END IF
15.       END IF
16.    ELSE
17.       Find a cluster which can be merged from ClusterList and
             assign it to canMerge
18.       IF canMerge is not the last element of ClusterList THEN
19.          Move clusters which are in front of canMerge to
                ClusterList's back
20.       ELSE
21.          IF the last element of ClusterList can be merged THEN
22.             Move clusters which are in front of the last element
                   of ClusterList to ClusterList's back
23.          ELSE
24.             canMergeFlag = FALSE
25.          ENDIF
26.       END IF
27.    END IF
28. UNTIL canMergeFlag is equal to FALSE
```

3.1.8 การวัดประสิทธิภาพของการจัดกลุ่มข้อมูล

ในการวัดความถูกต้องของการจัดกลุ่มข้อมูล ถ้าข้อมูลมีจำนวนมิติน้อยๆ เช่น 2 หรือ 3 มิติ เราสามารถนำผลการจัดกลุ่มที่ได้มาแสดงเป็นภาพแล้วตรวจสอบด้วยตาได้ แต่ในกรณีที่ข้อมูลมีจำนวนมิติมากๆ เราจะไม่สามารถนำผลการจัดกลุ่มมาแสดงเป็นรูปภาพได้ ดังนั้นจึงจำเป็นต้องมีวิธีการวัดที่ให้ผลออกมาเป็นตัวเลขซึ่งง่ายต่อการตรวจสอบ ซึ่งในการวัดประสิทธิภาพของการจัดกลุ่มข้อมูลโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลมนี้เราจะนำค่าสัมประสิทธิ์ประสิทธิภาพของการจัดกลุ่ม (Coefficient of Clustering Efficiency: CCE) จากงานวิจัย [5] มาปรับเปลี่ยนใหม่

ค่าสัมประสิทธิ์ประสิทธิภาพของการจัดกลุ่มข้อมูลโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลมหาได้จากสมการที่ (3.3)

$$CCE = k \times SSE_{norm} \times Eps \tag{3.3}$$

โดยที่

k คือ จำนวนกลุ่มข้อมูลที่ได้จากการจัดกลุ่ม

SSE_{norm} คือ Normalize of Sum Square Error ของการจัดกลุ่ม

Normalize of Sum Square Error คือค่าเฉลี่ยของผลรวมความผิดพลาดของอัตราส่วนคลาสของแต่ละกลุ่มข้อมูลหน่วยกับอัตราส่วนคลาสของกลุ่มข้อมูลที่กลุ่มข้อมูลหน่วยนั้นเป็นสมาชิก SSE_{norm} หาได้จากสมการที่ (3.4)

$$SSE_{norm} = \frac{\sum_{i=1}^{k} \sum_{x \in C_i} DCR(x, C_i)^2}{N} \tag{3.4}$$

โดยที่

k คือ จำนวนกลุ่มข้อมูลที่ได้จากการจัดกลุ่ม

C คือ กลุ่มข้อมูล

N คือ จำนวนกลุ่มข้อมูลหน่วยทั้งหมด

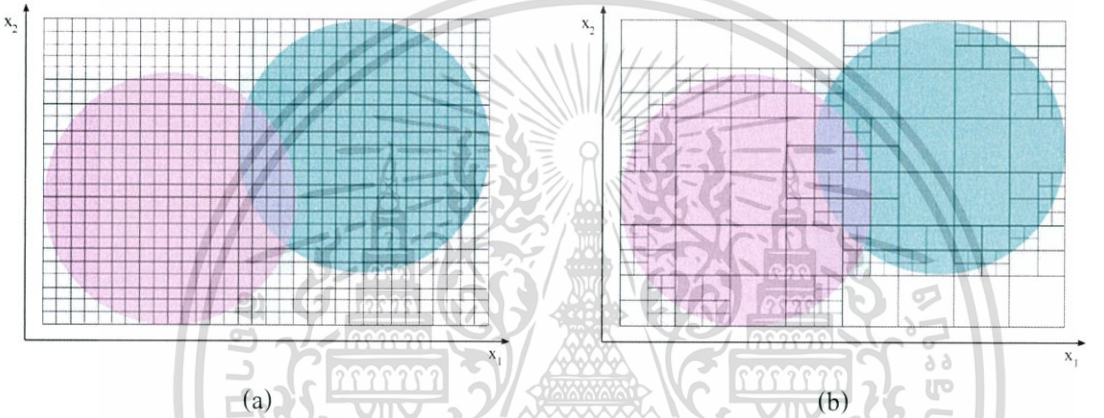
x คือ กลุ่มข้อมูลหน่วย

3.1.9 วิเคราะห์การจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลม

จากการศึกษาและพัฒนาการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลมเราได้พบว่าการใช้กลุ่มข้อมูลหน่วยแบบวงกลมนี้ทำให้เวลาที่ใช้ในการประมวลผลเพิ่มขึ้น ทั้งนี้ก็เนื่องมาจากในขั้นตอนการสร้างกลุ่มข้อมูลหน่วยแบบวงกลมซึ่งมีขอบเขตที่ไม่แน่นอนเหมือนกับขอบเขตของกริด จุดข้อมูลจะต้องถูกนำไปเทียบกับกลุ่มข้อมูลหน่วยเกือบทั้งหมด ต่างจากการใช้กริดที่จุดข้อมูลจะไม่ถูกนำไปเทียบกับกริดที่มีอยู่ทั้งหมดเพียงแค่นำพิกัดของจุดข้อมูลมาเทียบค่า Eps เราก็ทราบได้ทันทีว่าจุดข้อมูลนั้นอยู่ในกริดไหน แต่อย่างไรก็ตามเราก็ได้ข้อน่าสังเกตว่าเวลาที่ใช้ในการรวมกลุ่มข้อมูลหน่วยนั้นมักจะมากกว่าเวลาที่ใช้ในการสร้างกลุ่มข้อมูลหน่วยและจำนวนกลุ่มข้อมูลหน่วยก็มีผลโดยตรงกับเวลาที่ใช้ในการรวมกลุ่มข้อมูล ดังนั้นจึงเกิดแนวคิดขึ้นมาว่าถ้าเราใช้กลุ่มข้อมูลหน่วยที่มีขนาดแตกต่างกันในแต่ละพื้นที่ของสเปซก็จะทำให้จำนวนกลุ่มข้อมูลหน่วยลดลงและน่าจะทำให้เวลาที่ใช้ในการรวมกลุ่มข้อมูลลดลงด้วย แต่โชคไม่ดีที่การใช้กลุ่มข้อมูลหน่วยแบบวงกลมซึ่งมีขอบเขตที่ไม่แน่นอนทำให้การใช้กลุ่มข้อมูลหน่วยที่มีขนาดแตกต่างกันทำได้ลำบาก ดังนั้นจึงได้ยุติการพัฒนาการใช้กลุ่มข้อมูลหน่วยแบบวงกลมไว้เพียงเท่านี้และได้คิดวิธีใหม่ขึ้นมาซึ่งจะได้นำเสนอในหัวข้อถัดไป

3.2 การจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริดที่มีความละเอียดหลายระดับ

หัวข้อนี้จะนำเสนอวิธีการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยการใช้กริดที่มีขนาดต่างกันในแต่ละพื้นที่ของสเปซ สาเหตุที่กลับมาใช้กริดก็เพราะว่ากริดมีขอบเขตที่แน่นอน เราสามารถแตกกริดลงไปได้อีกโดยที่ไม่มีผลกระทบกับกริดข้างเคียง จากรูปที่ 3.8 เราจะเห็นได้ว่าการใช้กริดที่มีขนาดแตกต่างต่างกันตามแต่ละพื้นที่ของสเปซ (b) จะทำให้จำนวนของกริดทั้งหมดมีน้อยกว่าการใช้กริดที่มีขนาดเท่ากันหมด (a) ซึ่งการที่จำนวนกริดมีน้อยกว่านี้จะส่งผลให้เวลาที่ใช้ในการรวมกลุ่มข้อมูลและปริมาณหน่วยความจำที่ใช้ลดลงตามไปด้วย

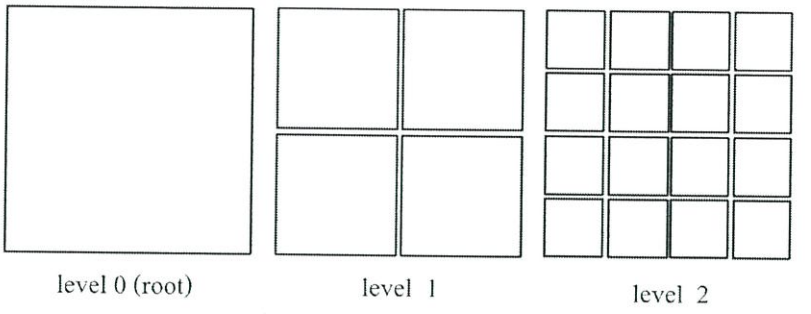


รูปที่ 3.8 แสดงการเปรียบเทียบการแบ่งสเปซด้วยกริดระดับเดียวกับกริดหลายระดับ

3.2.1 กริดที่มีความละเอียดหลายระดับ

ในการสร้างกริดที่มีขนาดเท่ากันหมดนั้นกริดทั้งหมดจะถูกสร้างขึ้นมาพร้อมกันทีเดียว แต่ในการสร้างกริดที่มีความละเอียดหลายระดับนี้ กริดจะถูกสร้างเป็นลำดับชั้นแบบบนลงล่าง จากกริดใหญ่ไปหากริดเล็ก โดยเริ่มแรกจะกำหนดให้ทั้งสเปซเป็นกริดระดับที่ 0 หลังจากนั้นก็จะพิจารณาต่อว่ากริดระดับที่ 0 สามารถแตกต่อไปได้อีกหรือไม่ ถ้ากริดระดับที่ 0 สามารถแตกต่อไปได้อีกก็จะพิจารณาอีกว่ากริดลูกของระดับที่ 0 (ระดับที่ 1) สามารถแตกต่อไปได้อีกหรือไม่ ทำอย่างนี้ไปเรื่อย ๆ จนกว่าจะไม่สามารถแตกกริดใดได้อีก และเมื่อการแตกกริดสิ้นสุดลงเราก็จะได้สเปซที่ถูกแบ่งโดยกริดที่มีขนาดต่างต่างกันดังรูปที่ 3.8 (b)

เพื่อความง่ายต่อการศึกษา วิทยานิพนธ์นี้จึงขอสมมติว่าข้อมูลที่นำมาจัดกลุ่มมีความกว้างในแต่ละมิติเท่ากันโดยประมาณหรืออาจแตกต่างกันเล็กน้อย และการแตกกริดจะทำโดยการแบ่งครึ่งในแต่ละมิติ ดังนั้นถ้าข้อมูลมี 2 มิติ ในการแตกกริดแต่ละครั้งก็จะได้กริดลูกจำนวน 4 กริด และถ้าข้อมูลมี 3 มิติก็จะได้กริดลูก จำนวน 8 กริด รูปที่ 3.9 แสดงการแตกกริดในแต่ละระดับเมื่อข้อมูลมี 2 มิติ



รูปที่ 3.9 แสดงกริดที่ระดับต่าง ๆ

3.2.2 กริดของข้อมูลรบกวน

กริดของข้อมูลรบกวนคือกริดที่มีจุดข้อมูลอยู่น้อยเกินไปและจะต้องกำจัดทิ้งไป ซึ่งสำหรับการใช้กริดที่มีความละเอียดหลายระดับนี้ เนื่องจากขนาดของกริดไม่ได้เท่ากันดังนั้นการจะพิจารณาว่ากริดใดเป็นกริดของข้อมูลรบกวนโดยใช้พารามิเตอร์ MinPts เหมือนกับการใช้กริดขนาดเท่ากันหมดจึงทำไม่ได้ ดังนั้นวิทยานิพนธ์นี้จึงใช้ความหนาแน่นของจุดข้อมูลมาเป็นตัวตัดสินแทน โดยความหนาแน่นที่ว่านี้จะเป็นความหนาแน่นต่อหนึ่งหน่วยพื้นที่ซึ่งสามารถหาได้จากการหารจำนวนจุดข้อมูลภายในกริดด้วยพื้นที่ของกริด และการกำหนดว่ากริดใดเป็นกริดของข้อมูลรบกวนก็จะทำโดยนำความหนาแน่นของกริดนั้นมาเทียบกับพารามิเตอร์ MinDens ซึ่งถ้ากริดใดมีความหนาแน่นน้อยกว่าค่า MinDens ก็จะถือว่ากริดนั้นเป็นกริดของข้อมูลรบกวน

3.2.3 อัลกอริธึมการสร้างกริดหลายระดับ

วิทยานิพนธ์นี้จะขอนำเสนอวิธีการสร้างกริดที่มีความละเอียดหลายระดับ 2 วิธี คือ 1) การสร้างกริดโดยพิจารณาการแตกกริดจากจำนวนจุดข้อมูลที่อยู่ภายในกริด และ 2) การสร้างกริดโดยพิจารณาการแตกกริดจากความต่างของอัตราส่วนคลาส

3.2.3.1 การสร้างกริดโดยพิจารณาการแตกกริดจากจำนวนจุดข้อมูลภายในกริด (GS1)

การสร้างกริดวิธีนี้จะทำการแตกกริดโดยใช้พารามิเตอร์ MinFrS และ MinDens มาเป็นตัวตัดสินว่าควรจะแตกกริดหรือไม่ โดยขั้นแรกจะพิจารณาก่อนว่ากริดที่จะแตก g มีความหนาแน่นน้อยกว่าค่า MinDens หรือไม่ ซึ่งถ้ากริด g มีความหนาแน่นน้อยกว่า MinDens ก็แสดงว่ากริด g เป็นกริดของข้อมูลรบกวนและจะต้องตรวจสอบต่อไปว่ากริดลูกของกริด g เป็นกริดของข้อมูลรบกวนด้วยหรือไม่ ถ้ากริดลูกของกริด g กริดของข้อมูลรบกวนหมด กริด g ก็จะไม่ถูกแตก แต่ถ้ามีกริดลูกของกริด g แม้เพียงหนึ่งกริดไม่ได้เป็นกริดของข้อมูลรบกวน กริด g ก็จะถูกแตก แต่ถ้ากริด g มีความหนาแน่นมากกว่าหรือเท่ากับ MinDens ก็แสดงว่ากริด g ไม่ได้เป็นกริดของข้อมูลรบกวนและจะต้องแตก และในกรณีที่กริด g ถูกแตก กริดลูกของ g จะถูกนำมาตรวจสอบว่ามีจำนวนจุดข้อมูลมากกว่าหรือเท่ากับค่า MinFrS หรือไม่ ถ้าใช่ก็นำกริดลูกของกริด g นั้นไปใส่ในคิวเพื่อจะได้ทำการตรวจสอบว่าสามารถแตกได้อีกหรือไม่ต่อไป

อัลกอริธึมของการสร้างกริดโดยพิจารณาการแตกกริดจากจำนวนจุดข้อมูลภายในกริดเป็นดังอัลกอริธึมที่ 3.3

อัลกอริธึมที่ 3.3 อัลกอริธึมการสร้างกริดแบบ GS1

```
1.  v_splitQueue = an empty queue
2.  v_grid = an empty list
3.  Assign all points to be a member of the root grid
4.  Push the root to v_splitQueue's back
5.  REPEAT
6.     Split v_splitQueue[0]
7.     splitFlag = FALSE
8.     IF v_splitQueue[0].density() is less than MINDENS
9.        FOR EACH child c of v_splitQueue[0] DO
10.          IF c.density() is equal to or greater than MINDENS
11.             splitFlag = TRUE
12.             BREAK
13.          END IF
14.       END FOR
15.    ELSE
16.       splitFlag = TRUE
17.    END IF
18.    IF splitFlag is equal to TRUE
19.       FOR EACH child c of v_splitQueue[0] DO
20.          IF c.numMembers() is equal to or greater than MINFRS
21.             Push c to v_splitQueue's back
22.          ELSE
23.             Push c to v_grid's back
24.          END IF
25.       END FOR
26.       v_splitQueue.dequeue()
27.    ELSE
28.       Push v_splitQueue[0] to v_grid's back
29.       v_splitQueue.dequeue()
30.    END IF
31. UNTIL v_splitQueue is empty
```

3.2.3.2 การสร้างกริดโดยพิจารณาการแตกกริดจากความต่างของอัตราส่วนคลาส (GS2)

การสร้างกริดวิธีนี้สามารถแบ่งออกเป็นวิธีย่อยอีก 2 วิธี คือ 1) ทำการแตกกริดโดยพิจารณาจากความต่างของอัตราส่วนคลาสของกริดลูกกับกริดลูก (GS2.1) และ 2) ทำการแตกกริดโดยพิจารณาจากความต่างของอัตราส่วนคลาสของกริดแม่กับกริดลูก (GS2.2) ซึ่งหลักการโดยรวมของการสร้างกริดวิธีนี้ก็จะยังคงคล้ายกับการสร้างกริดโดยพิจารณาการแตกกริดจากจำนวนจุดข้อมูลภายในกริด แต่จะต่างกันตรงที่แทนที่จะพิจารณาการแตกกริดแม่จากความหนาแน่นอย่างเดียวก็จะพิจารณาจากความต่างของอัตราส่วนคลาสด้วย โดยการพิจารณาจากความต่างของอัตราส่วนคลาสจะใช้พารามิเตอร์ MinDiff มาเป็นตัวตัดสิน คือ ถ้ามีกริดลูกอย่างน้อยหนึ่งคู่ (GS2.1) หรือ กริดแม่กับกริดลูก (GS2.2) ที่ความต่างของอัตราส่วนคลาสมากกว่าหรือเท่ากับค่า MinDiff ก็จะแตกกริดแม่นั้น และการพิจารณาว่าจะนำกริดลูกไป

แตกต่อหรือไม่จะไม่ใช้พารามิเตอร์ MinFrS แต่จะใช้พารามิเตอร์ MaxLevel แทน คือ ถ้ากริดลูกอยู่ในระดับเดียวกับ MaxLevel แล้วกริดลูกนั้นก็จะไม่ถูกนำไปใส่ในคิวเพื่อที่จะทำการแตกต่อ

อัลกอริธึมของการสร้างกริดโดยพิจารณาการแตกกริดจากความต่างของอัตราส่วนคลาสของกริดลูกกับกริดลูก (GS2.1) เป็นดังอัลกอริธึมที่ 3.4 และอัลกอริธึมของการสร้างกริดโดยพิจารณาการแตกกริดจากความต่างของอัตราส่วนคลาสของกริดแม่กับกริดลูก (GS2.2) เป็นดังอัลกอริธึมที่ 3.5

อัลกอริธึมที่ 3.4 อัลกอริธึมการสร้างกริดแบบ GS2.1

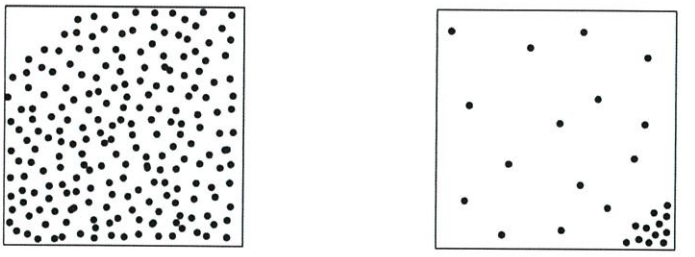
```
1.  v_splitQueue = an empty queue
2.  v_grid = an empty list
3.  Assign all points to be a member of the root grid
4.  Push the root to v_splitQueue's back
5.  REPEAT
6.     Split v_splitQueue[0]
7.     splitFlag = FALSE
8.     IF v_splitQueue[0].density() is less than MINDENS
9.        FOR EACH child c of v_splitQueue[0] DO
10.          IF c.density() is equal to or greater than MINDENS
11.             splitFlag = TRUE
12.             BREAK
13.          END IF
14.       END FOR
15.    ELSE
16.       FOR EACH pair (c1,c2) of childs of v_splitQueue[0] DO
17.          IF c1.density() is less than MINDENS or
                c2.density() is less than MINDENS or
                dcr(c1,c2) is equal to or greater than MINDIFF
18.             splitFlag = TRUE
19.             BREAK
20.          END IF
21.       END FOR
22.    END IF
23.    IF splitFlag is equal to TRUE
24.       FOR EACH child c of v_splitQueue[0] DO
25.          IF c.level() is less than MAXLEVEL
26.             Push c to v_splitQueue's back
27.          ELSE
28.             Push c to v_grid's back
29.          END IF
30.       END FOR
31.       v_splitQueue.dequeue()
32.    ELSE
33.       Push v_splitQueue[0] to v_grid's back
34.       v_splitQueue.dequeue()
35.    END IF
36. UNTIL v_splitQueue is empty
```

อัลกอริธึมที่ 3.5 อัลกอริธึมการสร้างกริดแบบ GS2.2

```
1.  v_splitQueue = an empty queue
2.  v_grid = an empty list
3.  Assign all points to be a member of the root grid
4.  Push the root to v_splitQueue's back
5.  REPEAT
6.     Split v_splitQueue[0]
7.     splitFlag = FALSE
8.     IF v_splitQueue[0].density() is less than MINDENS
9.        FOR EACH child c of v_splitQueue[0] DO
10.          IF c.density() is equal to or greater than MINDENS
11.             splitFlag = TRUE
12.             BREAK
13.          END IF
14.       END FOR
15.    ELSE
16.       FOR EACH child c of v_splitQueue[0] DO
17.          IF c.density() is less than MINDENS or
                dcr(v_splitQueue[0],c) is equal to or greater than
                MINDIFF
18.             splitFlag = TRUE
19.             BREAK
20.          END IF
21.       END FOR
22.    END IF
23.    IF splitFlag is equal to TRUE
24.       FOR EACH child c of v_splitQueue[0] DO
25.          IF c.level() is less than MAXLEVEL
26.             Push c to v_splitQueue's back
27.          ELSE
28.             Push c to v_grid's back
29.          END IF
30.       END FOR
31.       v_splitQueue.dequeue()
32.    ELSE
33.       Push v_splitQueue[0] to v_grid's back
34.       v_splitQueue.dequeue()
35.    END IF
36. UNTIL v_splitQueue is empty
```

3.2.4 การแก้ปัญหาเมื่อกริดขนาดใหญ่ไม่ยอมแตก

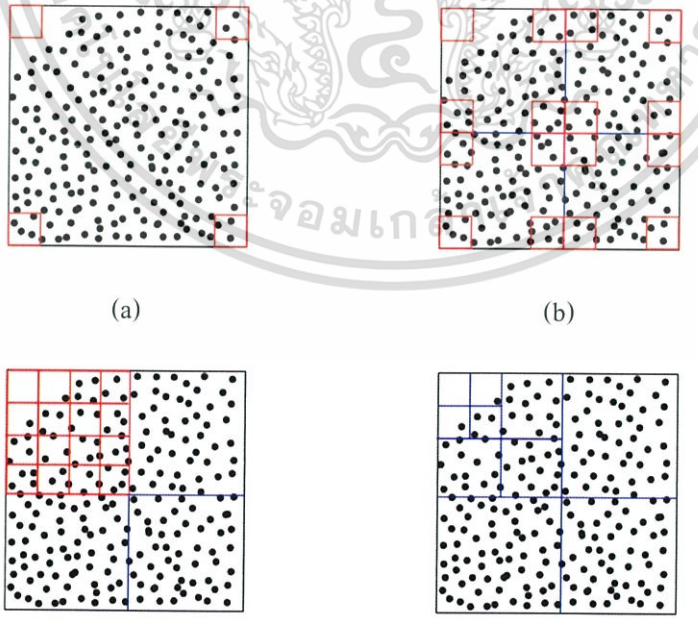
เนื่องจากการสร้างกริดเป็นแบบบนลงล่าง คือ จะทำการแตกกริดลงไปหนึ่งระดับถ้ากริดนั้นสามารถแตกได้ ดังนั้นถ้ากริดที่จะแตกนั้นใหญ่มากและการกระจายตัวหรือลักษณะการซ้อนทับกันของข้อมูลในกริดลูกไม่มีความแตกต่างกัน แต่แตกต่างกันในระดับที่ลึกกว่ากริดลูก กริดที่มีขนาดใหญ่นั้นก็จะไม่ถูกแตกและจะทำให้เกิดความผิดพลาดในการจัดกลุ่มข้อมูล รูปที่ 3.10 แสดงกริดขนาดใหญ่ที่จะไม่ถูกแตกเพราะกริดลูกไม่แตกต่างกัน โดยกริดในรูปที่ 3.10 (a) จะไม่ถูกแตกถ้าเรากำหนดค่า MinDens ไว้ต่ำเกินไป เพราะความหนาแน่นของกริดลูกจะมากกว่าค่า MinDens หมด ส่วนกริดในรูปที่ 3.10 (b) ก็จะไม่ถูกแตกถ้าเรากำหนดค่า MinDens สูงเกินไป เพราะความหนาแน่นของกริดลูกจะน้อยกว่าค่า MinDens หมด



(a) (b)

รูปที่ 3.10 แสดงกริดขนาดใหญ่ที่ไม่ยอมแตก

วิธีแก้ไขปัญหาดังกล่าวที่วิทยานิพนธ์นี้จะนำเสนอก็คือ ต้องหากริดที่มีขนาดเหมาะสมมาสุ่มดูลักษณะการกระจายตัวของข้อมูลภายในกริดใหญ่ว่ามีความแตกต่างกันหรือไม่ ซึ่งถ้าพบว่ามีความแตกต่างกันก็ให้ทำการแตกกริดใหญ่นั้น แต่เราจะทราบได้อย่างไรว่ากริดที่มีขนาดเหมาะสมจะมีขนาดเท่าไรและการนำกริดขนาดเล็กมาสุ่มดูพื้นที่ภายในกริดใหญ่ก็จะต้องทำให้เวลาที่ใช้ในการประมวลผลเพิ่มขึ้นแน่นอน เพราะฉะนั้นวิทยานิพนธ์นี้จึงขอเสนอว่าหลังจากที่สร้างกริดขึ้นมาเสร็จแล้ว ให้ทำการนับจำนวนกริดในแต่ละระดับดูว่ากริดระดับใดมีจำนวนมากที่สุดแล้วจึงถือว่ากริดระดับนั้นเป็นกริดที่น่าจะมีขนาดเหมาะสม และการสุ่มดูลักษณะการกระจายตัวของข้อมูลภายในกริดก็ให้ทำเฉพาะตรงมุมของกริดเท่านั้นพอ



(a) (b)

(c) (d)

รูปที่ 3.11 แสดงการแก้ปัญหาเมื่อกริดขนาดใหญ่ไม่ยอมแตกเพราะกริดลูกไม่แตกต่างกัน

ตัวอย่างในรูปที่ 3.11 (a) ถ้าเรานำกริดที่มีขนาดเหมาะสมมาตรวจดูที่มุมของกริดนั้นเราก็จะพบว่ากริดใหญ่กริดนั้นสมควรที่จะต้องแตก และในรูปที่ 3.11 (b) ถ้าเรานำกริดที่มีขนาดเหมาะสมมาตรวจดูที่มุมของกริดลูกของกริดใหญ่เราก็จะพบว่า มีเพียงกริดบนซ้ายเท่านั้นที่สมควรจะต้องแตก และเมื่อเราทำต่อไปจนถึงระดับ MaxLevel แล้วเราก็จะได้กริดที่มีความเหมาะสมดังรูปที่ 3.11 (d)

3.2.5 การรวมกริดให้เป็นกลุ่มข้อมูล

สำหรับการรวมกริดเข้าด้วยกันให้เป็นกลุ่มข้อมูลนั้นจะใช้วิธีเดียวกันกับการจัดกลุ่มข้อมูลโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลมและจะไม่กล่าวซ้ำในที่นี่อีก ให้ดูที่หัวข้อ 3.1.7

3.2.6 การวัดประสิทธิภาพของการจัดกลุ่มข้อมูล

ในการวัดประสิทธิภาพของการจัดกลุ่มข้อมูลโดยใช้กริดที่มีความละเอียดหลายระดับนี้จะใช้ค่าสัมประสิทธิ์ประสิทธิภาพของการจัดกลุ่มเหมือนการจัดกลุ่มข้อมูลโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลม แต่จะต้องมีการดัดแปลงให้เหมาะสมเสียก่อน

ค่าสัมประสิทธิ์ประสิทธิภาพของการจัดกลุ่มข้อมูลโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลมหาได้จากสมการที่ (3.5)

$$CCE = k \times SSE_{norm} \times EGS \tag{3.5}$$

โดยที่

k คือ จำนวนกลุ่มข้อมูลที่ได้จากการจัดกลุ่ม

SSE_{norm} คือ Normalize of Sum Square Error ของการจัดกลุ่ม

EGS คือ ความกว้างของกริดที่มีจำนวนมากที่สุด

Normalize of Sum Square Error คือค่าเฉลี่ยของผลรวมความผิดพลาดของอัตราส่วนคลาสของแต่ละกริดกับอัตราส่วนคลาสของกลุ่มข้อมูลที่กริดนั้นเป็นสมาชิก โดยในที่นี้กริดที่มีขนาดใหญ่กว่ากริดที่มีจำนวนมากที่สุดจะต้องถูกทำโทษทั้งนี้ก็เพื่อไม่ให้มีความลำเอียง (bias) เกิดขึ้น SSE_{norm} หาได้จากสมการที่ (3.6)

$$SSE_{norm} = \frac{\sum_{i=1}^{k}\sum_{x\in C_i} N_x^2 \times DCR(x,C_i)^2}{\sum_{i=1}^{k}\sum_{x\in C_i} N_x} \quad ; \quad N_x = \begin{cases} 1 & \text{if } L_x \geq L_{\max} \\ 2^{d(L_{\max}-L_x)} & \text{if } L_x < L_{\max} \end{cases} \tag{3.6}$$

โดยที่

k	คือ จำนวนกลุ่มข้อมูลที่ได้จากการจัดกลุ่ม
C	คือ กลุ่มข้อมูล
L_x	คือ ระดับของกริด x
$L_{\max}$	คือ ระดับของกริดส่วนใหญ่
x	คือ กริด
d	คือ จำนวนมิติของข้อมูล

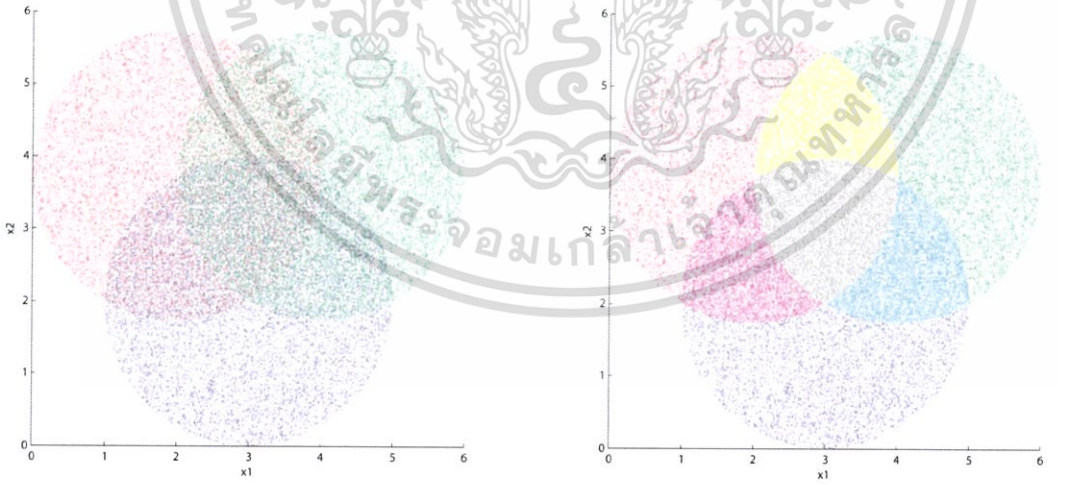
บทที่ 4
การทดลองจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลม

บทนี้เป็นการนำเสนอผลการทดลองจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลม โดยตอนแรกจะเป็นการทดลองปรับค่าพารามิเตอร์ต่างๆ เพื่อดูว่ามีผลอย่างไรบ้างต่อการจัดกลุ่มข้อมูล จากนั้นก็จะเป็นการทดลองเปรียบเทียบการจัดกลุ่มข้อมูลเทียบกับการใช้กริดที่มีขนาดเท่ากันหมด

4.1 ชุดข้อมูลที่ใช้ในการทดลอง

4.1.1 ข้อมูลที่มีการกระจายอย่างสม่ำเสมอ

ชุดข้อมูลที่มีการกระจายอย่างสม่ำเสมอที่ใช้ทดลองในวิทยานิพนธ์นี้เป็นชุดข้อมูลที่สังเคราะห์ขึ้นโดยใช้โปรแกรม MATLAB จุดข้อมูลมีทั้งหมด 91,725 จุด แต่ละจุดเกิดจากการเรียกใช้ฟังก์ชัน unifrnd มีคลาสทั้งหมด 3 คลาส แต่ละคลาสมีจุดข้อมูล 30,575 จุด แยกออกเป็นข้อมูลรบกวนเสีย 575 จุด และลักษณะการจัดเรียงตัวของจุดข้อมูลเป็นดังแสดงในรูปที่ 4.1 (a)



(a) (b)

รูปที่ 4.1 แสดงชุดข้อมูลที่มีการกระจายอย่างสม่ำเสมอ

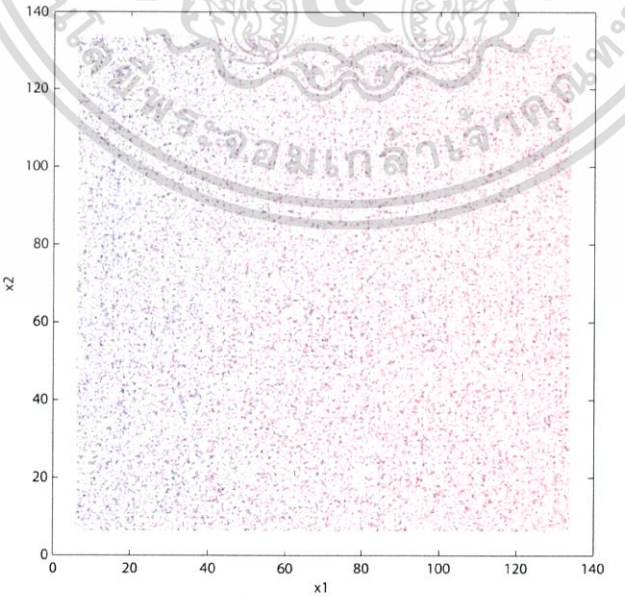
เมื่อพิจารณาการจัดเรียงตัวของจุดข้อมูลในรูปที่ 4.1 (a) แล้วสามารถเดาผลการจัดกลุ่มได้ทันทีว่าต้องแบ่งออกเป็น 7 กลุ่ม ดังแสดงในรูปที่ 4.1 (b) ซึ่งในรูปที่ 4.1 (b) ข้อมูลรบกวนทั้ง 1,725 จุดได้ถูกกำจัดออกไปแล้ว และในการวัดประสิทธิภาพของการจัดกลุ่มของชุดข้อมูลชุดนี้เราจะนำผลการจัดกลุ่ม

ที่ได้มาเปรียบเทียบกับผลการจัดกลุ่มที่คาดหวังที่แสดงในรูปที่ 4.1 (b) ด้วย โดยการวัดจะดูจากจำนวนจุดที่ถูกจัดกลุ่มผิด ซึ่งจุดที่ถูกจัดกลุ่มผิดก็จะประกอบไปด้วย

1. จุดที่เป็นข้อมูลรบกวนแต่ถูกกำจัดออกไม่หมดหรือถูกจัดเป็นสมาชิกของกลุ่มข้อมูลกลุ่มใดกลุ่มหนึ่ง
2. จุดที่ควรจะถูกระบุให้เป็นสมาชิกของกลุ่มข้อมูลกลุ่มใดกลุ่มหนึ่งแต่ถูกจัดเป็นข้อมูลรบกวนเพราะอยู่ในกลุ่มข้อมูลหน่วยหรือกริดที่มีความหนาแน่นไม่พอ
3. จุดข้อมูลที่ถูกระบุให้เป็นสมาชิกของกลุ่มข้อมูลที่ไม่ใช่กลุ่มข้อมูลที่ควรจะเป็น

4.1.2 ข้อมูลที่มีการกระจายลดลงอย่างสม่ำเสมอ

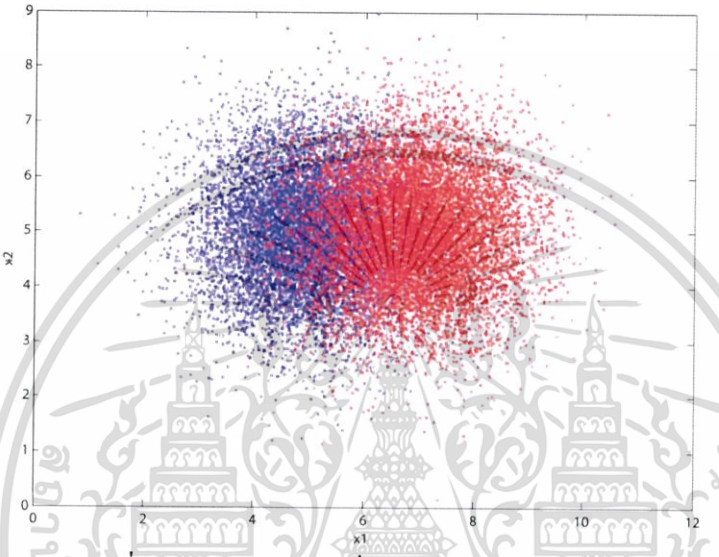
ชุดข้อมูลที่มีการกระจายลดลงอย่างสม่ำเสมอที่ใช้ทดลองในวิทยานิพนธ์นี้เป็นชุดข้อมูลที่สังเคราะห์ขึ้นโดยใช้โปรแกรม MATLAB จุดข้อมูลมีทั้งหมด 72,000 จุด แต่ละจุดเกิดจากการเรียกใช้ฟังก์ชัน unifrnd มีคลาสทั้งหมด 2 คลาส แต่ละคลาสมีจุดข้อมูล 36,000 จุด การกระจายตัวของข้อมูลจะแบ่งเป็นการกระจายแบบสม่ำเสมอจำนวน 8 ระดับซึ่งความกว้างของแต่ละระดับจะเท่ากันหมด โดยระดับแรกจะประกอบไปด้วยจุดข้อมูลสีน้ำเงินจำนวน 8,000 จุด และจุดข้อมูลสีแดงจำนวน 1,000 จุด ส่วนในระดับที่ 2 จุดข้อมูลสีน้ำเงินจะมี 7,000 จุด จุดข้อมูลสีแดงมีจำนวน 2,000 จุด และในระดับถัดไปก็จะมีลักษณะการเพิ่มขึ้นและลดลงของจุดข้อมูลเหมือนเช่นเดียวกันนี้ จนถึงระดับที่ 8 จุดข้อมูลสีน้ำเงินจะลดลงเหลือ 1,000 จุด ส่วนจุดข้อมูลสีแดงจะเพิ่มขึ้นเป็น 8,000 จุด ลักษณะข้อข้อมูลชุดนี้เป็นดังแสดงในรูปที่ 4.2



รูปที่ 4.2 แสดงชุดข้อมูลที่มีการกระจายลดลงอย่างสม่ำเสมอ

4.1.3 ข้อมูลที่มีการกระจายแบบปกติ

ชุดข้อมูลที่มีการกระจายแบบปกติที่ใช้ทดลองในวิทยานิพนธ์นี้เป็นชุดข้อมูลที่สังเคราะห์ขึ้นโดยใช้โปรแกรม MATLAB จุดข้อมูลมีทั้งหมด 20,000 จุด แต่ละจุดเกิดจากการเรียกใช้ฟังก์ชัน mvnrnd โดยกำหนดค่า sigma เป็น 1 ทั้งแกน x และ y จุดศูนย์กลางของกลุ่มข้อมูลอยู่ที่ (5,5) และ (7,5) มีคลาสทั้งหมด 2 คลาส แต่ละคลาสมีจุดข้อมูล 10,000 จุด



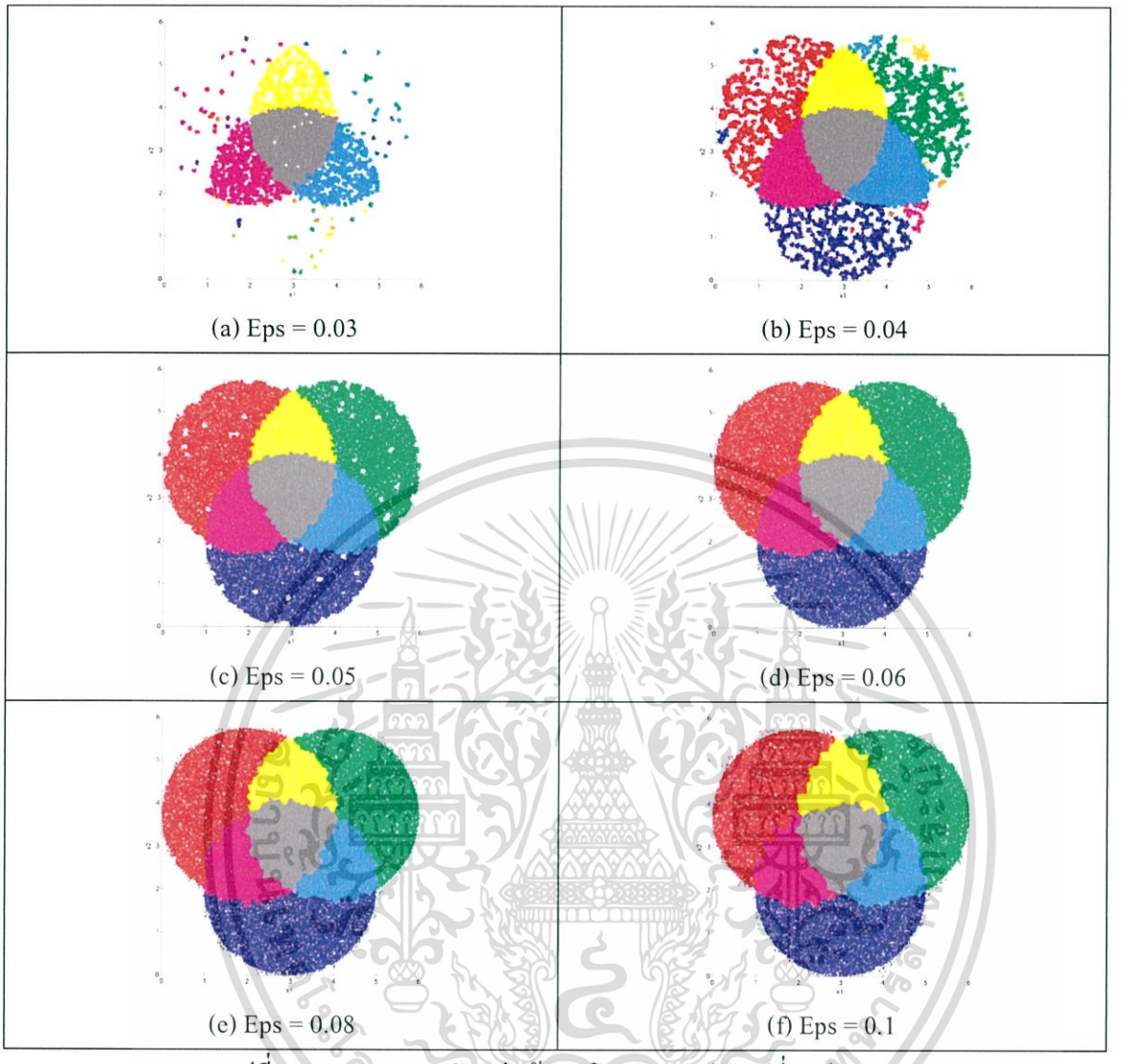
รูปที่ 4.3 แสดงชุดข้อมูลที่มีการกระจายแบบปกติ

4.2 การทดลองปรับค่าพารามิเตอร์ต่าง ๆ

การทดลองนี้จะเป็นการปรับค่าพารามิเตอร์ต่าง ๆ ของการจัดกลุ่มข้อมูลโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลม ซึ่งก็ได้แก่ค่า Eps MinPts และค่า MaxDiff เพื่อศึกษาดูว่าพารามิเตอร์แต่ละตัวมีความสัมพันธ์กันอย่างไรและมีผลต่อการจัดกลุ่มข้อมูลอย่างไรบ้าง โดยจะใช้ข้อมูลที่มีการกระจายอย่างสม่ำเสมอจากหัวข้อที่ 4.1.1 มาทำการทดลอง

4.2.1 การทดลองปรับเปลี่ยนค่า Eps

การทดลองนี้เป็นการปรับเปลี่ยนค่า Eps โดยที่กำหนดให้ค่า MinPts และ MaxDiff เป็นค่าคงที่ คือ จะทำการกำหนดค่า Eps เป็น 0.03 0.04 0.05 0.06 0.08 และ 0.1 ตามลำดับ ส่วนค่า MinPts และ MaxDiff กำหนดให้เท่ากับ 25 และ 0.55 ตามลำดับ ซึ่งผลการจัดกลุ่มเป็นดังแสดงในรูปที่ 4.4 และตารางที่ 4.1



(a) Eps = 0.03 (b) Eps = 0.04

(c) Eps = 0.05 (d) Eps = 0.06

(e) Eps = 0.08 (f) Eps = 0.1

รูปที่ 4.4 แสดงผลการจัดกลุ่มข้อมูลโดยทำการปรับเปลี่ยนค่า Eps

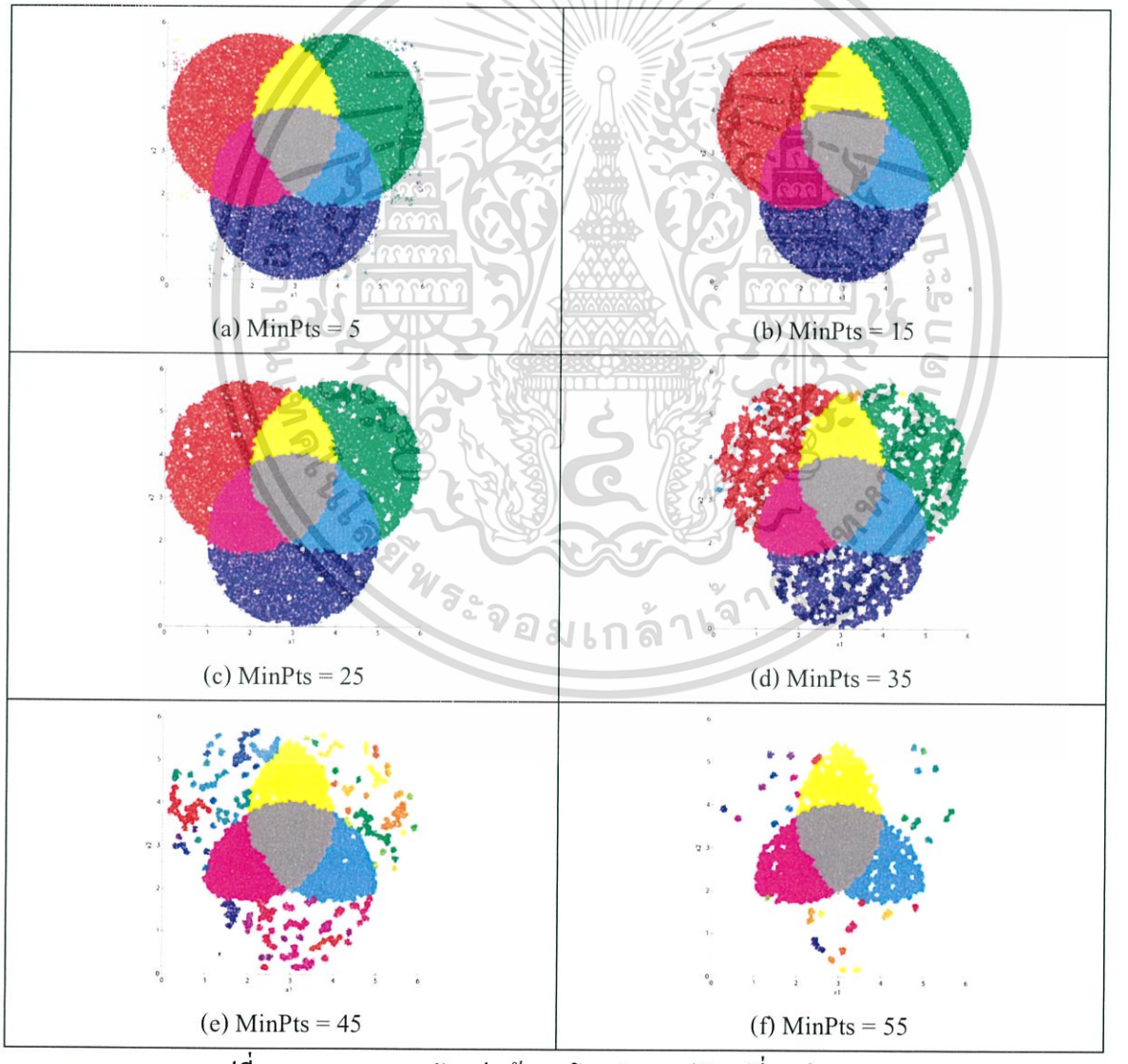
ตารางที่ 4.1 เปรียบเทียบผลการจัดกลุ่มข้อมูลโดยปรับเปลี่ยนขนาด Eps

Eps	จำนวนกลุ่มข้อมูล	SSE	SSE_{Norm}	CCE	Error (จุด)	Error (%)
0.03	75	50.1060	0.0106	0.0007	56908	62.04
0.04	23	33.1368	0.0116	0.0004	20360	22.20
0.05	7	23.4837	0.0123	0.0002	4133	4.50
0.06	7	13.4700	0.0100	0.0003	3237	3.53
0.08	7	7.6996	0.0097	0.0004	3969	4.33
0.1	7	4.6612	0.0090	0.0006	4588	5.00

จากผลการทดลองจะเห็นได้ว่าถ้าเราเลือกขนาด Eps ที่เล็กเกินไปแล้วกำหนดค่า MinPts ไม่เหมาะสม ก็จะมีกลุ่มข้อมูลหน่วยจำนวนมากที่ถูกกำหนดให้เป็นข้อมูลรบกวน ดังนั้นถ้าจะเลือกขนาด Eps เล็กก็จะต้องกำหนดค่า MinPts ให้น้อยตามไปด้วย และในกรณีที่เลือกขนาด Eps ใหญ่เกินไป ความผิดพลาดก็จะมีมากขึ้นตามลำดับ

4.2.2 การทดลองปรับเปลี่ยนค่า MinPts

การทดลองนี้เป็นการปรับเปลี่ยนค่า MinPts โดยที่กำหนดให้ค่า Eps และ MaxDiff เป็นค่าคงที่ คือ จะทำการกำหนดค่า MinPts เป็น 5 15 25 35 45 และ 55 ตามลำดับ ส่วนค่า Eps และ MaxDiff กำหนดให้เท่ากับ 0.05 และ 0.5 ตามลำดับ ซึ่งผลการจัดกลุ่มเป็นดังแสดงในรูปที่ 4.5 และตารางที่ 4.2



รูปที่ 4.5 แสดงผลการจัดกลุ่มข้อมูลโดยทำการปรับเปลี่ยนค่า MinPts

ตารางที่ 4.2 เปรียบเทียบผลการจัดกลุ่มข้อมูลโดยปรับเปลี่ยนค่า MinPts

MinPts	จำนวนกลุ่มข้อมูล	SSE	SSE_{Norm}	CCE	Error (จุด)	Error (%)
5	27	24.3724	0.0127	0.0172	3030	3.30
15	7	23.6000	0.0123	0.0043	2798	3.05
25	7	23.4837	0.0123	0.0043	4133	4.51
35	12	20.7702	0.0109	0.0065	14243	15.53
45	80	14.3460	0.0075	0.0300	40581	44.24
55	35	10.7276	0.0056	0.0098	52422	57.15

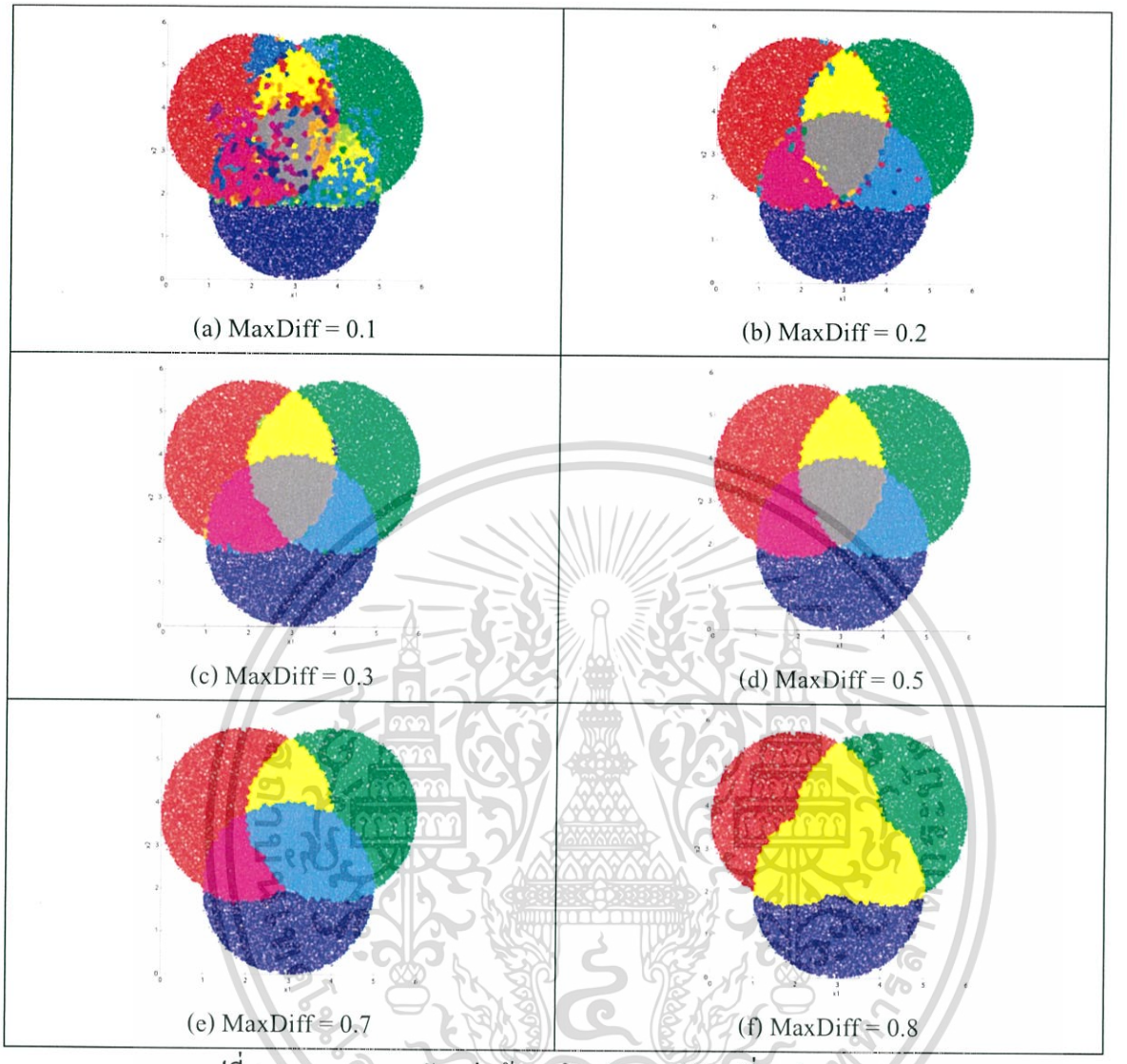
จากการทดลองก่อนหน้านี้เราได้พบว่าขนาด Eps มีความสัมพันธ์กับค่า MinPts นั้นถ้าเรากำหนดให้ขนาด Eps เป็นค่าคงที่ แล้วปรับเปลี่ยนค่า MinPts เมื่อ MinPts มีค่าน้อยเกินไป กลุ่มข้อมูลหน่วยที่มีความหนาแน่นน้อยก็จะไม่ถูกกำหนดให้เป็นข้อมูลรบกวน ดังเช่นในรูปที่ 4.5 (a) ที่มีข้อมูลรบกวนเป็นจำนวนมากไม่ถูกกำจัดทิ้งไป แต่ถ้ากำหนดค่า MinPts มากเกินไปกลุ่มข้อมูลหน่วยที่ไม่ใช่ข้อมูลรบกวนก็จะถูกกำจัดทิ้งไปด้วย

4.2.3 การทดลองปรับเปลี่ยนค่า MaxDiff

การทดลองนี้เป็นการปรับเปลี่ยนค่า MaxDiff โดยที่กำหนดให้ค่า Eps และ MinPts เป็นค่าคงที่ คือ จะทำการกำหนดค่า MaxDiff เป็น 0.1 0.2 0.3 0.5 0.7 และ 0.8 ตามลำดับ ส่วนค่า Eps และ MinPts กำหนดให้เท่ากับ 0.05 และ 15 ตามลำดับ ซึ่งผลการจัดกลุ่มเป็นดังแสดงในรูปที่ 4.6 และตารางที่ 4.3

ตารางที่ 4.3 เปรียบเทียบผลการจัดกลุ่มข้อมูลโดยปรับเปลี่ยนค่า MaxDiff

MaxDiff	จำนวนกลุ่มข้อมูล	SSE	SSE_{Norm}	CCE	Error (จุด)	Error (%)
0.1	199	2.6160	0.0014	0.0136	34934	38.09
0.2	49	13.6721	0.0071	0.0175	5632	6.14
0.3	20	18.0269	0.0094	0.0094	3513	3.83
0.5	7	23.6000	0.0123	0.0043	2798	3.05
0.7	6	56.9857	0.0298	0.0089	22038	24.03
0.8	4	191.0270	0.0998	0.0200	41686	45.45



รูปที่ 4.6 แสดงผลการจัดกลุ่มข้อมูลโดยทำการปรับเปลี่ยนค่า MaxDiff

MaxDiff เป็นพารามิเตอร์สำหรับกำหนดความต่างของอัตราส่วนคลาสสูงสุดที่จะยังถือว่ากลุ่มข้อมูลหน่วยที่นำมาเปรียบเทียบกันนั้นสามารถรวมเข้าด้วยกันได้อยู่ ดังนั้นถ้าเราเลือกค่า MaxDiff ต่ำเกินไปกลุ่มข้อมูลหน่วยที่มีความต่างของอัตราส่วนคลาสปานกลางถึงมากก็จะไม่ถูกรวมเข้าด้วยกันดังตัวอย่างในรูปที่ 4.6 (a) และ (b) แต่ถ้าเลือกค่าสูงเกินไปกลุ่มข้อมูลหน่วยที่มีความต่างของอัตราส่วนคลาสน้อยถึงปานกลางก็จะถูกรวมเข้าด้วยกันดังตัวอย่างในรูปที่ 4.6 (e) และ (f)

4.2.4 สรุปความสัมพันธ์ของพารามิเตอร์ต่าง ๆ

จากการทดลองพบว่าพารามิเตอร์ Eps จะมีความสัมพันธ์กับ MinPts กล่าวคือ ถ้ากำหนดขนาด Eps เล็กก็จะต้องกำหนดค่า MinPts น้อยตามไปด้วย มิเช่นนั้นแล้วกลุ่มข้อมูลหน่วยที่มีจำนวนข้อมูลน้อยกว่าค่า MinPts จะถูกกำหนดให้เป็นกลุ่มข้อมูลหน่วยของข้อมูลรบกวน แต่ถ้าเลือกขนาด Eps ใหญ่ก็จะต้องกำหนดค่า MinPts มากตามไปด้วย มิเช่นนั้นแล้วข้อมูลรบกวนก็จะถูกกำจัดทิ้งไปไม่หมด ส่วน

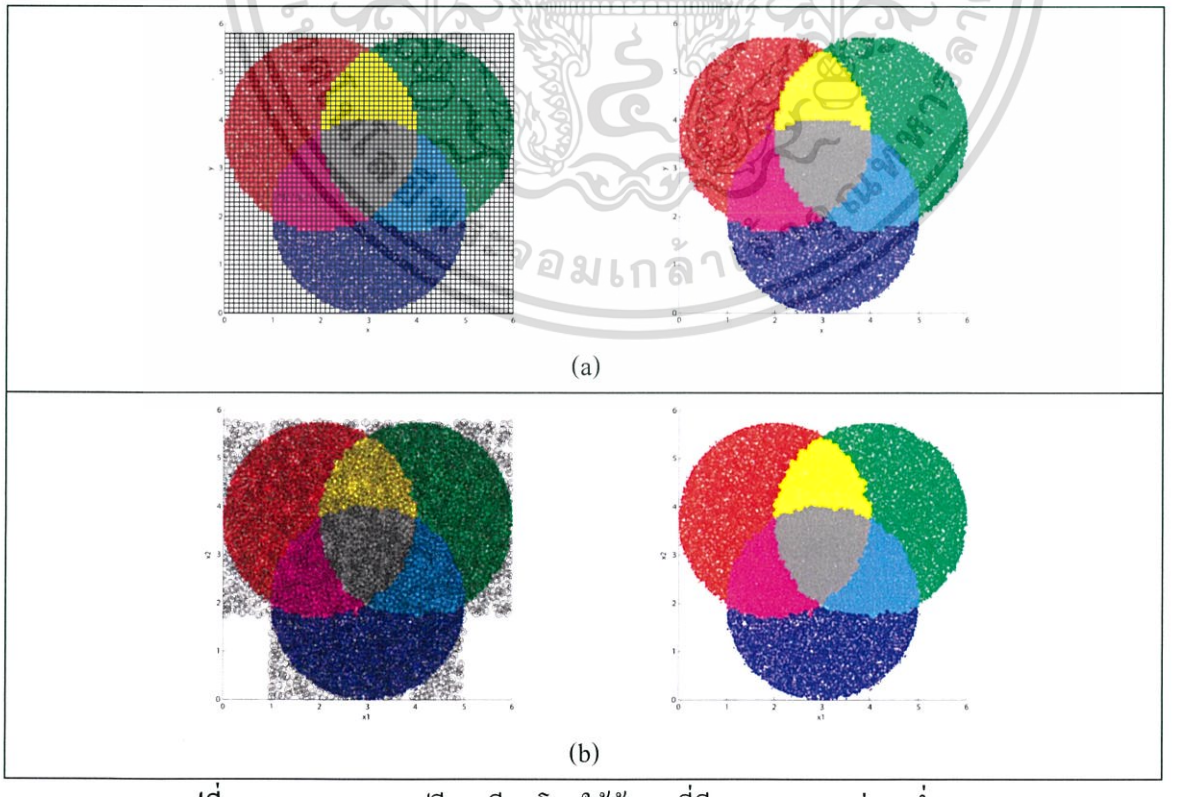
พารามิเตอร์ MaxDiff ไม่ได้มีความสัมพันธ์กับ Eps และ MinPts เลย และสำหรับการปรับค่าพารามิเตอร์เราจะต้องปรับขนาด Eps จนได้ขนาดที่เหมาะสมก่อนจากนั้นจึงค่อยปรับค่า MinPts ตาม และเมื่อได้ค่า Eps กับ MinPts ที่เหมาะสมแล้วจึงค่อยปรับค่า MaxDiff ทีหลัง

4.3 การทดลองเปรียบเทียบกับวิธีการอื่น

การทดลองในหัวข้อนี้จะเป็นการเปรียบเทียบวิธีการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลมกับการใช้กริดที่มีขนาดเท่ากันหมด โดยจะทำการเปรียบเทียบทั้งทางด้านเวลาที่ใช้ในการประมวลผลและจำนวนกลุ่มข้อมูลหน่วยที่สร้างขึ้น

4.3.1 การทดลองเปรียบเทียบโดยใช้ข้อมูลที่มีการกระจายอย่างสม่ำเสมอ

ข้อมูลที่ใช้ในการทดลองนี้เป็นข้อมูลที่มีการกระจายอย่างสม่ำเสมอซึ่งได้กล่าวถึงใน หัวข้อที่ 4.1.1 และการกำหนดค่าของพารามิเตอร์ต่าง ๆ จะกำหนดให้สอดคล้องกัน คือ สำหรับการจัดกลุ่มโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลมจะกำหนดให้ขนาด Eps เท่ากับ 0.04 MinPts กำหนดให้เท่ากับ 10 และ MaxDiff กำหนดให้เท่ากับ 0.5 สำหรับการจัดกลุ่มข้อมูลโดยใช้กริดจะกำหนดให้ขนาด Eps เท่ากับ 0.1 เพราะว่าเป็นขนาดที่ใกล้เคียงกับขนาดของกลุ่มข้อมูลหน่วยแบบวงกลม ส่วนค่า MinPts กับค่า MaxDiff จะกำหนดให้เท่ากันกับการจัดกลุ่มโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลม ซึ่งผลการทดลองที่ได้เป็นดังแสดงในรูปที่ 4.7 และตารางที่ 4.4



รูปที่ 4.7 แสดงผลการเปรียบเทียบโดยใช้ข้อมูลที่มีการกระจายอย่างสม่ำเสมอ

ตารางที่ 4.4 เปรียบเทียบผลการจัดกลุ่มข้อมูลโดยใช้ข้อมูลที่มีการกระจายอย่างสม่ำเสมอ

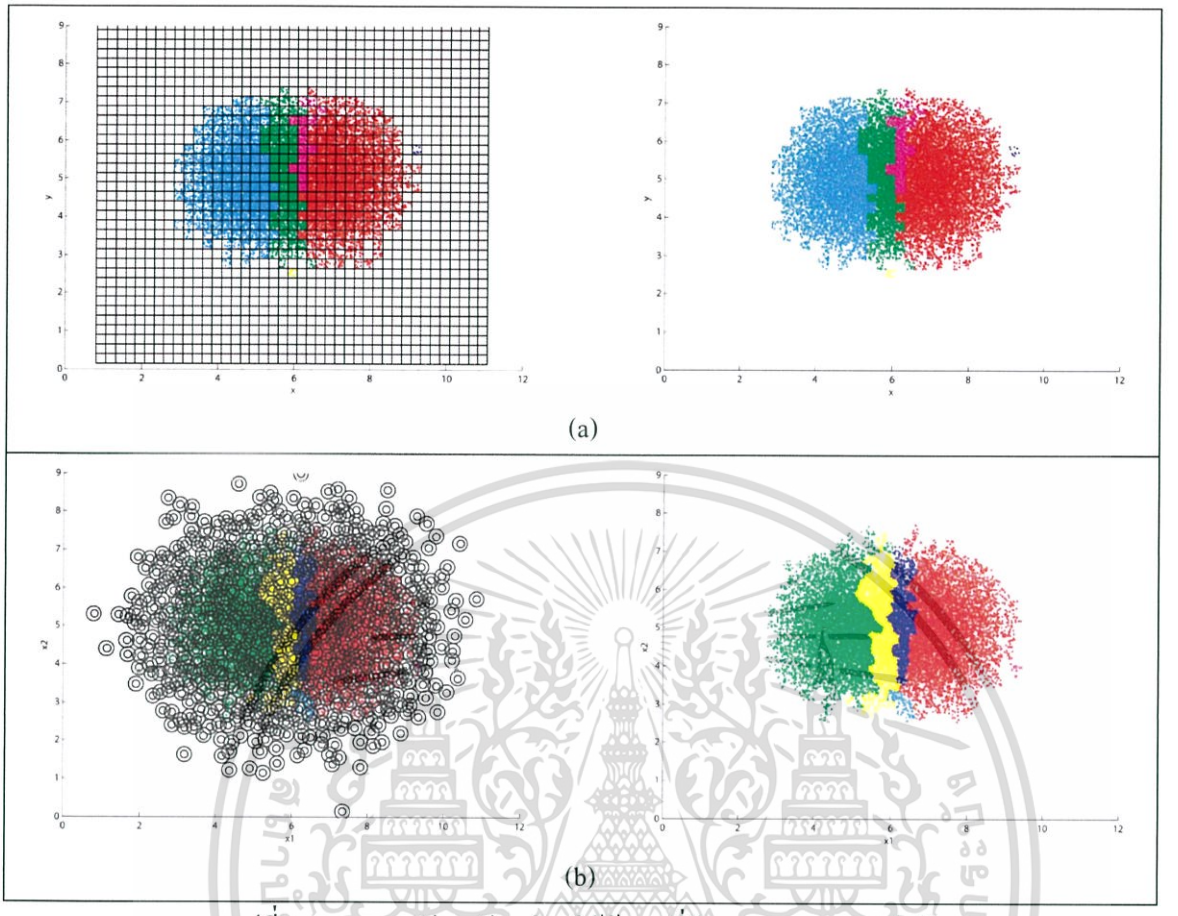
วิธีการ	จำนวนกลุ่มข้อมูลหน่วยทั้งหมด	จำนวนกลุ่มข้อมูลหน่วยที่ไม่ใช่ข้อมูลรบกวน	เวลาสร้าง กลุ่มข้อมูลหน่วย (วินาที)	เวลารวมกลุ่มข้อมูล (วินาที)	รวมเวลาทั้งหมด (วินาที)
กริดขนาดเท่ากันหมด	3480	2602	1.265	14.250	15.515
กลุ่มข้อมูลหน่วยแบบวงกลม	2846	2538	19.562	16.485	36.047

4.3.2 การทดลองเปรียบเทียบโดยใช้ข้อมูลที่มีการกระจายลดลงอย่างสม่ำเสมอ

ข้อมูลที่ใช้ในการทดลองนี้เป็นข้อมูลที่มีการกระจายลดลงอย่างสม่ำเสมอซึ่งได้กล่าวถึงในหัวข้อที่ 4.1.2 และการกำหนดค่าของพารามิเตอร์ต่าง ๆ จะกำหนดให้สอดคล้องกัน คือ สำหรับการจัดกลุ่มโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลมจะกำหนดให้ขนาด Eps เท่ากับ 2 MinPts กำหนดให้เท่ากับ 1 และ MaxDiff กำหนดให้เท่ากับ 0.3 สำหรับการจัดกลุ่มข้อมูลโดยใช้กริดจะกำหนดให้ขนาด Eps เท่ากับ 5 ส่วนค่า MinPts กับค่า MaxDiff จะกำหนดให้เท่ากันกับการจัดกลุ่มโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลม ซึ่งผลการทดลองที่ได้เป็นดังแสดงในรูปที่ 4.8 และตารางที่ 4.5

ตารางที่ 4.5 เปรียบเทียบผลการจัดกลุ่มข้อมูลโดยใช้ข้อมูลที่มีการกระจายลดลงอย่างสม่ำเสมอ

วิธีการ	จำนวนกลุ่มข้อมูลหน่วยทั้งหมด	จำนวนกลุ่มข้อมูลหน่วยที่ไม่ใช่ข้อมูลรบกวน	เวลาสร้าง กลุ่มข้อมูลหน่วย (วินาที)	เวลารวมกลุ่มข้อมูล (วินาที)	รวมเวลาทั้งหมด (วินาที)
กริดขนาดเท่ากันหมด	676	676	1.016	0.938	1.954
กลุ่มข้อมูลหน่วยแบบวงกลม	695	695	4.110	1.234	5.344



รูปที่ 4.9 ผลการเปรียบเทียบโดยใช้ข้อมูลที่มีการกระจายแบบปกติ

4.3.4 สรุปผลการทดลอง

จากผลการทดลองจะพบว่าการจัดกลุ่มข้อมูลโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลมจะทำสร้างกลุ่มข้อมูลหน่วยขึ้นเฉพาะในบริเวณที่มีข้อมูลอยู่เท่านั้น ต่างจากการใช้กริดขนาดเท่ากันหมดที่ กริดจะถูกสร้างขึ้นมาทั่วทั้งสเปซ แต่อย่างไรก็ตามเมื่อพิจารณาทางด้านเวลาที่ใช้ในการประมวลผลกลับพบว่าการจัดกลุ่มข้อมูลโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลมใช้เวลามากกว่าการใช้กริดขนาดเท่ากันหมดโดยเฉพาะเวลาที่ใช้ในการสร้างกลุ่มข้อมูลหน่วยที่มีความแตกต่างกันอย่างเห็นได้ชัด ส่วนผลการจัดกลุ่มก็จะเห็นได้ว่าทั้งสองวิธีสามารถจัดกลุ่มข้อมูลได้ผลดีเหมือนกันทั้งคู่

จากผลการทดลองเราจะเห็นว่าการสร้างกลุ่มข้อมูลหน่วยแบบวงกลมใช้เวลามากกว่าการสร้างกริดขนาดเท่ากันหมดแต่ปรากฏการณ์นี้จะเกิดขึ้นกับข้อมูลที่มีจำนวนมิติน้อยเท่านั้น เพราะว่าเมื่อจำนวนมิติของข้อมูลเพิ่มขึ้นจำนวนของกริดที่สร้างขึ้นจะเพิ่มขึ้นแบบเอ็กซ์โพเน็นเชียลดังนั้นจึงจะใช้เวลามากกว่าการสร้างกลุ่มข้อมูลหน่วยแบบวงกลมซึ่งจะถูกสร้างขึ้นเฉพาะในบริเวณที่มีข้อมูลอยู่เท่านั้น

บทที่ 5
การทดลองจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริดที่มีความละเอียดหลายระดับ

บทนี้เป็นการนำเสนอผลการทดลองจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริดที่มีความละเอียดหลายระดับ โดยตอนแรกจะเป็นการทดลองปรับค่าพารามิเตอร์ต่างๆ เพื่อดูว่ามีผลอย่างไรบ้างต่อการจัดกลุ่มข้อมูล ต่อจากนั้นก็จะเป็นการทดลองเปรียบเทียบวิธีสร้างกริดแบบต่างๆ ว่าแบบไหนให้ผลดีกว่ากันอย่างไรและภายใต้เงื่อนไขอะไรบ้าง และสุดท้ายก็จะเป็นการทดลองเปรียบเทียบการจัดกลุ่มข้อมูลระหว่างการใช้กริดที่มีความละเอียดหลายระดับกับการใช้กริดที่มีขนาดเท่ากันหมด

5.1 การทดลองปรับค่าพารามิเตอร์ต่างๆ

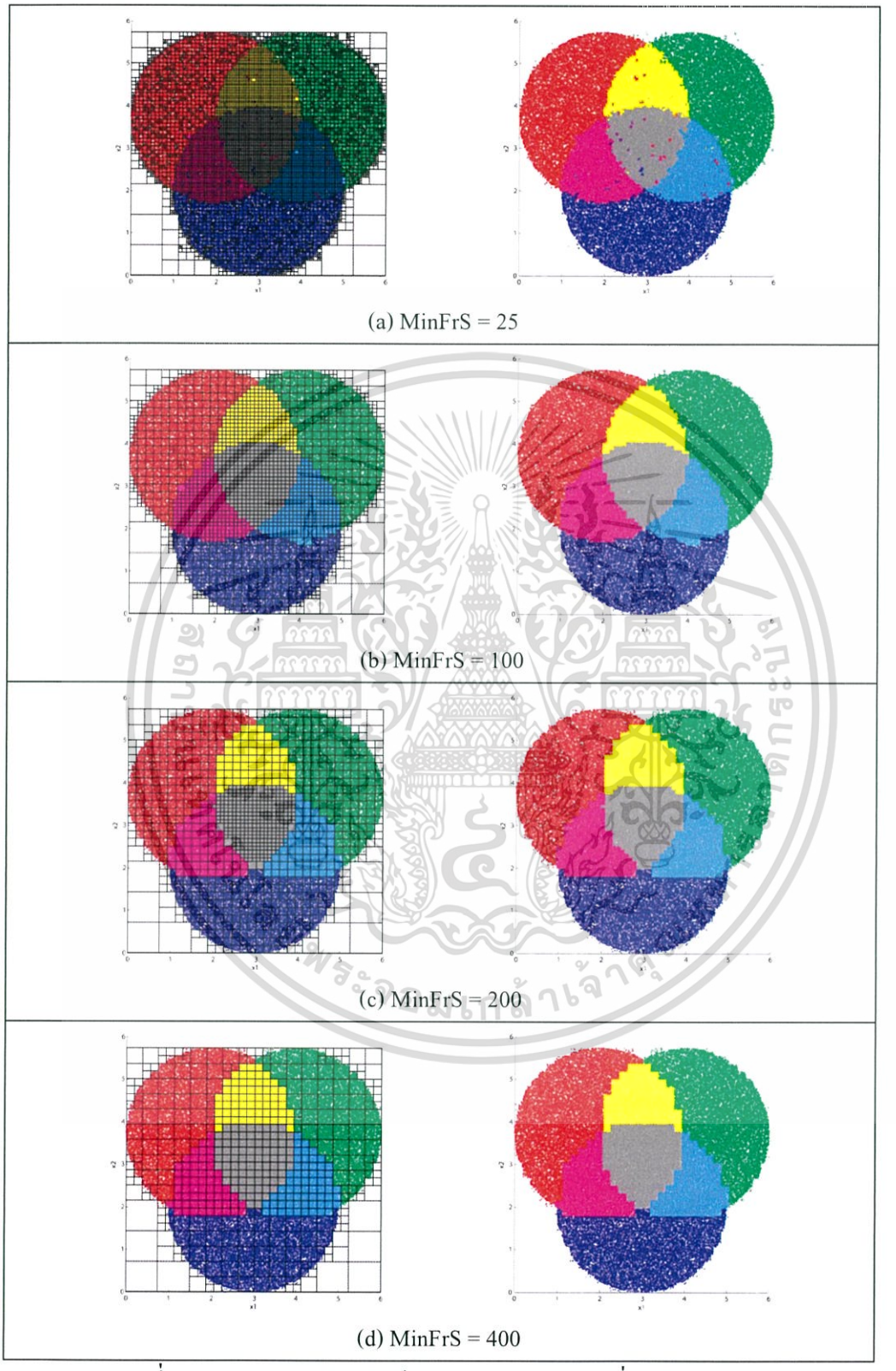
หัวข้อนี้จะเป็นการทดลองปรับเปลี่ยนค่าพารามิเตอร์ต่างๆ ซึ่งมีผลต่อการจัดกลุ่มข้อมูล เพื่อดูว่าแต่ละพารามิเตอร์มีผลต่อการจัดกลุ่มข้อมูลอย่างไรบ้าง อีกทั้งยังเป็นการหาความสัมพันธ์ระหว่างพารามิเตอร์ต่างๆ ด้วย โดยชุดข้อมูลที่ใช้จะเป็นชุดข้อมูลที่มีการกระจายอย่างสม่ำเสมอจากหัวข้อที่ 4.1.1

5.1.1 การทดลองปรับเปลี่ยนค่า MinFrS

การทดลองนี้จะเป็นการทดลองปรับเปลี่ยนค่า MinFrS ซึ่งเป็นพารามิเตอร์ของการสร้างกริดโดยพิจารณาจากจำนวนจุดข้อมูล ในขณะที่กำหนดให้พารามิเตอร์อื่นๆ เป็นค่าคงที่ โดยจะทำการกำหนดค่า MinFrS เป็น 25 100 200 และ 400 ตามลำดับ ส่วนค่า MinDens กับ MaxDiff กำหนดให้เท่ากับ 700 และ 0.5 ตามลำดับ ซึ่งผลการจัดกลุ่มข้อมูลที่ได้เป็นดังแสดงในตารางที่ 5.1 และรูปที่ 5.1

ตารางที่ 5.1 เปรียบเทียบผลการจัดกลุ่มข้อมูลโดยเปลี่ยนค่า MinFrS

MinFrS	จำนวนกลุ่มข้อมูล	CCE	Error (จุด)	จำนวนกริดที่ไม่ใช่ข้อมูลรบกวน	เวลาสร้างกริด (วินาที)	เวลารวมกลุ่มข้อมูล (วินาที)
25	103	0.2682	3110	8087	1.453	146.781
100	7	0.0207	2648	1784	1.360	6.718
200	7	0.0254	3845	1033	1.343	2.282
400	7	0.0377	4221	463	1.344	0.469



(a) MinFrS = 25

(b) MinFrS = 100

(c) MinFrS = 200

(d) MinFrS = 400

รูปที่ 5.1 แสดงผลการจัดกลุ่มข้อมูลโดยทำการปรับเปลี่ยนค่า MinFrS

จากรูปที่ 5.1 จะพบว่าถ้ากำหนดค่า MinFrS น้อยจะทำให้ได้กริดขนาดเล็กและมีจำนวนมาก เวลาที่ใช้ในการรวมกลุ่มข้อมูลก็จะมากด้วย แต่ถ้ากำหนดให้ค่า MinFrS มากก็จะทำให้ได้กริดขนาดใหญ่และมีจำนวนน้อยซึ่งก็จะส่งผลให้ใช้เวลาการรวมกลุ่มข้อมูลน้อยด้วย แต่ทั้งนี้ในการสร้างกริดโดยพิจารณาจากจำนวนจุดข้อมูล ขนาดของกริดจะขึ้นอยู่กับความหนาแน่นของข้อมูลด้วย นั่นคือ ในบริเวณที่มีข้อมูลอยู่หนาแน่นกว่ากริดก็จะมีขนาดเล็กกว่า

5.1.2 การทดลองปรับเปลี่ยนค่า MaxLevel

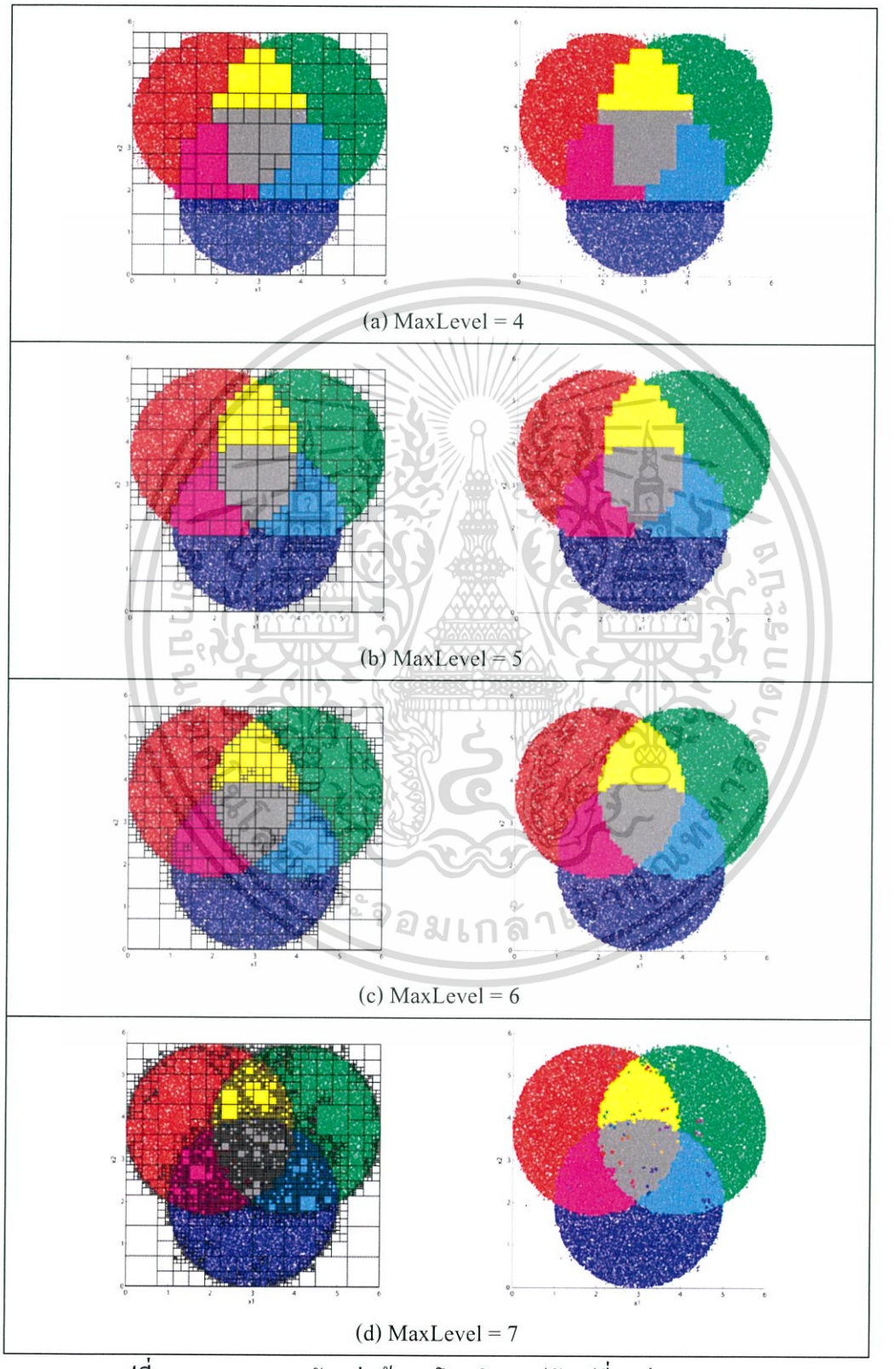
การทดลองนี้จะเป็นการทดลองปรับเปลี่ยนค่า MaxLevel ซึ่งเป็นพารามิเตอร์ของการสร้างกริดโดยพิจารณาจากความต่างของอัตราส่วนคลาส ในขณะที่กำหนดให้พารามิเตอร์อื่นๆ เป็นค่าคงที่ โดยจะใช้การสร้างกริดแบบ DCR 2 และทำการกำหนดค่า MaxLevel เป็น 4 5 6 และ 7 ตามลำดับ ส่วนค่า MinDiff MinDens และ MaxDiff จะกำหนดให้เท่ากับ 0.3 700 และ 0.5 ตามลำดับ ซึ่งผลการจัดกลุ่มข้อมูลที่ได้เป็นดังแสดงในตารางที่ 5.2 และรูปที่ 5.2

ตารางที่ 5.2 เปรียบเทียบผลการจัดกลุ่มข้อมูลโดยเปลี่ยนค่า MaxLevel

MaxLevel	จำนวนกลุ่มข้อมูล	CCE	Error (จุด)	จำนวนกริดที่ไม่ใช่ข้อมูลรบกวน	เวลาสร้างกริด (วินาที)	เวลารวมกลุ่มข้อมูล (วินาที)
4	7	0.0494	7756	123	1.313	0.031
5	7	0.0357	4176	268	1.359	0.157
6	7	0.0276	2453	672	1.375	0.953
7	122	0.5510	3222	4386	1.516	44.469

ดังที่ได้กล่าวไปแล้วว่าพารามิเตอร์ MaxLevel เป็นสิ่งที่ใช้กำหนดระดับความลึกของการแตกกริดสูงสุดที่อนุญาต ดังนั้นถ้ากำหนดค่า MaxLevel น้อยเราก็จะได้กริดในระดับลึกที่สุดที่มีขนาดใหญ่ แต่ในทางตรงกันข้ามถ้ากำหนดค่า MaxLevel มาก เราก็จะได้ได้กริดในระดับลึกที่สุดที่มีขนาดเล็ก ซึ่งก็สอดคล้องกับผลการทดลองในรูปที่ 5.2 เมื่อ MaxLevel เท่ากับ 4 เราจะได้กริดในระดับลึกสุดที่มีขนาดใหญ่ทั้งๆ ที่กริดนั้นสมควรที่จะถูกแตกต่อไปอีกแต่ก็ไม่สามารถแตกต่อไปได้เพราะติดที่ค่า MaxLevel แต่เมื่อกำหนดให้ MaxLevel เท่ากับ 5 และ 6 เราก็จะพบว่าผลการแตกกริดได้ดีขึ้นตามลำดับ กริดที่สมควรจะถูกแตกต่อไปได้ถูกแตกจนได้ผลที่น่าพอใจ ส่วนกริดที่ไม่สมควรจะถูกแตกต่อไปก็ไม่ถูกแตก ซึ่งก็แสดงให้เห็นว่าค่า MaxLevel ไม่มีผลต่อกริดเหล่านั้นเลย แต่เมื่อกำหนดให้ MaxLevel เท่ากับ 7 กลับพบว่ากริดที่ไม่สมควรจะถูกแตกต่อไปถูกแตก ทั้งนี้ก็เป็นเพราะว่าขนาดของกริดส่วนใหญ่ซึ่งใน

ที่นี้พบว่าเป็นขนาดกริดที่เล็กที่สุดไม่ใช่ขนาดกริดที่เหมาะสม ดังจะสังเกตได้จากกลุ่มข้อมูลที่จัดได้ที่มีแต่กลุ่มข้อมูลขนาดเล็กๆ เต็มไปหมด



รูปที่ 5.2 แสดงผลการจัดกลุ่มข้อมูลโดยทำการปรับเปลี่ยนค่า MaxLevel

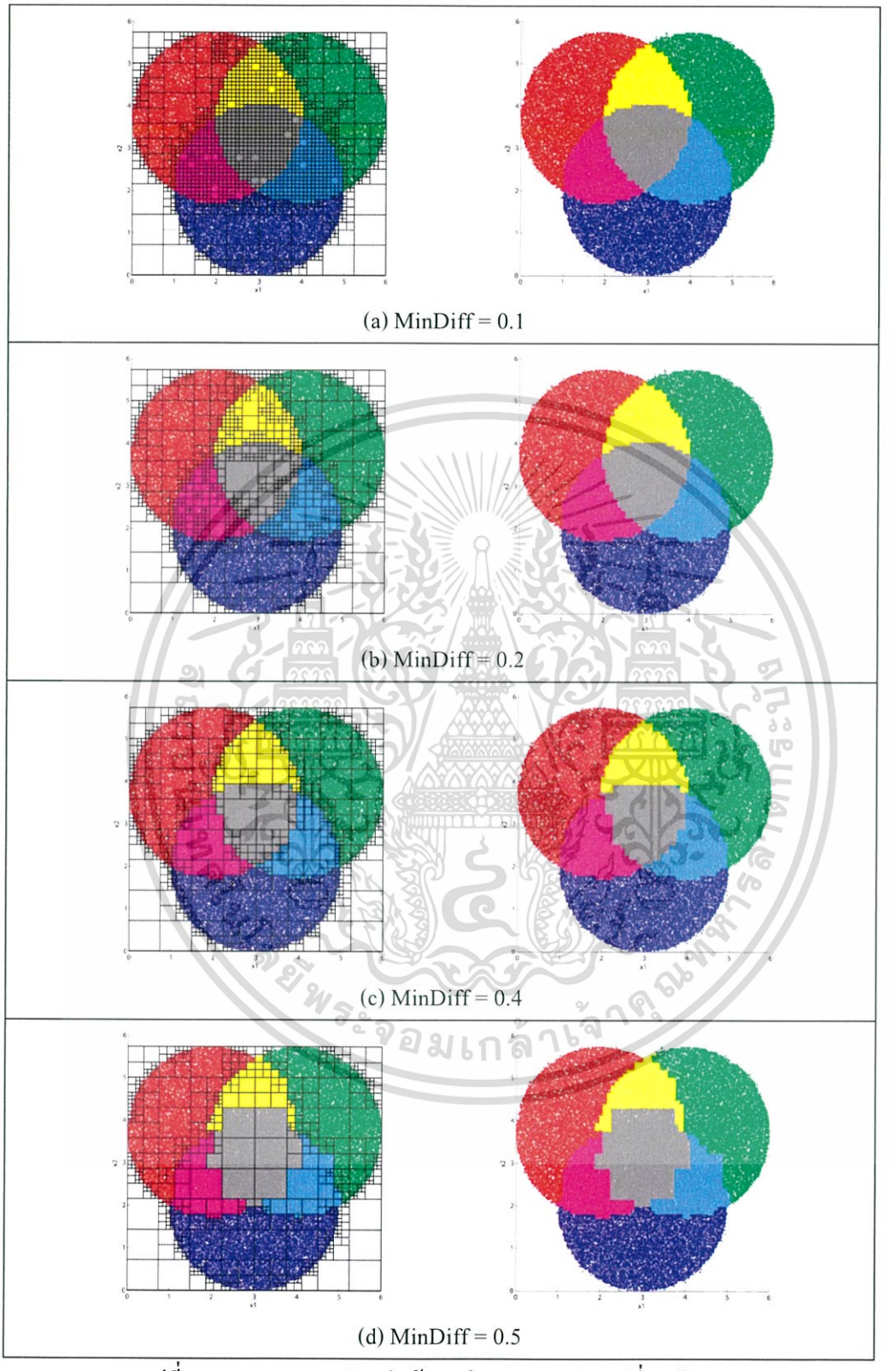
5.1.3 การทดลองปรับเปลี่ยนค่า MinDiff

การทดลองนี้จะเป็นการทดลองปรับเปลี่ยนค่า MinDiff ซึ่งเป็นพารามิเตอร์ของการสร้างกริดโดยพิจารณาจากความต่างของอัตราส่วนคลาส ในขณะที่กำหนดให้พารามิเตอร์อื่นๆ เป็นค่าคงที่ โดยจะใช้การสร้างกริดแบบ DCR 2 และทำการกำหนดค่า MinDiff เป็น 0.1 0.2 0.4 และ 0.5 ตามลำดับ ส่วนค่า MaxLevel MinDens และ MaxDiff จะกำหนดให้เท่ากับ 6 700 และ 0.5 ตามลำดับ ซึ่งผลการจัดกลุ่มข้อมูลที่ได้เป็นดังแสดงในตารางที่ 5.3 และรูปที่ 5.3

ตารางที่ 5.3 เปรียบเทียบผลการจัดกลุ่มข้อมูลโดยเปลี่ยนค่า MinDiff

MinDiff	จำนวนกลุ่มข้อมูล	CCE	Error (จุด)	จำนวนกริดที่ไม่ใช่ข้อมูลรบกวน	เวลาสร้างกริด (วินาที)	เวลารวมกลุ่มข้อมูล (วินาที)
0.1	7	0.0236	2381	1598	1.375	5.516
0.2	7	0.0265	2375	1028	1.391	2.219
0.4	8	0.0449	3194	561	1.391	0.656
0.5	8	0.2080	7646	438	1.375	0.406

MinDiff เป็นพารามิเตอร์อีกตัวหนึ่งที่ใช้กำหนดการอนุญาตให้มีการแตกกริดโดยจะยอมให้แตกกริดต่อไปได้ถ้าความต่างของอัตราส่วนคลาสระหว่างกริดแม่กับกริดลูกมีค่ามากกว่าหรือเท่ากับพารามิเตอร์ตัวนี้ ดังนั้นถ้ากำหนดให้ค่าของพารามิเตอร์ตัวนี้น้อยก็จะหมายถึงการอนุญาตให้กริดที่มีความต่างของอัตราส่วนคลาสน้อยสามารถแตกต่อไปได้อีก ส่งผลให้ได้กริดขนาดเล็กจำนวนมาก ซึ่งก็สอดคล้องกับผลการทดลองที่แสดงในรูปที่ 5.3 ที่เมื่อกำหนดให้ MinDiff เท่ากับ 0.1 กริดที่อยู่ในพื้นที่ที่มีการซ้อนทับกันของข้อมูลถูกแตกจนเกือบหมด ต่อมาเมื่อกำหนดให้ MinDiff เพิ่มขึ้นเป็น 0.2 ก็จะพบว่ามีบางกริดที่อยู่ในพื้นที่ที่มีการซ้อนทับกันของข้อมูลไม่ถูกแตก แต่ถ้ากำหนดค่าของ MinDiff มากเกินไปกริดที่สมควรจะถูกแตกก็อาจจะไม่ถูกแตกก็ได้ ดังรูปที่ 5.3 (c) และ (d) เมื่อกำหนดให้ MinDiff เท่ากับ 0.4 และ 0.5 ตามลำดับ จำนวนกริดที่อยู่ในพื้นที่ที่มีการซ้อนทับกันของข้อมูลไม่ถูกแตกเพิ่มขึ้นตามลำดับถึงแม้ว่ากริดนั้นจะเป็นกริดที่สมควรจะถูกแตกต่อไปก็ตาม



(a) MinDiff = 0.1

(b) MinDiff = 0.2

(c) MinDiff = 0.4

(d) MinDiff = 0.5

รูปที่ 5.3 แสดงผลการจัดกลุ่มข้อมูลโดยทำการปรับเปลี่ยนค่า MinDiff

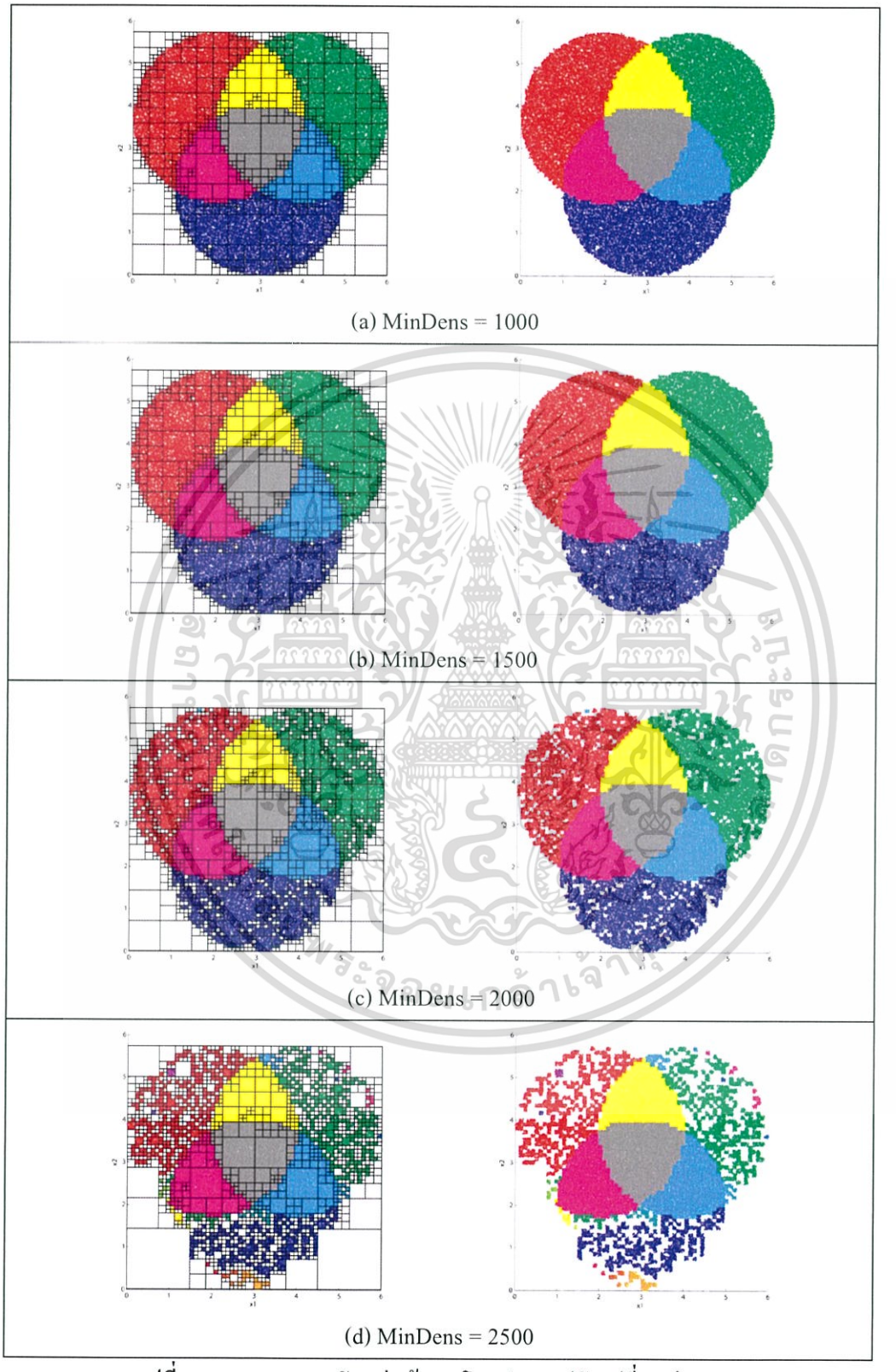
5.1.4 การทดลองปรับเปลี่ยนค่า MinDens

การทดลองนี้จะเป็นการทดลองปรับเปลี่ยนค่า MinDens ซึ่งเป็นพารามิเตอร์ของการสร้างกริดทั้งสองแบบ ในขณะที่กำหนดให้พารามิเตอร์อื่นๆ เป็นค่าคงที่ โดยจะใช้การสร้างกริดแบบ DCR 2 และทำการกำหนดค่า MinDens เป็น 1000 1500 2000 และ 2500 ตามลำดับ ส่วนค่า MaxLevel MinDiff และ MaxDiff จะกำหนดให้เท่ากับ 6 0.3 และ 0.5 ตามลำดับ ซึ่งผลการจัดกลุ่มข้อมูลที่ได้เป็นดังแสดงในตารางที่ 5.4 และรูปที่ 5.4

ตารางที่ 5.4 เปรียบเทียบผลการจัดกลุ่มข้อมูลโดยเปลี่ยนค่า MinDens

MinDens	จำนวนกลุ่มข้อมูล	CCE	Error (จุด)	จำนวนกริดที่ไม่ใช่ข้อมูลรบกวน	เวลาสร้างกริด (วินาที)	เวลารวมกลุ่มข้อมูล (วินาที)
1000	7	0.0249	2574	684	1.390	0.969
1500	7	0.0242	3156	736	1.375	1.141
2000	8	0.0208	7352	1085	1.406	2.500
2500	32	0.0763	24332	1075	1.391	2.484

เนื่องจาก Mindens เป็นพารามิเตอร์ที่กำหนดว่ากริดไหนควรจะจัดเป็นกริดของข้อมูลรบกวน โดยการพิจารณาจะกระทำโดยกำหนดให้กริดที่มีความหนาแน่นน้อยกว่าค่าของพารามิเตอร์ตัวนี้เป็นกริดของข้อมูลรบกวน ดังนั้นเมื่อกำหนดให้พารามิเตอร์ตัวนี้มีค่าน้อยเกินไปกริดที่ควรจะถูกจัดให้เป็นกริดของข้อมูลรบกวนก็จะไม่ถูกจัดให้เป็นกริดของข้อมูลรบกวน ส่งผลให้ข้อมูลรบกวนไม่ถูกกำจัดทิ้งไป แต่ถ้ากำหนดให้พารามิเตอร์ตัวนี้มีค่ามากเกินไปกริดที่ไม่ควรจะถูกจัดให้เป็นกริดของข้อมูลรบกวนก็จะถูกจัดให้เป็นกริดของข้อมูลรบกวน ส่งผลให้ข้อมูลที่ไม่ใช่ข้อมูลรบกวนถูกกำจัดทิ้งไป ซึ่งก็สอดคล้องกับผลการทดลองที่แสดงในรูปที่ 5.4 ที่เมื่อค่าของพารามิเตอร์ Mindens เพิ่มขึ้นก็เกิดรูพรุนในกลุ่มข้อมูลมากขึ้นเป็นลำดับ



รูปที่ 5.4 แสดงผลการจัดกลุ่มข้อมูลโดยทำการปรับเปลี่ยนค่า MinDens

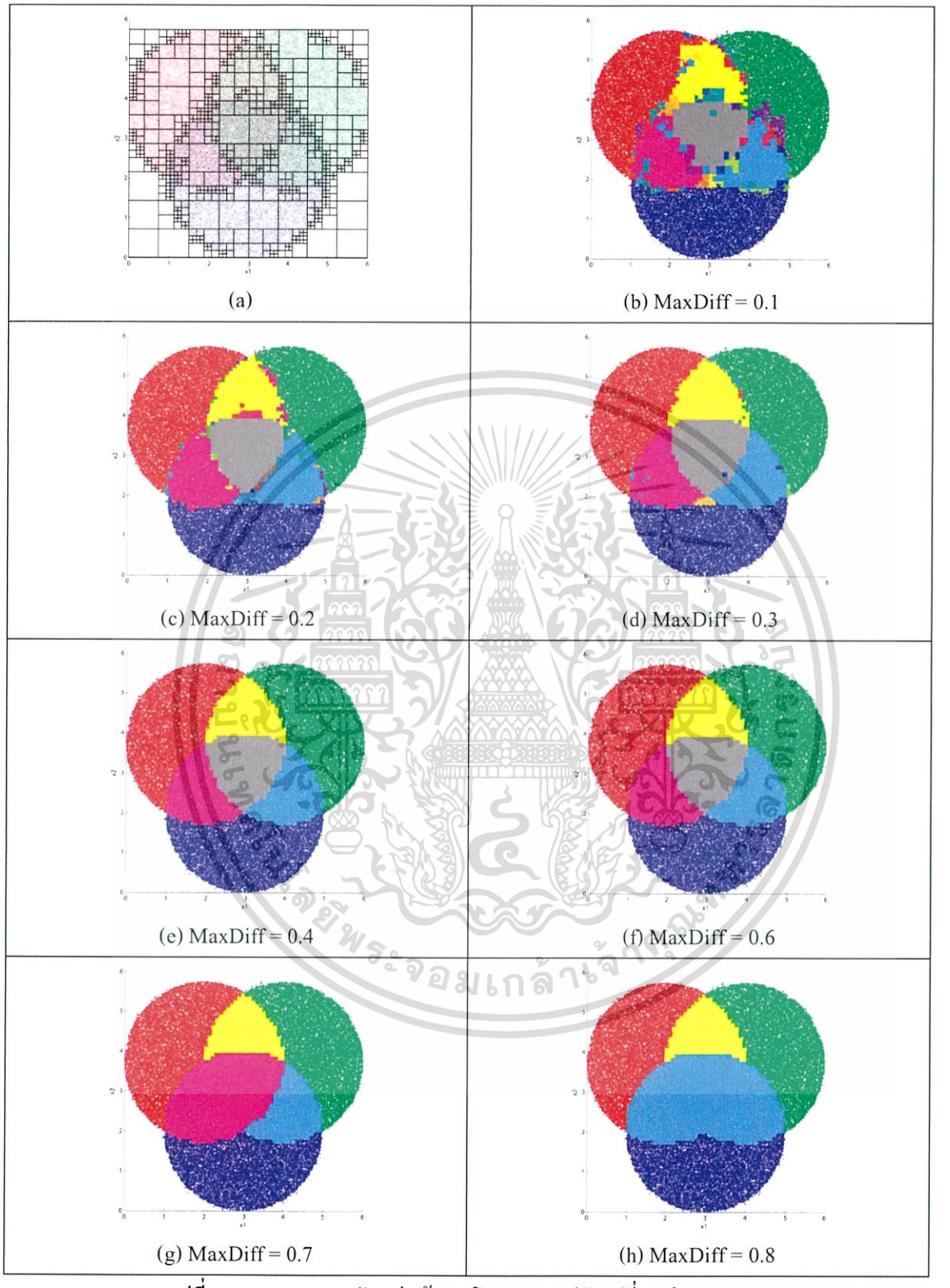
5.1.5 การทดลองปรับเปลี่ยนค่า MaxDiff

การทดลองนี้จะเป็นการทดลองปรับเปลี่ยนค่า MaxDiff ในขณะที่กำหนดให้พารามิเตอร์อื่น ๆ เป็นค่าคงที่ ถึงแม้ว่า MaxDiff จะเป็นพารามิเตอร์สำหรับการรวมกริดซึ่งผู้แต่งวิทยานิพนธ์นี้ไม่ได้คิดค้นเองก็ตาม แต่เพื่อความครบถ้วนสมบูรณ์ของวิทยานิพนธ์จึงจำเป็นต้องนำเสนอการทดลองนี้ด้วย โดยการทดลองนี้จะใช้การสร้างกริดแบบ DCR 2 และทำการกำหนดค่า MaxDiff เป็น 0.1 0.2 0.3 0.4 0.6 0.7 และ 0.8 ตามลำดับ ส่วนค่า MaxLevel MinDiff และ MinDens จะกำหนดให้เท่ากับ 6 0.3 และ 700 ตามลำดับ ซึ่งผลการจัดกลุ่มข้อมูลที่ได้เป็นดังแสดงในตารางที่ 5.5 และรูปที่ 5.5

ตารางที่ 5.5 เปรียบเทียบผลการจัดกลุ่มข้อมูลโดยเปลี่ยนค่า MaxDiff

MaxDiff	จำนวนกลุ่มข้อมูล	CCE	Error (จุด)	เวลารวมกลุ่มข้อมูล (วินาที)
0.1	195	0.3766	14855	1.532
0.2	77	0.2111	6197	1.125
0.3	34	0.1057	3998	1.000
0.4	9	0.0354	2492	0.937
0.6	7	0.0276	2453	0.937
0.7	6	0.2977	21713	0.937
0.8	5	0.5708	31679	0.937

MaxDiff เป็นพารามิเตอร์ที่ใช้สำหรับกำหนดความแตกต่างของอัตราส่วนคลาสสูงสุดที่ยังทำให้กริดสามารถรวมเข้าด้วยกันได้ ดังนั้นถ้ากำหนดค่าของ MaxDiff ไว้ต่ำเกินไปกริดที่มีความแตกต่างของอัตราส่วนคลาสมากหรือปานกลางก็จะไม่ถูกรวมเข้าด้วยกันซึ่งอาจจะส่งผลให้ได้กลุ่มข้อมูลเล็กๆ จำนวนมาก ดังรูปที่ 5.5 (b) และ (c) เมื่อเพิ่มค่าของ MaxDiff ขึ้นมาอีกหน่อยกริดที่มีความแตกต่างของอัตราส่วนคลาสปานกลางก็จะถูกรวมเข้าด้วยกัน ทำให้จำนวนกลุ่มข้อมูลที่มีขนาดเล็กลดลง ดังรูปที่ 5.5 (e) และ (f) แต่ถ้ากำหนดค่าของ MaxDiff ไว้สูงเกินไปกริดที่มีความแตกต่างของอัตราส่วนคลาสมากก็จะถูกรวมเข้าด้วยกันทำให้ได้จำนวนกลุ่มข้อมูลน้อย ดังรูปที่ 5.5 (b) และ (c)



รูปที่ 5.5 แสดงผลการจัดกลุ่มข้อมูลโดยทำการปรับเปลี่ยนค่า MaxDiff

5.2 การทดลองเปรียบเทียบวิธีการสร้างกริดแบบต่างๆ

การทดลองในหัวข้อนี้จะเป็นการเปรียบเทียบวิธีการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้การสร้างกริดที่มีความละเอียดหลายระดับแบบต่างๆ เพื่อดูว่าวิธีการสร้างกริดแต่ละแบบแบบไหนดีกว่ากัน หรือแบบไหนเหมาะกับข้อมูลที่เป็นอย่างไร

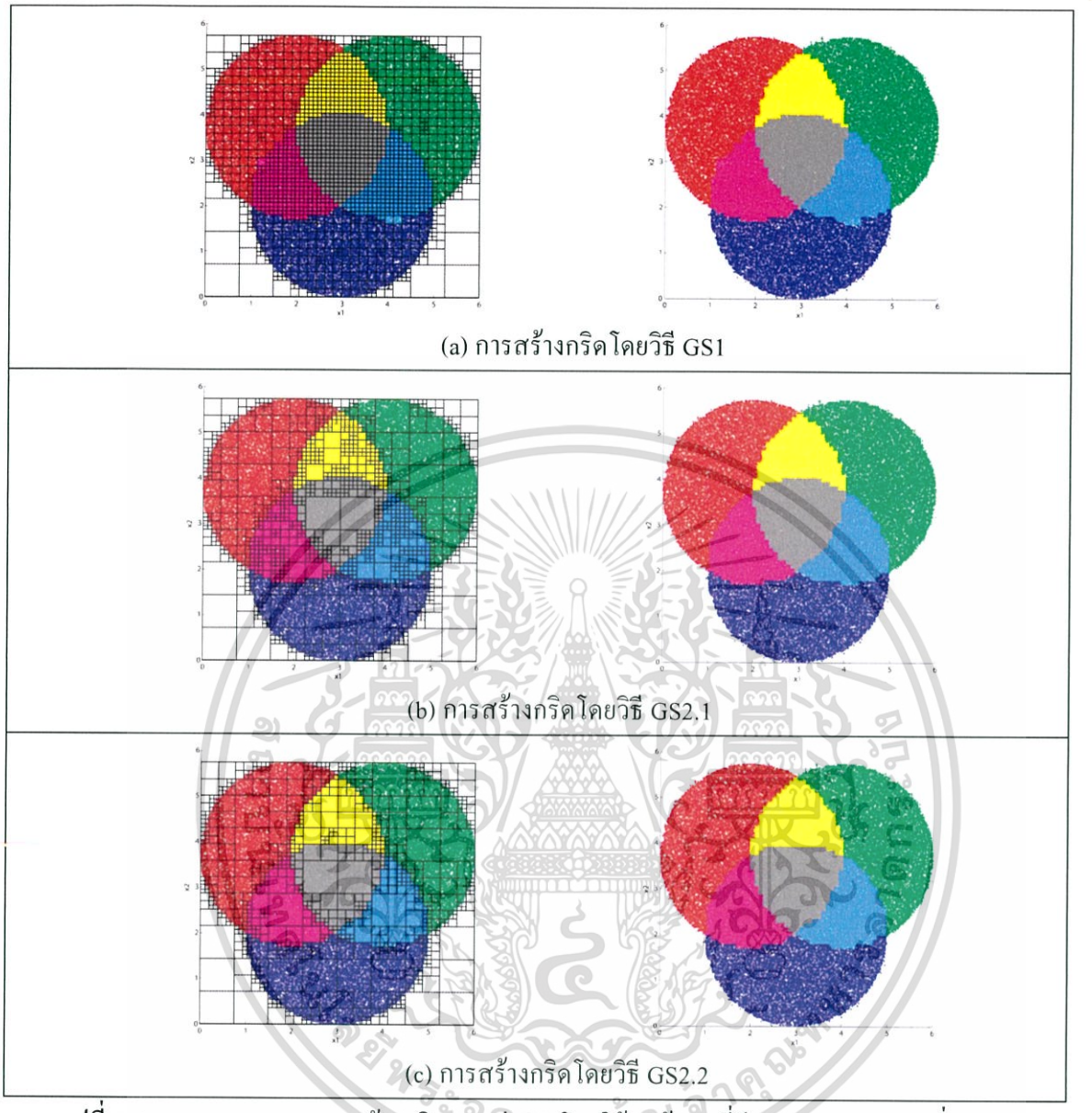
5.2.1 การทดลองเปรียบเทียบโดยใช้ข้อมูลที่มีการกระจายอย่างสม่ำเสมอ

การทดลองนี้จะเป็นการเปรียบเทียบวิธีสร้างกริดแบบต่าง ๆ โดยใช้ชุดข้อมูลที่มีการกระจายอย่างสม่ำเสมอจากหัวข้อที่ 4.1.1 ซึ่งการกำหนดค่าพารามิเตอร์ของวิธีสร้างกริดแบบต่าง ๆ จะเป็นดังนี้ คือ วิธีสร้างกริดแบบ GS1 ได้กำหนดให้ค่า MinFrS เท่ากับ 100 ค่า MinDens เท่ากับ 700 และค่า MaxDiff เท่ากับ 0.5 สำหรับวิธีสร้างกริดแบบ GS2 ทั้งสองแบบจะกำหนดให้ค่าพารามิเตอร์ต่าง ๆ เหมือนกัน คือ กำหนดให้ค่า MaxLevel เท่ากับ 6 ค่า MinDiff เท่ากับ 0.3 ส่วนค่า MinDens กับ MaxDiff นั้นจะใช้ค่าเดียวกันกับวิธี GS1 และเมื่อจัดกลุ่มข้อมูลออกมาแล้วได้ผลดังแสดงในรูปที่ 5.6 และตารางที่ 5.6

ตารางที่ 5.6 เปรียบเทียบการสร้างกริดแบบต่าง ๆ โดยใช้ข้อมูลที่มีการกระจายอย่างสม่ำเสมอ

วิธีการ	จำนวนกริดทั้งหมด	จำนวนกริดที่ไม่ใช่ข้อมูลรบกวน	เวลาสร้างกริด (วินาที)	เวลารวมกลุ่มข้อมูล (วินาที)	รวมเวลาทั้งหมด (วินาที)
GS1	1954	1784	1.328	6.562	7.890
GS2.1	1201	1032	1.343	2.204	3.547
GS2.2	841	672	1.343	0.938	2.281

จากผลการทดลองจะพบว่าสำหรับชุดข้อมูลนี้การสร้างกริดแบบ GS2.2 ให้ผลการสร้างกริดที่ดีที่สุด รองลงมาก็เป็นการสร้างกริดแบบ GS2.1 ส่วนการสร้างกริดแบบ GS1 ให้ผลการสร้างกริดที่ไม่ค่อยดี แต่เมื่อทำการรวมกริดเข้าด้วยกันแล้วทั้งสามวิธีก็ให้ผลใกล้เคียงกัน เมื่อพิจารณาผลทางด้านเวลาที่ใช้ในการประมวลผลกับปริมาณหน่วยความจำที่ใช้ก็จะพบว่ามีความสอดคล้องกันกับผลการสร้างกริด นั่นก็คือการสร้างกริดแบบ GS2.2 สามารถสร้างกริดได้อย่างเหมาะสมดังนั้นจึงให้ได้จำนวนกริดน้อยซึ่งก็ส่งผลให้เวลาที่ใช้ในการประมวลผลและปริมาณหน่วยความจำที่ใช้น้อยกว่าการสร้างกริดแบบ GS1.1 และแบบ GS1



(a) การสร้างกริดโดยวิธี GS1

(b) การสร้างกริดโดยวิธี GS2.1

(c) การสร้างกริดโดยวิธี GS2.2

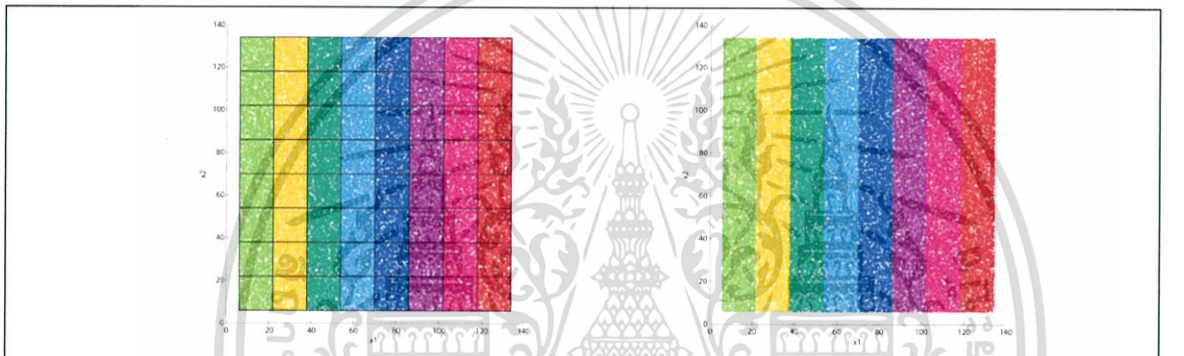
รูปที่ 5.6 แสดงผลการทดลองสร้างกริดแบบต่าง ๆ โดยใช้ชุดข้อมูลที่มีการกระจายแบบสม่ำเสมอ

5.2.2 การทดลองเปรียบเทียบโดยใช้ข้อมูลที่มีการกระจายลดลงอย่างสม่ำเสมอ

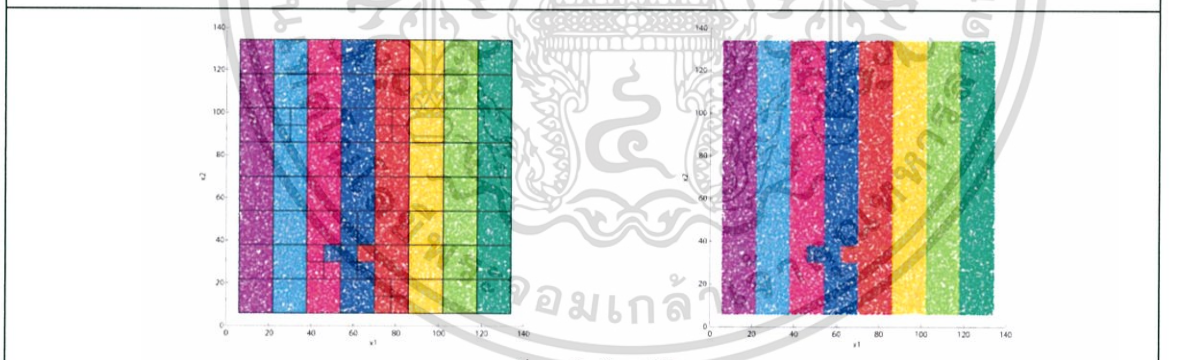
การทดลองนี้จะเป็นการเปรียบเทียบวิธีสร้างกริดแบบต่าง ๆ โดยใช้ชุดข้อมูลที่มีการกระจายลดลงอย่างสม่ำเสมอจากหัวข้อที่ 4.1.2 ซึ่งการกำหนดค่าพารามิเตอร์ของวิธีสร้างกริดแบบต่าง ๆ จะเป็นดังนี้ คือ วิธีสร้างกริดแบบ GS1 ได้กำหนดให้ค่า MinFrS เท่ากับ 2000 ค่า MinDens เท่ากับ 0 และค่า MaxDiff เท่ากับ 0.2 สำหรับวิธีสร้างกริดแบบ GS2 ทั้งสองแบบจะกำหนดให้ค่าพารามิเตอร์ต่าง ๆ เหมือนกัน คือ กำหนดให้ค่า MaxLevel เท่ากับ 4 ค่า MinDiff เท่ากับ 0.2 ส่วนค่า MinDens กับ MaxDiff นั้นจะใช้ค่าเดียวกันกับวิธี GS1 และเมื่อจัดกลุ่มข้อมูลออกมาแล้วได้ผลดังแสดงในรูปที่ 5.7 และตารางที่ 5.7

ตารางที่ 5.7 เปรียบเทียบการสร้างกริดแบบต่างๆ โดยใช้ข้อมูลที่มีการกระจายลดลงอย่างสม่ำเสมอ

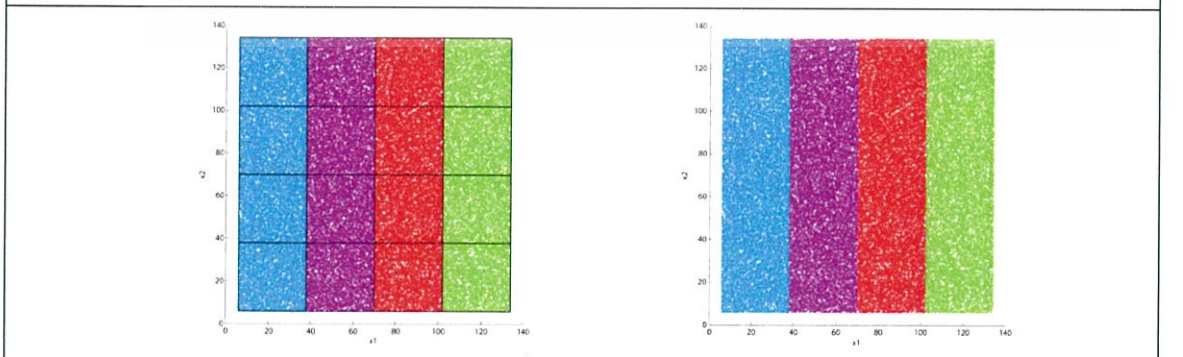
วิธีการ	จำนวนกริดทั้งหมด	จำนวนกริดที่ไม่ใช่ข้อมูลรบกวน	เวลาสร้างกริด (วินาที)	เวลารวมกลุ่มข้อมูล (วินาที)	รวมเวลาทั้งหมด (วินาที)
GS1	64	64	1.031	0.016	1.047
GS2.1	82	82	1.016	0.015	1.031
GS2.2	16	16	0.985	0.005	0.990



(a) การสร้างกริดโดยวิธี GS1



(b) การสร้างกริดโดยวิธี GS2.1



(c) การสร้างกริดโดยวิธี GS2.2

รูปที่ 5.7 แสดงผลการทดลองสร้างกริดแบบต่าง ๆ โดยใช้ชุดข้อมูลที่มีการกระจายลดลงอย่างสม่ำเสมอ

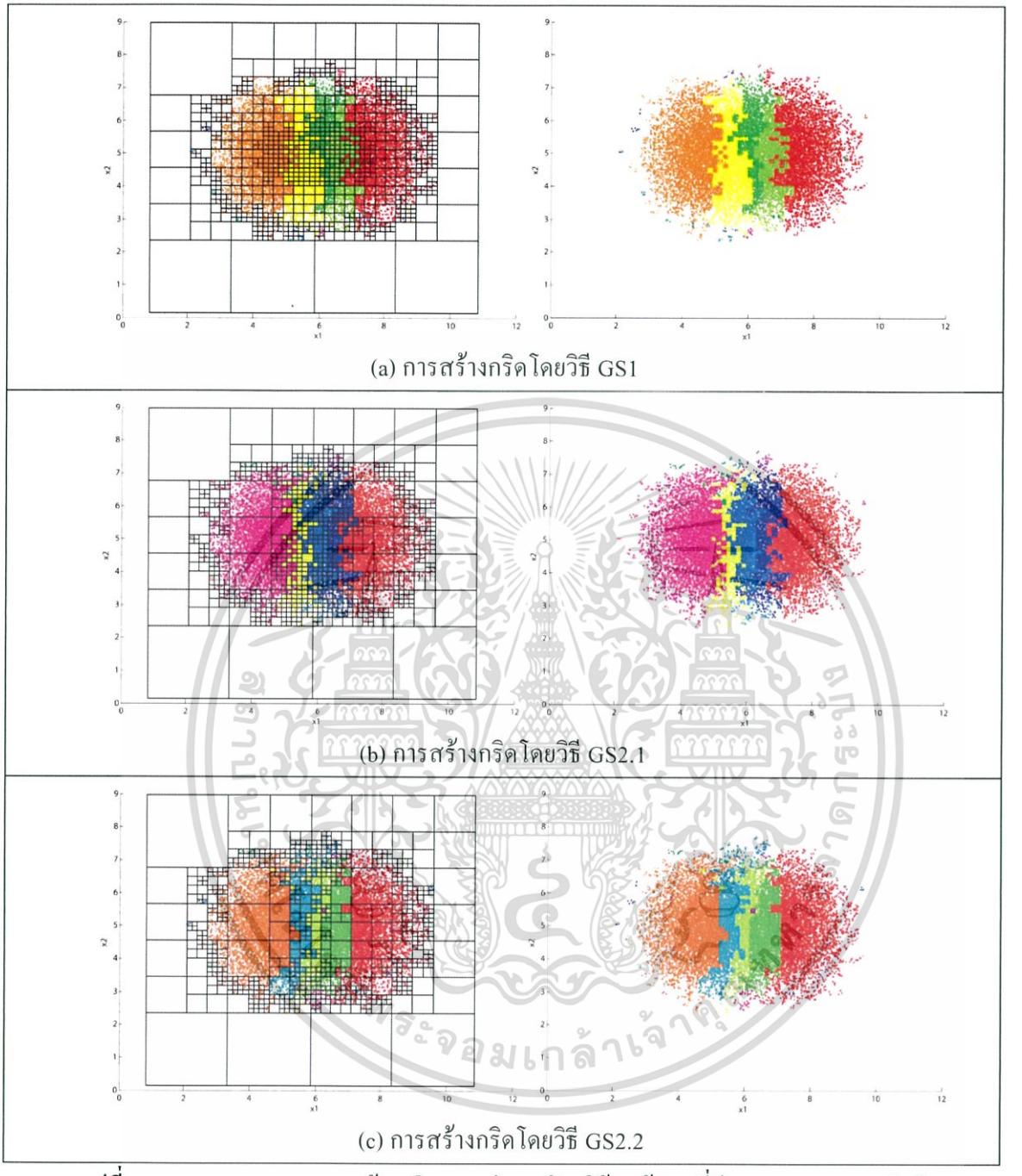
จากผลการทดลองจะพบว่าสำหรับข้อมูลชุดนี้การสร้างกริดทั้งสามวิธีให้ผลการจัดกลุ่มข้อมูลเหมาะสมกับลักษณะการกระจายตัวของข้อมูล แต่เมื่อมาวิเคราะห์การสร้างกริดของแต่ละวิธีจะพบว่าการสร้างกริดแบบ GS2.2 ให้ผลหยาบกว่าวิธี GS2.1 ทั้ง ๆ ที่ได้กำหนดให้พารามิเตอร์ต่าง ๆ ของวิธีการสร้างกริดแบบ GS2 ทั้งสองแบบเหมือนกัน ซึ่งสาเหตุนั้นก็มาจากการสร้างกริดแบบ GS2.2 จะพิจารณาการแตกกริดจากความต่างของอัตราส่วนคลาสของกริดแม่กับกริดลูก ซึ่งอัตราส่วนคลาสของ กริดแม่จริง ๆ แล้วก็คือค่าเฉลี่ยของกริดลูก และสำหรับข้อมูลชุดนี้ในกริดระดับที่ 2-3 อัตราส่วนคลาส ของกริดแม่กับกริดลูกไม่ได้แตกต่างกันมากดังนั้นกริดระดับที่ 2 จึงไม่ถูกแตก ส่วนการสร้างกริดแบบ GS2.1 นั้นจะพิจารณาการแตกกริดจากความต่างของอัตราส่วนคลาสของกริดลูก ซึ่งความต่างของอัตราส่วนคลาสระหว่างกริดลูกด้วยกันเองกับระหว่างกริดแม่กับกริดลูกนั้น ระหว่างกริดลูกด้วยกันเองจะมีความแตกต่างกันมากกว่า ดังนั้นการสร้างกริดแบบ GS2.1 จึงแตกกริดลงไปในระดับที่ลึกกว่าแบบ GS2.2 สำหรับการสร้างกริดแบบ GS1 นั้นจะพิจารณาการแตกกริดจากจำนวนจุดข้อมูลที่อยู่ในกริดอยู่แล้ว ดังนั้นเมื่อข้อมูลมีความหนาแน่นเท่ากันหมดกริดที่ได้จึงอยู่ในระดับเดียวกันหมด

5.2.3 การทดลองเปรียบเทียบโดยใช้ข้อมูลที่มีการกระจายแบบปกติ

การทดลองนี้จะเป็นการเปรียบเทียบวิธีสร้างกริดแบบต่าง ๆ โดยใช้ชุดข้อมูลที่มีการกระจายแบบปกติจากหัวข้อที่ 4.1.3 ซึ่งการกำหนดค่าพารามิเตอร์ของวิธีสร้างกริดแบบต่าง ๆ จะเป็นดังนี้ คือ วิธีสร้างกริดแบบ GS1 ได้กำหนดให้ค่า MinFrS เท่ากับ 100 ค่า MinDens เท่ากับ 100 และค่า MaxDiff เท่ากับ 0.3 สำหรับวิธีสร้างกริดแบบ GS2 ทั้งสองแบบจะกำหนดให้ค่าพารามิเตอร์ต่าง ๆ เหมือนกัน คือ กำหนดให้ค่า MaxLevel เท่ากับ 6 ค่า MinDiff เท่ากับ 0.25 ส่วนค่า MinDens กับ MaxDiff นั้นจะใช้ค่าเดียวกันกับวิธี GS1 และเมื่อจัดกลุ่มข้อมูลออกมาแล้วได้ผลดังแสดงในรูปที่ 5.8 และตารางที่ 5.8

ตารางที่ 5.8 เปรียบเทียบการสร้างกริดแบบต่าง ๆ โดยใช้ข้อมูลที่มีการกระจายแบบปกติ

วิธีการ	จำนวนกริดทั้งหมด	จำนวนกริดที่ไม่ใช่ข้อมูลรบกวน	เวลาสร้างกริด (วินาที)	เวลารวมกลุ่มข้อมูล (วินาที)	รวมเวลาทั้งหมด (วินาที)
GS1	922	649	0.328	0.969	1.297
GS2.1	934	655	0.297	1.078	1.375
GS2.2	751	473	0.313	0.593	0.906



(a) การสร้างกริดโดยวิธี GS1

(b) การสร้างกริดโดยวิธี GS2.1

(c) การสร้างกริดโดยวิธี GS2.2

รูปที่ 5.8 แสดงผลการทดลองสร้างกริดแบบต่าง ๆ โดยใช้ชุดข้อมูลที่มีการกระจายแบบปกติ

จากผลการทดลองพบว่าการสร้างกริดทั้งสามวิธีสามารถจัดกลุ่มข้อมูลชุดนี้ได้อย่างเหมาะสม แต่อาจจะแตกต่างกันบ้างซึ่งก็ไม่ได้ถือเป็นสาระสำคัญ แต่ถ้าพิจารณาการสร้างกริดของทั้งสามวิธีจะพบว่าการสร้างกริดแบบ GS2 ทั้งสองวิธีจะสามารถแตกกริดบริเวณที่ข้อมูลมีความบริสุทธิ์มากได้เหมาะสมกว่าการสร้างกริดแบบ GS1

5.3 การทดลองเปรียบเทียบกับวิธีการอื่น

การทดลองในหัวข้อนี้จะเป็นการเปรียบเทียบวิธีการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริดขนาดเดียวกับกริดที่มีความละเอียดหลายระดับ เพื่อแสดงให้เห็นว่าการใช้กริดที่มีความละเอียดหลายระดับใช้หน่วยความจำและเวลาในการประมวลผลน้อยกว่าการใช้กริดขนาดเดียว ในขณะที่ให้ผลการจัดกลุ่มข้อมูลใกล้เคียงกัน

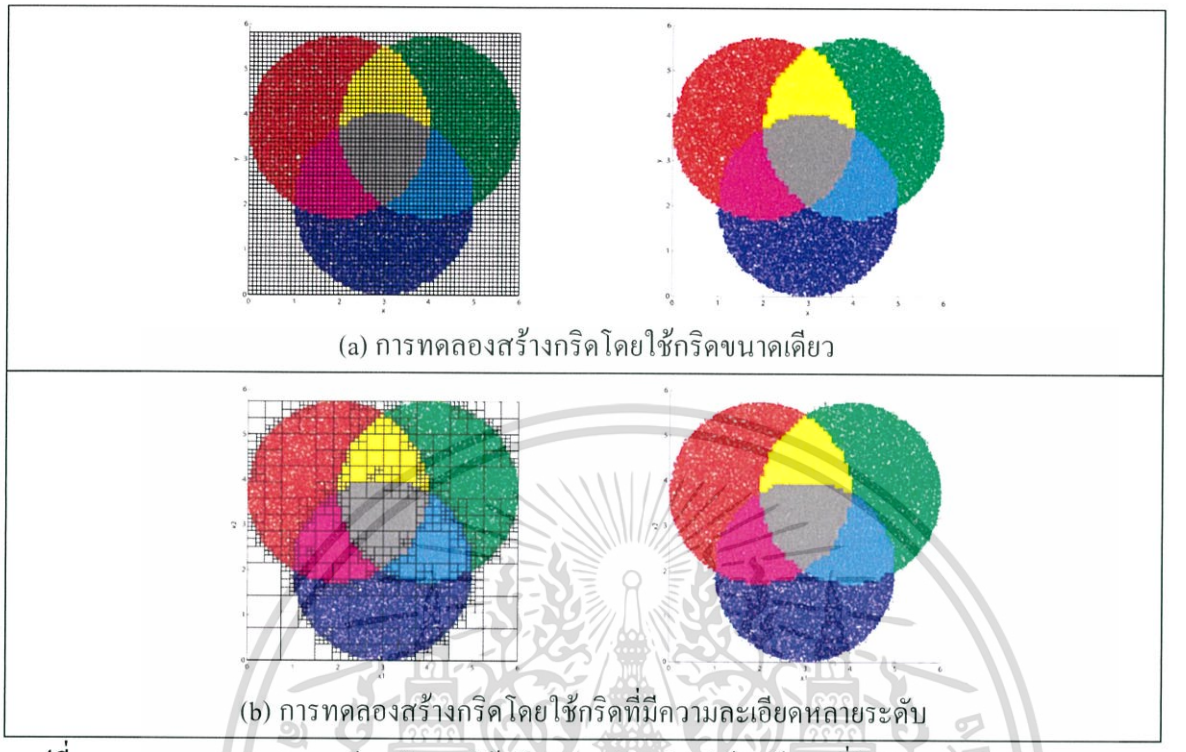
5.3.1 การทดลองเปรียบเทียบโดยใช้ข้อมูลที่มีการกระจายอย่างสม่ำเสมอ

ข้อมูลที่ใช้ในการทดลองนี้เป็นข้อมูลที่มีการกระจายอย่างสม่ำเสมอซึ่งได้กล่าวถึงในหัวข้อที่ 4.1.1 และวิธีสร้างกริดที่มีความละเอียดหลายระดับที่นำมาใช้คือการสร้างกริดแบบ GS2.2 โดยได้กำหนดให้ค่า MaxLevel เท่ากับ 6 ค่า MinDiff เท่ากับ 0.3 ค่า MinDens เท่ากับ 700 และค่า MaxDiff เท่ากับ 0.5 ส่วนการใช้กริดขนาดเดียว ได้กำหนดให้ค่า Eps เท่ากับ 0.09375 เนื่องจากเป็นขนาดที่เท่ากับขนาดกริดส่วนใหญ่ของวิธีที่ใช้กริดที่มีความละเอียดหลายระดับ ส่วนค่า MinPts กำหนดให้เท่ากับ 10 และค่า MaxDiff กำหนดให้เท่ากับ 0.5 ซึ่งผลการจัดกลุ่มข้อมูลเป็นดังแสดงในรูปที่ 5.9 และตารางที่ 5.9

ตารางที่ 5.9 เปรียบเทียบผลการจัดกลุ่มข้อมูลโดยใช้ข้อมูลที่มีการกระจายอย่างสม่ำเสมอ

วิธีการ	จำนวนกริดทั้งหมด	จำนวนกริดที่ไม่ใช่ข้อมูลรบกวน	เวลาสร้างกริด (วินาที)	เวลารวมกลุ่มข้อมูล (วินาที)	รวมเวลาทั้งหมด (วินาที)
กริดขนาดเท่ากันหมด	3968	2942	1.250	17.640	18.890
MRG	841	672	1.375	0.922	2.297

จากผลการทดลองจะเห็นได้ว่าเมื่อกลุ่มข้อมูลมีขอบเขตที่ชัดเจน และจำนวนข้อมูลรบกวนก็มีไม่มากจนเกินไปดังเช่นชุดข้อมูลที่ใช้นี้ การใช้กริดที่มีความละเอียดหลายระดับจะได้ผลดี กล่าวคือในพื้นที่ข้างในกลุ่มข้อมูลซึ่งเป็นพื้นที่ที่เนื้อข้อมูลมีความคล้ายคลึงกัน กริดจะมีขนาดใหญ่ ส่วนบริเวณขอบของกลุ่มข้อมูลซึ่งเป็นพื้นที่ที่เนื้อข้อมูลมีความแตกต่างกัน กริดจะมีขนาดเล็ก ส่งผลให้จำนวนกริดทั้งหมดมีน้อยกว่าการใช้ กริดขนาดเดียว และในเมื่อจำนวนกริดมีน้อยเวลาที่ใช้ในการรวมกริดก็จะน้อยด้วย แต่มีข้อน่าสังเกตที่เวลาที่ใช้ในการสร้างกริดที่การใช้กริดที่มีความละเอียดหลายระดับจะใช้เวลามากกว่าการใช้กริดขนาดเดียว อย่างไรก็ตามเวลาที่ใช้ในการสร้างกริดก็ต่างกันไม่มากเมื่อเทียบกับเวลาที่ใช้ในการรวมกริดและเมื่อพิจารณาจากเวลาโดยรวมแล้วก็จะพบว่าการใช้กริดที่มีความละเอียดหลายระดับใช้เวลาน้อยกว่าการใช้กริดขนาดเดียว



(a) การทดลองสร้างกริดโดยใช้กริดขนาดเดียว

(b) การทดลองสร้างกริดโดยใช้กริดที่มีความละเอียดหลายระดับ

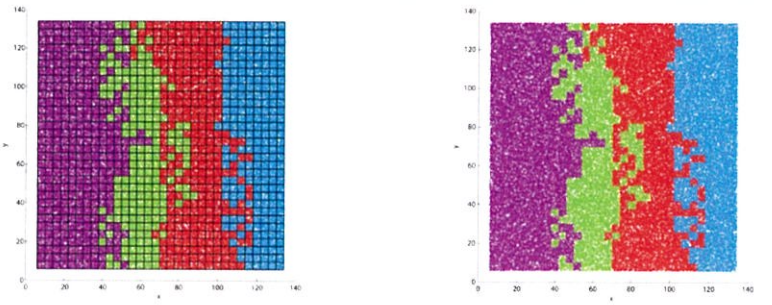
รูปที่ 5.9 แสดงผลการทดลองเทียบกับการใช้กริดขนาดเดียวโดยใช้ชุดข้อมูลที่มีการกระจายแบบสม่ำเสมอ

5.3.2 การทดลองเปรียบเทียบโดยใช้ข้อมูลที่มีการกระจายลดลงอย่างสม่ำเสมอ

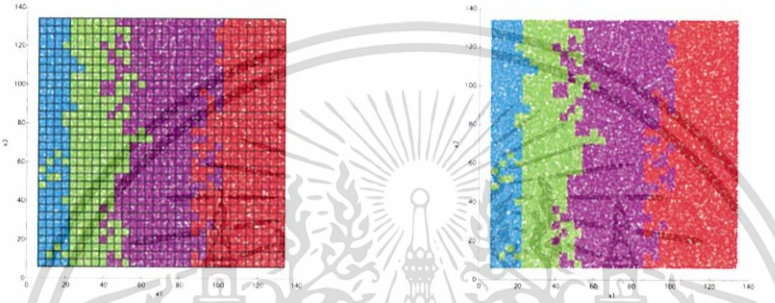
ข้อมูลที่ใช้ในการทดลองนี้เป็นข้อมูลที่มีการกระจายลดลงอย่างสม่ำเสมอซึ่งได้กล่าวถึงในหัวข้อที่ 4.1.2 และวิธีสร้างกริดที่มีความละเอียดหลายระดับที่นำมาใช้คือการสร้างกริดแบบ GS1 โดยได้กำหนดให้ค่า MinFrS เท่ากับ 110 ค่า MinDens เท่ากับ 1 และค่า MaxDiff เท่ากับ 0.3 ส่วนการใช้กริดขนาดเดียว ได้กำหนดให้ค่า Eps เท่ากับ 4 เนื่องจากเป็นขนาดที่เท่ากับขนาดกริดส่วนใหญ่ของวิธีที่ใช้กริดที่มีความละเอียดหลายระดับ ส่วนค่า MinPts กำหนดให้เท่ากับ 10 และค่า MaxDiff กำหนดให้เท่ากับ 0.3 ซึ่งผลการจัดกลุ่มข้อมูลเป็นดังแสดงในรูปที่ 5.10 และตารางที่ 5.10

ตารางที่ 5.10 เปรียบเทียบผลการจัดกลุ่มข้อมูลโดยใช้ข้อมูลที่มีการกระจายลดลงอย่างสม่ำเสมอ

วิธีการ	จำนวนกริดทั้งหมด	จำนวนกริดที่ไม่ใช่ข้อมูลรบกวน	เวลาสร้างกริด (วินาที)	เวลารวมกลุ่มข้อมูล (วินาที)	รวมเวลาทั้งหมด (วินาที)
กริดขนาดเท่ากันหมด	1024	1024	1.016	2.094	3.110
MRG	1024	1024	1.047	2.140	3.187



a) การทดลองสร้างกริดโดยใช้กริดขนาดเดียว



(b) การทดลองสร้างกริดโดยใช้กริดที่มีความละเอียดหลายระดับ

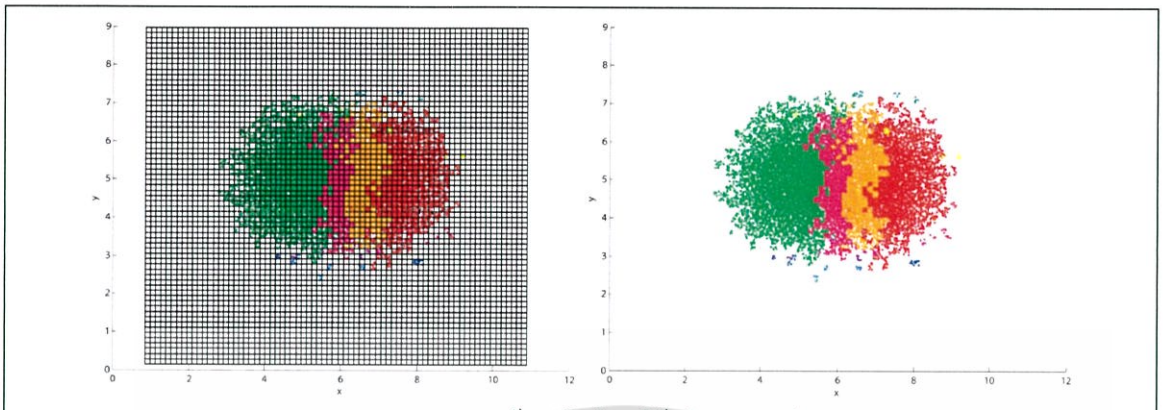
รูปที่ 5.10 แสดงผลการทดลองเทียบกับการใช้กริดขนาดเดียวโดยใช้ชุดข้อมูลที่มีการกระจายลดลงอย่างสม่ำเสมอ

จากการทดลองจะเห็นได้ว่าเมื่อความหนาแน่นของข้อมูลเท่ากันหมดทั่วทั้งสเปซและขอบเขตของกลุ่มไม่มีความชัดเจนดังเช่นชุดข้อมูลที่ใช้นี้ การใช้กริดที่มีความละเอียดหลายระดับจะ ไม่ได้ผลกล่าวคือจะให้ผลไม่แตกต่างจากการใช้กริดขนาดเท่ากันหมด อีกทั้งยังใช้เวลาในการประมวลผลและปริมาณหน่วยความจำมากกว่าการใช้กริดขนาดเดียวเล็กน้อย

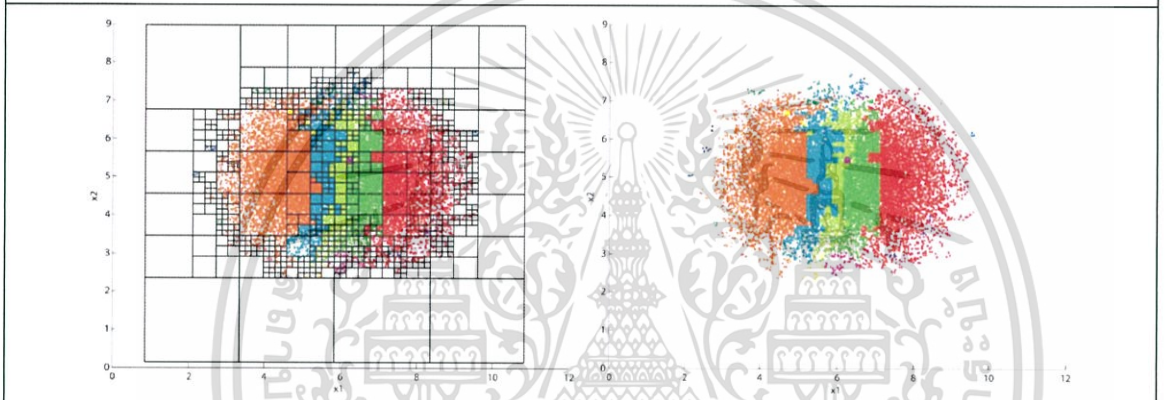
จากผลการจัดกลุ่มที่แสดงในรูปที่ 5.10 การที่ลักษณะของกลุ่มข้อมูลที่ได้จากทั้งสองวิธีไม่เหมือนกันนั้นเป็นเพราะลำดับการรวมกริดของทั้งสองวิธีนี้ไม่เหมือนกันและไม่ได้ถือเป็นสาระสำคัญ ซึ่งถ้าผู้อ่านมีความสนใจก็สามารถศึกษาเพิ่มเติมได้จากวิทยานิพนธ์ [6]

5.3.3 การทดลองเปรียบเทียบโดยใช้ข้อมูลที่มีการกระจายแบบปกติ

ข้อมูลที่ใช้ในการทดลองนี้เป็นข้อมูลที่มีการกระจายแบบปกติซึ่งได้กล่าวถึงในหัวข้อที่ 4.1.3 และวิธีสร้างกริดที่มีความละเอียดหลายระดับที่นำมาใช้คือการสร้างกริดแบบ GS2.2 โดยได้กำหนดให้ค่า MaxLevel เท่ากับ 6 ค่า MinDiff เท่ากับ 0.3 ค่า MinDens เท่ากับ 100 และค่า MaxDiff เท่ากับ 0.3 ส่วนการใช้กริดขนาดเดียว ได้กำหนดให้ค่า Eps เท่ากับ 0.138 เนื่องจากเป็นขนาดที่เท่ากับขนาดกริดส่วนใหญ่ของวิธีที่ใช้กริดที่มีความละเอียดหลายระดับ ส่วนค่า MinPts กำหนดให้เท่ากับ 5 และค่า MaxDiff กำหนดให้เท่ากับ 0.3 ซึ่งผลการจัดกลุ่มข้อมูลเป็นดังแสดงในรูปที่ 5.11 และตารางที่ 5.11



(a) การทดลองสร้างกริดโดยใช้กริดขนาดเดียว



(b) การทดลองสร้างกริดโดยใช้กริดที่มีความละเอียดหลายระดับ

รูปที่ 5.11 แสดงผลการทดลองเทียบกับการใช้กริดขนาดเดียวโดยใช้ชุดข้อมูลที่มีการกระจายแบบปกติ

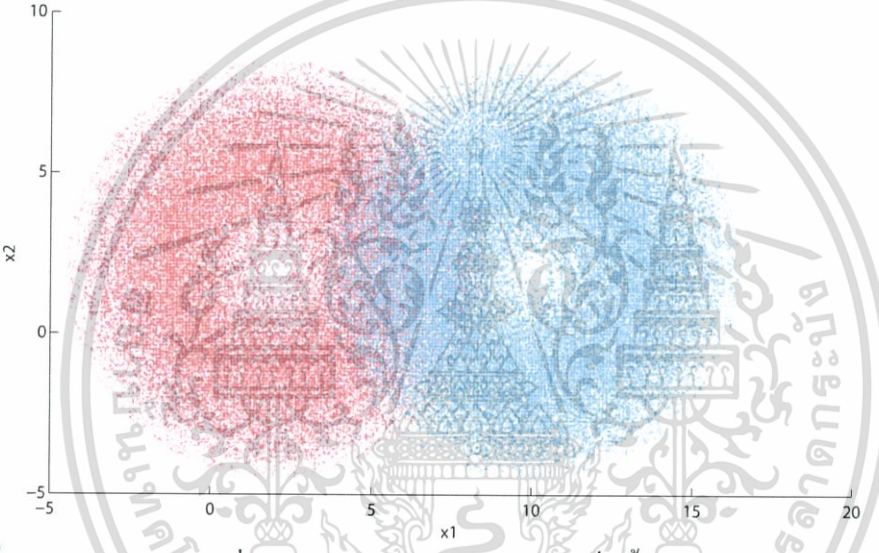
จากผลการทดลองพบว่าสำหรับข้อมูลชุดนี้การจัดกลุ่มข้อมูลโดยใช้กริดที่มีความละเอียดหลายระดับสามารถจัดกลุ่มข้อมูลได้ใกล้เคียงกับการใช้กริดขนาดเท่ากันหมด อีกทั้งยังใช้เวลาในการประมวลผลน้อยกว่าอีกด้วย สาเหตุที่สามารถจัดกลุ่มข้อมูลชุดนี้ได้อย่างเหมาะสมก็เนื่องจากว่าข้อมูลชุดนี้เกือบจะมีขอบเขตที่ชัดเจนเหมือนชุดข้อมูลที่มีการกระจายตัวอย่างสม่ำเสมอ

ตารางที่ 5.11 เปรียบเทียบผลการจัดกลุ่มข้อมูลโดยใช้ข้อมูลที่มีการกระจายแบบปกติ

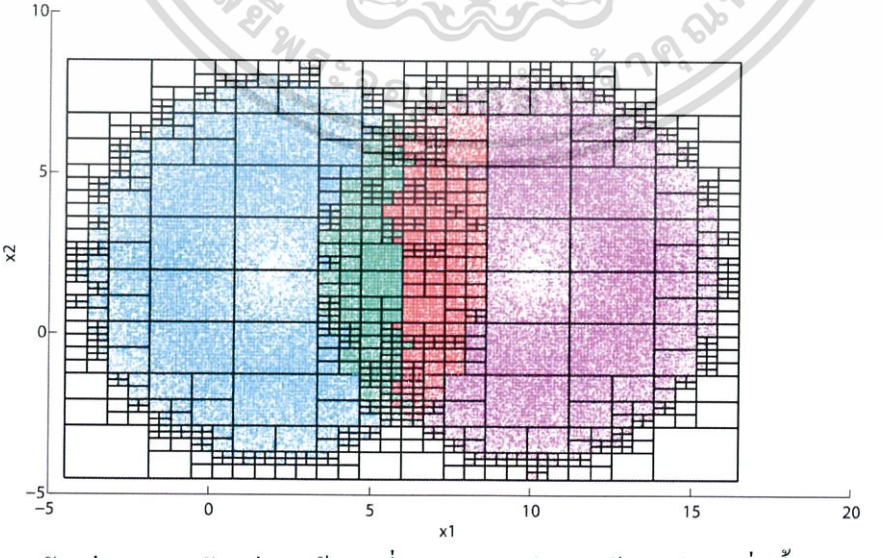
วิธีการ	จำนวนกริดทั้งหมด	จำนวนกริดที่ไม่ใช่ข้อมูลรบกวน	เวลาสร้างกริด (วินาที)	เวลารวมกลุ่มข้อมูล (วินาที)	รวมเวลาทั้งหมด (วินาที)
กริดขนาดเท่ากันหมด	4672	1061	0.297	2.516	2.813
MRG	715	437	0.343	0.500	0.843

5.4 การทดลองจัดกลุ่มข้อมูลที่มีความหนาแน่นแตกต่างกัน

การทดลองในหัวข้อนี้จะเป็นการทดลองจัดกลุ่มข้อมูลที่ความหนาแน่นของข้อมูลมีความแตกต่างกันและความต่างของอัตราส่วนคลาสของบริเวณที่อยู่ใกล้กันค่อยๆ มีการเปลี่ยนแปลงทีละน้อย ไม่ใช้มีการเปลี่ยนแปลงแบบก้าวกระโดด ชุดข้อมูลที่ใช้ชุดแรกจะเป็นข้อมูลที่มี 2 คลาสแต่ละคลาสมีลักษณะเป็นวงกลมโดยความหนาแน่นของข้อมูลบริเวณจุดศูนย์กลางของวงกลมจะต่ำและจะค่อยๆ เพิ่มขึ้นทีละน้อยตามแนวรัศมีของวงกลมจากนั้นก็จะค่อยๆ ลดลงตามแนวรัศมีของวงกลม ลักษณะของชุดข้อมูลนี้เป็นดังแสดงในรูปที่ 5.12



รูปที่ 5.12 แสดงชุดข้อมูลที่ความหนาแน่นของข้อมูลมีการเพิ่มขึ้นและลดลงตามแนวรัศมี

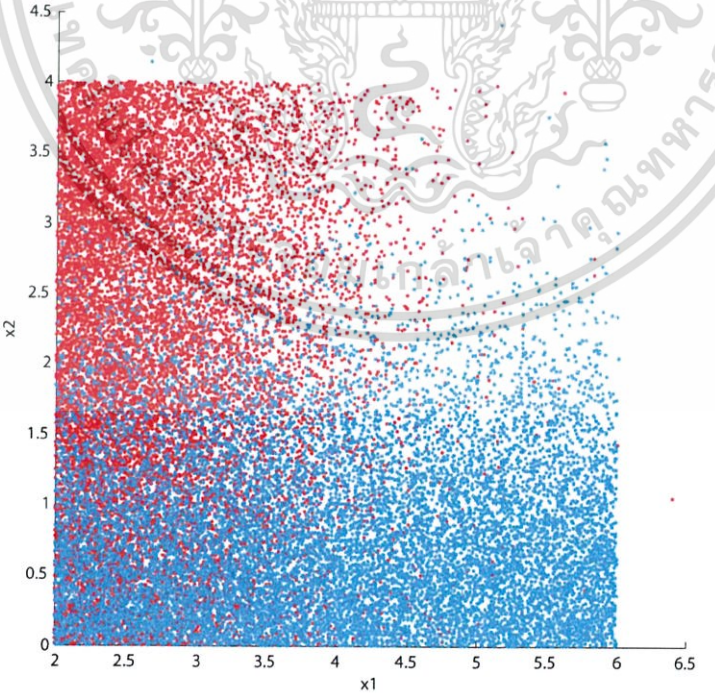


รูปที่ 5.13 แสดงตัวอย่างผลการจัดกลุ่มชุดข้อมูลที่ความหนาแน่นของข้อมูลมีการเพิ่มขึ้นและลดลงตามแนวรัศมี

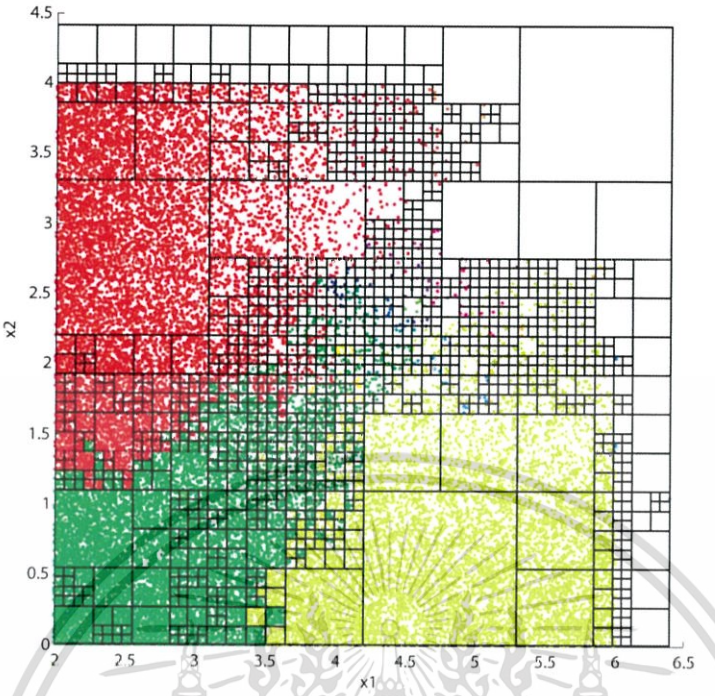
ตัวอย่างผลการจัดกลุ่มข้อมูลชุดนี้โดยใช้วิธีการแตกกริดแบบ GS2.1 เป็นดังแสดงในรูปที่ 5.13 ซึ่งจากรูปจะเห็นได้ว่าการแตกกริดไม่ได้ขึ้นอยู่กับความหนาแน่นของข้อมูล ทั้งนี้ก็เพราะว่าความหนาแน่นถูกนำมาใช้เพื่อกำจัดข้อมูลที่มีความหนาแน่นน้อยเท่านั้น จริงๆ แล้วการแตกกริดจะพิจารณาจากอัตราส่วนคลาสเป็นหลักเพราะว่าเป็นการจัดกลุ่มข้อมูลที่มีการซ้อนทับกัน ไม่ใช่การจัดกลุ่มข้อมูลตามความหนาแน่นของข้อมูล

จากผลการรวมกลุ่มข้อมูลจะเห็นว่าในบริเวณที่ข้อมูลมีการซ้อนทับกันจะได้กลุ่มข้อมูล 2 กลุ่ม ซึ่งก็เป็นไปตามผลการจัดกลุ่มที่ควรจะได้ ทั้งนี้ก็เนื่องจากในบริเวณที่ข้อมูลมีการซ้อนทับกันอัตราส่วนคลาสของข้อมูลจะค่อยๆ เปลี่ยนทีละน้อยตามความหนาแน่นของข้อมูลของแต่ละคลาสซึ่งมีการเปลี่ยนแปลงทีละน้อย ดังนั้นเมื่อความต่างของอัตราส่วนคลาสน้อยข้อมูลที่อยู่ใกล้กันก็เลยถูกจัดเป็นกลุ่มเดียวกันและขยายออกไปเป็นบริเวณกว้างจนกว่าจะไปถึงบริเวณที่อัตราส่วนคลาสมีการเปลี่ยนแปลงแบบก้าวกระโดด

ชุดข้อมูลที่ใช้ทดลองชุดที่สองเป็นชุดข้อมูลที่มีจำนวนคลาส 2 คลาส คลาสแรกการกระจายตัวของข้อมูลในแนวแกน x จะเป็นแบบลดลงแบบการกระจายแบบปกติ ส่วนแนวแกน y จะเป็นการกระจายแบบสม่ำเสมอ และสำหรับคลาสที่สองการกระจายตัวของข้อมูลในแนวแกน x จะเป็นการกระจายแบบสม่ำเสมอ ส่วนแนวแกน y จะเป็นการกระจายลดลงแบบการกระจายแบบปกติ ลักษณะของชุดข้อมูลนี้เป็นดังแสดงในรูปที่ 5.14



รูปที่ 5.14 แสดงชุดข้อมูลที่ความหนาแน่นของข้อมูลมีการลดลงตามแนวแกน x และตามแนวแกน y



รูปที่ 5.13 แสดงตัวอย่างผลการจัดกลุ่มชุดข้อมูลที่ความหนาแน่นของข้อมูลมีการลดลงตามแนวแกน x และตามแนวแกน y

สำหรับชุดข้อมูลนี้ความหนาแน่นของข้อมูลก็ค่อยๆ เปลี่ยนแปลงทีละน้อยเช่นเดียวกับชุดข้อมูลแรก นอกจากนี้อัตราส่วนคลาสก็ค่อยๆ เปลี่ยนแปลงที่ละน้อยเช่นเดียวกัน ดังนั้นจากการจัดกลุ่มข้อมูลจึงได้กลุ่มข้อมูลที่อัตราส่วนคลาสของพื้นที่ที่อยู่ใกลกันอาจจะมีความแตกต่างกันมากบ้าง ซึ่งก็สามารถอธิบายได้เช่นเดียวกันกับชุดข้อมูลแรก

บทที่ 6
สรุปและข้อเสนอแนะ

6.1 สรุป

วิทยานิพนธ์นี้ได้นำเสนอการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลมและโดยใช้กริดที่มีความละเอียดหลายระดับ และได้ทำการทดลองจัดกลุ่มข้อมูลโดยใช้ข้อมูลที่มีลักษณะการกระจายตัวแบบต่าง ๆ ซึ่งจากผลการทดลองพบว่าวิธีการจัดกลุ่มข้อมูลทั้งสองวิธีให้ผลการจัดกลุ่มข้อมูลที่ใกล้เคียงกับการจัดกลุ่มข้อมูลโดยใช้กริดที่มีขนาดเท่ากันหมด นอกจากนี้ยังใช้กริดหรือกลุ่มข้อมูลหน่วยแบบวงกลมน้อยกว่าการใช้กริดขนาดเท่ากันหมดด้วย ซึ่งนั่นก็หมายความว่าถ้ามีการออกแบบโครงสร้างข้อมูลที่ดีพอแล้ววิธีการที่นำเสนอในวิทยานิพนธ์นี้จะใช้หน่วยความจำน้อยกว่าการใช้กริดขนาดเท่ากันหมด แต่เมื่อพิจารณาด้านเวลาที่ใช้ในการประมวลผลกลับพบว่าวิธีการจัดกลุ่มข้อมูลโดยใช้กลุ่มข้อมูลหน่วยแบบวงกลมใช้เวลามากกว่าการใช้กริดขนาดเท่ากันหมด มีเพียงการจัดกลุ่มข้อมูลโดยใช้กริดที่มีความละเอียดหลายระดับเท่านั้นที่ใช้เวลาในการประมวลผลน้อยกว่าการใช้กริดขนาดเท่ากันหมด แต่ทั้งนี้ข้อมูลที่นำมาจัดกลุ่มจะต้องมีจำนวนมิติน้อยๆ เท่านั้น เพราะถ้าหากจำนวนมิติของข้อมูลมีมากๆ แล้วการใช้กริดทั้งสองวิธีก็จะใช้เวลาในการสร้างกริดมากกว่าการสร้างกลุ่มข้อมูลหน่วยแบบวงกลม นอกจากนี้ถ้าข้อมูลมีการกระจายตัวในลักษณะที่อัตราส่วนคลาสของพื้นที่ที่อยู่ใกล้กันค่อยๆ มีการเปลี่ยนแปลงทีละน้อยกลุ่มข้อมูลที่ได้ก็อาจจะมีพื้นที่ที่อยู่ใกล้กันที่อัตราส่วนคลาสมีความแตกต่างกันมากก็ได้

ข้อเสียของการใช้กริดที่มีความละเอียดหลายระดับอีกประการหนึ่งก็คือ การแตกกริดออกเป็น 2 ส่วนในแต่ละมิติ เพราะว่ากริดขนาดเล็กที่สุดอาจจะไม่ใช่ขนาดกริดที่เหมาะสมก็ได้ เช่น ถ้ากริดระดับที่ n มีความกว้าง 4 หน่วย ดังนั้นเมื่อแตกกริดต่อไป กริดระดับที่ n+1 ก็จะมีความกว้าง 2 หน่วย แต่สำหรับข้อมูลชุดนั้นขนาดกริดที่ให้ผลการจัดกลุ่มที่ดีอาจจะเป็น 3 หน่วยก็ได้

6.2 ข้อเสนอแนะ

สำหรับการจัดกลุ่มข้อมูลที่นำเสนอในวิทยานิพนธ์นี้ เพื่อที่จะให้ได้ผลการจัดกลุ่มข้อมูลที่เหมาะสม เรายังคงต้องอาศัยผู้ใช้งานในการปรับค่าพารามิเตอร์ต่าง ๆ แต่คงจะดีไม่น้อยถ้าหากเราไม่ต้องคอยปรับค่าพารามิเตอร์ต่าง ๆ เอง ดังนั้นจึงอยากขอเสนอแนวทางในการทำวิจัยต่อว่า จะเป็นไปได้หรือไม่ที่จะพัฒนาวิธีการที่นำเสนอนี้ให้มีความสามารถในการเรียนรู้เพื่อปรับเปลี่ยนค่าพารามิเตอร์ต่าง ๆ ได้เอง

สำหรับวิธีการที่ใช้กริดที่มีความละเอียดหลายระดับที่นำเสนอนี้ยังไม่สามารถนำไปใช้กับข้อมูลที่ความกว้างของแต่ละมิติมีความแตกต่างกันมากได้เพราะว่ากริดที่ได้จะมีลักษณะเป็นรูปสี่เหลี่ยมผืนผ้า ซึ่งก็จะส่งผลให้เกิดความผิดพลาดขึ้นมามากได้ วิธีแก้ไขก็คือการแตกกริดระดับแรกจะต้องไม่แตกเป็น 2 ส่วนในแต่ละมิติ แต่จะต้องแตกในลักษณะที่ทำให้กริดในระดับที่ 1 มีความกว้างในแต่ละมิติเท่ากันโดยประมาณ หลังจากนั้นการแตกกริดในระดับต่อไปก็สามารถแตกออกเป็น 2 ส่วนได้ตามปกติ

เอกสารอ้างอิง

[1] Christoph F. Eick, Nidal Zeidat, and Zhenghong Zhao. "Supervised Clustering – Algorithms and Benefits." Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004).

[2] S. H. Al-Harbiand and V. J. Rayward-Smith. "Adapting k-means for supervised clustering." Appl. Intell. (2006) 24. pp. 219–226.

[3] Thomas Finley and Thorsten Joachims. "Supervised Clustering with Support Vector Machines." Proceedings of the 22 nd International Conference on Machine Learning, Bonn, Germany, 2005

[4] Boontee Kruatrachue, Kulwarun Warunsin, and Kritawan Siriboon. "The Classified Method for Overlapping Data." ICCA 2004.

[5] ธนวัฒน์ ภัทรวรเมธ. 2548. "วิธีการจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้เทคนิคความหนาแน่น." วิทยานิพนธ์วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ บัณฑิตวิทยาลัย, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

[6] พัฒนพล รัตนพงษ์พร. 2549. "การจัดกลุ่มข้อมูลที่มีการซ้อนทับกันโดยใช้กริด." วิทยานิพนธ์วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ บัณฑิตวิทยาลัย, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

[7] Pang-Ning Tan, Micheal Steinbach, and Vipin Kumar. **Introduction to Data Mining. Pearson International Edition.** 2006.

ภาคผนวก

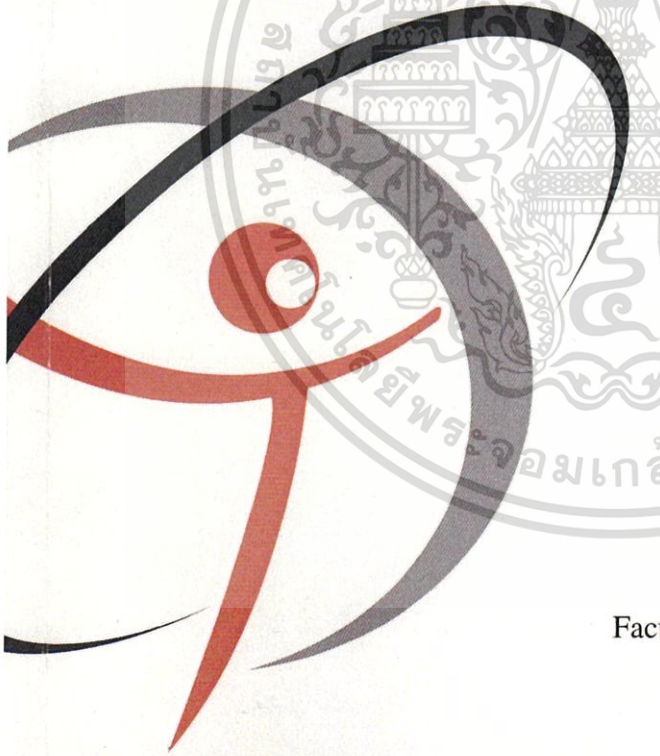
งานวิจัยที่ได้รับการตีพิมพ์

Proceedings of NCIT2008

The 2nd National Conference on Information Technology 2008

"การวิจัยเทคโนโลยีสารสนเทศเพื่อการพัฒนาประเทศที่ยั่งยืน"

6-7 พฤศจิกายน 2551



Faculty of Information Technology

Rangsit University

ISBN 978-974-377-856-8

The 2nd National Conference on Information Technology 2008 (NCIT 2008)

การแบ่งกลุ่มข้อมูลแบบมีผู้สอนโดยใช้ความแตกต่างระหว่างอัตราส่วนคลาส

SUPERVISED CLUSTERING USING DISSIMILARITY OF CLASS RATIO

บุญธีร์ เครือตราชู และ *กิตติคุณ นันตา*

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

อีเมล์: boontee@yahoo.com , setsunakami@yahoo.com

บทคัดย่อ

การแบ่งกลุ่มข้อมูลแบบมีผู้สอนซึ่งเป็นการแบ่งกลุ่มข้อมูลที่มีการจำแนกคลาสของข้อมูลมาใช้ด้วยนั่นทำให้เรารู้ไว้ก่อนและทั้งในฟีเจอร์สเปซมีจุดข้อมูลของคลาสไหนอยู่บ้าง ต่างกับการแบ่งกลุ่มข้อมูลแบบไม่มีผู้สอนที่ใช้อัลกอริทึมในปีเจอร์สเปซอย่างเดียวเท่านั้น เทคนิคการแบ่งกลุ่มข้อมูลแบบมีผู้สอนโดยทั่วไปนั้นจะมีเป้าหมายที่การหากลุ่มข้อมูลที่มีความบริสุทธิ์มากที่สุดเท่าที่จะเป็นไปได้ ซึ่งในบางกรณีการซ้อนทับกันมาก ๆ แล้วกลุ่มข้อมูลที่ได้ก็มีความบริสุทธิ์มากเฉพาะที่บางพื้นที่ที่มีความบริสุทธิ์มากแต่บางพื้นที่มีการผสมผสานระหว่างข้อมูลที่อยู่คนละคลาสกัน ในบทความนี้เราได้นำเสนอเทคนิคการแบ่งกลุ่มข้อมูลแบบมีผู้สอนที่สามารถบอกได้ว่าบริเวณใดในฟีเจอร์สเปซที่มีการปนกันของข้อมูลเกิดขึ้นและปนกันในสัดส่วนเท่าไร โดยใช้วิธีตรวจสอบจากอัตราส่วนของคลาสในพื้นที่ย่อย ๆ กล่าวคือ พื้นที่ที่อยู่ในบริเวณใกล้เคียงกันและมีส่วนผสมของข้อมูลที่ไม่แตกต่างกันมากนักจะถูกจัดให้เป็นกลุ่มเดียวกัน ดังนั้นกลุ่มข้อมูลที่ได้จากการแบ่งด้วยเทคนิคนี้จึงเป็นกลุ่มข้อมูลที่มีสัดส่วนของข้อมูลแต่ละคลาสเหมือนกันทั้งกลุ่ม

คำสำคัญ— การแบ่งกลุ่มข้อมูล, การแบ่งกลุ่มข้อมูลแบบมีผู้สอน, การแบ่งประเภทข้อมูลแบบไม่มีผู้สอน

ABSTRACT

Unlike traditional clustering, supervised clustering which applied on classified data lets we know the distribution of classes. Most of supervised clustering has the goal of identifying class-uniform clusters. What will happen if the classes enormously overlap? This paper proposes a novel supervised clustering technique with an ability of indentifying an overlapping region and its mixture. Connected regions with similar class ratio are merged together to forms a cluster. Hence, the cluster resulting from this technique has uniform mixture and can be of any shape.

Index Terms— Clustering, supervised clustering, overlapping class data, unsupervised classification

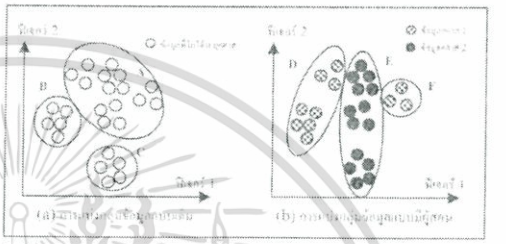
I. บทนำ

การแบ่งกลุ่มข้อมูล (clustering) หรือการวิเคราะห์กลุ่มข้อมูล (cluster analysis) เป็นวิธีการทางการคำนวณ (computation) อย่างหนึ่งที่มีบทบาทสำคัญทั้งในด้านการศึกษา การวิจัย และในสาขาวิชาชีพต่าง ๆ โดยทำหน้าที่เป็นเครื่องมือที่ช่วยอำนวยความสะดวกให้แก่งานในด้านนั้น ๆ ตัวอย่างของงานที่มีการนำอัลกอริทึมแบ่งกลุ่มข้อมูลไปใช้ ได้แก่ ชีววิทยา สถิติ การแพทย์ การรู้จำรูปแบบ (pattern recognition) การทำเหมืองข้อมูล (data mining) การสืบค้นข้อมูล (information retrieval) และการเรียนรู้ของเครื่อง (machine learning) เป็นต้น

ความหมายของการแบ่งกลุ่มข้อมูลก็คือ การจัดข้อมูลซึ่งอยู่ในรูปของจุดในฟีเจอร์สเปซ (feature space) ให้เป็นกลุ่มโดยพิจารณาจากความคล้ายกัน (similarity) ของข้อมูลเหล่านั้น หรือ จาก

ด้านหนึ่งของจุดข้อมูลในฟีเจอร์สเปซ (feature space) ซึ่งนั่นก็หมายความว่า ข้อมูลที่ถูกจัดให้อยู่ในกลุ่มเดียวกันย่อมต้องมีความคล้ายคลึงกัน (หรืออยู่ในด้านหนึ่งในลักษณะเดียวกัน) มากกว่าข้อมูลที่อยู่ในกลุ่มอื่น และหากจะมองว่าการแบ่งกลุ่มข้อมูลเป็นการแบ่งประเภทข้อมูล (classification) แบบหนึ่ง ก็สามารถทำได้เหมือนกัน คือ เป็นการแบ่งประเภทข้อมูลแบบไม่มีผู้สอน (unsupervised classification) ซึ่งเราสามารถนำมาใช้ในการกำหนดคลาส (class) ของข้อมูลได้โดยที่ไม่ต้องเรียนรู้จากชุดข้อมูลตัวอย่างซึ่งมีการบอกคลาสของข้อมูลมาด้วยแบบ แต่ทำการกำหนดคลาสของข้อมูลโดยใช้หลักการที่ว่า ข้อมูลที่มีความคล้ายคลึงกันก็ควรจะถูกจัดให้อยู่ในกลุ่มเดียวกัน และควรกำหนดคลาสเดียวกัน (class label) ให้กับข้อมูลเหล่านั้น ต่างไปจากการพูดถึงการแบ่งกลุ่มข้อมูลจะไม่มีการพูดถึงว่าข้อมูลอยู่ในคลาสใด เพียงแต่บอกว่าข้อมูลอยู่ในด้านหนึ่งในฟีเจอร์สเปซ สำหรับการระบุว่าเป็นคลาสใดจัดเป็นผลจากการจัดกลุ่ม เช่น รูปที่ 1 (a) จากด้านหนึ่งในฟีเจอร์สเปซสามารถแบ่งข้อมูลได้เป็น 3 กลุ่ม (3 คลาส) คือ กลุ่ม A B และ C

เดิมทีการที่ทำการแบ่งกลุ่มข้อมูลนั้นก็เพื่อต้องการหากลุ่มของข้อมูลที่มีความคล้ายกันเท่านั้น มีหลายครั้งเมื่อทำการแบ่งกลุ่มข้อมูลแล้วพบว่ามีจุดข้อมูลซึ่งอยู่ใกล้กันมากแต่กลับถูกจัดให้อยู่คนละกลุ่มกันและการกระจายตัวของข้อมูลเหล่านั้นก็มีลักษณะที่ไม่น่าจะแยกออกจากกันได้ ดังนั้นจึงได้มีการคิดหาวิธีการแบ่งกลุ่มข้อมูลที่สามารถจัดการกับข้อมูลที่ไม่สามารถแบ่งออกเป็นกลุ่มที่แยกจากกันอย่างชัดเจนได้ ซึ่งเทคนิคที่ว่านี้สามารถแบ่งได้เป็น 2 ประเภท คือ 1) การแบ่งกลุ่มข้อมูลแบบซ้อนทับกัน (overlapping clustering) ตัวอย่างเช่น [3], [4] เป็นต้น และ 2) การแบ่งกลุ่มข้อมูลแบบฟัซซี (fuzzy clustering) เช่น fuzzy c-mean [1] เป็นต้น หลักการของการแบ่งกลุ่มข้อมูลแบบซ้อนทับกันก็คือ จุดข้อมูลใด ๆ ก็ตามสามารถเป็นสมาชิกของกลุ่มข้อมูลได้มากกว่าหนึ่งกลุ่มขึ้นไป ส่วนการแบ่งกลุ่มข้อมูลแบบฟัซซีนั้น จุดข้อมูลใด ๆ จะถูกกำหนดให้เป็นสมาชิกของกลุ่มข้อมูลทุกกลุ่ม โดยใช้ค่าความเป็นสมาชิกของกลุ่ม (cluster membership) เป็นตัวกำหนดว่าจุดข้อมูลใดมีความเป็นสมาชิกของกลุ่มข้อมูลใดเท่าไร



รูปที่ 1 แสดงความแตกต่างของการแบ่งกลุ่มข้อมูลแบบเดิม (a) กับการแบ่งกลุ่มข้อมูลแบบมีผู้สอน (b)

แต่ในการจัดจำแนกข้อมูลจะรู้ก่อนแล้วว่าข้อมูลชุดนั้นมีกี่คลาส โดยข้อมูลเหล่านั้นจะได้ถูกทำการแบ่งประเภทไว้ก่อนแล้วในอีกฟีเจอร์สเปซหนึ่ง และสามารถนำมาช่วยในการแบ่งกลุ่มข้อมูลให้เหมาะสมขึ้นได้ เช่น รูปที่ 1 (b) ถ้าใช้คลาสเลเบิลด้วยก็จะได้กลุ่มข้อมูลที่แตกต่างกันหนึ่ง โดยข้อมูลที่อยู่คนละคลาสกันก็ไม่ควรจะอยู่ในกลุ่มเดียวกัน การแบ่งกลุ่มข้อมูลแบบนี้เรียกว่า การแบ่งกลุ่มข้อมูลแบบมีผู้สอน (supervised clustering)

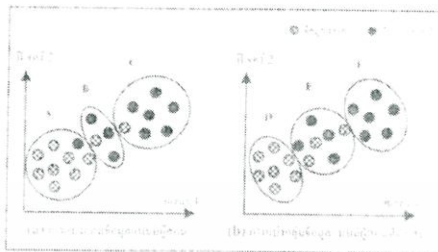
ข้อแตกต่างของการแบ่งกลุ่มข้อมูลแบบมีผู้สอนกับการแบ่งกลุ่มข้อมูลแบบเดิมก็คือ ข้อมูลที่มีความคล้ายคลึงกัน (โดยมีด้านหนึ่งในฟีเจอร์สเปซใกล้เคียงกัน) และมีคลาสเลเบิลเดียวกันเหมือนกัน ก็ควรจะถูกจัดให้อยู่ในกลุ่มเดียวกัน แต่ถ้าอยู่ในด้านหนึ่งมีค่าต่างกันแต่มีคลาสเลเบิลเดียวกันก็ควรจะจัดให้อยู่คนละกลุ่ม

[2] ได้เสนอว่า เกณฑ์ที่ใช้ในการประเมินคุณภาพของการแบ่งกลุ่มข้อมูลแบบมีผู้สอน มี 2 อย่าง คือ จำนวนกลุ่มข้อมูลที่ได้จากการแบ่ง กับ ความบริสุทธิ์ของกลุ่มข้อมูลเหล่านั้น กล่าวคือ การแบ่งกลุ่มข้อมูลแบบมีผู้สอนนั้นต้องการกลุ่มข้อมูลที่สมาชิก

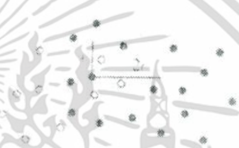
ภายในกลุ่มมีความคล้ายคลึงกันและมีสมาชิกที่เป็นคลาสใดคลาสหนึ่งมากกว่าสมาชิกที่เป็นคลาสอื่นมาก ๆ กลุ่มข้อมูลต้องมีความบริสุทธิ์มาก ๆ ดังแสดงในรูปที่ 2 (a) แต่ถ้าข้อมูลที่นำมาจัดกลุ่มนั้นมีการซ้อนทับกันมากแล้ว การที่จะแบ่งข้อมูลเหล่านั้นออกเป็นกลุ่มที่มีความบริสุทธิ์ของคลาสมาก ๆ จะเป็นสิ่งที่เป็นไปได้ยาก ดังนั้นจึงทำให้เกิดความคิดเรื่องการซ้อนทับกันของคลาสขึ้นมา

บทความนี้นำเสนอแนวคิดสำหรับแก้ปัญหาซ้อนทับกันของคลาส โดยมีหลักการ คือ ภายในกลุ่มข้อมูลเดียวกัน แต่ละพื้นที่ในกลุ่มข้อมูลนั้นจะต้องมีส่วนผสมของคลาสที่แตกต่างกัน ดังแสดงในรูปที่ 2 (b) ผลจากการแบ่งกลุ่มข้อมูลโดยใช้วิธีนี้นอกจากจะบอกว่าข้อมูลมีกี่กลุ่มและอยู่ตำแหน่งใดบ้างแล้ว ยังบอกอีกด้วยว่ากลุ่มข้อมูลนั้น ๆ มีส่วนผสมที่เป็นข้อมูลของแต่ละคลาสเดิมอยู่ในอัตราส่วนเท่าไร ซึ่งการจัดกลุ่มแบบนี้จะมีประโยชน์อย่างมากเมื่อต้องการทราบว่าข้อมูลกลุ่มใหม่มีความสัมพันธ์กับกลุ่มเดิมอย่างไร เช่น ในการรู้จำอักขระ (character recognition) ถ้าต้องการผลของการรู้จำอักขระ ถ้าคลาสเดิมของชุดข้อมูลเป็นคลาสอะไร เราสามารถทำได้โดยนำข้อมูลทั้งหมดมาจัดกลุ่มใหม่แล้วพิจารณาว่าชุดข้อมูลนั้นอยู่ในกลุ่มใหม่กลุ่มใด ซึ่งจะทำให้สามารถบอกได้ถูกต้องมากขึ้นว่าชุดข้อมูลนั้นควรเป็นข้อมูลของคลาสใด เช่น ในรูปที่ 2 (b) ถ้าชุดข้อมูลที่ต้องการทราบคลาสอยู่ในกลุ่มใหม่ D ก็สามารถระบุได้ว่าเป็นคลาส 1 เดิม แต่ถ้าอยู่กลุ่ม F ก็จะเป็นข้อมูลคลาส 2 เดิม แต่ถ้าเป็นกลุ่ม E ก็มีโอกาสที่จะเป็นได้ทั้งคลาส 1 และ คลาส 2 โดยมีโอกาสเท่ากับ 3/7 และ 4/7 ตามลำดับ

เนื้อหาที่จะนำเสนอต่อไป มีดังต่อไปนี้ หัวข้อที่ 2 กล่าวถึงรายละเอียดของเทคนิคการแบ่งกลุ่มข้อมูลแบบมีผู้สอนโดยใช้ความแตกต่างระหว่างอัตราส่วนคลาส หัวข้อที่ 3 อธิบายอัลกอริทึมของเทคนิคนี้ จากนั้นจึงนำเสนอผลการทดลองกับชุดข้อมูลสังเคราะห์ในหัวข้อที่ 4 และในหัวข้อที่ 5 จะเป็นการสรุปภาพรวมของงานวิจัยนี้



รูปที่ 2 แสดงการแบ่งกลุ่มข้อมูลแบบมีผู้สอนโดยทั่วไป (a) ไป การแบ่งกลุ่มข้อมูลแบบมีผู้สอนที่นำเสนอ (b)



รูปที่ 3 แสดงไฮเปอร์สเฟียร์ในฟีเจอร์สเปซ 2 มิติ

2. การแบ่งกลุ่มข้อมูลแบบมีผู้สอนโดยใช้ความแตกต่างระหว่างอัตราส่วนคลาส

เทคนิคการแบ่งกลุ่มข้อมูลแบบมีผู้สอนที่นำเสนอในบทความนี้เป็นเทคนิคที่ทำการแบ่งกลุ่มข้อมูลโดยใช้ไฮเปอร์สเฟียร์ในการแบ่งพื้นที่ในฟีเจอร์สเปซให้เป็นพื้นที่เล็ก ๆ โดยจะคัดเลือกไฮเปอร์สเฟียร์ที่มีจำนวนสมาชิกน้อยกว่าค่าคงที่ค่าหนึ่ง (*MinPts*) คือไฮเปอร์สเฟียร์ที่มีสมาชิกเป็นข้อมูลรบกวน (noise) และจะไม่นำไปใช้ในการสร้างกลุ่มข้อมูล การสร้างกลุ่มข้อมูลจะเป็นการรวมไฮเปอร์สเฟียร์ที่อยู่ติดกันและมีความแตกต่างของอัตราส่วนคลาส (สมการที่ 2) น้อยกว่าค่าคงที่ค่าหนึ่ง (*MaxDiff*) เข้าด้วยกัน

ไฮเปอร์สเฟียร์เป็นพื้นที่ในฟีเจอร์สเปซที่มีลักษณะเป็นทรงกลม พื้นที่ของไฮเปอร์สเฟียร์นั้นแบ่งออกเป็น 2 ชั้นคือ พื้นที่ชั้นในซึ่งเป็นพื้นที่ที่อยู่ภายในรัศมี r (r เป็นค่าที่กำหนดโดย

ผู้ใช้งาน) กับพื้นที่ชั้นนอกที่อยู่ระหว่าง r_1 กับ $2r_1$ ดังแสดงในรูปที่ 3 ไฮเปอร์สเฟียร์ลำดับที่ j นิยามโดย

$$H_j = \{C, I, O, N\} \quad (1)$$

สำหรับทุก $j = 0,1,\ldots,m$ เมื่อ C คือจุดศูนย์กลางของไฮเปอร์สเฟียร์นั้น I คือเซตของสมาชิกที่อยู่ในพื้นที่ชั้นในของไฮเปอร์สเฟียร์นั้น O คือเซตของสมาชิกที่อยู่ในพื้นที่ชั้นนอกของไฮเปอร์สเฟียร์นั้น และ N คือเซตของคู่ลำดับ หมายถึง ไฮเปอร์สเฟียร์ข้างเคียงกับระยะห่างระหว่างไฮเปอร์สเฟียร์นั้นกับไฮเปอร์สเฟียร์ข้างเคียง (นิยามที่ 3)

นิยามที่ 1 (อัตราส่วนคลาส) อัตราส่วนคลาสของพื้นที่ A ใดๆ หมายถึง อัตราส่วนระหว่างจำนวนข้อมูลของแต่ละคลาสที่อยู่ในพื้นที่นั้นกับจำนวนข้อมูลทั้งหมดของคลาสนั้นในพื้นที่ A อัตราส่วนคลาสของพื้นที่ A นิยามได้ดังนี้

$$CR(A) = \{r_1^A, r_2^A, \ldots, r_n^A\} \quad (2)$$

โดยที่

$$r_i^A = \frac{n_i^A}{n_i} \quad (3)$$

เมื่อ n_i^A คือจำนวนชุดข้อมูลของคลาส i ที่อยู่ในพื้นที่ A

นิยามที่ 2 (ความแตกต่างระหว่างอัตราส่วนคลาส: ความแตกต่างระหว่างอัตราส่วนคลาสเป็นค่าที่แสดงถึงความแตกต่างระหว่างอัตราส่วนคลาสของสองพื้นที่ใดๆ ความแตกต่างระหว่างอัตราส่วนคลาสของพื้นที่ A กับพื้นที่ B หาได้จาก

$$DCR(A,B) = |CR(A) - CR(B)| = \sum_{i=1}^{n} |r_i^A - r_i^B| \quad (4)$$

นิยามที่ 3 (ไฮเปอร์สเฟียร์ข้างเคียง) ไฮเปอร์สเฟียร์ A จะเป็นไฮเปอร์สเฟียร์ข้างเคียงของไฮเปอร์สเฟียร์ B ก็ต่อเมื่อ ระยะห่างระหว่างจุดศูนย์กลางของไฮเปอร์สเฟียร์ทั้งสองน้อยกว่าค่าคงที่ค่าหนึ่ง (ในบทความนี้เรากำหนดให้มีค่าเท่ากับ $4r_1$)

3. อัลกอริธึม

อัลกอริธึมการแบ่งกลุ่มข้อมูลที่นำเสนอนี้สามารถแบ่งออกได้เป็น 2 ขั้นตอน คือ *ขั้นตอนเริ่มต้น (Initialization)* ซึ่งเป็นขั้นตอนการสร้างไฮเปอร์สเฟียร์ขึ้นในฟีเจอร์สเปซ และ *ขั้นตอนการรวมไฮเปอร์สเฟียร์ (Merging)* รูปที่ 4 แสดงตัวอย่างของการสร้างไฮเปอร์สเฟียร์

3.1. ขั้นตอนเริ่มต้น

อัลกอริธึมสำหรับขั้นการไฮเปอร์สเฟียร์เมื่อมีการป้อนข้อมูลอินพุตเข้าไป มีขั้นตอนดังนี้คือไปนี้

1. เมื่อมีข้อมูลอินพุต X_i ถูกป้อนเข้าไป จะทำการคำนวนหาระยะห่างระหว่างอินพุต X_i นั้นกับไฮเปอร์สเฟียร์ที่มีอยู่ในขณะนั้นทั้งหมด ซึ่งถ้าพบว่าไฮเปอร์สเฟียร์ใดจะกลายเป็นไฮเปอร์สเฟียร์ข้างเคียง (ดูนิยามที่ 3) ของไฮเปอร์สเฟียร์ใหม่ที่จะถูกสร้างขึ้นเมื่ออินพุต X_i นั้นเข้ามา ก็ให้เป็นสมาชิกของไฮเปอร์สเฟียร์นั้นๆ โดยที่จะทำการสร้างคู่ลำดับ $(j, dist(C_j, X_i))$ ขึ้นเก็บไว้ โดยที่ j คือหมายเลขลำดับของไฮเปอร์สเฟียร์นั้น และ $dist(C_j, X_i)$ คือระยะห่างระหว่างจุดศูนย์กลางของไฮเปอร์สเฟียร์นั้นกับอินพุต X_i (คู่ลำดับนี้จะทำให้รู้ว่าไฮเปอร์สเฟียร์ที่สร้างขึ้นมาใหม่นั้นมีไฮเปอร์สเฟียร์ไหนเป็นไฮเปอร์สเฟียร์ข้างเคียงบ้าง)
2. ระยะห่างที่ได้คำนวนในข้อหนึ่ง จะทำให้รู้ว่าอินพุต X_i นั้นอยู่ในพื้นที่ชั้นในของไฮเปอร์สเฟียร์ H_j ไหนหรือไม่ ซึ่งถ้าพบว่าอินพุตนั้นอยู่ในพื้นที่ชั้นในของไฮเปอร์สเฟียร์ใดก็จะกำหนดให้เป็นสมาชิกในพื้นที่ชั้นในของไฮเปอร์สเฟียร์ H_j นั้น ดังรูปที่ 4 (c) (ถ้าอินพุตใหม่ป้อนเข้ามาอีกก็กลับไปทำงานขั้นตอนที่

รูปที่ 4 แสดงตัวอย่างการสร้างไฮเปอร์สเฟียร์

3. ถ้าอินพุทนั้นไม่ได้อยู่ในพื้นที่ชั้นในของไฮเปอร์สเฟียร์ใดเลย แต่กลับอยู่ในพื้นที่ชั้นนอกของไฮเปอร์สเฟียร์ใด ก็ตาม ดังรูปที่ 4 (a) ก็จะยังไม่กำหนดให้ X_i เป็นสมาชิกของไฮเปอร์สเฟียร์ใด แต่จะเก็บพักไว้ก่อน ทั้งนี้จะมีการเก็บระยะห่างระหว่างอินพุทนั้นกับไฮเปอร์สเฟียร์ที่ใกล้ที่สุด และ หมายเลขไฮเปอร์สเฟียร์สุดท้ายที่มีอยู่ในขณะนั้น ไว้ด้วย (ถ้ายังมีอินพุทใหม่เป็นอินเข้ามาอีกก็กลับไปทำตามขั้นตอนที่ 1)
4. ถ้าพบว่าอินพุท X_i นั้นไม่ได้อยู่ทั้งในพื้นที่ชั้นในและชั้นนอกของไฮเปอร์สเฟียร์ใดเลย ก็ทำการสร้างไฮเปอร์สเฟียร์ใหม่ขึ้นมา โดยใช้อินพุท X_i นั้นเป็นจุดศูนย์กลาง ดังรูปที่ 4 (b)
5. ทำซ้ำขั้นที่ 1 จนกว่าจะไม่มีข้อมูลป้อนเข้ามาอีก
6. นำอินพุทที่ได้เก็บไว้ในขั้นตอนที่ 3 มากำหนดเป็นสมาชิกของไฮเปอร์สเฟียร์ โดยกำหนดให้เป็นสมาชิกของไฮเปอร์สเฟียร์ที่ใกล้ที่สุด ดังรูปที่ 4 (c), (d) และ (e)

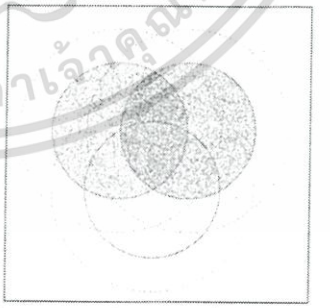
3.2. ขั้นตอนการรวมไฮเปอร์สเฟียร์

การรวมไฮเปอร์สเฟียร์ให้เป็นกลุ่มข้อมูลมีขั้นตอนดังต่อไปนี้

1. เลือกไฮเปอร์สเฟียร์ใดก็ได้ที่มีจำนวนสมาชิกมากกว่าหรือเท่ากับ $MinPts$ มาเป็นจุดเริ่มต้นสำหรับการรวม โดยนำไฮเปอร์สเฟียร์นั้นไปใส่ในลิสต์ L แล้วทำการสร้างกลุ่มข้อมูลขึ้นมาใหม่และกำหนดให้ไฮเปอร์สเฟียร์นั้นเป็นสมาชิกของกลุ่มข้อมูลที่สร้างขึ้นใหม่นี้
2. สำหรับแต่ละไฮเปอร์สเฟียร์ H_i ในลิสต์ L ทำการตรวจไฮเปอร์สเฟียร์ข้างเคียง H_j ของ H_i ว่ามีจำนวนสมาชิกมากกว่าหรือเท่ากับ $MinPts$ และความแตกต่างระหว่างอัตราส่วนคลาสของกลุ่มข้อมูลนี้กับไฮเปอร์สเฟียร์ H_i น้อยกว่าหรือเท่ากับ $Maxdiff$ หรือไม่ ถ้าเงื่อนนี้เป็นจริงก็จะกำหนดให้ไฮเปอร์สเฟียร์ H_j เป็นสมาชิกของกลุ่มข้อมูลนี้ และเพิ่มไฮเปอร์สเฟียร์ H_j เข้าไปในลิสต์ L แต่ถ้า H_j มีจำนวนสมาชิกน้อยกว่า $MinPts$ ที่กำหนดให้ H_j เป็นไฮเปอร์สเฟียร์ของข้อมูลรบกวน ทำขั้นตอนนี้กับไฮเปอร์สเฟียร์ข้างเคียง H_j ของ H_i จนครบทั้งหมดแล้วจึงเอาไฮเปอร์สเฟียร์ H_i ออกจากลิสต์ L
3. ทำขั้นตอนที่ 2 ซ้ำจนลิสต์ L ว่าง
4. ทำซ้ำขั้นตอนที่ 1 จนไม่มีไฮเปอร์สเฟียร์ที่ไม่ใช่ไฮเปอร์สเฟียร์ของข้อมูลรบกวนที่ยังไม่ได้รวมเหลืออยู่อีก

4. การทดลอง

การทดลองที่นำเสนอนี้เป็นการทดลองเบื้องต้นที่ทำกับชุดข้อมูลสังเคราะห์ที่มีจำนวนมิติเท่ากับ 2 มิติ ดังแสดงในรูปที่ 4 ชุดข้อมูลนี้ประกอบไปด้วยจุดข้อมูลที่กระจุกตัวรวมกันเป็นรูปร่างแปลก 3 รูงที่วางซ้อนกันอยู่ โดยที่วงกลมแต่ละวงจะมีความหนาแน่นที่มีจุดข้อมูลกระจุกตัวกันอย่างหนาแน่นซึ่งจุดข้อมูลในบริเวณนี้เป็นจุดข้อมูลจริงกับบริเวณจุดข้อมูลที่เป็นข้อมูลรบกวนกระจายตัวอยู่อย่างเบาบาง ทั้งข้อมูลจริงและข้อมูลรบกวนมีการกระจายตัวแบบยูนิฟอร์ม (uniform distribution) ผลของการแบ่งกลุ่มข้อมูลด้วยวิธีที่นำเสนอนี้โดยใช้ $\varepsilon = 0.01$, $MaxDiff = 0.45$ และ $MinPts = 10$ เป็นค่าแสดงในรูปที่ 5



รูปที่ 4 แสดงชุดข้อมูลสังเคราะห์ที่ใช้ในการทดลอง

จากรูปที่ 5 จะเห็นได้ว่าเทคนิคนี้นำเสนอนี้ให้ผลการแบ่งกลุ่มข้อมูลที่น่าพอใจสำหรับชุดข้อมูลสังเคราะห์ในรูปที่ 4 และสามารถขยายออกพื้นที่ที่มีการซ้อนทับกันของข้อมูลได้ตามวัตถุประสงค์ ถึงแม้ว่าผลการแบ่งกลุ่มที่นำมาแสดงนี้จะได้จากการทดลองปรับค่าพารามิเตอร์ต่าง ๆ หลายครั้งและการทดลองนี้ยังเป็นการทำกับข้อมูลสังเคราะห์อยู่แต่ก็นับได้ว่าเป็นจุดเริ่มต้นที่ดีซึ่งจะได้ใช้เป็นแนวทางในการพัฒนาต่อไป

รูปที่ 5 แสดงผลการแบ่งกลุ่มข้อมูลของชุดข้อมูลในรูปที่ 4

5. สรุปงานวิจัย

บทความนี้ได้กล่าวถึงปัญหาการซ้อนทับกันของคลาสที่เกิดขึ้นในการแบ่งกลุ่มข้อมูลแบบมีผู้สอน และได้เสนอทางเลือกในการแก้ปัญหาโดยเสนอให้ทำการระบุพื้นที่ที่มีการซ้อนทับกันของคลาส โดยการขยายขอบเขตกลุ่มข้อมูลนี้จุดข้อมูลอยู่ในกลุ่มอยู่ใกล้กันและแต่ละพื้นที่ภายในกลุ่มมีอัตราส่วนคลาสของข้อมูลใกล้เคียงกัน เทคนิคการแบ่งกลุ่มข้อมูลแบบมีผู้สอนโดยใช้ความแตกต่างระหว่างอัตราส่วนคลาสที่นำเสนอในบทความนี้ยังอยู่ในขั้นต้นของการพัฒนาและยังต้องทำการปรับปรุงอีกในหลายประเด็น และต้องทดลองปรับเทียบกับวิธีการอื่นและกับข้อมูลจริง

6. เอกสารอ้างอิง

[1] J.C. Bezdek, R. Ehrlich, W. Full, "FCM: the fuzzy c-means clustering algorithm," *Computers and Geosciences*, Vol. 10, 1984, pp. 191–203.

[2] C. Frick, N. Zeidat, and Z. Zhao, "Supervised Clustering – Algorithms and Benefits," in *Proc. ICTAI 2004*, Boca Raton, Florida, November 2004.

[3] Y.-I. Chen, H.-J. Hu, "An overlapping cluster algorithm to provide non-exhaustive clustering," *European Journal of Operational Research* 173, 2006, pp. 762–780.

[4] A. Banerjee, C. Krumpelman, J. Ghosh, S. Basu, and R. J. Mooney, "Model-based Overlapping Clustering," in *Proc. KDD 2005*, Chicago, IL, August 2005, pp. 532-537.

ประวัติผู้เขียน

ชื่อ-นามสกุล นายกิตติคุณ นันตา

วันเดือนปีเกิด 18 มิถุนายน พ.ศ. 2525 ที่จังหวัดเชียงใหม่

ที่อยู่ 58 หมู่ 6 บ้านใหม่โพธิ์งาม ต.บ้านหลวง อ.แม่อาย จ.เชียงใหม่ 50280

โทร. 086-1798432

ประวัติการศึกษา

พ.ศ. 2544 จบการศึกษาระดับมัธยมศึกษาจากโรงเรียนรังษีวิทยา อ.ฝาง จ.เชียงใหม่

พ.ศ. 2549 จบการศึกษาวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ จากคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง