การเลือกโปรโตไทป์โดยวิธีการจัดกลุ่มซ้อนฐานแบบมีการสอน

PROTOTYPE SELECTION BY SUPERVISED CLUSTERING

กมลรัฐ กางการ
KAMONNAT KANGKAN

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2559

การเลือกโปรโตไทป์โดยวิธีการจัดกลุ่มข้อมูลแบบมีการสอน

PROTOTYPE SELECTION BY SUPERVISED CLUSTERING

กมลณัฐ กางการ

KAMONNAT KANGKAN

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2550

PROTOTYPE SELECTION BY SUPERVISED CLUSTERING

KAMONNAT KANGKAN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN COMPUTER ENGINEERING

SCHOOL OF GRADUDATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2007

หัวข้อวิทยานิพนธ์	การเลือกโปรโตไทป์โดยวิธีการจัดกลุ่มข้อมูลแบบมีการสอน
นักศึกษา	นายกมลณัฐ กางการ
รหัสนักศึกษา	47060829
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
พ.ศ.	2550
อาจารย์ที่ปรึกษาวิทยานิพนธ์	รศ.ดร.บุญธีร์ เครือตราชู

บทคัดย่อ

วิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule ถูกนำมาใช้อย่างแพร่หลายทั้งในเชิงการศึกษาวิจัยและการนำไปประยุกต์ใช้ในงานด้านการรู้จำรูปแบบข้อมูล ทั้งนี้เพราะวิธีการนี้เป็นวิธีที่เรียบง่ายและมีประสิทธิภาพในการแยกแยะข้อมูลที่ดี อย่างไรก็ตามในกรณีที่โปรโตไทป์อ้างอิงของวิธีการนี้มีจำนวนมาก วิธีการนี้ต้องพบกับปัญหาการใช้การคำนวนสูงและปัญหาการใช้หน่วยความจำในปริมาณสูง วิทยานิพนธ์นี้นำเสนอวิธีการเลือกโปรโตไทป์อ้างอิงจากตัวแทนของชุดข้อมูลตั้งต้นโดยใช้หลักการการจัดกลุ่มข้อมูลแบบมีการสอน ซึ่งมีเป้าหมายเพื่อลดจำนวนของโปรโตไทป์อ้างอิงลงโดยที่ยังคงความสามารถในการแยกแยะข้อมูลให้ใกล้เคียงกับการใช้ชุดข้อมูลตั้งต้นทั้งหมดเป็นโปรโตไทป์อ้างอิง จากผลการทดลองได้แสดงให้เห็นว่าวิธีการเลือกโปรโตไทป์อ้างอิงที่ได้นำเสนอนี้สามารถลดจำนวนโปรโตไทป์อ้างอิงให้กับชุดข้อมูลการรู้จำที่มีมิติและขนาดที่หลากหลายได้อย่างมีประสิทธิภาพ

Thesis Title	Prototype Selection by Supervised Clustering
Student	Mr. Kamonnat Kangkan
Student ID	47060829
Degree	Master of Engineering
Program	Computer Engineering
Year	2007
Thesis Advisor	Assoc. Prof. Dr. Boontee Kruatrachue

ABSTRACT

The nearest neighbor rule assigns an unclassified sample to the same class as the nearest classified samples (reference prototypes). Over the last 40 years, this simple classification rule has been intensively used in board range of pattern recognition applications. In contrast to its simplicity, the rule has a good behavior when applied to non-trivial problems. However, in case of large quantity of reference prototypes, the nearest neighbor rule requires large memory space and expensive computation time. This thesis proposes a condensing prototype selection algorithm, which is based on supervised clustering schemes. The obtained prototype set size is significantly reduced from the original dataset size but it preserves consistency property with the original dataset. The experimental results using the proposed technique show good prototype-reducing performance over a variety of recognition datasets.

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญตาราง (ต่อ)

ตารางที่ หน้า

สารบัญตาราง (ต่อ)

ตารางที่ หน้า

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

วิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule (NN rule) ถือว่าเป็นวิธีการแยกแยะข้อมูลโดยใช้ระยะทางในการตัดสินใจ (distance-based classification rules) ซึ่งเป็นวิธีการที่ไม่จำเป็นต้องทราบพารามิเตอร์ด้านสถิติที่อธิบายกลุ่มข้อมูล (non parametric approach) จึงถูกนำไปใช้อย่างแพร่หลายตลอด 40 กว่าปีที่ผ่านมา ทั้งในเชิงการศึกษาวิจัยและการนำไปใช้งานจริง โดยเฉพาะในงานด้านการรู้จำรูปแบบของข้อมูล (pattern recognition) ทั้งนี้ก็เพราะกฎนี้มีคุณสมบัติที่น่าสนใจ 2 ประการดังนี้ คุณสมบัติข้อแรกคือวิธีการนี้มีหลักการที่เรียบง่ายไม่ซับซ้อนสามารถทำความเข้าใจได้ง่ายและการเขียนโปรแกรมก็ทำได้โดยง่าย ซึ่งตามหลักการแล้ววิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule ต้องการกระบวนการทำงานเพียง 2 กระบวนการหลักนั่นคือ กระบวนการในการเก็บข้อมูลตั้งต้นทั้งหมดไว้ใช้เป็นโปรโตไทป์อ้างอิงและกระบวนการชี้เฉพาะโปรโตไทป์อ้างอิงตัวที่อยู่ใกล้กับออบเจ็คที่ต้องการทราบชนิดมากที่สุดเท่านั้น ส่วนคุณสมบัติข้อที่สองคือ วิธีการนี้มีประสิทธิภาพในการแยกแยะข้อมูลที่ดี ซึ่งในทางทฤษฎีแล้วเมื่อ nearest neighbor rule มีจำนวนโปรโตไทป์สำหรับอ้างอิงมากพอ(ลู่เข้าสู่อนันต์) กฎนี้จะมีความสามารถในการแยกแยะข้อมูลได้ดีไม่น้อยไปกว่าวิธีการแยกแยะข้อมูลอื่นๆ [1][2]

อย่างไรก็ตามวิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule ก็มีข้อด้อยอยู่ด้วยเช่นกัน ในกรณีที่กฎ nearest neighbor rule มีโปรโตไทป์อ้างอิงเป็นจำนวนมาก ซึ่งเป็นผลทำให้เกิดข้อด้อยที่สำคัญ 2 ประการตามมาดังนี้ ข้อด้อยแรกคือปัญหาความต้องการมีหน่วยความจำเป็นจำนวนมากเพื่อให้เพียงพอต่อการเก็บข้อมูลของโปรโตไทป์อ้างอิงที่มีอยู่ทั้งหมด (high memory demands problem) ส่วนข้อด้อยที่สองคือปัญหาความจำเป็นต้องใช้การคำนวนในปริมาณสูง (high computational demands problem) เพราะกระบวนการชี้เฉพาะโปรโตไทป์อ้างอิงที่ใกล้กับออบเจ็คที่ต้องการทราบชนิดมากที่สุดนั้นก็คือกระบวนในการค้นหาโปรโตไทป์อ้างอิงตัวที่มีระยะห่างน้อยที่สุดจากออบเจ็คที่ต้องการทราบชนิดนั่นเอง ซึ่งการทำงานของกระบวนการนี้ประกอบไปด้วยการคำนวนหาระยะทางระหว่างออบเจ็คที่ต้องการทราบชนิดกับโปรโตไทป์อ้างอิงที่มีอยู่ทั้งหมดทุกตัว โดยใช้ฟังก์ชันการคำนวนระยะทาง (distance function) ที่ได้กำหนดไว้ก่อนแล้ว ซึ่งจะเห็นได้ว่าขั้นตอนการแยกแยะข้อมูล (classification phase) ของวิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule นี้จะมีขนาดใหญ่ขึ้น (huge classification phase) ตามจำนวนของโปรโตไทป์อ้างอิง

เป็นผลทำให้วิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule ต้องใช้เวลาในการทำงานนาน เมื่อมีโปรโตไทป์อ้างอิงเป็นจำนวนมาก

การเลือกโปรโตไทป์ (prototype selection) เป็นแนวทางหนึ่งที่ถูกนำมาใช้เพื่อแก้ไขปัญหาการสิ้นเปลืองหน่วยความจำและการคำนวนของวิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule ซึ่งการเลือกโปรโตไทป์สามารถแบ่งตามเป้าหมายได้เป็น 2 ประเภท [3] คือ การเลือกโปรโตไทป์เพื่อกำจัดข้อมูลที่ไม่เข้าพวก (editing prototype selection) และการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิง (condensing prototype selection) โดยเป้าหมายหลักของการเลือกโปรโตไทป์เพื่อกำจัดข้อมูลที่ไม่เข้าพวก (editing prototype selection) คือการกำจัดข้อมูลที่อยู่โดดไม่เข้าพวก (outliers) ซึ่งข้อมูลเหล่านี้โดยส่วนใหญ่จะเป็นข้อมูลที่ผิดพลาด (noise) ถ้าเราเก็บข้อมูลพวกนี้ซึ่งมีเป็นจำนวนมากไว้เป็นโปรโตไทป์อ้างอิงทั้งหมดจะทำให้การแยกแยะข้อมูลมีผลลัพท์ที่ผิดพลาดจากความเป็นจริงมาก หลังจากเราได้กำจัดข้อมูลที่ไม่เข้าพวกแล้วจะเป็นผลทำให้ข้อมูลที่เหลือวางตัวอยู่ในลักษณะเป็นกลุ่มก้อนซึ่งแต่ละกลุ่มมีข้อมูลชนิดเดียวกันเท่านั้น (cluster groups of homogenous prototypes) ซึ่งเป็นผลทำให้การแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule ทำงานได้ผลดีมากขึ้น [2] ส่วนเป้าหมายหลักของการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิง (condensing prototype selection) คือ การลดจำนวนโปรโตไทป์อ้างอิงของกฎ nearest neighbor rule ลงให้มากที่สุดเท่าที่จะทำได้โดยที่ไม่สูญเสียความสามารถในการแยกแยะข้อมูลไปมากนัก เพื่อนำไปช่วยแก้ปัญหาการสิ้นเปลืองหน่วยความจำและการคำนวนที่เกิดขึ้นกับวิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule ที่มีโปรโตไทป์อ้างอิงจำนวนมาก ซึ่งการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิง (condensing prototype selection) สามารถแบ่งย่อยตามลักษณะของโปรโตไทป์ผลลัพท์ ได้เป็น 2 ประเภท[3] ดังนี้ การเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (selection-condensing prototype selection) และการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นเซ็ตของโปรโตไทป์ที่สร้างขึ้นใหม่ (replacement-condensing prototype selection) ข้อดีประการหนึ่งของการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (selection-condensing prototype selection) คือสามารถใช้ได้กับชุดข้อมูลที่ไม่สามารถสร้างโปรโตไทป์ขึ้นมาใหม่ได้ เช่นข้อมูลลายมือเขียนที่มีลักษณะแบบ featureless แต่การเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นเซ็ตของโปรโตไทป์ที่สร้างขึ้นใหม่ (replacement-condensing prototype selection) ไม่สามารถใช้ได้

วิทยานิพนธ์นี้ได้นำเสนอวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (selection-condensing prototype selection) โดยใช้หลักการการจัดกลุ่มข้อมูลแบบมีการสอนมาช่วยในการพิจารณาเลือกตัวแทนของกลุ่มข้อมูล ซึ่งตัวแทนของกลุ่มข้อมูลที่เลือกได้จะถูกนำไปใช้เป็นโปรโตไทป์อ้างอิงแทนการใช้กลุ่มข้อมูลทั้งหมดเป็นโปรโต

ไทป์อ้างอิง โดยที่ยังคงความสามารถในการแยกแยะข้อมูลให้ใกล้เคียงเดิม พร้อมกันนั้นได้นำเสนอผลการทดลองเปรียบเทียบประสิทธิภาพในการลดจำนวนโปรโตไทป์ของวิธีการเลือกโปรโตไทป์ที่ได้นำเสนอกับวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (selection-condensing prototype selection) วิธีการอื่นๆ

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

ความมุ่งหมายและวัตถุประสงค์ของวิทยานิพนธ์นี้ เพื่อศึกษาหาวิธีการเลือกโปรโตไทป์เพื่อการลดจำนวนโปรโตไทป์อ้างอิงลงแต่ยังคงความสามารถในการแยกแยะข้อมูลให้ใกล้เคียงกับการใช้ข้อมูลตั้งต้นทั้งหมดเป็นโปรโตไทป์อ้างอิง เพื่อช่วยในการแก้ปัญหาความต้องการหน่วยความจำและความสามารถในการคำนวนในปริมาณที่สูงที่เกิดขึ้นกับวิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule ที่มีโปรโตไทป์อ้างอิงจำนวนมาก

1.3 สมมุติฐานของการศึกษา

เนื่องจากการที่วิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule เก็บชุดข้อมูลที่ใช้สอน (“training dataset” บ้างก็เรียกข้อมูลตั้งต้น “original dataset”) ทั้งหมดเป็นโปรโตไทป์อ้างอิง ดังนั้นในกรณีที่ชุดข้อมูลที่ใช้สอนมีขนาดใหญ่จึงทำให้เกิดปัญหาความต้องการหน่วยความจำและความสามารถในการคำนวนในปริมาณที่สูง ซึ่งการเก็บข้อมูลใช้สอนทั้งหมดเป็นโปรโตไทป์อ้างอิงนั้นเป็นการกระทำที่ก่อให้เกิดความซ้ำซ้อนและสิ้นเปลืองทั้งในเชิงหน่วยความจำและการคำนวน ถ้าหากมีวิธีการที่มีความสามารถพิจารณาเลือกเฉพาะตัวแทนของข้อมูลตั้งต้นไปเป็นโปรโตไทป์อ้างอิงได้อย่างมีประสิทธิภาพ จะช่วยลดความซ้ำซ้อนของโปรโตไทป์อ้างอิงเป็นผลให้เราสามารถลดจำนวนโปรโตไทป์อ้างอิงลงได้มากแต่ยังคงความสามารถในการแยกแยะข้อมูลให้ใกล้เคียงกับการใช้ข้อมูลตั้งต้นทั้งหมดเป็นโปรโตไทป์อ้างอิง

1.4 ขอบเขตของการศึกษา

ขอบเขตการศึกษาหลักของวิทยานิพนธ์นี้คือ การศึกษาการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของข้อมูลตั้งต้น (selection-condensing prototype selection) และได้นำเสนอวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนซึ่งถือได้ว่าเป็นวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้นวิธีการหนึ่ง

1.5 ขั้นตอนของการศึกษา

ศึกษางานวิจัยที่เกี่ยวกับการเลือกโปรโตไทป์จากเอกสารที่ได้มีการนำเสนอมาก่อน เพื่อให้เห็นภาพรวมของงานวิจัยด้านการเลือกโปโตไทป์ แล้วศึกษาเพิ่มเติมเฉพาะในหัวข้อการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของข้อมูลตั้งต้น (selection-condensing prototype selection) จากนั้นศึกษาให้เข้าใจถึงหลักการการจัดกลุ่มข้อมูลแบบมีการสอนแล้วนำหลักการจัดกลุ่มข้อมูลแบบมีการสอนมาใช้ในการเลือกโปรโตไทป์เพื่อให้ได้วิธีการเลือกโปรโตไทป์ที่ดีขึ้น ทำการทดลองเพื่อประเมินประสิทธิภาพในการลดจำนวนโปรโตไทป์ของวิธีการใหม่และทำการทดลองเพื่อประเมินประสิทธิภาพในการแยกแยะข้อมูลของโปรโตไทป์ที่เลือกได้ด้วยวิธีการใหม่ สรุปและวิเคราะห์ผลการทดลอง

1.6 รายละเอียดในแต่ละบท

ในวิทยานิพนธ์ฉบับนี้แบ่งเนื้อหาการนำเสนอออกเป็น 6 บทดังนี้

บทที่ 1 กล่าวถึงความเป็นมาและความสำคัญของปัญหา วัตถุประสงค์และขอบเขตของงานวิจัย

บทที่ 2 กล่าวถึงพื้นฐานการแยกแยะข้อมูลและการเลือกโปรโตไทป์เพื่อให้เข้าใจความหมายของการเลือกโปรโตไทป์ประเภทต่างๆและคุณสมบัติที่เกี่ยวข้อง

บทที่ 3 กล่าวถึงวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของข้อมูลตั้งต้นที่เคยได้มีการนำเสนอมาก่อน

บทที่ 4 นำเสนอแนวคิดและหลักการของวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนซึ่งเป็นวิธีการที่วิทยานิพนธ์ฉบับนี้นำเสนอ

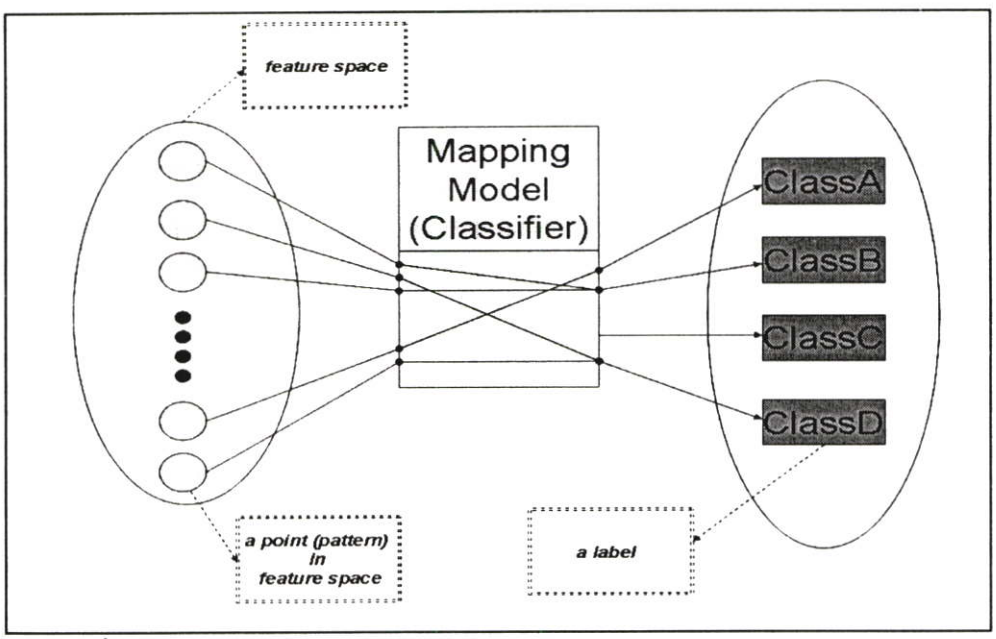
บทที่ 5 แสดงผลการทดลองเพื่อประเมินประสิทธิภาพในการลดจำนวนโปรโตไทป์ของวิธีการที่ได้นำเสนอ และผลการทดลองเพื่อประเมินประสิทธิภาพในการแยกแยะข้อมูลของโปรโตไทป์ที่เลือกได้ด้วยวิธีการที่นำเสนอ สรุปและวิเคราะห์ผลการทดลอง

บทที่ 6 สรุปการวิจัยและข้อเสนอแนะ

บทที่ 2
การแยกแยะข้อมูลและการเลือกโปรโตไทป์

2.1 การแยกแยะข้อมูล (Classification)

การแยกแยะข้อมูล คือ กระบวนการสร้างความสัมพันธ์เพื่อเชื่อมโยง (mapping) จาก space อันหนึ่งซึ่งเรียกว่า "feature space" (feature space เป็น discrete space หรือ continuous space ก็ได้) ไปยังเซ็ต (discrete set) อันหนึ่งซึ่งเรียกว่า "category set" ซึ่งมีสมาชิกเป็นป้ายชื่อ (label) ระบุชนิดของข้อมูล (category) ที่มีอยู่ทั้งหมด ซึ่งสามารถอธิบายการแยกแยะข้อมูลให้อยู่ในเชิงคณิตศาสตร์ [4] ได้ดังนี้ กำหนดให้ feature space แทนด้วยเซ็ตของเว็คเตอร์ $X = \{\vec{x}_1, ..., \vec{x}_n\}$ (ในที่นี้เป็น finite set ขนาด n) และ category set แทนด้วยเซ็ตของป้ายชื่อชนิด $Y = \{y_1, ..., y_c\}$ (ในที่นี้เป็น finite set ขนาด c) แล้วตัวแยกแยะข้อมูล (classifier) สามารถแทนได้ด้วยฟังชันก์ $h: X \rightarrow Y$ ซึ่งบอกความสัมพันธ์เชื่อมโยง (mapping) จากออบเจ็คที่อยู่ภายใน feature space ใดๆ $\vec{x}_i \in X$ ไปยังป้ายชื่อชนิดของออบเจ็คนั้น $y_j \in Y$

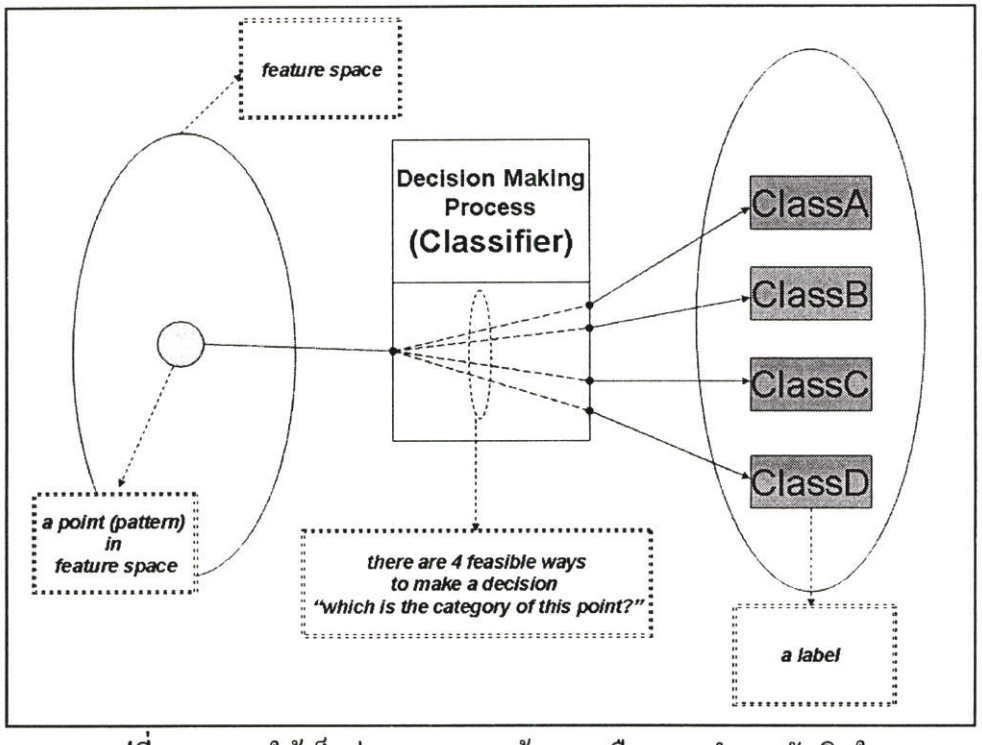


รูปที่ 2.1 แสดงให้เห็นว่าการแยกแยะข้อมูลเสมือนการสร้างความความสัมพันธ์

รูปที่ 2.1 แสดงให้เห็นว่าการแยกแยะข้อมูลนั้นเสมือนกับการสร้างความความสัมพันธ์ระหว่างออบเจ็คใน feature space กับป้ายชื่อชนิดที่มีอยู่ใน category set ทั้งหมด โดยแสดง feature space ด้วยวงรีซึ่งภายในประกอบไปด้วยจุดหลายจุดซึ่งก็เป็นตัวแทนของออบเจ็ค x_i ใดๆที่อยู่ภายใน

feature space ส่วนของ category set นั้นแสดงด้วยวงรีซึ่งภายในประกอบไปด้วยป้ายชื่อชนิดที่มีอยู่ทั้งหมด ส่วนตัวแยกแยะข้อมูล (classifier) นั้นแสดงด้วยกล่องที่มีเส้นเชื่อมโยงเพื่อบอกความสัมพันธ์ระหว่างออบเจ็ค x_i ใดๆที่อยู่ภายใน feature space ไปยังป้ายชื่อชนิดที่มีอยู่ใน category set

การแยกแยะข้อมูลในอีกมุมมองหนึ่งสามารถพิจารณาเป็นเสมือนกระบวนการตัดสินใจ (decision system) ซึ่งรับค่าคุณลักษณะเฉพาะ (feature value) เป็นอินพุท แล้วนำค่าที่ได้มาใช้ในการตัดสินใจว่าเป็นข้อมูลชนิดใด โดยพิจารณาจากชนิดที่เป็นไปได้ทั้งหมด (feasible category) ให้ผลลัพท์ออกมาเป็นชื่อชนิดของอินพุทนั้น ตัวอย่างเช่น การแยกแยะข้อมูลการยื่นขอทำสินเชื่อโดยรับค่าต่างๆเหล่านี้ ข้อมูลรายรับ อายุ สถานภาพการแต่งงาน ที่อยู่ ประวัติการเงินของบุคคลเป็นอินพุทเพื่อทำการตัดสินใจว่าจะตอบปฏิเสธหรือตอบตกลงการยื่นขอทำสินเชื่อของบุคคลนี้ เป็นต้น



รูปที่ 2.2 แสดงให้เห็นว่าการแยกแยะข้อมูลเสมือนการทำการตัดสินใจ

จากรูปที่ 2.2 แสดงให้เห็นว่าการแยกแยะข้อมูลเป็นเสมือนการทำการตัดสินใจ โดยการรับข้อมูลเกี่ยวกับลักษณะเฉพาะของออบเจ็บตัวหนึ่งที่อยู่ภายใน feature space เข้ามาพิจารณาตัดสินใจเลือกว่าจะกำหนดให้ออบเจ็คนั้นเป็นชนิดใด ซึ่งจากรูปนั้นมีชนิดทั้งหมด 4 ชนิด

2.2 กฎ Nearest Neighbor Rule (NN Rule)

กฎ nearest neighbor rule เป็นกฎที่ใช้เพื่อการตัดสินใจของวิธีการแยกแยะข้อมูลแบบใช้ระยะทางในการตัดสินใจ (distance-based classification rules) ซึ่งวิธีการนี้สามารถทำการตัดสินใจแยกแยะข้อมูลโดยที่ไม่จำเป็นต้องทราบพารามิเตอร์ด้านสถิติที่อธิบายกลุ่มข้อมูลก่อน (non parametric approach)[1][2] และวิธีการนี้ถือได้ว่ามีความสามารถในการเรียนรู้จากข้อมูลที่ใช้สอนซึ่งมีการระบุชนิดของข้อมูลไว้ก่อน (labeled training dataset) ดังนั้นวิธีการนี้จึงจัดได้ว่าอยู่ในประเภทการแยกแยะข้อมูลโดยการเรียนรู้แบบมีผู้สอน (supervised learning classification)

นิยาม กฎ nearest neighbor rule "กำหนดให้ออปเจ็คที่ไม่ทราบชนิด เป็นชนิดเดียวกันกับชนิดของโปรโตไทป์อ้างอิงตัวที่อยู่ใกล้กับออปเจ็คนั้นมากที่สุด" [1]

กฎ nearest neighbor rule [5]

$$\theta = \theta' \mid (x', \theta') \in PROTOTYPE \wedge \delta(x', x) = \min_{i=1}^{N} \delta(x_i, x) \tag{2.1}$$

โดยกำหนดให้

- เซ็ตชนิด (category set) คือ เซ็ตของชนิด ในที่นี้กำหนดให้มี M ชนิด

$$CATEGORY = \{y_1, y_2, ..., y_M\} \tag{2.2}$$

- สเปซลักษณะเฉพาะ(feature space) คือ สเปซของค่าลักษณะเฉพาะเป็นที่ไปได้ทั้งหมด ในที่นี้แทนด้วยเสปซจำนวนจริงขนาด d มิติ

$$FEATURE = \Re^d \tag{2.3}$$

- เซ็ตโปรโตไทป์อ้างอิง (prototype set or reference set) คือ เซ็ตของออบเจ็คที่ทราบชนิดอยู่แล้ว ซึ่งสามารถแสดงด้วยเซ็ตของคู่อันดับออบเจ็คคลาส (object-class pairs) จำนวน N คู่ ดังนี้

$$PROTOTYPES = \{(x_1, \theta_1), (x_2, \theta_2), ..., (x_N, \theta_N)\} \tag{2.4}$$

โดยกำหนดให้

- x_i คือ สัญลักษณ์แสดงออบเจ็ค ซึ่งออบเจ็คนั้นเทียบได้กับจุดระบุตำแหน่งหนึ่งจุดบน feature space ($FEATURE$)

$$x_i \in FEATURE \ for \ i = 1..N \tag{2.5}$$

- θ_i คือ สัญลักษณ์แสดงชนิดของออบเจ็ค x_i โดย θ_i ใดๆต้องเป็นสมาชิกของ category set

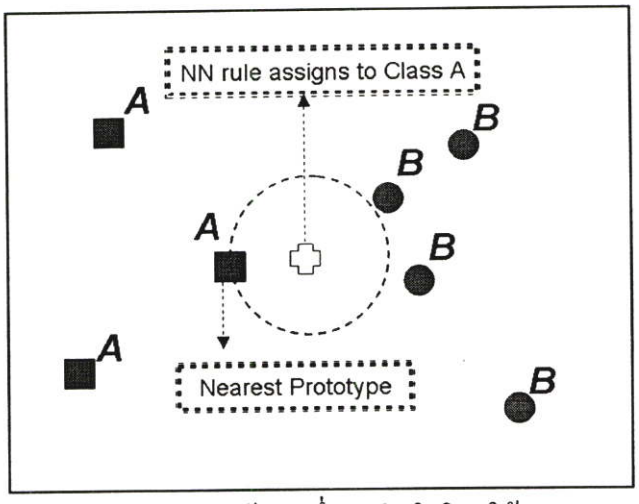
$$\theta_i \in CATEGORY \ for \ i = 1..N \tag{2.6}$$

- ออบเจ็คที่ไม่ทราบชนิด(unknown object) คือ คู่ลำดับออบเจ็คคลาส (object-class pairs) ที่ทราบค่าออบเจ็ต x (ทราบตำแหน่งในเสปซ) แต่ไม่ทราบว่าค่า θ (ไม่ทราบว่า x เป็นชนิดใด)

$$unknownPair = (x, \theta) \tag{2.7}$$

– ฟังก์ชันคำนวนระยะทาง(distance function) คือ ฟังก์ชันสำหรับคำนวนระยะห่างระหว่างจุดสองจุดบน feature space (ออบเจ็คคือจุดหนึ่งจุดบน feature space) ซึ่งในที่นี้แทนด้วยสัญลักษณ์ δ

$$Distance\ Function:\ \delta(x_j, x_k) \mid x_j, x_k \in FEATURE \tag{2.8}$$



รูปที่ 2.3 แสดงตัวอย่างการแยกแยะข้อมูลซึ่งตัดสินใจโดยใช้กฎ nearest neighbor rule

จากรูปที่ 2.3 แสดงตัวอย่างการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule โดยมีโปรโตไทป์อ้างอิงจำนวน 7 โปรโตไทป์ ประกอบไปด้วยโปรโตไทป์ที่เป็นชนิด A จำนวน 3 โปรโตไทป์ (ในที่นี้แสดงด้วยรูปสี่เหลี่ยมเล็ก) และโปรโตไทป์ที่เป็นชนิด B จำนวน 4 โปรโตไทป์ (ซึ่งในที่นี้แสดงด้วยรูปวงกลมเล็ก) ซึ่งโปรโตไทป์ทั้ง 7 โปรโตไทป์มีการวางตัวในตำแหน่งที่แตกต่างกันบน feature space ซึ่งแสดงได้ดังรูปที่ 2.3 เรากำหนดให้ออบเจ็คที่ไม่ทราบชนิดตัวหนึ่ง(ในที่นี้แสดงโดยรูปเครื่องหมายบวก) มีตำแหน่งบน feature space ดังที่แสดงในรูป เมื่อเราต้องการตัดสินใจว่าออบเจ็คนี้เป็นชนิดใดโดยใช้กฎ nearest neighbor rule แล้ว กฎนี้จะตัดสินใจกำหนดให้ออบเจ็คตัวนี้เป็นชนิดเดียวกันกับชนิดของโปรโตไทป์อ้างอิงตัวที่อยู่ใกล้กับออบเจ็คตัวนี้มากที่สุด ซึ่งจากรูปได้แสดงให้เห็นว่ากฎ nearest neighbor rule กำหนดให้ออบเจ็คที่ไม่ทราบชนิดเป็นชนิด A เช่นเดียวกับชนิดของโปรโตไทป์ที่อยู่ใกล้กับออบเจ็คตัวนั้นมากที่สุด

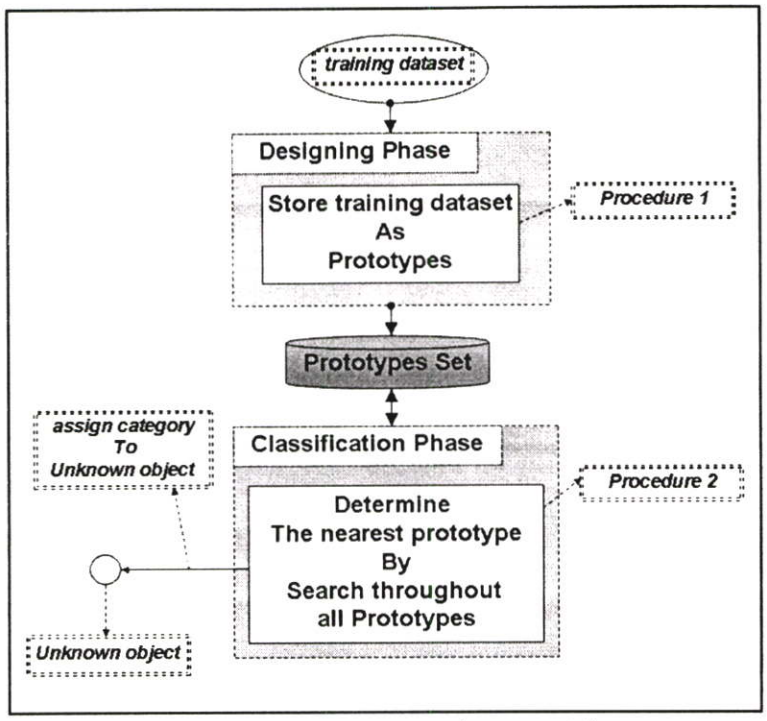
2.2.1 กระบวนการทำงานของการแยกแยะข้อมูลโดยใช้กฎ Nearest Neighbor Rule

กระบวนการทำงานของวิธีการแยกแยะข้อมูลสามารถแบ่งได้เป็น 2 ขั้นตอนหลัก [9] นั้นก็คือ ขั้นตอนการสร้างโมเดลสำหรับแยกแยะข้อมูล (designing phase) และขั้นตอนการแยกแยะข้อมูล (classification phase) โดยขั้นตอนการสร้างโมเดลสำหรับแยกแยะข้อมูลเป็นขั้นตอนการเรียนรู้เพื่อสร้างโมเดลขอบเขตการตัดสินใจ (decision boundary model) เพื่อนำใช้ในการแยกแยะข้อมูล ส่วนขั้นตอนการแยกแยะข้อมูลเป็นขั้นตอนที่ใช้โมเดลขอบเขตการตัดสินใจที่ได้สร้างขึ้นนั้นมาทำการแยกแยะข้อมูล ซึ่งขั้นตอนนี้ถือได้ว่าเป็นขั้นตอนที่มีการลงมือทำการแยกแยะข้อมูลจริงๆ ทำให้บางทีเรียกขั้นตอนนี้ว่าเป็นขั้นตอนปฏิบัติการ (operation phase) ส่วนในขั้นตอนสร้างโมเดลสำหรับแยกแยะข้อมูลไม่มีการทำการแยกแยะข้อมูลแต่อย่างใดเป็นเพียงการสร้างโมเดลขอบเขตการตัดสินใจเท่านั้น และที่สำคัญอีกประการหนึ่งคือกระบวนการทำงานที่อยู่ในขั้นตอนการแยกแยะข้อมูลจะวนซ้ำทำงานใหม่ทุกครั้งที่มีการแยกแยะข้อมูลหนึ่งข้อมูล ส่วนกระบวนการทำงานในขั้นตอนการสร้างโมเดลสำหรับแยกแยะข้อมูลจะมีการทำงานเพียงครั้งเดียวเท่านั้นนั้น นั่นก็คือในช่วงเริ่มต้นที่มีการเรียนรู้จากข้อมูลที่ใช้สอนเพื่อมาสร้างเป็นโมเดลขอบเขตการตัดสินใจเท่านั้น หลังจากได้โมเดลขอบเขตการตัดสินใจแล้วก็ไม่จำเป็นต้องมีการทำงานในขั้นตอนการสร้างโมเดลสำหรับแยกแยะข้อมูลอีกเป็นครั้งที่สอง

ในทำนองเดียวกันเราสามารถแบ่งกระบวนการทำงานของการแยะแยะข้อมูลโดยใช้กฎ nearest neighbor rule เป็น 2 ขั้นตอนหลัก [3][9] ได้ดังนี้

1. **ขั้นตอนการสร้างโมเดลสำหรับแยกแยะข้อมูล (Designing Phase)** การแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule นั้นไม่มีการสร้างโมเดลขอบเขตการตัดสินใจเพื่อใช้ในการแยะแยะข้อมูลแต่อย่างใด ดังนั้นจึงไม่มีกระบวนการทำงานใดๆในขั้นตอนนี้ นอกจากการเก็บข้อมูลตั้งต้นทั้งหมดไปเป็นโปรโตไทป์อ้างอิงเท่านั้น ซึ่งกระบวนการนี้สามารถแสดงได้ดัง Procedure 1 ในรูปที่ 2.4

2. **ขั้นตอนการแยกแยะข้อมูล (Classification Phase)** ดังที่ได้ทราบจากนิยามของกฎ nearest neighbor rule แล้วว่า "กฎ nearest neighbor rule จะกำหนดให้ออปเจ็คที่ไม่ทราบชนิด เป็นชนิดเดียวกันกับชนิดของโปรโตไทป์อ้างอิงตัวที่อยู่ใกล้กับออปเจ็คนั้นมากที่สุด" ดังนั้นในขั้นตอนการแยกแยะข้อมูลของวิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule จึงมีกระบวนการทำงานหลักเพียงขั้นตอนเดียวนั่นก็คือ กระบวนการชี้เฉพาะโปรโตไทป์ตัวที่อยู่ใกล้กับออปเจ็คที่ไม่ทราบชนิดมากที่สุดนั่นเอง ซึ่งกระบวนการนี้สามารถแสดงได้ดัง Procedure 2 ในรูปที่ 2.4



รูปที่ 2.4 แสดงภาพรวมของกระบวนการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule

จากรูปที่ 2.4 ได้แสดงให้เห็นภาพรวมของกระบวนการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule โดยจากรูปจะเห็นว่ามีการแบ่งกระบวนการทำงานทั้งหมดออกเป็นขั้นตอน 2 หลัก นั้นคือ ขั้นตอนการสร้างโมเดลสำหรับแยกแยะข้อมูล (designing phase) และขั้นตอนการแยกแยะข้อมูล (classification phase) ในขั้นตอนการสร้างโมเดลสำหรับแยกแยะข้อมูลมีกระบวนการทำงานอยู่เพียงกระบวนการเดียวนั้นคือกระบวนการเก็บข้อมูลตั้งต้นไปเป็นโปรโตไทป์อ้างอิง และในขั้นตอนการแยกแยะข้อมูลก็มีกระบวนการทำงานอยู่เพียงกระบวนการเดียวนั้นคือกระบวนการชี้เฉพาะโปรโตไทป์ตัวที่ใกล้กับออปเจ็คที่ไม่ทราบชนิดมากที่สุด ซึ่งการทำงานจะเริ่มต้นจากการรับข้อมูลที่ใช้สอน (training dataset) เข้าสู่ขั้นตอนการสร้างโมเดลสำหรับแยกแยะข้อมูลซึ่งในขั้นตอนนี้จะทำการเก็บข้อมูลตั้งต้นทั้งหมดไว้เป็นโปรโตไทป์อ้างอิง หลังจากที่ได้โปรโตไทป์อ้างอิงเรียบร้อยแล้วก็เข้าสู่ขั้นตอนการแยกแยะข้อมูล โดยหากต้องการแยกแยะออบเจ็คที่ไม่ทราบชนิดหนึ่งออบเจ็คก็จะต้องทำกระบวนการชี้เฉพาะโปรโตไทป์ตัวที่ใกล้กับออปเจ็คตัวนั้นมากที่สุดหนึ่งครั้ง

จะเห็นได้ว่ากระบวนการทำงานในขั้นตอนการสร้างโมเดลสำหรับแยกแยะข้อมูลนั้นจะมีการทำงานเพียงรอบเดียวนั้นก็คือช่วงเริ่มต้นเรียนรู้เพื่อสร้างโมเดลขอบเขตการตัดสินใจเท่านั้น แต่กระบวนการทำงานในขั้นตอนการแยกแยะข้อมูลจะมีการทำงานทุกครั้งที่มีการแยกแยะออบเจ็คหนึ่งออบเจ็ค ดังนั้นสำหรับการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule แล้วกระบวนการชี้

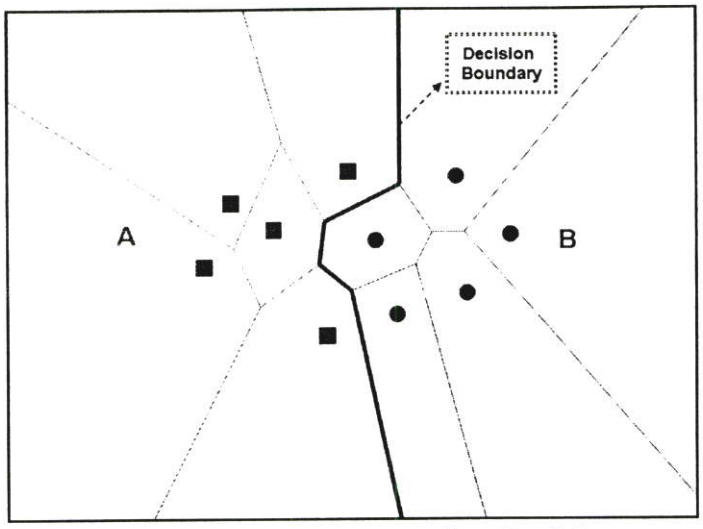
เฉพาะโปรโตไทป์ตัวที่ใกล้กับออปเจ็คที่ต้องการทราบชนิดมากที่สุดจึงต้องมีการทำงานทุกครั้งที่มีการแยกแยะออบเจ็คที่ไม่ทราบชนิดหนึ่งออบเจ็ค

กระบวนการชี้เฉพาะโปรโตไทป์ตัวที่ใกล้มากที่สุดกับออปเจ็คที่ไม่ทราบชนิดก็คือการค้นหาโปรโตไทป์อ้างอิงตัวที่มีระยะทางห่างจากออปเจ็คตัวนั้นน้อยที่สุดนั่นเอง ซึ่งจะต้องมีการคำนวนระยะทางระหว่างออปเจ็คที่ไม่ทราบชนิดนั้นกับโปรโตไทป์อ้างอิงทุกตัวเพื่อการเปรียบเทียบหาโปรโตไทป์อ้างอิงที่มีระยะทางห่างจากออบเจ็คนั้นน้อยที่สุด ซึ่งเห็นได้ว่าจำนวนรอบของการคำนวนฟังก์ชันหาระยะทาง (distance function) จะมีค่าแปรตามจำนวนของโปรโตไทป์อ้างอิงโดยตรง ดังนั้นในกรณีที่มีโปรโตไทป์อ้างอิงเป็นจำนวนมากจึงมีภาระในการคำนวนที่สูงขึ้นตามไปด้วย

จากปัจจัยที่การแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule จะต้องมีการคำนวนเพื่อชี้เฉพาะโปรโตไทป์ตัวที่ใกล้กับออปเจ็คที่ไม่ทราบชนิดมากที่สุดทุกครั้งที่มีการแยกแยะออบเจ็คและภาระการคำนวนเพื่อชี้เฉพาะโปรโตไทป์ตัวที่ใกล้ที่สุดมีค่าแปรตามจำนวนของโปรโตไทป์อ้างอิงโดยตรง ดังนั้นวิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule จึงเผชิญกับปัญหาความต้องการการคำนวนในปริมาณที่สูงและปัญหาการแยกแยะข้อมูลมีช่วงเวลาในการทำงานที่ยาวนาน (long operation time) เมื่อมีโปรโตไทป์อ้างอิงเป็นจำนวนมาก

2.2.2 ขอบเขตในการตัดสินใจของการแยกแยะข้อมูลโดยใช้กฎ Nearest Neighbor Rule [5]

เราสามารถแสดงขอบเขตในการตัดสินใจ (decision boundary) ของวิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule ออกมาในรูปแบบที่เรียกว่า "Voronoi Diagram" [5] (ตัวอย่างของ voronoi diagram แสดงในรูปที่ 2.5 ซึ่งถือว่าเป็นโมเดลในการแยกแยะข้อมูลของวิธีการนี้ (classification model) ถึงแม้ในขั้นตอนการสร้างโมเดลสำหรับแยกแยะข้อมูล (designing phase) ของวิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule จะไม่ได้ทำอะไรเลยนอกจากเก็บข้อมูลที่ใช้สอนทั้งหมดเป็นโปรโตไทป์อ้างอิง นั้นหมายถึงโมเดลนี้ไม่ได้ถูกสร้างขั้นในขั้นตอนนี้ ส่วนในขั้นตอนการแยกแยะข้อมูล (classification phase) ของวิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule นั้นก็ไม่ได้ทำการสร้างโมเดลนี้ขึ้นมาเพื่อแยกแยะข้อมูลอย่างชัดแจ้ง แต่เมื่อเราทำการแยกแยะตามกฎ nearest neighbor rule แล้ว ผลลัพท์ของการแยกแยะข้อมูลจะเสมือนกับว่าใช้ voronoi diagram เป็นขอบเขตในการตัดสินใจ แม้เราจะไม่ได้สร้างโมเดลของ voronoi diagram ขั้นมาเลยก็ตาม



รูปที่ 2.5 แสดงขอบเขตการตัดสินใจของการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule

จากรูปที่ 2.5 ถ้ายึด voronoi diagram เป็นโมเดลในการตัดสินใจจะได้ผลลัพท์ของการแยกแยะข้อมูลเช่นเดียวกับการใช้กฎ nearest neighbor rule โดยเมื่อมีออบเจ็คที่ไม่ทราบชนิดมีตำแหน่งอยู่ภายในช่องเซลล์ใดของ voronoi diagram กำหนดให้ออบเจ็คนั้นเป็นชนิดเดียวกันกับชนิดของโปรโตไทป์ของช่องนั้น ซึ่งจากรูปแสดงขอบเขตการตัดสินใจที่เกิดจากการรวมขอบเขตของช่องเซลล์ที่มีชนิดเดียวกัน เช่นเมื่อมีออบเจ็คที่ไม่ทราบชนิดซึ่งมีตำแหน่งอยู่ภายในบริเวณด้านซ้ายของขอบเขตในการตัดสินใจ (decision boundary) จะกำหนดให้ออบเจ็คนั้นเป็นชนิด A แต่ถ้าหากออบเจ็คนั้นมีตำแหน่งอยู่ในบริเวณด้านขวาของขอบเขตในการตัดสินใจจะกำหนดให้ออบเจ็คนั้นเป็นชนิด B เป็นต้น

2.2.3 ข้อดีและข้อเสียของการแยกแยะข้อมูลโดยใช้กฎ Nearest Neighbor Rule

วิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule นี้ถูกนำไปใช้อย่างแพร่หลาย ทั้งนี้เพราะวิธีการนี้มีความเรียบง่ายในแนวคิดพร้อมทั้งการเขียนโปรแกรมก็ทำได้โดยง่าย [2][3] และวิธีการนี้สามารถทำการแยกแยะข้อมูลได้โดยที่ไม่จำเป็นต้องทราบค่าพารามิเตอร์ทางสถิติที่อธิบายกลุ่มข้อมูลก่อน (non-parametric decision rule) [6] อีกทั้งวิธีการนี้มีความสามารถแยกแยะข้อมูลได้ดีใกล้เคียงกับวิธีการแยกแยะข้อมูลอื่นๆที่ต้องทราบค่าพารามิเตอร์ทางสถิติที่อธิบายกลุ่มข้อมูลก่อน และถ้าหากว่ามีข้อมูลที่ใช้สอนจำนวนมากพอค่าความน่าจะเป็นที่จะทำการแยกแยะข้อมูลผิดพลาดของกฎ nearest neighbor rule จะมีค่าใกล้เคียงกับค่าความน่าจะเป็นที่จะทำการแยกแยะข้อมูลผิดพลาดของเบย์ (Bayes probability of error) ซึ่งเป็นค่าความน่าจะเป็นในการตัดสินใจแยกแยะข้อมูลผิดพลาดที่น้อยที่สุด [1] แต่กฎ Bayes optimal decision rule นี้จำเป็นต้องทราบค่าของฟังก์ชัน class-conditional probability density function ของกลุ่มข้อมูลนั้น [5][6] ซึ่งในความเป็นจริงนั้นไม่สามารถทราบค่าที่แท้จริงของฟังก์ชันนี้ได้ Cover และ Hart [1] ได้แสดงให้เห็นว่าในกรณีที่มีข้อมูล

ที่ใช้สอนเป็นจำนวนมาก (ค่าลู่เข้าสู่อนันต์) แล้วค่าความผิดผลาดของกฎ nearest neighbor rule จะมีค่าอยู่ไม่เกิน 2 เท่าของค่าความผิดผลาดของเบย์ (Bayes optimal error) เสมอ [1][2][3][5][7][8] นั่นหมายถึงว่าวิธีการนี้สามารถสร้างได้โดยง่ายและมีประสิทธิภาพที่สูงใกล้เคียงกับวิธีการอื่นๆที่สร้างได้ยากกว่านั่นเอง อีกทั้งวิธีการนี้สามารถใช้ในการแยกแยะข้อมูลกับปัญหาที่ไม่เฉพาะเจาะจงได้ (non-trivial problem) [2] หรือก็คือวิธีการนี้สามารถใช้ได้ผลดีกับปัญหาหลากหลายรูปแบบนั่นเอง ด้วยเหตุนี้วิธีการนี้จึงถูกนำไปประยุกต์ใช้ในงานด้านต่างๆอย่างแพร่หลาย นอกจากนั้นกฎนี้มักถูกใช้เป็นตัวเปรียบเทียบเพื่อประเมินประสิทธิภาพให้วิธีการอื่นๆในการนำไปใช้ในงานต่างๆกันด้วย [3] ทั้งนี้ก็เพราะกฎ nearest neighbor rule นั่นง่ายและมีประสิทธิภาพที่ดีกับงานในหลากหลายรูปแบบและมีประสิทธิภาพที่ดีเพียงพอที่จะเป็นตัวเปรียบเทียบเพื่อประเมินประสิทธิภาพของวิธีการอื่น

อย่างไรก็ตามวิธีการที่ง่ายและมีประสิทธิภาพนี้ก็มีข้อด้อยอยู่ในตัวด้วย ซึ่งเป็นสาเหตุสำคัญอย่างหนึ่งที่ทำให้ในหลายงานไม่เลือกใช้วิธีการนี้หรือไม่สามารถใช้วิธีการนี้ได้เพราะไม่เหมาะสม เช่น เมื่อใช้กฎ nearest neighbor rule สำหรับแยกแยะข้อมูลที่มีมิติจำนวนมากจำเป็นต้องมีข้อมูลที่ใช้สอนเป็นจำนวนมากตามไปด้วย ปัญหาในลักษณะนี้เรียกว่า “Curse of Dimensionality” [6] ปัญหานี้สามารถแก้ไขได้โดยการหาวิธีสร้างข้อมูลที่ใช้สอนขึ้นมาใหม่เองจนมีจำนวนมากพอกับความต้องการ (แต่ก็จะทำให้เกิดปัญหาโปรโตไทป์อ้างอิงมากเกินไปตามมา) หรือ อีกวิธีคือพยายามแก้ปัญหาโดยการลดมิติของข้อมูลลง (dimensionality reduction) ให้มีน้อยที่สุดเท่าที่ทำได้ โดยใช้วิธีการจำพวก feature exaction เช่น PCA, ICA, MDS เป็นต้น[6] หรือใช้วิธีการจำพวก feature selection เช่น branch-and-bound search, sequential forward selection เป็นต้น[6] ข้อด้อยสำคัญประการหนึ่งของกฎ nearest neighbor rule คือ เพื่อให้การแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule มีประสิทธิภาพในการแยกแยะข้อมูลที่ดี (มีค่าความถูกต้องสูง “high classification accuracy” บ้างก็เรียก “high recognition rate”) จำเป็นต้องมีการเก็บโปรโตไทป์ไว้สำหรับอ้างอิงเป็นจำนวนมาก ด้วยเหตุนี้ทำให้มีความต้องการหน่วยความจำเป็นจำนวนมากให้เพียงพอต่อการจัดเก็บข้อมูลของตัวโปรโตไทป์อ้างอิงที่มีอยู่ทั้งหมด (high memory demands problem) และกระบวนการชี้เฉพาะโปรโตไทป์ที่ใกล้ที่สุดนั้นจำเป็นต้องทำการคำนวนฟังก์ชันคำนวนระยะทางจากออบเจ็คนั้นไปหาโปรโตไทป์ทุกๆตัว เป็นสาเหตุให้เกิดปัญหาความต้องการความสามารถในการคำนวนปริมาณสูง (high computational demands problem) และด้วยสาเหตุนี้ผนวกกับการทำงานของกฎ nearest neighbor rule มีลักษณะเป็นการเรียนรู้แบบขี้เกียจ (lazy learning algorithm) [10] ซึ่งจะทำการเรียนรู้ใหม่ทุกครั้งในขั้นตอนปฏิบัติการ (operation phase) เมื่อสองปัจจัยนี้มารวมกันคือต้องเรียนรู้ใหม่ทุกครั้งที่มีการทำการแยกแยะข้อมูลหนึ่งออบเจ็ค (การเรียนรู้ของกฎ nearest neighbor rule ก็คือการค้นหาเพื่อชี้เฉพาะโปรโตไทป์ที่ใกล้กับออบเจ็คที่ไม่ทราบชนิด) และการเรียนรู้แต่ละครั้งมีความต้องการการคำนวนที่สูง ผลที่เกิดขึ้นก็คือ การแยกแยะข้อมูลมีช่วงเวลา

ในการทำงานที่ยาวนาน (long operation time) ซึ่งเป็นช่วงเวลาที่ต้องรอคอยโดยตรง เป็นผลทำให้วิธีการนี้ไม่เหมาะสมกับงานบางรูปแบบ เช่นงานที่ต้องการการตอบสนองแบบฉับพลันแต่จำเป็นต้องมีโปรโตไทป์อ้างอิงเป็นจำนวนมาก เป็นต้น

2.3 การเลือกโปรโตไทป์ (Prototype Selection)

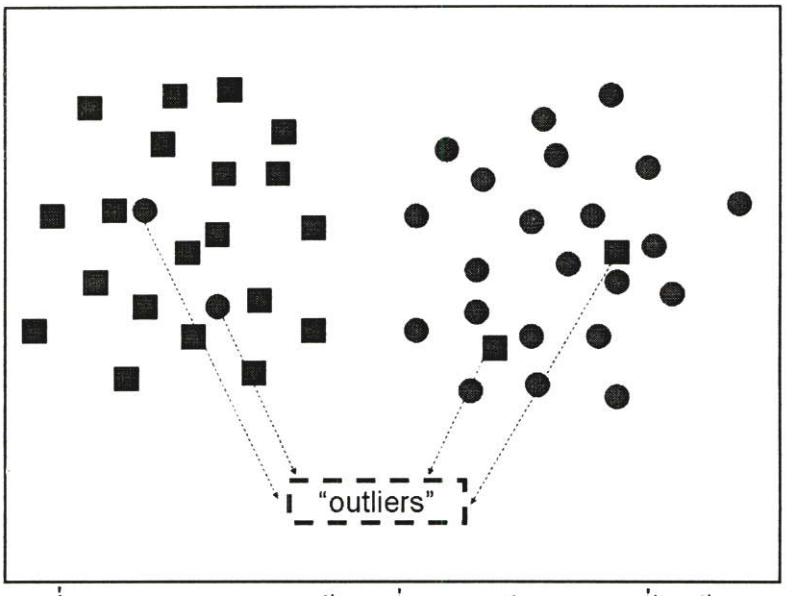
การเลือกโปรโตไทป์เป็นแนวทางหนึ่งที่ช่วยแก้ปัญหาที่เกิดขึ้นกับการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule เช่น ปัญหาความต้องการการคำนวนที่สูง (high computational demands problem) ปัญหาความต้องการหน่วยความจำจำนวนมาก (high memory demands problem) และปัญหาการลดข้อมูลที่ผิดพลาด (noise reduction problem) เป็นต้น ซึ่งเทคนิคการเลือกโปรโตไทป์แต่ละวิธีสามารถช่วยแก้ปัญหาที่แตกต่างกันขึ้นอยู่กับว่าเทคนิคในการเลือกโปรโตไทป์นั้นมีความต้องการเพื่อใช้ในการแก้ปัญหาอะไร เทคนิคการเลือกโปรโตไทป์ถูกนำเสนอออกมามากมายซึ่งสามารถแบ่งเทคนิคการเลือกโปรโตไทป์ออกเป็น 2 ประเภทตามจุดประสงค์ของเทคนิค ได้ดังนี้ [2] [3]

1. **วิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิง** (ศัพท์ทางเทคนิคเรียกว่า "Condensing Prototype Selection" หรือ "Condensing Technique") คือ เทคนิคของการเลือกโปรโตไทป์ซึ่งมีเป้าหมายเพื่อลดจำนวนโปรโตไทป์ที่ใช้สำหรับอ้างอิงโดยกฎ nearest neighbor rule ให้มีจำนวนน้อยลงมากที่สุดเท่าที่ทำได้โดยที่ยังคงความสามารถในการแยกแยะข้อมูลให้ใกล้เคียงกับกฎ nearest neighbor rule ที่ใช้ชุดข้อมูลตั้งต้น(original dataset) ทั้งหมดเป็นโปรโตไทป์อ้างอิง ซึ่งเทคนิคการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงนี้ช่วยในการแก้ปัญหาความต้องการความสามารถในการคำนวนที่สูง (high computational demands problem) และปัญหาความต้องการหน่วยความจำเป็นจำนวนมาก (high memory demands problem) ที่เกิดขึ้นกับการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule ที่มีโปรโตไทป์อ้างอิงเป็นจำนวนมาก เนื่องจากเทคนิคการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงนี้ มีเป้าหมายที่จะลดจำนวนโปรโตไทป์อ้างอิงของกฎ nearest neighbor rule ลงให้มากที่สุดโดยที่ยังคงความสามารถของการแยกแยะข้อมูลของกฎ nearest neighbor rule ให้ใกล้เคียงกับกฎ nearest neighbor rule ที่ใช้ข้อมูลตั้งต้นทั้งหมดเป็นโปรโตไทป์อ้างอิง ดังนั้นผลการแยกแยะข้อมูลของกฎ nearest neighbor rule ที่ใช้โปรโตไทป์ที่ลดจำนวนลงแล้วเป็นโปรโตไทป์อ้างอิงจะดีหรือไม่ดีนั้นจึงขึ้นกับข้อมูลตั้งต้นด้วยว่ามีความสามารถในการแยกแยะข้อมูลได้ดีแค่ไหน ซึ่งเทคนิคการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงไม่ได้คำนึงว่าผลลัพท์จะสามารถแยกแยะข้อมูลได้ดีหรือไม่ เพียงแต่ทำการลดจำนวนโปรโตไทป์อ้างอิงลงให้มากที่สุดโดยที่ยังคงความสามารถของการ

แยกแยะข้อมูลให้ใกล้เคียงกับการใช้ข้อมูลตั้งต้นทั้งหมดเป็นโปรโตไทป์อ้างอิงเท่านั้น ดังนั้นเพื่อจะให้โปรโตไทป์ที่ลดจำนวนลงแล้วสามารถใช้ในแยกแยะข้อมูลได้ดี ชุดข้อมูลตั้งต้นของวิธีการนี้ก็ต้องสามารถแยกแยะข้อมูลได้ดีเสียก่อน ตัวอย่างของเทคนิคการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิง[11] เช่น proximity graphs[5], CNN[14], genetic algorithm[15], TABU[3], MCA[16] เป็นต้น

2. **วิธีการเลือกโปรโตไทป์เพื่อกำจัดข้อมูลที่ไม่เข้าพวก** (ศัพท์ทางเทคนิคเรียกว่า "Editing Prototype Selection" หรือ "Editing Technique") คือ เทคนิคของการเลือกโปรโตไทป์ซึ่งมีเป้าหมายเพื่อลดข้อมูลที่ใช้สอนที่ผิดพลาดซึ่งแทรกอยู่ในข้อมูลที่ใช้สอนทั้งหมด โดยวิธีการเหล่านี้มีเป้าหมายเพื่อลดข้อมูลที่มีลักษณะไม่เข้าพวก (outliers) ซึ่งก็คือข้อมูลที่มีลักษณะการวางตัวอยู่ระหว่างข้อมูลชนิดอื่นไม่อยู่ในกลุ่มข้อมูลที่เป็นชนิดเดียวกัน โดยแต่ละเทคนิคก็มีวิธีการที่แตกต่างกันไปเพื่อใช้สำหรับชี้เฉพาะว่าข้อมูลนั้นมีลักษณะไม่เข้าพวก (แต่ละวิธีก็มีนิยามของลักษณะความไม่เข้าพวกที่แตกต่างกัน) [5] เมื่อเราลดข้อมูลซึ่งมีลักษณะไม่เข้าพวกเหล่านี้ไปจนหมด แล้วกลุ่มของโปรโตไทป์ที่เหลืออยู่จะมีลักษณะการวางตัวเป็นกลุ่มก้อน ซึ่งแต่ละกลุ่มจะมีโปรโตไทป์ชนิดเดียวกันรวมกันอยู่ ไม่มีโปรโตไทป์ที่ไปอยู่ปะปนกับกลุ่มโปรโตไทป์ที่เป็นคนละชนิดกัน ซึ่งลักษณะการวางตัวของโปรโตไทป์ที่เป็นกลุ่มก้อนซึ่งในกลุ่มนั้นมีแต่โปรโตไทป์ที่เป็นชนิดเดียวกันทั้งหมด กลุ่มโปรโตไทป์เหล่านี้เรียกว่ามีลักษณะ "well-clustered groups of homogenous prototype" ซึ่งข้อดีที่ตามมานอกจากช่วยลดข้อมูลที่ผิดพลาด (noise reduction) แล้ว ถ้าหากใช้โปรโตไทป์เหล่านี้เป็นโปรโตไทป์อ้างอิง กฎ k-nearest neighbor rule จะมีความสามารถในการแยกแยะข้อมูลใกล้เคียงกับกฎ nearest neighbor rule นั่นหมายถึงว่า ถ้าหากว่าแต่ละกลุ่มข้อมูล (clusters) มีโปรโตไทป์เป็นชนิดเดียวกันทั้งหมดแล้วจะสามารถแยกแยะข้อมูลเหล่านี้ได้โดยที่ใช้เพียงกฎ nearest neighbor rule ก็เพียงพอไม่จำเป็นต้องใช้กฎ k-nearest neighbor rule ซึ่งช่วยลดปริมานการคำนวนลงและยังช่วยลดปัญหา overfitting ของการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule ลงด้วย สรุปได้ว่าเทคนิคการเลือกโปรโตไทป์เพื่อกำจัดข้อมูลที่ไม่เข้าพวก (editing prototype selection technique) คือเทคนิคการเลือกโปรโตไทป์ที่มีเป้าหมายเพื่อต้องการลดข้อมูลที่มีลักษณะไม่เข้าพวกให้หมดไป ซึ่งช่วยลดข้อมูลที่ผิดพลาด (noise reduction) และช่วยแก้ปัญหา overfitting ที่เกิดกับการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule ลง โดยไม่ได้สนใจให้ความสำคัญต่อการลดจำนวนโปรโตไทป์ให้น้อยลงแต่อย่างใด ตัวอย่างของเทคนิคการเลือกโปรโตไทป์เพื่อกำจัดข้อมูลที่ไม่เข้าพวก เช่น proximity graphs [5], edited nearest neighbor (ENN) [12], All k-NN [13] เป็นต้น ในรูปที่ 2.6 แสดงตัวอย่างข้อมูลที่วางตัวอยู่ในลักษณะที่ไม่เข้าพวกซึ่งเทคนิคการเลือกโปรโตไทป์เพื่อกำจัดข้อมูลที่ไม่เข้าพวกต้องการกำจัดข้อมูลเหล่านี้ และหลังจากที่กำจัดข้อมูลที่ไม่เข้าพวกเหล่านี้ไปแล้วจะเห็นว่า

ข้อมูลที่เหลือจะมีการวางตัวเป็นกลุ่มก้อนซึ่งแต่ละกลุ่มมีเฉพาะข้อมูลที่เป็นชนิดเดียวกันอยู่รวมกัน

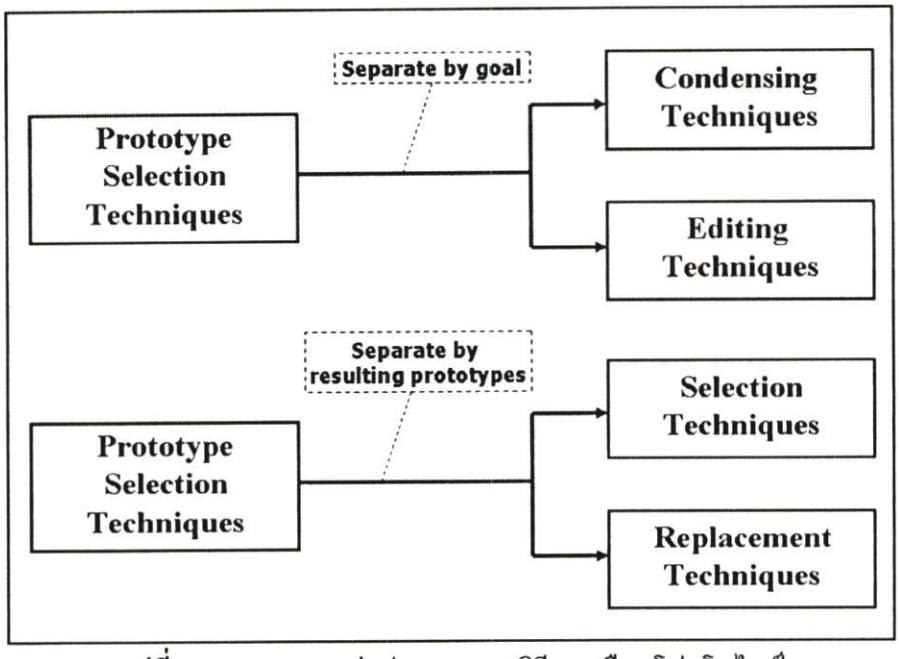


รูปที่ 2.6 แสดงตัวอย่างของข้อมูลที่วางตัวอยู่ในลักษณะที่ไม่เข้าพวก

นอกจากการแบ่งเทคนิคการเลือกโปรโตไทป์ตามจุดประสงค์ของเทคนิคดังที่ได้กล่าวไปแล้วนั้น เทคนิคการเลือกโปรโตไทป์ยังสามารถแบ่งตามลักษณะของผลลัพท์ได้อีกด้วย โดยสามารถแบ่งได้เป็น 2 ประเภท ดังนี้ [2] [3]

1. **วิธีการเลือกโปรโตไทป์ที่ให้ผลลัพท์เป็นเซ็ตของโปรโตไทป์ที่สร้างขึ้นใหม่** (ศัพท์ทางเทคนิคเรียกว่า "Replacement Prototype Selection" หรือ "Replacement Technique") คือ เทคนิคการเลือกโปรโตไทป์ซึ่งผลลัพท์ที่ได้เป็นเซ็ตของโปรโตไทป์ซึ่งได้มาจากชุดข้อมูลตั้งต้นหรือไม่ก็ได้ นั่นหมายความว่า เทคนิคการเลือกโปรโตไทป์นี้สามารถสร้างโปรโตไทป์ขึ้นมาใหม่ได้ ซึ่งถือเป็นข้อดีของวิธีนี้เพราะโปรโตไทป์ที่เป็นผลลัพท์ไม่จำเป็นต้องอยู่ในเซ็ตของข้อมูลตั้งต้นเสมอ ดังนั้นวิธีการนี้จึงมีโอกาสที่จะได้โปรโตไทป์ผลลัพท์ที่ดีกว่าวิธีการเลือกโปรโตไทป์ที่ให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้นหากว่าวิธีการนั้นสามารถสร้างโปรโตไทป์ที่ดีกว่าออบเจ็คที่อยู่ภายในเซ็ตข้อมูลตั้งต้นได้ (ทั้งนี้ก็ต้องขึ้นอยู่กับประสิทธิภาพของวิธีการและความสมบูรณ์ของเซ็ตของโปรไทป์ตั้งต้นนั้นด้วย) ตัวอย่างเทคนิคการเลือกโปรโตไทป์ที่ให้ผลลัพท์เป็นเซ็ตของโปรโตไทป์ที่สร้างขึ้นใหม่ เช่น MCA [16], LVQ [17] เป็นต้น

2. **วิธีการเลือกโปรโตไทป์ที่ให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น** (ศัพท์ทางเทคนิคเรียกว่า "Selection Prototype Selection" หรือ "Selection Technique") คือ เทคนิคการเลือกโปรโตไทป์ซึ่งผลลัพท์ที่ได้เป็นเซ็ตของโปรโตไทป์ที่ได้มาจากชุดข้อมูลตั้งต้น นั่นหมายถึง เทคนิคการเลือกโปรโตไทป์แบบนี้คือการเลือกโปรโตไทป์จากชุดข้อมูลตั้งต้นนั่นเอง ดังนั้นเซ็ตของโปรโตไทป์ที่เลือกได้จึงเป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้นเสมอ เพราะฉะนั้นโปรโตไทป์ผลลัพท์ที่ได้จากวิธีการนี้จึงมีความสามารถในการแยกแยะข้อมูลขึ้นอยู่กับเซ็ตข้อมูลตั้งต้นเป็นปัจจัยด้วย ข้อดีประการหนึ่งของเทคนิคการเลือกโปรโตไทป์ที่ให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้นคือเทคนิคนี้สามารถใช้กับชุดข้อมูลที่ไม่สามารถสร้างขึ้นมาใหม่เองได้ เช่น ข้อมูลลายมือเขียนซึ่งมีลักษณะเฉพาะที่เรียกว่าเป็นลักษณะเฉพาะแบบ "featureless" ซึ่งไม่สามารถสร้างโปรโตไทป์ของข้อมูลในลักษณะนี้ขึ้นมาใหม่เองได้ เป็นต้น ตัวอย่างเทคนิคการเลือกโปรโตไทป์ที่ให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น เช่น proximity graphs[5], CNN[14], GA[15], TABU[3] เป็นต้น



รูปที่ 2.7 แสดงการแบ่งประเภทของวิธีการเลือกโปรโตไทป์

จากรูปที่ 2.7 ได้แสดงให้เห็นว่าวิธีการเลือกโปรโตไทป์นั้นสามารถแบ่งได้โดยการพิจารณา 2 ปัจจัยนั่นคือ เป้าหมายของเทคนิคการเลือกโปรโตไทป์และลักษณะของผลลัพท์ที่ได้จากแต่ละเทคนิค โดยหากแบ่งการเลือกโปรโตไทป์ตามเป้าหมายจะแบ่งได้เป็น วิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิง (condensing technique) และวิธีการเลือกโปรโตไทป์เพื่อกำจัดข้อมูลที่ไม่เข้าพวก (editing technique) ถ้าหากว่าแบ่งการเลือกโปรโตไทป์ตามลักษณะของผลลัพท์ที่ได้จะแบ่งได้เป็น วิธีการเลือกโปรโตไทป์ที่ให้ผลลัพท์เป็นเซ็ตของโปรโตไทป์ที่สร้างขึ้นใหม่

(replacement technique) และวิธีการเลือกโปรโตไทป์ที่ให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (selection technique) ซึ่งเกณฑ์ในการทั้ง 2 นั้นสามารถนำมาใช้ร่วมกันได้ อาทิเช่น การเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (selection-condensing prototype selection) เป็นต้น

2.4 คุณสมบัติความสอดคล้อง (Consistency Property)

คุณสมบัติความสอดคล้อง (consistency property) เป็นแนวคิดหนึ่งที่ถูกนำมาใช้เป็นแกนหลักของเทคนิคการเลือกโปรโตไทป์เพื่อการลดจำนวนโปรโตไทป์อ้างอิง โดยแนวคิดของคุณสมบัติความสอดคล้องนี้ถูกนำเสนอในเอกสารที่ชื่อว่า "The Condensed Nearest Neighbor Rule" (CNN) [14] ซึ่งถือว่าเป็นเอกสารชิ้นแรกๆที่นำเสนอวิธีการเลือกโปรโตไทป์เพื่อการลดจำนวนโปรโตไทป์ ซึ่งเอกสารนี้ถูกอ้างอิงโดยเอกสารอื่นๆที่ได้นำเสนอวิธีการเลือกโปรโตไทป์เพื่อการลดจำนวนโปรโตไทป์อ้างอิงหรืองานอื่นๆที่เกี่ยวข้อง ซึ่งเนื้อหาที่นิยมนำมาอ้างอิงของเอกสารนี้มีทั้งวิธีการเลือกโปรโตไทป์เพื่อการลดจำนวนโปรโตไทป์ที่ชื่อว่า "CNN" และแนวคิดคุณสมบัติความสอดคล้องของโปรโตไทป์อ้างอิงที่เลือกได้กับข้อมูลตั้งต้น ซึ่งแนวคิดคุณสมบัติความสอดคล้องนี้ถูกนำมาปรับใช้และยึดถือเป็นแกนหลักเพื่อใช้ในการออกแบบวิธีการเลือกโปรโตไทป์เพื่อการลดจำนวนโปรโตไทป์อ้างอิงอย่างแพร่หลายมาจนถึงปัจจุบัน

นิยาม ความสอดคล้อง (Consistency Property) "เซ็ตของโปรโตไทป์ที่เลือกมาได้จะมีความสอดคล้องกับเซ็ตข้อมูลตั้งต้น ก็ต่อเมื่อ วิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule ซึ่งมีเซ็ตของโปรโตไทป์ที่เลือกมาได้เป็นเซ็ตอ้างอิง สามารถแยกแยะข้อมูลตั้งต้นได้อย่างถูกต้องทุกตัว" [14]

ดังที่ทราบกันแล้วว่าวิธีการเลือกโปรโตไทป์เพื่อการลดจำนวนโปรโตไทป์ (condensing prototype selection) คือวิธีการเลือกโปรโตไทป์ซึ่งมีเป้าหมายเพื่อลดจำนวนโปรโตไทป์อ้างอิงของกฎ nearest neighbor rule ให้มีจำนวนน้อยที่สุดเท่าที่ทำได้โดยยังคงความสามารถในการแยกแยะข้อมูลให้ใกล้เคียงกับกฎ nearest neighbor rule ที่ใช้ชุดข้อมูลตั้งต้น (original dataset) เป็นโปรโตไทป์อ้างอิง ซึ่งคำว่า "ความสามารถในการแยกแยะข้อมูลให้ใกล้เคียงเดิม" หมายถึงขอบเขตของการตัดสินใจแยกแยะข้อมูลใหม่ (decision boundary) ยังคงใกล้เคียงขอบเขตของการตัดสินใจเดิมมากที่สุดนั่นเอง หากแต่ว่าการตรวจสอบโดยตรงว่าขอบเขตในการตัดสินใจนั้นใกล้เคียงกับค่าเดิมนั้นทำได้ยาก

แนวคิดความสอดคล้องเป็นแนวคิดที่ใช้เพื่อยืนยันว่ากฎ nearest neighbor rule ซึ่งใช้โปรโตไทป์ที่หามาได้ใหม่เป็นโปรโตไทป์อ้างอิงมีความสามารถในการแยกแยะข้อมูลใกล้เคียงกับกฎ nearest neighbor rule ที่ใช้ชุดข้อมูลตั้งต้น (original dataset) ทั้งหมดเป็นโปรโตไทป์อ้างอิง โดยที่ไม่จำเป็นต้องมีการตรวจสอบขอบเขตในการตัดสินใจของกฎ nearest neighbor rule โดยแนวคิดความสอดคล้องนี้ได้เสนอว่า เพียงแต่ตรวจสอบว่ากฎ nearest neighbor rule ที่ใช้โปรโตไทป์ที่หามาได้ใหม่เป็นโปรโตไทป์อ้างอิงสามารถแยกแยะข้อมูลตั้งต้นได้อย่างถูกต้องทั้งหมด นั่นแสดงว่าโปรโตไทป์ที่หามาได้ใหม่มีความสอดคล้องกับชุดข้อมูลตั้งต้น ซึ่งสามารถยืนยันได้ว่าขอบเขตในการตัดสินใจใหม่นั้นใกล้เคียงกับขอบขอบเขตการตัดสินใจเดิม ซึ่งการตรวจสอบนี้สามารถทำได้โดยง่ายและมีประสิทธิภาพ ทำให้วิธีเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิง (condensing prototype selection) ใช้แนวคิดนี้เป็นหลักเพื่อรับรองว่าความสามารถในการแยกแยะข้อมูลของโปรโตไทป์ที่หามาได้ใหม่นั้นยังคงใกล้เคียงความสามารถในการแยกแยะข้อมูลของชุดข้อมูลตั้งต้นเดิมอยู่

สรุปแล้วในบทนี้ได้นำเสนอเนื้อหาต่างๆตั้งแต่อธิบายให้เข้าใจถึงคำจำกัดความของการแยกแยะข้อมูลเพื่อให้เข้าใจตรงกันว่าการแยกแยะข้อมูลนั้นคืออะไร และกล่าวอธิบายถึงวิธีการแยกแยะข้อมูลรูปแบบหนึ่งซึ่งเรียกว่า “วิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule” ทั้งได้แสดงนิยามทั้งในเชิงอรรถอธิบายและเชิงคณิตศาสตร์ของกฎ nearest neighbor rule และกระบวนการทำงานของวิธีการนี้ โดยแสดงให้เห็นภาพรวมว่าสามารถแบ่งกระบวนทำงานได้เป็น 2 ขั้นตอน นั้นคือ ขั้นตอนการออกแบบ (designing phase) และขั้นตอนแยกแยะข้อมูล (classification phase) ซึ่งกระบวนการทำงานในขั้นตอนแยกแยะข้อมูลนั้นจะต้องมีการทำงานซ้ำทุกครั้งที่มีการแยกแยะข้อมูลและความต้องการการคำนวนของกระบวนการทำงานในขั้นตอนแยกแยะข้อมูลของกฎ nearest neighbor rule จะมีมากขึ้นแปรตามจำนวนของโปรโตไทป์อ้างอิง เป็นผลทำให้วิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule เกิดปัญหาความต้องการหน่วยความจำและความสามารถในการคำนวนที่สูงในกรณีที่มีโปรโตไทป์อ้างอิงเป็นจำนวนมาก หลังจากนั้นได้สรุปข้อดีและข้อเสียของวิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule ให้เข้าใจอย่างชัดเจนมากยิ่งขึ้น แล้วกล่าวอธิบายถึงหนทางหนึ่งที่ใช้สำหรับการแก้ปัญหาที่เกิดขึ้นกับวิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule ซึ่งวิธีการแก้ปัญหานั้นก็คือ การเลือกโปรโตไทป์ (prototype selection) และได้แสดงการแบ่งประเภทของเทคนิคการเลือกโปรโตไทป์ว่านิยมแบ่งกันโดยใช้หลักเกณฑ์ 2 หลักก็คือ เกณฑ์เป้าหมายและเกณฑ์ลักษณะของผลลัพท์ที่ได้นั่นเอง สุดท้ายได้กล่าวถึงคุณสมบัติความสอดคล้อง (consistency property) ซึ่งถือได้ว่าเป็นหลักเกณฑ์ที่นำมาใช้ในการรับประกันว่าโปรโตไทป์ที่หาได้ใหม่นั้นมีความสามารถในการแยกแยะข้อมูลใกล้เคียงกับข้อมูลตั้งต้นเดิมนั่นเอง

เนื่องจากวิธีการเลือกโปรโตไทป์ที่วิทยานิพนธ์ได้นำเสนอนี้จัดได้ว่าเป็นประเภทการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (selection-condensing prototype selection) เนื้อหาในบทถัดไปจึงเป็นการกล่าวถึงวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้นที่มีการนำเสนอมาก่อนให้เข้าใจถึงการทำงานของวิธีการเหล่านี้เพื่อนำไปใช้อ้างอิงในการทดลองเพื่อการเปรียบเทียบต่อไป

บทที่ 3
การเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น

จากเนื้อหาในบทที่ 2 ได้อธิบายไปแล้วว่าการเลือกโปรโตไทป์สามารถจำแนกโดยใช้เป้าหมายของการเลือกโปรโตไทป์ได้เป็น 2 ประเภทนั่นคือ การเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์ (condensing prototype selection) และการเลือกโปรโตไทป์เพื่อการกำจัดข้อมูลที่ไม่เข้าพวก (editing prototype selection) อีกทางหนึ่งที่ใช้การจำแนกประเภทของการเลือกโปรโตไทป์คือการจำแนกการเลือกโปรโตไทป์ตามลักษณะของผลลัพท์ที่ได้ ซึ่งสามารถจำแนกได้เป็นการเลือกโปรโตไทป์ที่ให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (selection prototype selection) และการเลือกโปรโตไทป์ที่ให้ผลลัพท์เป็นเซ็ตของโปรโตไทป์ที่สร้างขึ้นเอง (replacement prototype selection)

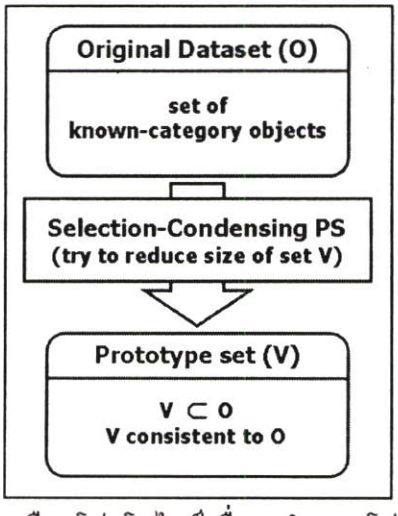
วิทยานิพนธ์นี้มีขอบเขตที่สนใจเฉพาะในส่วนของการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของข้อมูลตั้งต้นเป็นหลัก(selection-condensing prototype selection) ดังนั้นเนื้อหาในบทนี้จึงเป็นการนำเสนอคำอธิบายเกี่ยวกับการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของข้อมูลตั้งต้นและกล่าวถึงวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของข้อมูลตั้งต้นที่ได้เคยมีการนำเสนอมาก่อน ซึ่งในที่นี้คือวิธีการเลือกโปรโตไทป์ที่ชื่อว่า "condensed nearest neighbor rule (CNN)"[14] และวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยใช้ genetic algorithm (GA) [15]

3.1 การเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (Selection-Condensing Prototype Selection)

การเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทปอ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น เป็นการเลือกโปรโตไทป์แบบหนึ่งซึ่งมีเป้าหมายเพื่อลดจำนวนของโปรโตไทป์ที่ใช้ในการอ้างอิงโดยกฎ nearest neighbor rule ให้มีจำนวนโปรโตไทป์น้อยที่สุดแต่ยังคงความสามารถในการแยกแยะข้อมูลของกฎ nearest neighbor rule ให้ใกล้เคียงเดิมไม่เปลี่ยนแปลงไปมากนั้น โดยเซ็ตของโปรโตไทป์ผลลัพท์ที่เลือกมาได้ใหม่ต้องเป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น ซึ่งหมายความว่าโปรโตไทป์ที่ได้จะได้มาจากการเลือกสมาชิกในเซ็ตข้อมูลตั้งต้นมาใช้เป็นโปรโตไทป์เท่านั้นไม่ให้มีการสร้างโปรโตไทป์ขึ้นมาใหม่ ทำให้วิธีการนี้เลือกโปรโตไทป์แบบนี้สามารถใช้กับข้อมูลประเภทที่ไม่สามารถสร้างโปรโตไทป์ขึ้นมาใหม่ได้ เช่น ข้อมูลลายมือเขียนที่มีลักษณะเฉพาะแบบ

"featureless" เป็นต้น ซึ่งวิธีการเลือกโปรโตไทป์แบบให้ผลลัพท์เป็นเซ็ตของโปรโตไทป์ที่สร้างขึ้นเอง (replacement prototype selection) ไม่สามารถใช้ได้

จากที่ได้กล่าวไปแล้วว่าการเลือกโปรโตไทป์เพื่อเพื่อลดจำนวนโปรโตไทป์อ้างอิงมักใช้หลักคุณสมบัติความสอดคล้อง(consistency property) เพื่อรับรองว่าความสามารถในการแยกแยะข้อมูลของโปรโตไทป์ที่หาได้ใหม่ใกล้เคียงความสามารถในการแยกแยะข้อมูลของชุดข้อมูลตั้งต้นเดิม ดังนั้นเป้าหมายการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้นที่ยึดหลักความสอดคล้อง คือการพยามเลือกหาซับเซ็ตที่มีขนาดเล็กที่สุดของเซ็ตข้อมูลตั้งต้น โดยที่ซับเซ็ตที่เลือกได้ยังคงความสอดคล้องกับเซ็ตข้อมูลตั้งต้นอยู่นั่นเอง ซึ่งปัญหาในลักษณะนี้เรียกว่า "ปัญหาการเลือกซับเซ็ตสอดคล้องที่มีขนาดเล็กที่สุด (minimal consistent subset selection problem)" [3][15] อย่างไรก็ตามการหาคำตอบที่แท้จริงของปัญหานี้สามารถทำได้ยากเพราะปัญหานี้เป็นปัญหาการจัดหมู่ที่ยาก (hard combinatorial problem) [3] ดังนั้นการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้นที่ยึดหลักความสอดคล้องจึงไม่ใช่การหาคำตอบที่แท้จริงของปัญหานี้ แต่เป็นเพียงการหาคำตอบเชิงประมาณที่ใกล้เคียงกับคำตอบที่แท้จริงเท่านั้น ซึ่งหมายความว่าการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้นที่ยึดหลักความสอดคล้องมีความพยายามหาซับเซ็ตที่มีความสอดคล้องกับเซ็ตข้อมูลตั้งต้นที่มีขนาดเล็กที่สุดเท่าที่จะทำได้โดยอาศัยการคำนวนไม่มากเกินไป



รูปที่ 3.1 แสดงเป้าหมายของการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น

ในรูปที่ 3.1 แสดงให้เห็นว่าการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้นที่ยึดหลักความสอดคล้องมีเป้าหมายเพื่อลดขนาดของเซ็ตโปรโตไทป์ (V) โดยที่มีเงื่อนไขว่าเซ็ตโปรโตไทป์ (V) ต้องเป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (O) และเซ็ตโปรโตไทป์ (V) ต้องมีความสอดคล้องกับเซ็ตข้อมูลตั้งต้น (O) ด้วย

3.2 วิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้นที่เคยได้มีการนำเสนอมาก่อน

ในหัวข้อที่ผ่านมาได้อธิบายให้เข้าใจถึงความหมายและเป้าหมายของการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น(selection-condensing prototype selection) ในหัวข้อนี้นำเสนอวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้นที่เคยได้มีการนำเสนอมาก่อน ซึ่งในที่นี้นำเสนอ 2 วิธีนั้นคือ วิธีการ condensed nearest neighbor rule (CNN) [14] และวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยใช้ genetic algorithm (GA) [15] ซึ่งสามารถนำเสนอได้ดังต่อไปนี้

3.2.1 วิธีการ Condensed Nearest Neighbor Rule (CNN)

วิธีการนี้ถูกนำเสนอในเอกสารชื่อ "The Condense Nearest Neighbor Rule" [14] โดย Peter E. Hart ซึ่งเป็นเอกสารเดียวกับเอกสารที่นำเสนอหลักการคุณสมบัติความสอดคล้อง (consistency property) ที่ได้กล่าวไปแล้วนั่นเอง วิธีการนี้ถือได้ว่าเป็นวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (selection-condensing prototype selection) เพราะวิธีการนี้มีเป้าหมายเพื่อลดจำนวนของโปรโตไทป์อ้างอิงลงโดยที่เซ็ตของโปรโตไทป์ที่ได้ใหม่เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้นและรับประกันว่าเซ็ตของโปรโตไทป์ที่หาได้ใหม่จะมีความสอดคล้องกับเซ็ตของข้อมูลตั้งต้นเสมอ ซึ่งวิธีการนี้นิยมนำมาใช้เพื่อเปรียบเทียบประสิทธิภาพในการลดจำนวนโปรโตไทป์กับวิธีการเลือกโปรโตไทป์อื่นๆและเป็นที่รู้จักกันอย่างแพร่หลาย

3.2.1.1 กระบวนการทำงานของวิธีการ Condensed Nearest Neighbor Rule (CNN)

หัวข้อนี้แสดงกระบวนการทำงานของวิธีการ condensed nearest neighbor rule โดยสามารถแสดงในรูปแบบคำสั่งเทียม (pseudo code) ได้ดังรูปที่ 3.2

```
1   NearestNeighbor(prototypeList, data)
2     RETURN the nearest prototypeList's member with data.
3   END NearestNeighbor
4
5   NN(prototypeList, data)
6     IF(NearestNeighbor(prototypeList, data).label == data.label)
7       RETURN TRUE
8     ELSE
9       RETURN FALSE
10    END IF
11  END NN
12
13  CNN_MAIN
14    prototypeList.Add(dataList[1])
15    FOR i = 2 TO dataList.Size()
16      IF(NN(prototypeList, dataList[i]) == TRUE) THEN
17        redundanceList.Add(dataList[i])
18      ELSE
19        protoypeList.Add(dataList[i])
20      END IF
21    END FOR
22    DO
23      flag = TRUE
24      j = 1
25      WHILE j ≤ redundanceList.Size() DO
26        IF(NN(prototypeList, redundanceList[j]) == TRUE) THEN
27          j = j+1
28        ELSE
29          prototypeList.Add(redundanceList[j])
30          redundanceList.Remove(j)
31          flag = FALSE
32        END IF
33      END WHILE
34    UNTIL flag == TRUE
35  END CNN_MAIN
```

รูปที่ 3.2 แสดงกระบวนทำงานของวิธีการ condensed nearest neighbor rule

จากรูปที่ 3.2 ได้แสดงกระบวการทำงานของวิธีการ condensed nearest neighbor rule ในรูปแบบคำสั่งเทียม (pseudo code) ซึ่งสามารถอธิบายความหมายของคำสั่งเทียมในรูปที่ 3.2 ได้ดังนี้

กำหนดค่าเริ่มต้นให้ลิสต์ dataList เก็บออบเจ็คที่ทราบชนิดทั้งหมด (ชุดข้อมูลตั้งต้น) และกำหนดให้ลิสต์ prototypeList เป็นลิสต์ว่างและกำหนดให้ลิสต์ redundanceList เป็นลิสต์ว่าง

- คำสั่งในบรรทัดที่ 1 ถึง 3 หมายความว่ากำหนดให้มีฟังก์ชันชื่อ NearestNeighbor ซึ่งรับค่าลิสต์ prototypeList และค่าออบเจ็ค data แล้วคืนค่ากลับเป็นโปรโตไทป์ในลิสต์ prototypeList ตัวที่อยู่ใกล้ออบเจ็ค data มากที่สุด
- คำสั่งในบรรทัดที่ 5 ถึง 11 หมายความว่ากำหนดให้มีฟังก์ชันชื่อ NN ซึ่งรับค่าลิสต์ prototypeList และค่าออบเจ็ค data แล้วทำการตรวจสอบว่าโปรโตไทป์ในลิสต์ prototypeList ตัวที่อยู่ใกล้ออบเจ็ค data มากที่สุด(โดยเรียกใช้ฟังก์ชัน NearestNeighbor) เป็นชนิดเดียวกันกับชนิดของออบเจ็ค data หรือไม่ กรณีที่เป็นชนิดเดียวทำการคืนค่า "จริง" กรณีที่เป็นคนละชนิดทำการคืนค่ากลับเป็น "เท็จ"

คำสั่งในบรรทัดที่ 13 ถึง 35 เป็นขั้นตอนการทำงานหลักของวิธีการ condensed nearest neighbor rule ซึ่งมีความหมายดังนี้

- คำสั่งในบรรทัดที่ 14 หมายความว่าให้สำเนาออบเจ็คแรกในลิสต์ dataList มาเก็บไว้ในลิสต์ prototypeList
- คำสั่งในบรรทัดที่ 15 ถึง 21 หมายความว่าให้พิจารณาตั้งแต่ออบเจ็คลำดับที่ 2 จนถึงออบเจ็คลำดับสุดท้ายในลิสต์ dataList ทีละตัวว่า กฎ nearest neighbor rule ที่ใช้ลิสต์ prototypeList ในการอ้างอิงจะสามารถแยกแยะออบเจ็คนั้นได้ถูกต้องหรือไม่(โดยการเรียกใช้ฟังก์ชัน NN) ในกรณีที่แยกแยะได้ถูกต้องให้สำเนาออบเจ็คนั้นไปไว้ที่ลิสต์ redundanceList ส่วนกรณีแยกแยะผิดให้สำเนาออบเจ็คนั้นไปไว้ที่ลิสต์ prototypeList
- คำสั่งในบรรทัดที่ 22 ถึง 34 หมายความว่าให้วนทำคำสั่งที่อยู่ในบรรทัดที่ 23 ถึง 33 ซ้ำไปเรื่อยๆจนกว่าค่า flag ในบรรทัดที่ 34 จะมีค่าเป็นจริง
- คำสั่งในบรรทัดที่ 23 หมายความว่ากำหนดให้ค่า flag มีค่าเป็นจริง
- คำสั่งในบรรทัดที่ 24 หมายความว่ากำหนดให้ค่า j มีค่าเป็น 1
- คำสั่งในบรรทัดที่ 25 ถึง 33 หมายความว่าให้พิจารณาตั้งแต่ออบเจ็คแรกจนถึงออบเจ็คสุดท้ายในลิสต์ redundanceList ที่ละตัวว่า กฎ nearest neighbor rule ที่ใช้ลิสต์ prototypeList ในการอ้างอิงจะสามารถแยกแยะออบเจ็คนั้นได้ถูกต้องหรือไม่(โดยการเรียกใช้ฟังก์ชัน NN) ในกรณีที่แยกแยะได้ถูกต้องให้เลื่อนไปพิจารณาออบเจ็คที่อยู่ในลำดับถัดไปในลิสต์ redundanceList เลย (เพิ่มค่า j ขึ้นหนึ่ง) ส่วนกรณีที่แยกแยะผิดให้ย้ายออบเจ็คนั้นจากลิสต์ redundanceList ไปไว้ที่ลิสต์ prototypeList (สำเนาออบเจ็คนั้นลงในลิสต์ prototypeList แล้วลบออบเจ็คนั้นจากลิสต์ redundanceList) แล้วกำหนดให้ค่า flag มีค่า

เป็นเท็จ โดยกรณีนี้ไม่ต้องเพิ่มค่า j เพราะออบเจ็คในลิสต์ redundanceList ถูกลบไปหนึ่งตัวทำให้ออบเจ็คที่อยู่ในลำดับหลัง j ถูกเลื่อนมาหนึ่งลำดับ

เมื่อสิ้นสุดการทำงานจะได้ออบเจ็คที่อยู่ในลิสต์ prototypeList เป็นโปรโตไทป์ที่หาได้ใหม่ ส่วนออบเจ็คที่อยู่ภายในลิสต์ redundanceList สามารถละทิ้งได้ โดยที่รับประกันว่าเมื่อกฎ nearest neighbor rule ที่ใช้ออบเจ็คในลิสต์ prototypeList เป็นโปรโตไทป์อ้างอิงจะสามารถแยกแยะออบเจ็คในชุดข้อมูลตั้งต้น (dataList) ได้ถูกต้องทั้งหมด ทั้งนี้ก็เพราะว่าจากคำสั่งในบรรทัดที่ 14 ถึง 21 ออบเจ็คทุกตัวในลิสต์ dataList จะถูกสำเนาไปไว้ที่ลิสต์ prototypeList หรือไม่ก็ลิสต์ redundanceList ดังนั้นออบเจ็คในลิสต์ dataList จึงเท่ากับออบเจ็คในลิสต์ prototypeList รวมกับออบเจ็คในลิสต์ redundanceList (dataList = prototypeList + redundanceList) และจากคำสั่งในบรรทัดที่ 22 ถึง 34 ซึ่งจะทำงานวนซ้ำเพื่อย้ายออบเจ็คจากลิสต์ redundanceList ไปสู่ลิสต์ prototypeList จนกว่ากฎ nearest neighbor rule ที่ใช้ออบเจ็คในลิสต์ prototypeList เป็นโปรโตไทป์อ้างอิงจะสามารถแยกแยะออบเจ็คในลิสต์ redundanceList ได้ถูกต้องทั้งหมดจึงจะหยุดการทำงาน ดังนั้นเมื่อสิ้นสุดการทำงานแล้วกฎ nearest neighbor rule ที่ใช้ออบเจ็คในลิสต์ prototypeList เป็นโปรโตไทป์อ้างอิงย่อมสามารถแยกแยะออบเจ็คในลิสต์ redundanceList ได้ถูกต้องทั้งหมดและย่อมสามารถแยกแยะออบเจ็คในลิสต์ prototypeList ได้ถูกต้องด้วย (เพราะกฎ nearest neighbor rule ย่อมแยกแยะโปรโตไทป์ที่ใช้อ้างอิงได้ถูกต้องเสมอ) เพราะฉะนั้นเมื่อสิ้นสุดการทำงานแล้วกฎ nearest neighbor rule ที่ใช้ออบเจ็คในลิสต์ prototypeList เป็นโปรโตไทป์อ้างอิงจึงสามารถแยกแยะออบเจ็คที่อยู่ในชุดข้อมูลตั้งต้น (dataList) ได้ถูกต้องทั้งหมด

3.2.1.2 ข้อดีและข้อเสียของวิธีการ Condensed Nearest Neighbor Rule (CNN)

ข้อดีและข้อเสียของวิธีการ condensed nearest neighbor rule มีดังต่อไปนี้

ข้อดี

1. วิธีการ condensed nearest neighbor rule ทำการเลือกโปรโตไทป์จากออบเจ็คในเซ็ตข้อมูลตั้งต้น ดังนั้นวิธีการนี้จึงสามารถใช้ได้กับข้อมูลที่ไม่สามารถสร้างขึ้นมาใหม่ได้
2. กระบวนการทำงานของวิธีการ condensed nearest neighbor rule ไม่ซับซ้อนสามารถเข้าใจได้ง่ายและสามารถเขียนโปรแกรมเพื่อประยุกต์ใช้งานได้โดยง่าย
3. วิธีการ condensed nearest neighbor rule สามารถทำการเลือกโปรโตไทป์ได้อย่างรวดเร็ว

ข้อเสีย

1. โปรโตไทป์ผลลัพท์ที่เลือกได้โดยวิธีการ condensed nearest neighbor rule ขึ้นกับลำดับของออบเจ็คในลิสต์ของข้อมูลตั้งต้น อาทิเช่นในตอนแรกวิธีการ condensed nearest neighbor rule สามารถเลือกโปรโตไทป์จากลิสต์ข้อมูลตั้งต้นได้ผลลัพท์เป็นโปรโตไทป์ที่มีจำนวนน้อย แต่เมื่อเปลี่ยนลำดับของออบเจ็คในลิสต์ข้อมูลตั้งต้นใหม่วิธีการ condensed nearest

neighbor rule อาจเลือกได้โปรโตไทป์ผลลัพท์ที่มีจำนวนไม่เท่าเดิม(มากกว่าหรือน้อยลงกว่าเดิมก็ได้)

2. วิธีการ condensed nearest neighbor rule ยังลดจำนวนโปรโตไทป์ได้ไม่มากนัก ยังเป็นไปได้ที่จะสามารถลดจำนวนโปรโตไทป์อ้างอิงลงไปได้อีก

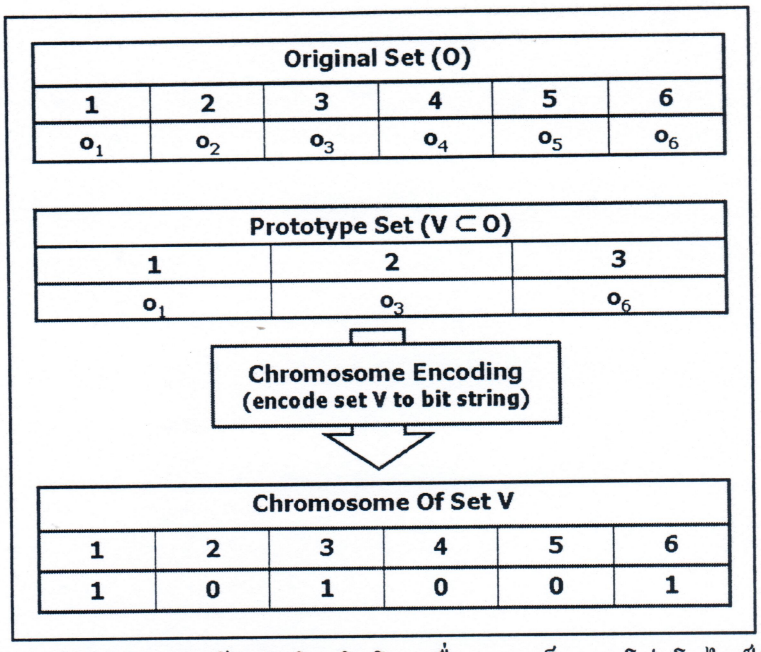
3.2.2 วิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยใช้ Genetic Algorithm

หัวข้อนี้อธิบายวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยอาศัยหลักการ genetic algorithm ซึ่งหลักการทำงานที่แสดงในที่นี้ยึดตามเอกสารชื่อ "Nearest Prototype Classifier: Clustering, Genetic Algorithms, or Random Search?" [15] โดย Ludmila I. Kuncheva และ James C. Bezdek ซึ่งเอกสารนี้นำเสนอการทดลองนำวิธีการ genetic algorithm (GA) มาใช้ในการเลือกโปรโตไทป์จากเซ็ตข้อมูลตั้งต้นให้มีจำนวนโปรโตไทป์น้อยลงและโปรโตไทป์ที่หาได้ยังคงความสอดคล้องกับเซ็ตข้อมูลตั้งต้นอยู่ ซึ่งวิธีการนี้ถือได้ว่าเป็นวิธีการเลือกโปรโตไทป์เพื่อลดโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (selection-condensing prototype selection) ซึ่งจากผลการทดลองในเอกสารได้แสดงให้เห็นว่าวิธีการนี้สามารถลดจำนวนโปรโตไทป์อ้างอิงได้ดีกับชุดข้อมูลตั้งต้นที่มีขนาดเล็ก วิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยใช้ genetic algorithm สามารถอธิบายได้ดังต่อไปนี้

3.2.2.1 การเข้ารหัสโครโมโซม (Chromosome Encoding)

การเข้ารหัสโครโมโซมโดยการแทนเซ็ตโปรโตไทป์ให้อยู่ในรูปบิตสตริง (bit string) ที่มีความยาวเท่ากับจำนวนออบเจ็คในเซ็ตข้อมูลตั้งต้น โดยมีข้อกำหนดว่าหากเลือกออบเจ็คตำแหน่งใดในเซ็ตข้อมูลตั้งต้นไปเป็นโปรโตไทป์แล้วกำหนดให้ค่าบิตในตำแหน่งนั้นมีค่าเป็น "1" ส่วนตำแหน่งอื่นๆที่ออบเจ็คไม่ได้ถูกเลือกไปเป็นโปรโตไทป์กำหนดให้มีค่าเป็น "0"

ในรูปที่ 3.3 แสดงตัวอย่างการเข้ารหัสโครโมโซมเพื่อแสดงเซ็ตของโปรโตไทป์ที่เลือกได้ โดยตารางด้านบนของรูปที่ 3.3 แสดงถึงเซ็ตข้อมูลตั้งต้นเซ็ตซึ่งมีสมาชิก 6 ตัวนั้นคือ o_1, o_2, o_3, o_4, o_5, o_6 ซึ่งเรียงอยู่ในตำแหน่งที่ 1 ไปจนถึง 6 ในเซ็ตข้อมูลตั้งต้นตามลำดับ ส่วนตารางตรงกลางของรูปที่ 3.3 แสดงถึงเซ็ตโปรโตไทป์ที่เลือกได้ซึ่งมีสมาชิกอยู่ทั้งหมด 3 ตัวนั้นคือ o_1, o_2, o_6 เซ็ตของโปรโตไทป์ที่เลือกได้นี้สามารถเข้ารหัสโครโมโซมให้อยู่ในรูปแบบบิตสตริงที่มีความยาวจำนวน 6 บิตเท่ากับจำนวนสมาชิกของเซ็ตข้อมูลตั้งต้น โดยที่บิตสตริงมีบิตในตำแหน่งที่ 1 , 2 และ 6 มีค่าเป็น "1" ส่วนตำแหน่งอื่นมีค่าเป็น "0" เพราะออบเจ็คที่อยู่ในตำแหน่งที่ 1, 2 และ 6 ของเซ็ตข้อมูลตั้งต้นถูกเลือกไปเป็นสมาชิกของเซ็ตโปรโตไทป์ ซึ่งบิตสตริงนี้แสดงได้ดังตารางด้านล่างของรูปที่ 3.3



รูปที่ 3.3 แสดงตัวอย่างการเข้ารหัสโครโมโซมเพื่อแสดงเซ็ตของโปรโตไทป์ที่เลือกได้

3.2.2.2 ฟิทเนสฟังก์ชัน (Fitness Function)

ฟิทเนสฟังก์ชัน คือ ฟังก์ชันที่มีไว้ใช้ในการประเมินค่าโครโมโซมของ genetic algorithm ซึ่งฟิทเนสฟังก์ชันจะแตกต่างกันไปขึ้นอยู่ว่านำเอา genetic algorithm ไปใช้แก้ปัญหาอะไร ฟิทเนสฟังก์ชันของวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยใช้ genetic algorithm สามารถแสดงได้ดังนี้

$$\mathbb{F}(S) = \frac{classifiedNum(S) - \alpha * size(S)}{N\_OriginalDataset} \qquad (3.1)$$

โดย

$\mathbb{F}(S)$ คือ ฟิทเนสฟังก์ชันของโครโมโซม S

S คือ โครโมโซมซึ่งเข้ารหัสแทนเซ็ตโปรโตไทป์ V

$classifiedNum(S)$ คือ ฟังก์ชันแสดงจำนวนของออบเจ็ค (ในเซ็ตข้อมูลตั้งต้น) ที่สามารถแยกแยะได้อย่างถูกต้องด้วยกฎ nearest neighbor rule ที่อ้างอิงเซ็ตโปรโตไทป์ V (ที่โครโมโซม S เข้ารหัสแทนอยู่)

$size(S)$ คือ ขนาดของเซ็ตโปรโตไทป์ V ที่โครโมโซม S เข้ารหัสแทนอยู่

$N\_OriginalDataset$ คือ ขนาดของเซ็ตข้อมูลตั้งต้น

α คือ สัมประสิทธิ์ถ่วงน้ำหนัก (weighted coefficient) เพื่อถ่วงความสำคัญของพจน์ $size(S)$ กับพจน์ $classifiedNum(S)$ ให้สมดุล (สำหรับการทดลองในที่นี้เรากำหนดให้ α มีค่า

เท่ากับ $1/N\_OriginalDataset$ เพราะให้ความสำคัญกับพจน์ $classifiedNum(S)$ มากกว่า)

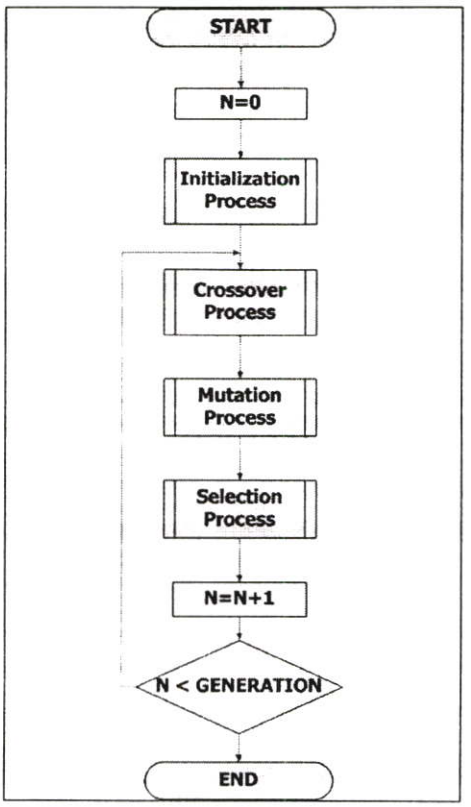
ซึ่งจะเห็นได้ว่าค่าฟิทเนสฟังก์ชันนี้จะมีค่าอยู่ในช่วงตั้งแต่ -1 ถึง 1 โดยหากค่าของฟังก์ชัน $classifiedNum(S)$ มีค่ามากนั่นคือกฎ nearest neighbor rule ที่อ้างอิงเซ็ตโปรโตไทป์ V (ที่โครโมโซม S เข้ารหัสแทนอยู่)สามารถแยกแยะออบเจ็คในเซ็ตข้อมูลตั้งต้นได้ถูกต้องเป็นจำนวนมาก และค่า $size(S)$ มีค่าน้อยนั่นคือขนาดของเซ็ตโปรโตไทป์ V (เซ็ตที่โครโมโซม S เข้ารหัสแทนอยู่)มีค่าน้อย แล้วค่าฟิตเนสฟังก์ชันจะมีค่ามาก (ใกล้ค่า 1) ซึ่งหมายความว่าโครโมโซม S เป็นโครโมโซมที่ดีควรได้รับการพิจารณานำไปใช้เป็นพ่อพันธุ์แม่พันธุ์เพื่อให้ได้ลูกหลานที่ดีขึ้น ในทางกลับกับหากฟังก์ชัน $classifiedNum(S)$ มีค่าน้อยนั่นคือกฎ nearest neighbor rule ที่อ้างอิงเซ็ตโปรโตไทป์ V (เซ็ตที่โครโมโซม S เข้ารหัสแทนอยู่)สามารถแยกแยะออบเจ็คในเซ็ตข้อมูลตั้งต้นได้ถูกต้องน้อยและค่า $size(S)$ มีค่ามากนั่นคือขนาดของเซ็ตโปรโตไทป์ V (เซ็ตที่โครโมโซม S เข้ารหัสแทนอยู่)มีค่ามาก แล้วค่าฟิตเนสฟังก์ชันก็จะมีค่าน้อย (ใกล้ค่า -1) ซึ่งหมายความว่าโครโมโซม S เป็นโครโมโซมที่ไม่ดีจึงไม่ควรนำไปใช้เป็นพ่อพันธุ์แม่พันธุ์

3.2.2.3 กระบวนการทำงานของวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยใช้ Genetic Algorithm (GA)

กระบวนการทำงานหลักของวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยใช้ genetic algorithm มีอยู่ 4 กระบวนการหลัก ดังนี้

1. กระบวนการกำหนดค่าเริ่มต้น (initialization process)
2. กระบวนการครอสโอเวอร์ (crossover process)
3. กระบวนการมิวเทชัน (mutation process)
4. กระบวนการเลือกประชากรในรุ่นถัดไป (selection process)

โดยสามารถอธิบายภาพรวมของกระบวนการทำงานหลักของวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยใช้ genetic algorithm ในรูปแบบผังงาน (flowchart) ได้ดังรูปที่ 3.4



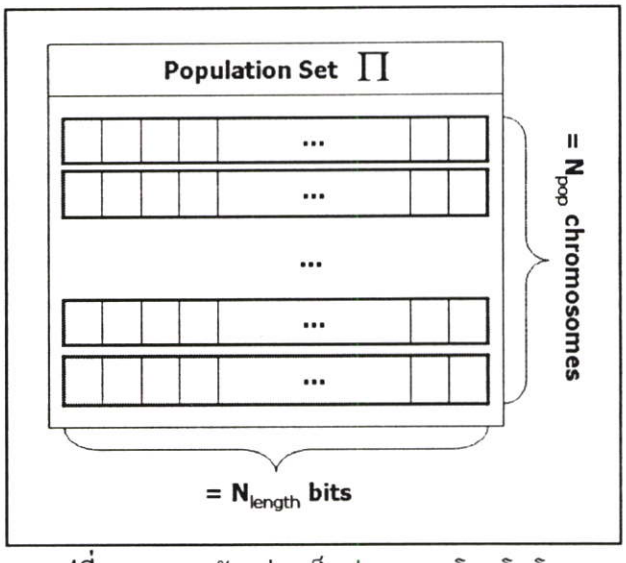
รูปที่ 3.4 แสดงภาพรวมของกระบวนการทำงานของวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยใช้ genetic algorithm ในรูปแบบผังงาน

จากรูปที่ 3.4 จะเห็นว่าการทำงานของวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยใช้ genetic algorithm เริ่มต้นโดยการทำกระบวนการกำหนดค่าเริ่มต้น (initialization process) เพียงหนึ่งครั้ง แล้ววนทำซ้ำกระบวนการครอสโอเวอร์ (crossover process) กระบวนการมิวเทชัน (mutation process) และกระบวนการเลือกประชากรในรุ่นถัดไป เป็นจำนวน GENERATION รอบ (ซึ่งค่า GENERATION มีการกำหนดไว้ก่อน) ซึ่งก็เทียบได้กับการดำรงเผ่าพันธุ์โดยสืบทอดโครโมโซมเป็นจำนวน GENERATION รุ่น เราสามารถอธิบายถึงรายละเอียดของการทำงานของกระบวนการหลักของวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยใช้ genetic algorithm ได้ดังต่อไปนี้

3.2.2.3.1 กระบวนการกำหนดค่าเริ่มต้น (Initialization Process)

กระบวนการกำหนดค่าเริ่มต้นถือได้ว่าเป็นกระบวนการทำงานแรกสุดและกระบวนการนี้จะมีการทำงานเพียงครั้งเดียวในตอนแรกนี้เท่านั้น ซึ่งกระบวนการกำหนดค่าเริ่มต้นทำหน้าที่สร้างโครโมโซมประชากรเริ่มต้นโดยสุ่มกำหนดค่าเริ่มต้นให้กับประชากรเหล่านั้น ซึ่งสามารถอธิบายรายละเอียดการทำงานได้ดังนี้

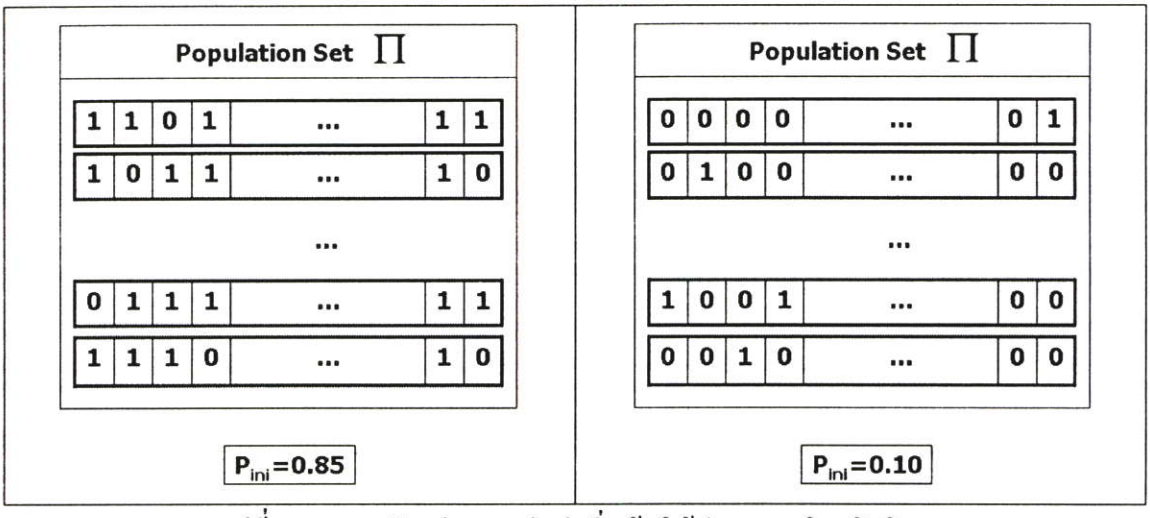
1. สร้างเซ็ตของประชากรโครโมโซม (population set) เริ่มต้นจำนวน N_{pop} โครโมโซม โดยในที่นี้ใช้สัญลักษณ์ Π แทนเซ็ตของประชากรโครโมโซม ($\Pi = \{S_1, ..., S_{N_{pop}}\}$) รูปที่ 3.5 แสดงตัวอย่างเซ็ตของประชากรโครโมโซม Π ซึ่งจะเห็นว่าในเซ็ต Π มีโครโมโซมทั้งหมด N_{pop} โครโมโซม (ค่า N_{pop} เป็นค่าคงที่ที่ถูกกำหนดไว้ก่อน เพื่อกำหนดจำนวนประชากรโครโมโซมในแต่ละ generation) แต่ละโครโซมแทนด้วยบิตสริง (bit string) ความยาว N_{length} บิต (N_{length} มีค่าเท่ากับขนาดของเซ็ตข้อมูลตั้งต้น)



รูปที่ 3.5 แสดงตัวอย่างเซ็ตประชากรโครโมโซม

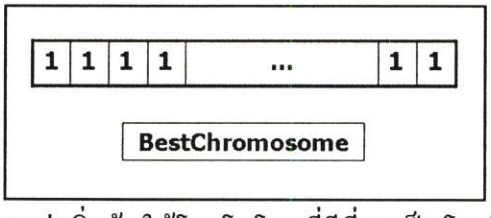
2. สุ่มค่าเริ่มต้นให้กับแต่ละบิตในโครโมโซมทั้งหมด โดยแต่ละบิตในโครโมโซมทำการสุ่มด้วยค่าความน่าจะเป็นที่ได้กำหนดไว้ก่อน (P_{ini}) นั่นคือแต่ละบิตในโครโมโซมมีโอกาสที่จะสุ่มให้มีค่าเป็น "1" เท่ากับ P_{ini} และมีโอกาสที่จะสุ่มให้มีค่าเป็น "0" เท่ากับ 1- P_{ini} ซึ่งการที่มีค่า P_{ini} ทำให้สามารถควบคุมลักษณะการสร้างประชากรโครโมโซมเริ่มต้นได้ เช่น ต้องการให้ประชากรโครโมโซมเริ่มต้นที่มีลักษณะหนาแน่น (แต่ละบิตมีโอกาสที่จะเป็น 1 ได้สูง) เราก็กำหนดให้ $P_{ini} = 0.85$ (กำหนดให้ค่า P_{ini} ใกล้ 1) หากว่าเราต้องการให้ประชากรโครโมโซมเริ่มต้นมีลักษณะเบาบาง (แต่ละบิตมีโอกาสที่จะเป็น 1 ได้ต่ำ) เราก็กำหนดให้ $P_{ini} = 0.10$ (กำหนดให้ค่า P_{ini} ใกล้ 0) เป็นต้น ซึ่งหากประชากรโครโมโซมเริ่มต้นหนาแน่นค่าความถูกต้องในการแยกแยะข้อมูล $classifiedNum(S)$ จะมีค่าสูง แต่จำนวนโปรโตไทป์ที่ถูกเลือก $size(S)$ นั้นก็จะมีจำนวนมากด้วย ในทางตรงกันข้ามหากประชากรโครโมโซมเริ่มต้นเบาบางค่าความถูกต้องในการแยกแยะข้อมูล $classifiedNum(S)$ จะมีค่าต่ำ แต่จำนวนโปรโตไทป์ที่ถูกเลือก $size(S)$ นั้นก็จะมีจำนวนน้อย ซึ่งทำให้เราสามารถปรับค่าเริ่มต้นที่เหมาะสมได้ ในรูปที่ 3.6 แสดงตัวอย่างสุ่มค่าเริ่มต้นให้กับแต่ละบิตในโครโมโซมประชากรทั้งหมด โดยในด้านซ้ายของรูปที่ 3.6 แสดง

เซ็ตประชากรโครโมโซมที่ถูกสุ่มค่าเริ่มต้นโดยใช้ความน่าจะเป็น $P_{ini} = 0.85$ ซึ่งจะสังเกตเห็นได้ว่าประชากรโครโมโซมมีบิตที่มีค่าเป็น 1 อยู่อย่างหนาแน่น ส่วนในด้านขวาของรูปที่ 3.6 แสดงเซ็ตประชากรโครโมโซมที่ถูกสุ่มค่าเริ่มต้นโดยใช้ความน่าจะเป็น $P_{ini} = 0.10$ ซึ่งจะสังเกตเห็นได้ว่าประชากรโครโมโซมมีบิตที่มีค่าเป็น 1 อยู่อย่างเบาบาง



รูปที่ 3.6 แสดงตัวอย่างการสุ่มค่าเริ่มต้นให้ประชากรโครโมโซม

3. สร้าง "โครโมโซมที่ดีที่สุด (*BestChromosome*)" ซึ่งเป็นโครโมโซมที่ใช้เก็บค่าโครโมโซมที่มีความสอดคล้องกับชุดข้อมูลตั้งต้นที่มีขนาดเล็กมากที่สุดในโครโมโซมทั้งหมดที่เคยได้มีการสร้างขึ้นมา โดยกำหนดค่าเริ่มต้นให้โครโมโซมที่ดีที่สุดมีค่าเป็น 1 ทุกบิต ซึ่งหมายถึงโครโมโซมที่ดีที่สุดในตอนเริ่มต้น คือโครโมโซมที่เข้ารหัสแทนเซ็ตโปรโตไทป์ที่มีออบเจ็คในเซ็ตข้อมูลตั้งต้นทั้งหมดเป็นสมาชิก ซึ่งรับรองว่าโครโมโซมที่ดีที่สุดอันนี้จะมีความสอดคล้องกับชุดข้อมูลตั้งต้นอย่างแน่นอนแต่มีโปรโตไทป์จำนวนมากในตอนเริ่มต้น ภายหลังถ้าเกิดมีโครโมโซมที่มีความสอดคล้องกับเซ็ตข้อมูลตั้งต้นและมีขนาดเล็กมากกว่าโครโมโซมที่ดีที่สุดแล้วโครโมโซมที่ดีที่สุดจะถูกปรับปรุงค่าให้เท่ากับโครโมโซมอันนั้นแทน (การปรับปรุงค่าโครโมโซมที่ดีที่สุดอยู่ในกระบวนการเลือกประชากรในรุ่นถัดไป)



รูปที่ 3.7 แสดงการกำหนดค่าเริ่มต้นให้โครโมโซมที่ดีที่สุดเป็นโครโมโซมที่มีทุกบิตเป็น 1

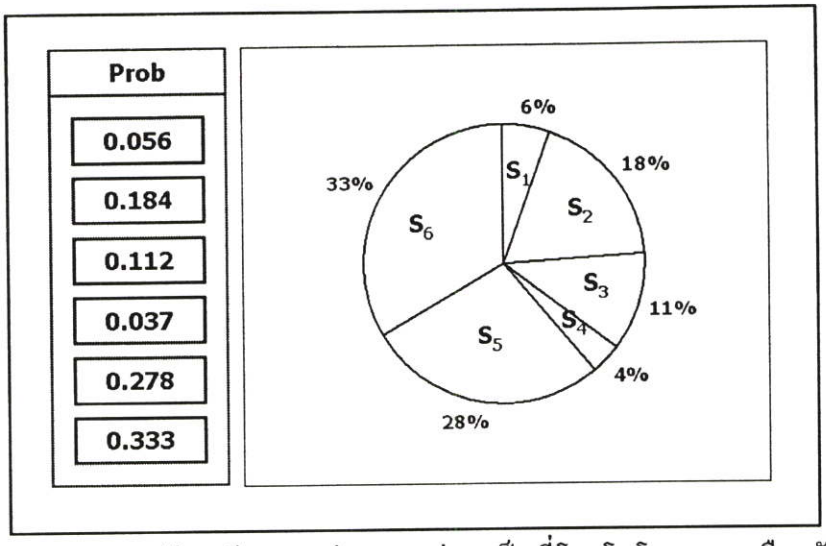
3.2.2.3.2 กระบวนการครอสโอเวอร์ (Crossover Process)

จากขั้นตอนที่แล้วได้ทำการสร้างและกำหนดค่าให้ประชากรโครโมโซมทั้ง N_{pop} โครโมโซม กระบวนการครอสโอเวอร์เป็นกระบวนการที่ทำการเลือกสุ่มเลือกโครโมโซมที่ดีจากกลุ่มประชากรโครโมโซมมาใช้เป็นพ่อพันธุ์แม่พันธุ์เพื่อสร้างโครโมโซมลูกหลานในรุ่นต่อไป ซึ่งสามารถอธิบายรายละเอียดการทำงานได้ดังนี้

1. สร้าง mating set (M) ของเซ็ตประชากรโครโมโซม Π โดยใช้หลักการรูเล็ท (roulette principle) นั่นคือเมื่อสุ่มเลือกโครโมโซม (จากโครโมโซมที่มีอยู่ในเซ็ตประชากรโครโมโซม Π) ผ่านทาง mating set (M) นี้ โอกาสที่โครโมโซมแต่ละตัวจะถูกเลือกนั้นจะเป็นอัตราส่วนเชิงเส้นกับค่าของฟิตเนสฟังก์ชันของโครโมโซมตัวนั้น ซึ่งนั้นแปลว่าโครโมโซมที่มีค่าฟิตเนสฟังก์ชันสูงจะมีความน่าจะเป็นที่ถูกสุ่มเลือกได้สูงเช่นกัน รูปที่ 3.8 แสดงตัวอย่างในการสร้าง mating set ซึ่งจะเห็นได้ว่าเซ็ตประชากรโครโมโซม Π มีโครโมโซม 6 ตัว (S_1 ถึง S_6) เมื่อนำไปคำนวนหาค่าฟิทเนชฟังก์ชันจะได้ค่าดังแสดงในตาราง $\mathbb{F}(S)$ เมื่อนำค่าฟิทเนชฟังก์ชันมารวมแบบสะสม(accumulation)จากบนลงล่างจะสามารถแสดงได้ดังตาราง Acc จากนั้นนำค่าในตาราง Acc มาทำนอร์มอลไลท์ (normalize) (โดยในที่นี้หารทุกด้วยค่า 2.7 เพื่อให้ค่าไม่เกิน1) ถึงตอนนี้ถ้าเราสุ่มค่าที่อยู่ในช่วงตั้งแต่ 0 ถึง 1 แล้วความน่าจะเป็นที่ค่าสุ่มนั้นจะอยู่ในช่วงของโครโมโซม S_1 คือ 0 ถึง 0.056 มีค่า 0.056 และความน่าจะเป็นที่ค่าสุ่มนั้นจะอยู่ในช่วงของโครโมโซม S_2 คือ 0.056 ถึง 0.240 มีค่า 0.184 ในทำนองเดียวกันนี้เราสามารถคำนวนค่าความน่าจะเป็นที่จะสุ่มค่าไปตกอยู่ในช่วงของ S_1 ถึง S_6 ซึ่งสามารถแสดงค่าความน่าจะเป็นเหล่านั้นได้ดังตาราง Prob นั่นเอง และในรูปที่ 3.9 แสดงการเปรียบเทียบระหว่างความน่าจะเป็นที่โครโมโซมจะถูกเลือกกับรูเล็ท ซึ่งจะเห็นได้ว่าโครโมโซมที่มีค่าฟิทเนชฟังก์ชันมากช่องรูเล็ทของโครโมโซมนั้นก็จะกว้างมีโอกาสถูกเลือกได้สูงนั้นเอง

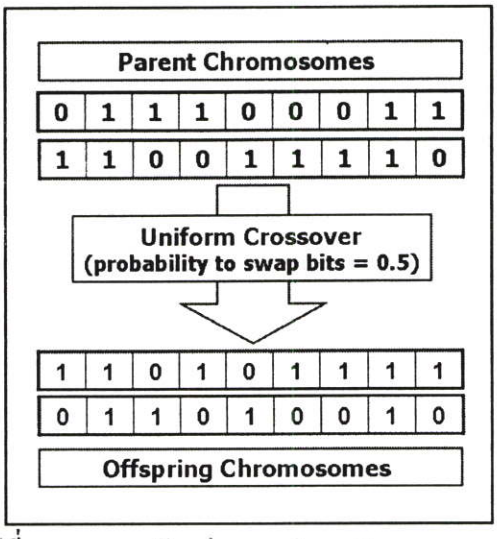
Π	$\mathbb{F}(S)$	Acc	Norm	Prob
S_1	0.15	0.15	0.056	0.056
S_2	0.50	0.65	0.240	0.184
S_3	0.30	0.95	0.352	0.112
S_4	0.10	1.05	0.389	0.037
S_5	0.75	1.80	0.667	0.278
S_6	0.90	2.70	1	0.333

รูปที่ 3.8 แสดงตัวอย่างการสร้าง mating set



รูปที่ 3.9 แสดงการเปรียบเทียบระหว่างความน่าจะเป็นที่โครโมโซมจะถูกเลือกกับรูเล็ท

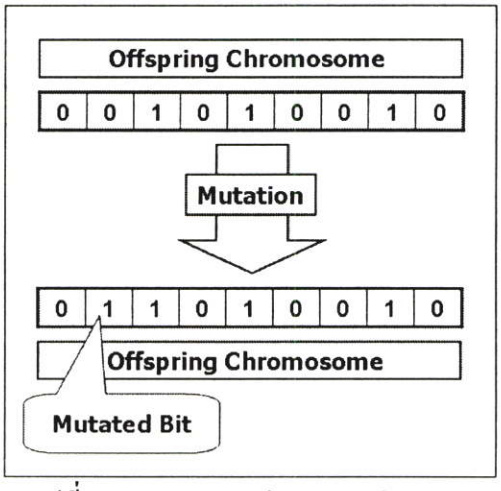
2. สุ่มเลือกโครโมโซมทีละคู่ผ่านทาง mating set (**M**) ที่ได้สร้างไว้ แล้วนำคู่โครโมโซมที่เลือกได้มาทำการคลอสโอเวอร์แบบ "uniform crossover" โดยนำโครโมโซม 2 ตัวมาทำการสุ่มเพื่อสลับบิตกัน โดยความน่าจะเป็นที่จะทำการสลับค่ากันมีค่าเท่ากับ 0.5 นั่นคือแต่ละบิตมีโอกาสเท่ากันที่จะทำการสลับบิตหรือไม่สลับบิต ซึ่งจะเห็นได้ว่าการทำ uniform crossover หนึ่งครั้งใช้โครโมโซมพ่อแม่ 2 โครโมโซม (parent chromosomes) และให้ผลลัพท์เป็นโครโมโซมลูกหลาน 2 โครโมโซม (offspring chromosome) ขั้นตอนนี้จะทำการสุ่มเลือกคู่โครโมโซมขึ้นมาทำครอสโอเวอร์เป็นจำนวน N_{pop} ครั้ง ทำให้ได้ผลลัทพ์เป็นโครโมโซมลูกหลานจำนวน $2N_{pop}$ โครโมโซม ซึ่งโครโมโซมลูกหลานทั้งหมดนี้จะถูกเก็บรวมกันไว้ในเซ็ตโครโมโซมลูกหลาน O (Offspring Set) รูปที่ 3.10 แสดงตัวอย่างการทำ uniform crossover ซึ่งจะเห็นได้ชัดเจนว่าการทำ uniform crossover หนึ่งครั้งใช้โครโมโซมพ่อแม่ 2 โครโมโซมและให้โครโมโซมลูกหลาน 2 โครโมโซม ซึ่งในรูปแสดงโครโมโซมลูกหลานที่เกิดจากโครโมโซมพ่อแม่ที่สลับบิตในลำดับที่ 1, 3, 6 และ 7



รูปที่ 3.10 แสดงตัวอย่างการทำ uniform crossover

3.2.2.3.3 กระบวนการมิวเทชัน (Mutation Process)

กระบวนการมิวเทชันคือกระบวนการกลายพันธุ์ซึ่งจะแตกต่างกับกระบวนการครอสโอเวอร์ตรงที่บิตในโครโมโซมลูกหลานไม่ได้สืบทอดมาจากบิตของโครโมโซมพ่อแม่ ซึ่งกระบวนการทำมิวเทชันในที่นี้คือการสุ่มเปลี่ยนค่าบิตในโครโมโซมลูกหลานโดยแต่ละบิตในโครโมโซมลูกหลานทั้งหมดมีโอกาสเปลี่ยนค่าจาก "0" เป็น "1" หรือจาก "1" เป็น "0" (กลายพันธุ์) ด้วยค่าความน่าจะเป็น "mutation rate (P_m)" ที่ได้กำหนดไว้ก่อน รูปที่ 3.11 แสดงตัวอย่างการทำมิวเทชันโครโมโซมลูกหลานตัวหนึ่งซึ่งจะเห็นว่าบิตที่ 2 ของโครโมโซมลูกหลานเป็นบิตที่เกิดการกลายพันธุ์เพราะถูกสุ่มเปลี่ยนค่าบิตจาก "0" เป็น "1"

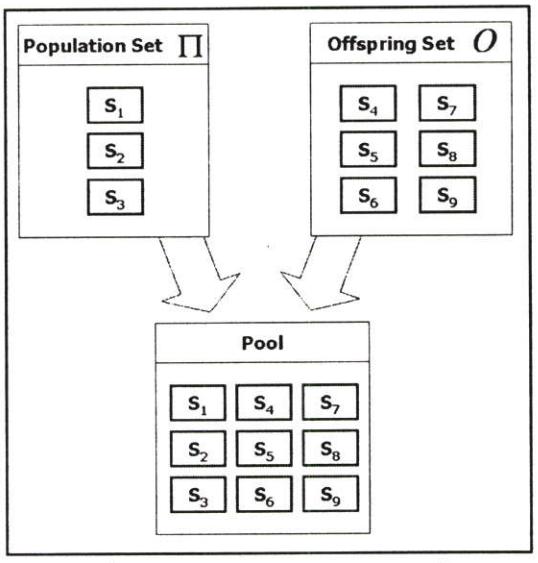


รูปที่ 3.11 แสดงตัวอย่างการทำมิวเทชัน

3.2.2.3.4 กระบวนการเลือกประชากรในรุ่นถัดไป (Selection Process)

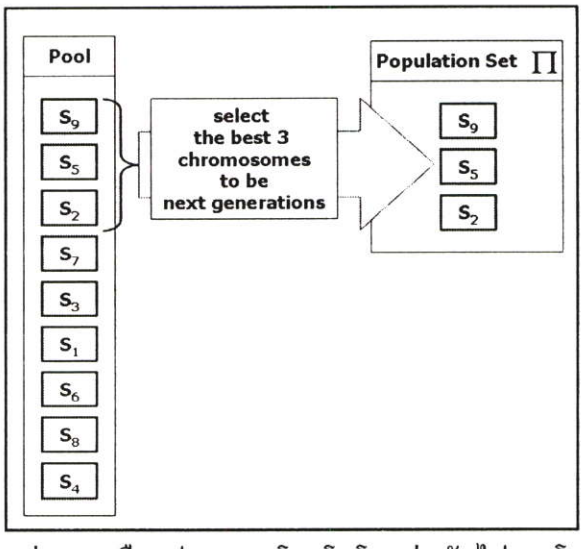
กระบวนการนี้จะทำการเลือกเก็บโครโมโซมบางส่วนจากโครโมโซมที่มีอยู่ทั้งหมดไปเป็นประชากรโครโมโซมในรุ่นถัดไปและทำการปรับปรุงค่าโครโมโซมที่ดีที่สุด สามารถอธิบายรายละเอียดของกระบวนการเลือกประชากรในรุ่นถัดไปได้ดังนี้

1. นำโครโมโซมในเซ็ตประชากรโครโมโซม Π และเซ็ตโครโมโซมลูกหลาน O ทั้งหมดมารวมกันไว้เป็นโครโมโซมกองกลาง (chromosome pool) แสดงตัวอย่างได้ดังรูปที่ 3.12



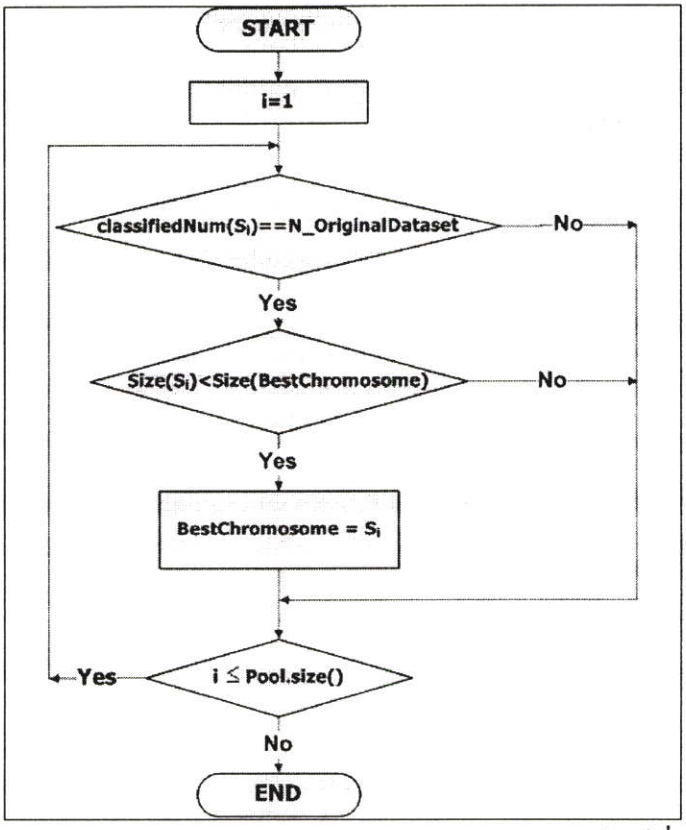
รูปที่ 3.12 แสดงตัวอย่างการรวมเซ็ตประชากรโครโมโซมและเซ็ตโครโมโซมลูกหลานไว้ด้วยกัน

2. เลือกเฉพาะโครโมโซมที่มีค่าฟิทเนสมากที่สุดในโครโมโซมกองกลางจำนวน N_{pop} ตัวไว้เป็นประชากรโครโมโซมในรุ่นถัดไปตามวิถีทาง “elitist strategy” แสดงตัวอย่างได้ดังรูปที่ 3.13



รูปที่ 3.13 แสดงตัวอย่างการเลือกประชากรโครโมโซมรุ่นถัดไปจากโครโมโซมกองกลาง

3. ทำการปรับปรุงค่าโครโมโซมที่ดีที่สุด (*BestChromosome*) โดยในที่นี้ต้องการปรับปรุงค่าโครโมโซมที่ดีที่สุดให้เก็บค่าเซ็ตโปรโตไทป์ที่มีความสอดคล้องกับเซ็ตข้อมูลตั้งต้นและมีจำนวนโปรโตไทป์น้อยที่สุดเท่าที่จะหาได้ ซึ่งสามารถทำได้โดยการพิจารณาโครโมโซมกองกลาง (chromosome pool) ทีละโครโมโซมว่ามีโครโมโซมใดบ้างที่มีค่า *classifiedNum*(S) (จำนวนออบเจ็คที่แยกแยะถูกต้อง) เท่ากับ $N\_OriginalDataset$ (ขนาดชุดข้อมูลตั้งต้น) ซึ่งถ้าเท่ากันแสดงว่าโครโมโซมนั้นเข้ารหัสแทนเซ็ตโปรโตไทป์ที่มีความสอดคล้องกับเซ็ตข้อมูลตั้งต้น จากนั้นพิจารณาว่าเซ็ตโปรโตไทป์ (ที่โครโมโซมนั้นแทนอยู่) มีขนาดที่เล็กว่าเซ็ตโปรโตไทป์(ที่โครโมโซมที่ดีที่สุดแทนอยู่) หรือไม่ ถ้าเล็กกว่ากำหนดให้โครโมโซมที่ดีที่สุด (*BestChromosome*) มีค่าเท่ากับโครโมโซมนั้น แต่ถ้าใหญ่กว่าให้ผ่านไปพิจารณาโครโมโซมถัดไป แสดงผังงานการปรับรุงค่าโครโมโซมที่ดีที่สุดได้ดังรูปที่ 3.14



รูปที่ 3.14 แสดงผังงานของกระบวนการปรับปรุงค่าโปรโตไทป์ที่ดีที่สุด

3.2.2.4 ข้อดีและข้อเสียของวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยใช้ Genetic Algorithm (GA)

ข้อดีและข้อเสียของวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยใช้ genetic algorithm มีดังต่อไปนี้

ข้อดี

วิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยใช้ genetic algorithm สามารถหาเซ็ตโปรโตไทป์ที่มีขนาดเล็กได้ดีสำหรับชุดข้อมูลตั้งต้นที่มีขนาดเล็ก หรือกล่าวอีกอย่างได้ว่าวิธีการนี้สามารถใช้หาคำตอบที่ใกล้เคียงกับคำตอบที่แท้จริงของปัญหาการหาซับเซ็ตสอดคล้องที่มีขนาดเล็กที่สุดได้ แต่ชุดข้อมูลตั้งต้นต้องมีขนาดที่เล็กเท่านั้น

ข้อเสีย

1. วิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยใช้ genetic algorithm ถือเป็นวิธีการเชิงสุ่ม ดังนั้นวิธีการนี้จึงให้ผลลัพท์ที่ไม่เหมือนเดิมทุกๆครั้ง ซึ่งหมายความว่า ถ้าใช้วิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยใช้ genetic algorithm ในการหาโปรโตไทป์อ้างอิงจากชุดข้อมูลตั้งต้นชุดเดียวกันแต่มีการทำงานสองครั้ง ในการทำงานครั้งแรกอาจจะให้ผลลัพท์เป็นโปรโตไทป์อ้างอิงที่มีขนาดเล็กมาก แต่ในการทำงานครั้งที่สองอาจได้ผลลัทพ์เป็นโปรโตไทป์อ้างอิงที่มีขนาดใหญ่มากก็ได้ วิธีการนี้จึงไม่ใช่วิธีการที่ให้ผลลัพท์ที่แน่นอนทุกครั้งที่มีการทำงาน
2. วิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยใช้ genetic algorithm ไม่เหมาะที่จะใช้ในการลดจำนวนโปรโตไทป์อ้างอิงให้กับชุดข้อมูลที่มีขนาดใหญ่ ทั้งนี้ก็เพราะ genetic algorithm นั้นถือเป็นวิธีการค้นคำตอบวงกว้าง (global search algorithm) ซึ่งทำงานโดยการเฟ้นสุ่มเพื่อค้นคำตอบที่ดีจากปริภูมิที่เป็นไปได้ทั้งหมด (possible space) (ซึ่งมีวิธีการ crossover, mutation และ selection ช่วยในการค้นหาคำตอบ) แต่สำหรับชุดข้อมูลตั้งต้นขนาดใหญ่จะมีปริภูมิที่เป็นไปได้ทั้งหมด (possible space) ขนาดมหาศาล เช่น สำหรับชุดข้อมูลตั้งต้นขนาด 150 ออบเจ็คจะมีปริภูมิที่เป็นไปได้ทั้งหมด (possible space) ขนาด $2^{150} = 1.427 \times 10^{45}$ ดังนั้นจึงเป็นไปได้ยากที่จะค้นพบคำตอบในระดับที่ดีสำหรับข้อมูลที่มีขนาดใหญ่ เพื่อบรรเทาปัญหานี้ genetic algorithm ต้องมี heuristic ที่ดีมาช่วยในการค้นคำตอบ แต่สำหรับวิธีการที่นำเสนอนี้เป็นเพียง genetic algorithm อย่างง่ายที่ใช้ในการเลือกโปรโตไทป์อ้างอิงสำหรับชุดข้อมูลที่มีขนาดเล็กเท่านั้น

3.3 วิธีการ Learning Vector Quantization (LVQ)

วิธีการ learning vector quantization (LVQ) [17][20][21] เป็นวิธีการเรียนรู้แบบมีผู้สอนวิธีการหนึ่ง (supervised learning technique) ซึ่งอัลกอริทึมในการเรียนรู้ของวิธีการ learning vector quantization เป็นอัลกอริทึมการเรียนรู้ในลักษณะที่เรียกว่า "การเรียนรู้แบบแข่งขัน (competitive learning)" โดยผู้ชนะเท่านั้นที่มีสิทธิในการเรียนรู้ในแต่ละรอบ (winner take all) ซึ่งถือได้ว่าเป็นอัลกอลิทึมในการเรียนรู้ลักษณะเช่นเดียวกับวิธีการ self organizing map (SOM) [17] หากแต่ว่าวิธีการ self organizing map เป็นวิธีการในการเรียนรู้แบบไม่มีผู้สอน (unsupervised learning) จึงทำให้วิธีการ self organizing map เหมาะกับการนำไปใช้ในการเรียนรู้เพื่อการจัดกลุ่มข้อมูล (data clustering) เพราะสามารถทำการเรียนรู้เพื่อจัดกลุ่มของข้อมูลได้โดยที่ไม่จำเป็นต้องทราบชนิดของข้อมูลก่อนนั่นเอง และเพราะว่าวิธีการ learning vector quantization นี้เป็นวิธีการเรียนรู้แบบมีผู้สอนดังนั้นวิธีนี้จึงจำเป็นต้องทราบชนิดของข้อมูลเสียก่อน โดยวิธีการ learning vector quantization จะทำการเรียนรู้จากข้อมูลที่ทราบชนิดเหล่านี้เพื่อนำสิ่งที่เรียนรู้ได้ไปใช้ในการปรับค่าให้กับ codebook vector (ซึ่ง codebook vector ก็เปรียบเสมือนกับโปรโตไทป์ของกลุ่มข้อมูลนั้นเอง โดยโปรโตไทป์ในที่นี้ไม่ได้มาจากการเลือกจากชุดข้อมูลตั้งต้นเหมือนในวิธีการที่ผ่านมา แต่ได้จากการคำนวนค่าขึ้นมาใหม่) ดังนั้นวิธีการ learning vector quantization จึงเหมาะแก่การนำไปใช้ในการเรียนรู้เพื่อคำนวนหาค่าตำแหน่งที่เหมาะสมให้กับโปรโตไทป์ของกลุ่มข้อมูลที่ทราบชนิดอยู่แล้ว เพราะฉะนั้นจึงถือได้ว่าวิธีการ learning vector quantization เป็นวิธีการเลือกโปรโตไทป์ (prototype selection) วิธีการหนึ่ง [22]

3.3.1 อัลกอริทึมในการเรียนรู้ของวิธีการ Learning Vector Quantization (LVQ)

การเรียนรู้ของวิธีการ learning vector quantization เกิดขึ้นได้จากการค้นหา codebook vector ที่เป็นผู้ชนะจาก codebook vector ที่มีอยู่ทั้งหมด แล้วทำการปรับค่าให้กับ codebook vector อันนั้น โดยสามารถอธิบายอัลกอลิทึมในการเรียนรู้ของวิธีการ learning vector quantization ได้ดังนี้

กำหนดให้ *CodeBook* เป็นเซ็ตของ codebook vector ที่มีอยู่ทั้งหมด ซึ่งในที่นี้กำหนดให้มีสมาชิก n เวคเตอร์ โดยสามารถแสดงในรูปสัญลักษณ์ได้ว่า $CodeBook = \{m_1, m_2, ..., m_n\}$ ดังนั้นเราสามารถแสดง codebook vector ที่อยู่ภายในเซ็ต *CodeBook* ได้โดยสัญลักษณ์ m_i ซึ่งค่า i มีค่าได้ตั้งแต่ 1 ถึง n และกำหนดให้สัญลักษณ์ x แทน input vector (ซึ่งเป็นเวคเตอร์ที่จะรับเข้ามาเรียนรู้เพื่อใช้ปรับค่าให้กับ codebook vector) และกำหนดให้สัญลักษณ์ m_c แทน codebook vector ที่มีอยู่ใกล้กับ input vector (x) มากที่สุด ซึ่งเรียก codebook vector ได้อีกอย่างว่า "codebook vector ผู้ชนะ (winning codebook vector)"

$$m_c \mid c = \arg\min_i \{\| x - m_i \|\} \tag{3.2}$$

ดังที่ได้กล่าวไปแล้วว่าวิธีการ learning vector quantization เป็นวิธีการเรียนรู้จากข้อมูลที่ทราบชนิดอยู่ก่อน (ในที่นี้ก็คือ input vector (x)) เพื่อนำไปใช้ในการปรับค่าตำแหน่งให้กับ codebook vector ซึ่งหลักการในการเรียนรู้ (leaning rule) เพื่อปรับค่าให้กับ codebook vector ของวิธีการ learning vector quantization สามารถแสดงได้ดังนี้ (ในที่นี้แสดงเฉพาะหลักการเรียนรู้แบบที่ 1 ของ LVQ (type one learning vector quantization: LVQ1)[17][21])

หลักการเรียนรู้ของวิธีการ Learning Vector Quantization (LVQ)

กรณีที่ codebook $m_c(t)$ มีชนิดเดียวกับ input vector $x(t)$ มีการปรับค่า codebook $m_c(t)$ ดังนี้

$$m_c(t+1) = m_c(t) + \alpha(t)[x(t) - m_c(t)] \tag{3.3}$$

กรณีที่ codebook $m_c(t)$ มีชนิดต่างกับ input vector $x(t)$ มีการปรับค่า codebook $m_c(t)$ ดังนี้

$$m_c(t+1) = m_c(t) - \alpha(t)[x(t) - m_c(t)] \tag{3.4}$$

กรณีของ codebook ตัวอื่นๆที่ไม่ใช่ codebook $m_c(t)$ ไม่มีการปรับค่าใดๆ

$$m_i(t+1) = m_i(t) \quad ; \quad i \neq c \tag{3.5}$$

โดยในที่นี้

สัญลักษณ์ $x(t)$ หมายถึง input vector ที่ได้รับเข้ามาในรอบที่ t

สัญลักษณ์ $m_c(t)$ หมายถึง codebook ชนะซึ่งก็คือ codebook อยู่ใกล้กับ $x(t)$ มากที่สุดในรอบที่ t

สัญลักษณ์ $m_c(t+1)$ หมายถึง codebook ชนะซึ่งถูกปรับค่าแล้วกลายเป็น codebook ในรอบถัดไป $t+1$

สัญลักษณ์ $m_i(t)$ หมายถึง codebook ใดๆในเซ็ตของ codebook ในรอบที่ t

สัญลักษณ์ $m_i(t+1)$ หมายถึง codebook ใดๆในเซ็ตของ codebook ในรอบที่ $t+1$

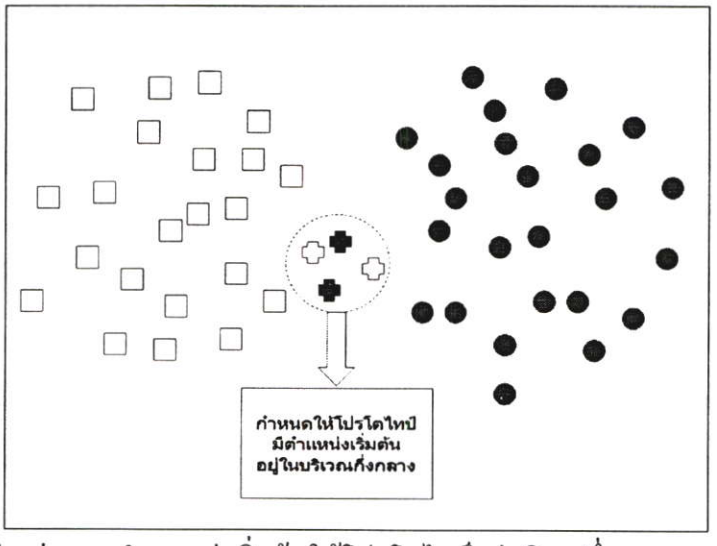
สัญลักษณ์ $\alpha(t)$ หมายถึง อัตราส่วนในการเรียนรู้ (learning rate) ในรอบที่ t ซึ่งอัตราส่วนในการเรียนรู้ก็คือ ค่าสัมประสิทธิ์ที่มีการปรับค่าได้ค่าหนึ่ง ซึ่งจะมีค่าอยู่ในช่วง $1 < \alpha(t) < 1$ โดยที่ค่า $\alpha(t)$ จะค่อยลดลงเรื่อยๆในทุกๆรอบของการเรียนรู้ ซึ่งวิธีการกำหนดการลดลงของค่า $\alpha(t)$ มีหลายวิธี เช่น กำหนดให้เริ่มต้นค่า $\alpha(0)$ มีค่าเป็น 0.1 แล้วลดลงเรื่อยจนถึง 0 ในรอบที่ 1,000 $\alpha(t) = 0.1 - \frac{0.1t}{1,000}$ เป็นต้น

อัลกอริทึมในการเรียนรู้ของวิธีการ learning vector quantization จะทำการรับค่า input vector เข้ามาแล้วทำการเรียนรู้โดยการปรับปรุงค่า codebook ตามหลักการเรียนรู้ (learning rule) ที่ได้กล่าวไปข้างต้น โดยจะวนทำซ้ำ (รับค่าอินพุทแล้วปรับค่า codebook) ไปเรื่อยๆจนกว่าจะเป็นไปตามเงื่อนไขสิ้นสุดการทำงาน ซึ่งเงื่อนไขนี้อาจจะเป็นการกำหนดจำนวนรอบในการทำงานไว้ล่วงหน้า หรือใช้การพิจารณาว่า codebook vector มีการเปลี่ยนแปลงค่าน้อยมากจนแทบจะไม่มีการเปลี่ยนแปลงแล้ว เป็นต้น

3.3.2 กระบวนการทำงานของวิธีการเลือกโปรโตไทป์โดยใช้หลักการ Learning Vector Quantization (LVQ)

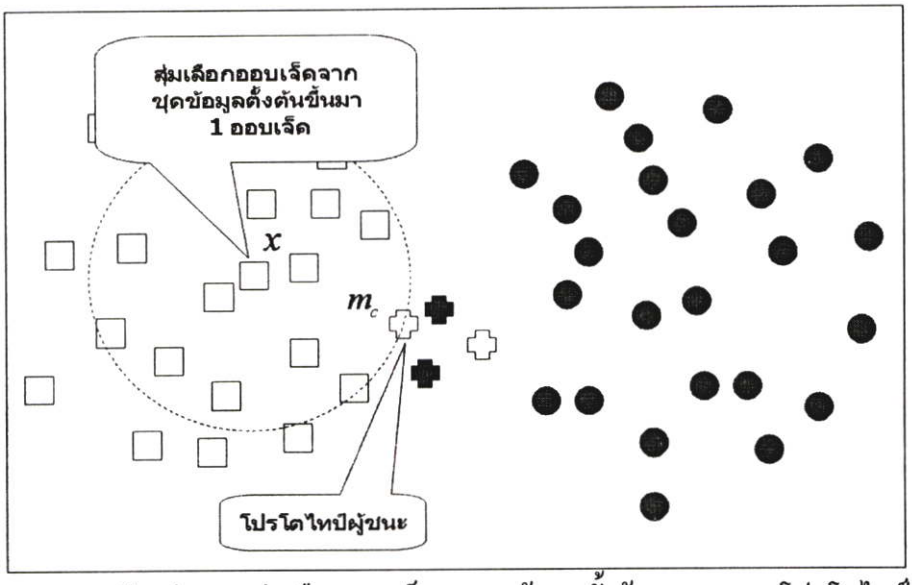
ในส่วนนี้อธิบายตัวอย่างการนำหลักการ learning vector quantization มาใช้เพื่อการคำนวนหาตำแหน่งที่เหมาะสมให้กับโปรโตไทป์ของกลุ่มข้อมูล ซึ่งสามารถอธิบายรายละเอียดของกระบวนการทำงาน ได้ดังต่อไปนี้

1. กำหนดจำนวนของโปรโตไทป์ (ซึ่งเทียบได้กับ codebook นั้นเอง) แต่ละชนิด และกำหนดค่าตำแหน่งเริ่มต้นให้กับโปรโตไทป์ทุกตัว ซึ่งการกำหนดค่าเริ่มต้นให้กับโปรโตไทป์นี้มีหลายวิธี อาทิเช่น กำหนดให้โปรโตไทป์อยู่บริเวณจุดกึ่งกลางของกลุ่มข้อมูลทั้งหมด หรือใช้การจัดกลุ่มข้อมูลเพื่อหาตำแหน่งเริ่มต้นให้กับโปรโตไทป์ (เช่น k-mean เป็นต้น) หรืออาจเป็นการสุ่มตำแหน่งเริ่มต้นให้อยู่ในบริเวณขอบเขตที่เป็นไปได้ เป็นต้น ซึ่งการกำหนดค่าเริ่มต้นนั้นที่เหมาะสมกับลักษณะของข้อมูลจะช่วยให้การเรียนรู้ทำได้รวดเร็วและได้ผลลัพท์ที่ดีขึ้น จากตัวอย่างในรูปที่ 3.15 จะเห็นว่ามีโปรโตไทป์ทั้งหมดจำนวน 4 ตัว (ชนิดละ 2 ตัว) และสุ่มให้โปรโตไทป์มีตำแหน่งเริ่มต้นอยู่ภายในบริเวณกึ่งกลางของกลุ่มข้อมูล



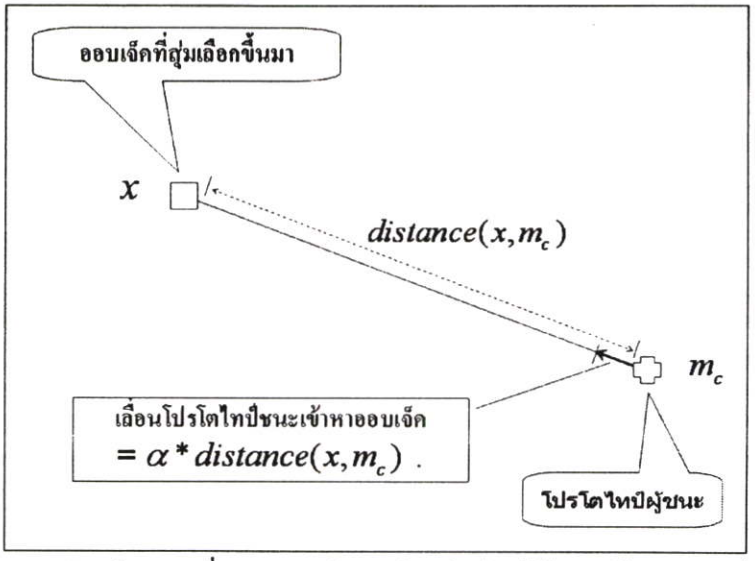
รูปที่ 3.15 แสดงตัวอย่างการกำหนดค่าเริ่มต้นให้โปรโตไทป์อยู่บริเวณกึ่งกลางของกลุ่มข้อมูลตั้งต้น

2. สุ่มเลือกออบเจ็คในเซ็ตข้อมูลตั้งต้นขึ้นมาหนึ่งออบเจ็ค (ซึ่งก็เทียบได้กับการรับค่า input vector เข้ามา) แล้วทำการค้นหาโปรโตไทป์ (ซึ่งเทียบได้กับ codebook) ที่อยู่ใกล้ออบเจ็คนั้นมากที่สุด ซึ่งเรียกโปรโตไทป์นี้ว่า "โปรโตไทป์ผู้ชนะ" ในรูปที่ 3.16 แสดงตัวอย่างการสุ่มเลือกออบเจ็คจากชุดข้อมูลตั้งต้นขึ้นมาหนึ่งตัว x แล้วทำการค้นหาโปรโตไทป์ที่อยู่ใกล้ออบเจ็คนั้นมากที่สุด m_c และกำหนดให้โปโตไทป์นั้นเป็น "โปรโตไทป์ผู้ชนะ"



รูปที่ 3.16 แสดงตัวอย่างการสุ่มเลือกออบเจ็คจากชุดข้อมูลตั้งต้นและการหาโปรโตไทป์ผู้ชนะ

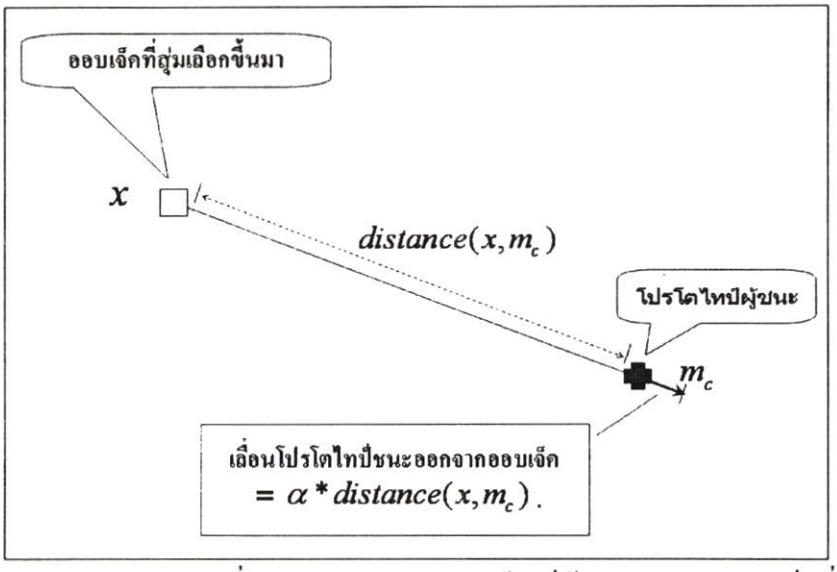
3. ทำการปรับค่าโปรโตไทป์ตามหลักการเรียนรู้ที่ได้กล่าวไปแล้ว โดยพิจารณาที่โปรโตไทป์ผู้ชนะว่ามีชนิดเหมือนกับออบเจ็คที่สุ่มเลือกขึ้นมาหรือไม่
   - กรณีที่โปรโตไทป์ผู้ชนะมีชนิดเหมือนกับออบเจ็คที่สุ่มเลือกขึ้นมา ให้ทำการปรับตำแหน่งของโปรโตไทป์ผู้ชนะตามสมการที่ (3.3) ซึ่งสามารถอธิบายได้ว่าเป็นเสมือนกับการเลื่อนตำแหน่งของโปรโตไทป์ผู้ชนะเข้าหาออบเจ็คที่ถูกสุ่มเลือกขึ้นมาเป็นระยะทางเท่ากับอัตราส่วนการเรียนรู้คูณกับระยะห่างระหว่างโปรโตไทป์ผู้ชนะกับออบเจ็คที่ถูกสุ่มเลือก ในรูปที่ 3.17 แสดงตัวอย่างการปรับค่าตำแหน่งของโปรโตไทป์ผู้ชนะให้เลื่อนเข้าหาออบเจ็คที่ถูกสุ่มเลือก (ซึ่งเป็นกรณีที่โปรโตไทป์ผู้ชนะมีชนิดเหมือนกับออบเจ็คที่ถูกสุ่มเลือก) โดยการเลื่อนนี้จะมีระยะทางเท่ากับอัตราส่วนการเรียนรู้คูณกับระยะห่างระหว่างโปรโตไทป์ผู้ชนะกับออบเจ็คที่สุ่มเลือกขึ้นมา เช่น ถ้าอัตราส่วนการเรียนรู้ (α) มีค่าเท่ากับ 0.1 ก็จะมีระยะทางในการเลื่อนเท่ากับ $\frac{1}{10}$ เท่าของระยะห่างระหว่างโปรโตไทป์ผู้ชนะกับออบเจ็คที่สุ่มเลือกขึ้นมา เป็นต้น



รูปที่ 3.17 แสดงตัวอย่างการเลื่อนตำแหน่งของโปรโตไทป์ผู้ชนะเข้าหาออบเจ็คที่ถูกเลือก

– กรณีที่โปรโตไทป์ผู้ชนะมีชนิดต่างกับออบเจ็คที่สุ่มเลือกขึ้นมา ให้ทำการปรับตำแหน่งของโปรโตไทป์ผู้ชนะตามสมการที่ (3.4) ซึ่งสามารถอธิบายได้ว่าเป็นเสมือนกับการเลื่อนตำแหน่งของโปรโตไทป์ผู้ชนะให้ออกห่างจากออบเจ็คที่ถูกสุ่มเลือกขึ้นมาเป็นระยะทางเท่ากับอัตราส่วนการเรียนรู้คูณกับระยะห่างระหว่างโปรโตไทป์ผู้ชนะกับออบเจ็คที่ถูกสุ่มเลือกขึ้นมา รูปที่ 3.18 แสดงตัวอย่างการปรับค่าตำแหน่งของโปรโตไทป์ผู้ชนะให้เลื่อนออกจากออบเจ็คที่ถูกสุ่มเลือกขึ้นมา (ซึ่งเป็นกรณีที่โปรโตไทป์ผู้ชนะมีชนิดต่างกับออบเจ็คที่ถูกสุ่มเลือกขึ้นมา) โดยการเลื่อนนี้จะมีระยะทางเท่ากับอัตราส่วนการเรียนรู้คูณกับระยะห่างระหว่างโปรโตไทป์ผู้ชนะกับออบเจ็คที่สุ่มเลือกขึ้นมา

สำหรับโปรโตไทป์ตัวอื่นๆที่ไม่ใช่โปรโตไทป์ผู้ชนะจะไม่มีการปรับค่าตำแหน่งแต่อย่างใด ซึ่งก็เป็นไปตามสมการที่ (3.5) และหลักการเรียนรู้แบบแข่งขันที่ผู้ชนะจึงจะมีโอกาสในการเรียนรู้



รูปที่ 3.18 แสดงตัวอย่างการเลื่อนตำแหน่งของโปรโตไทป์ผู้ชนะออกจากออบเจ็คที่ถูกเลือก

4. ทำการลดค่าอัตราส่วนการเรียนรู้ลง ซึ่งการลดอัตราส่วนการเรียนรู้นั้นมีหลายวิธี อาทิเช่น วิธีการลดอัตราส่วนการเรียนรู้แบบเชิงเส้น α โดยมีการกำหนดจำนวนรอบไว้ก่อน ตัวอย่างเช่น กำหนดให้เริ่มต้นค่า $\alpha(0)$ มีค่าเป็น 0.1 แล้วค่อยๆลดลงแบบเชิงเส้นจนถึง 0 ในรอบที่ 1,000 ($\alpha(t) = \alpha(0) - \frac{\alpha(0)*t}{1,000} = 0.1 - \frac{0.1t}{1,000}$) [17] เป็นต้น หรือวิธีการลดอัตราส่วนการเรียนรู้แบบเอกโพเนนเทียล (exponential decay) ตัวอย่างเช่น กำหนดให้เริ่มต้นค่า $\alpha(0)$ มีค่าเป็น 0.01 แล้วค่อยๆลดลงแบบเอกโพเนนเทียลในทุกๆรอบ โดยสมมุติว่ามีอัตราการลดแบบเอกโพเนนเทียล k เท่ากับ 0.1 ($\alpha(t) = \alpha(0) * k^t = 0.01 * 0.1^t$) [23] เป็นต้น ซึ่งการลดอัตราส่วนการเรียนรู้มีผลต่อผลลัพท์สุดท้ายเพราะ ถ้าอัตราส่วนการเรียนรู้มีค่ามากและลดลงช้าเกินไปโปรโตไทป์ก็จะมีการเลื่อนไปมาไม่ลู่เข้าหาตำแหน่งที่เหมาะสม หรือถ้าอัตราส่วนการเรียนรู้มีค่าน้อยและลดลงเร็วเกินไปโปรโตไทป์ก็จะมีการเลื่อนน้อยมากจนไม่สามารถเลื่อนไปถึงตำแหน่งที่เหมาะสมได้
5. วนทำซ้ำกระบวนการ 2-4 ใหม่อีกครั้งจนกว่าจะพบเงื่อนไขสิ้นสุด ซึ่งเงื่อนไขสิ้นสุดอาจเป็นการจำนวนรอบในการเรียนรู้ไว้ก่อน หรือการสังเกตว่าโปรโตไทป์มีการเปลี่ยนแปลงตำแหน่งน้อยมากก็ได้

เมื่อสิ้นสุดการทำงาน ผลลัพท์ที่ได้คือตำแหน่งสุดท้ายของโปรโตไทป์ที่คำนวนได้นั้นเอง โดยจะเห็นได้ว่าในทุกรอบการทำงานโปรโตไทป์จะค่อยปรับตำแหน่งเพื่อเลื่อนเข้าหาตำแหน่งที่เหมาะสมที่โปรโตไทป์ควรจะอยู่ ในช่วงรอบแรกๆของการทำงานโปรโตไทป์จะมีการเลื่อนตำแหน่งเป็นระยะทางที่มากเพราะมีอัตราการเรียนรู้ที่สูง ส่วนในรอบหลังของการทำงานโปรโตไทป์จะมีการเลื่อนตำแหน่งเป็นระยะทางที่น้อยมากเพราะมีอัตราการเรียนรู้มีค่าลดลงเหลือน้อยมาก

แล้วนั่นเอง จนในที่สุดเมื่อสิ้นสุดการทำงานโปรโตไทป์จะเลื่อนสู่ตำแหน่งที่เหมาะสม เพราะฉะนั้นจึงถือว่าตำแหน่งสุดท้ายของโปรโตไทป์คือผลลัพท์ที่คำนวนได้ ซึ่งจะเห็นได้ว่าการได้ผลลัพท์สุดท้ายจะดีหรือไม่นั้น ขึ้นกับการกำหนดค่าพารามิเตอร์เป็นสำคัญ เช่น ถ้ากำหนดพารามิเตอร์การลดค่าอัตราการเรียนรู้ไม่เหมาะสมก็อาจทำให้โปรโตไทป์หยุดเลื่อนก่อนจะถึงจุดที่เหมาะสมได้ เป็นต้น

3.3.3 ข้อดีและข้อเสียของวิธีการเลือกโปรโตไทป์โดยใช้หลักการ Learning Vector Quantization

ข้อดีและข้อเสียของวิธีการเลือกโปรโตไทป์โดยใช้หลักการ Learning Vector Quantization (LVQ) มีดังต่อไปนี้ [24]

ข้อดี

1. วิธีการเลือกโปรโตไทป์โดยใช้หลักการ learning vector quantization มีหลักการเรียนรู้แบบแข่งขันซึ่งเป็นหลักการเรียนรู้ที่เรียบง่ายและมีประสิทธิภาพ
2. วิธีการเลือกโปรโตไทป์โดยใช้หลักการ learning vector quantization สามารถใช้งานกับข้อมูลที่มีข้อมูลที่ผิดพลาด (noise) ปนอยู่ได้ดี นั่นคือวิธีการนี้จะสามารถทำการหาค่าตำแหน่งที่เหมาะสมให้กับโปรโตไทป์ของชุดข้อมูลตั้งต้นได้ดี แม้ว่าจะมีข้อมูลที่ผิดพลาดปนอยู่
3. วิธีการเลือกโปรโตไทป์โดยใช้หลักการ learning vector quantization สามารถกำหนดจำนวนโปรโตไทป์ได้ตั้งแต่เริ่มต้น จะเห็นว่าโปรโตไทป์ผลลัพท์ของวิธีการนี้ไม่ได้จากการเลือกจากชุดข้อมูลตั้งต้น แต่เป็นการกำหนดจำนวนโปรโตไทป์แต่ละชนิดตั้งแต่เริ่มต้น แล้วทำการเรียนรู้เพื่อปรับหาค่าตำแหน่งที่เหมาะสมให้กับโปรโตไทป์เหล่านั้น ดังนั้นวิธีการนี้จึงสามารถกำหนดจำนวนโปรโตไทป์ได้ตามต้องการ

ข้อเสีย

1. วิธีการเลือกโปรโตไทป์โดยใช้หลักการ learning vector quantization จะให้ผลลัพท์ที่ดีหรือไม่ดีนั้นขึ้นอยู่กับการกำหนดค่าพารามิเตอร์ให้กับการทำงาน ทั้งการกำหนดตำแหน่งเริ่มต้นของโปรโตไทป์และการกำหนดการลดลงของอัตตราส่วนการเรียนรู้และการกำหนดเงื่อนไขการสิ้นสุดการทำงาน ซึ่งการกำหนดค่าเริ่มต้นเหล่านี้จำเป็นต้องใช้ประสบการณ์ในการปรับค่าให้เหมาะกับลักษณะของชุดข้อมูลตั้งต้น จึงจะทำให้ได้โปรโตไทป์ผลลัพท์ที่ดี
2. วิธีการเลือกโปรโตไทป์โดยใช้หลักการ learning vector quantization ถือเป็นวิธีการเลือกโปรโตไทป์แบบ replacement technique นั่นคือ โปรโตไทป์ผลลัพท์ได้จากการคำนวนค่าขึ้นมาใหม่ไม่ใช่การเลือกจากเซ็ตข้อมูลตั้งต้น ซึ่งนั่นทำให้วิธีการนี้ใช้ได้กับเฉพาะชุดข้อมูลที่สามารถคำนวนเพื่อเลื่อนตำแหน่งได้ แต่สำหรับข้อมูลบางประเภทไม่สามารถ

คำนวนเพื่อสร้างขึ้นมาใหม่ได้ ดังนั้นวิธีการนี้จึงไม่สามารถใช้กับข้อมูลประเภทนี้ได้ เช่น ข้อมูลลายมือเขียน เป็นต้น

3. วิธีการเลือกโปรโตไทป์โดยใช้หลักการ learning vector quantization ไม่ได้รับประกันว่าโปรโตไทป์ผลลัพท์ที่ได้จะมีความสอดคล้องกับชุดข้อมูลตั้งต้นที่ใช้สอนแต่อย่างใด

แม้ว่าวิธีการ learning vector quantization จะถือได้ว่าเป็นการเลือกโปรโตไทป์วิธีการหนึ่งแต่ก็ว่าไม่ใช่วิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (selection-condensing prototype selection) (ดังนั้นจึงไม่จัดให้วิธีการนี้รวมอยู่ในหัวข้อที่ 3.2) และวิธีการนี้ไม่ได้มีเป้าหมายหลักเพื่อการลดจำนวนโปรโตไทป์เช่นดียวกันกับวิธีการที่กล่าวในหัวข้อที่ 3.2 (ไม่ได้ลดโปรโตไทป์โดยตรงแต่สามารถกำหนดจำนวนโปรโตไทป์ได้) และโปรโตไทป์ผลลัพท์ของวิธีการ learning vector quantization ก็ไม่ได้รับประกันว่าจะต้องมีความสอดคล้องกับเซ็ตข้อมูลตั้งต้นอีกด้วย ดังนั้นวิธีการ learning vector quantization จึงไม่สามารถนำไปเปรียบเทียบในเชิงประสิทธิภาพของการลดจำนวนโปรโตไทป์ (โดยที่ยังคงความสอดคล้องกับชุดข้อมูลตั้งต้น) กับวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (selection-condensing prototype selection) แต่สามารถนำมาเปรียบเทียบเชิงประสิทธิภาพในการแยกแยะข้อมูลกับวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้นได้ ดังนั้นวิธีการ learning vector quantization นี้จึงถูกนำไปใช้ในการทดลองเพื่อเปรียบเทียบเชิงประสิทธิภาพในการแยกแยะข้อมูลกับวิธีการเลือกโปรโตไทป์โดยใช้หลักการจัดกลุ่มข้อมูลแบบมีการสอน (ซึ่งวิธีการนี้จะถูกนำเสนอในบทที่ 4)

บทที่ 4
การเลือกโปรโตไทป์โดยใช้วิธีการจัดกลุ่มข้อมูลแบบมีการสอน

4.1 การใช้สัญลักษณ์พื้นฐาน (Common Notation)

หัวข้อนี้แสดงข้อกำหนดการใช้สัญลักษณ์พื้นฐาน ซึ่งสัญลักษณ์เหล่านี้จะถูกใช้ในการอธิบายกฎและนิยามที่จะได้นำเสนอในหัวข้อถัดไป ดังนั้นเนื้อหาในส่วนนี้ถือเป็นการประกาศให้ทราบถึงการใช้สัญลักษณ์ให้เข้าใจเป็นพื้นฐานเดียวกันก่อนดังนี้

ออบเจ็ค (Object) คือ สิ่งของหรือวัตถุใดๆที่เรากำลังกล่าวถึง ซึ่งวัตถุในโลกแห่งความเป็นจริงนั้นไม่สามารถนำมาใช้ในการคำนวนได้ ดังนั้นสิ่งที่เราใช้ในการแสดงถึง (represent) วัตถุนั้นคือค่าปริมาณที่วัดได้จากกคุณลักษณะเฉพาะของวัตถุนั้น เช่น สมมุติว่าเราแสดงถึงมนุษย์ด้วยพิจารณาถึงค่าปริมาณลักษณะเฉพาะเพียง 2 ค่านั่นคือ ค่าปริมาณความสูง (เซ็นติเมตร) และ ค่าปริมาณน้ำหนัก (กิโลกรัม) ซึ่งจะเห็นว่าในตอนนี้เราสามารถพิจารณามนุษย์เป็นเพียงข้อมูลที่มี 2 มิตินั่นคือค่าความสูงและค่าน้ำหนักเท่านั้นที่เราพิจารณา เป็นต้น จากที่เราทราบแล้วว่าเราสามารถแทนวัตถุใดๆด้วยข้อมูลที่มีมิติเท่าจำนวนลักษณะเฉพาะของวัตถุนั้น ดังนั้นเราจึงสามารถพิจารณาว่าวัตถุเป็นเสมือนจุดพิกัดระบุตำแหน่ง (coordinate) จุดหนึ่งบนสเปซของลักษณะเฉพาะ (feature space: $\Re^d$) ซึ่งก็คือสเปซที่มีมิติขนาด d ที่ครอบคลุมจุดพิกัดของออบเจ็คที่เป็นไปได้ทั้งหมดนั้นเอง โดยในที่นี้เรากำหนดให้แสดงออบเจ็คใดๆด้วยสัญลักษณ์ x ซึ่งอยู่ภายในสเปซของลักษณะเฉพาะ $\Re^d$ ได้ดังนี้

Object: $x \in \Re^d$	(4.1)

การเลือกค่าปริมาณลักษณะเฉพาะที่ใช้แสดงวัตถุด้วยนั้นแตกต่างกันไปขึ้นกับการนำไปใช้ เช่น เลือกค่าปริมาณความสูงและค่าปริมาณน้ำหนักอาจจะเพียงพอต่อการนำไปใช้แยกแยะเชิงประมาณว่ามนุษย์คนนั้นมีรูปร่างสมส่วนหรือไม่ แต่ถ้าเราต้องการแยกแยะว่ามนุษย์คนนั้นเป็นคนเชื้อสายใด เอเชีย ยุโรป หรือ แอฟริกา อาจจำเป็นต้องเลือกลักษณะเฉพาะอย่างอื่นแทน เช่น สีผิว สีผม สีนัยน์ตา ฯลฯ เป็นต้น จะเห็นว่าวัตถุคือมนุษย์เหมือนกันแต่เป้าหมายการนำไปใช้แตกต่างกันวัตถุนั้นก็สามารถแสดง (represent) ได้ด้วยค่าลักษณะเฉพาะที่แตกต่างกัน

ป้ายชื่อชนิด (Class Label) คือ ป้ายชื่อระบุชนิดของออบเจ็คนั่นเอง ในที่นี้เรากำหนดให้แสดงป้ายชื่อชนิดด้วยสัญลักษณ์ θ โดยค่าป้ายชื่อชนิดที่เป็นไปได้ทั้งหมดจะต้องอยู่ภายในเซ็ตของชนิด

$\mathbb{C}$ (category) เท่านั้น ซึ่งในที่นี้กำหนดให้เซ็ต $\mathbb{C}$ มีสมาชิกเป็นป้ายชื่อชนิดจำนวน c ป้าย (เท่ากับจำนวนชนิดที่มีอยู่ทั้งหมด) เราสามารถแสดงนิยามของป้ายชื่อชนิดได้ดังนี้

$$\text{Class Label: } \theta \in \mathbb{C} = \{class_1, class_2, ..., class_c\} \tag{4.2}$$

ตัวอย่างของเซ็ตชนิด $\mathbb{C}$ เช่น จากตัวอย่างการแบ่งประเภทมนุษย์ ถ้าต้องการแยกแยะว่ามนุษย์คนนั้นมีรูปร่างแบบใด เซ็ตชนิดรูปร่างมนุษย์ $\mathbb{C}$ อาจเป็นเซ็ต {อ้วน, ผอม, สมส่วน} หรือถ้าต้องการแยกแยะว่ามนุษย์คนนั้นมีเชื้อสายใด เซ็ตชนิดเชื้อสายมนุษย์ $\mathbb{C}$ อาจเป็นเซ็ต {เอเชีย, ยุโรป, แอฟริกา} เป็นต้น ซึ่งถ้าเราต้องการแยกแยะว่ามนุษย์คนนั้นมีรูปร่างแบบใดและกำหนดให้เซ็ตชนิดรูปร่างมนุษย์ $\mathbb{C}$ เป็นเซ็ต {อ้วน, ผอม, สมส่วน} แล้วค่าของป้ายชื่อ θ จะมีค่าที่เป็นไปได้คือ อ้วน ผอม หรือสมส่วนเท่านั้นเป็นอย่างอื่นไม่ได้ ดังนิยาม (4.2) นั่นเอง

ฟังก์ชันวัดระยะทาง (Distance Function) คือ ฟังก์ชันที่ใช้วัดระยะห่างระหว่างออบเจ็ค (จุดพิกัด) 2 ออบเจ็คที่อยู่ภายในสเปซของลักษณะเฉพาะ ซึ่งฟังก์ชันนี้ดำเนินการบนสเปซของลักษณะเฉพาะ $\Re^d$ อันเดียวกันกับสเปซของออบเจ็คทั้งคู่ โดยในที่นี้เรากำหนดให้แสดงฟังก์ชันวัดระยะทางด้วยสัญลักษณ์ δ ซึ่งสามารถแสดงนิยามของฟังก์ชันวัดระยะทางได้ดังนี้

$$\text{Distance Funtion: } \delta(x_i, x_j) \mid x_i, x_j \in \Re^d \tag{4.3}$$

4.2 นิยามที่เกี่ยวกับกฎ Nearest Neighbor Rule (Nearest Neighbor Definition)

ในหัวข้อที่ผ่านมาได้แสดงให้ทราบถึงข้อกำหนดการใช้สัญลักษณ์พื้นฐานที่เราได้กำหนดขึ้นมา ในหัวข้อนี้เป็นการใช้สัญลักษณ์พื้นฐานที่ได้กล่าวไปแล้วนั้นในการแสดงข้อกำหนดที่เกี่ยวกับกฎ nearest neighbor rule ซึ่งสามารถแสดงได้ดังต่อไปนี้

นิยาม เซ็ตโปรโตไทป์อ้างอิง (Reference Set or Prototype Set) คือ เซ็ตของโปรโตไทป์ที่กฎ nearest neighbor rule ใช้อ้างอิงเพื่อการตัดสินใจแยกแยะข้อมูลนั่นเอง ในที่นี้เราแทนเซ็ตโปรโตไทป์อ้างอิง ที่มีสมาชิก m โปรโตไทป์ได้ด้วยเซ็ตของคู่อันดับออบเจ็คคลาส (object-class pairs : V) ที่มีสมาชิก m คู่ โดยแต่ละคู่อันดับประกอบด้วยออบเจ็คและป้ายชื่อชนิดของออบเจ็คอันนั้น ซึ่งสามารถแสดงนิยามของเซ็ตโปรโตไทป์อ้างอิงได้ดังนี้

$$\text{Prototype Set: } V = \{v_1, v_2, ..., v_m\} = \{(x_{v1}, \theta_{v1}), (x_{v2}, \theta_{v2}), ..., (x_{vm}, \theta_{vm})\} \tag{4.4}$$

นิยาม ออบเจ็คไม่ทราบชนิด (Unknown Object) คือ ออบเจ็คที่เราไม่ทราบว่าเป็นชนิดใด (ซึ่งในที่นี้ก็คือออบเจ็คที่ต้องการให้กฎ nearest neighbor rule ช่วยตัดสินใจว่าควรเป็นชนิดใด) ในที่นี้เราแสดงออบเจ็คที่ไม่ทราบชนิดด้วยคู่อันดับออบเจ็คคลาส (object-class pairs) ซึ่งแทนด้วยสัญลักษณ์ (x',θ') ซึ่ง x' ก็คือสัญลักษณ์แสดงจุดพิกัดของออบเจ็คบน feature space ส่วน θ' คือสัญลักษณ์แสดงป้ายชื่อชนิดของออบเจ็ค x' ซึ่งในตอนนี้เรายังไม่ทราบค่า θ' เราสามารถแสดงนิยามออบเจ็คไม่ทราบชนิดได้ดังนี้

$$\text{Unknow Object: } (x',\theta') \mid x' \in \Re^d \tag{4.5}$$

นิยาม โปรโตไทป์อ้างอิงที่ใกล้กับออบเจ็คมากที่สุด คือ โปรโตไทป์ที่อยู่ภายในเซ็ตโปรโตไทป์อ้างอิงที่อยู่ใกล้กับออบเจ็คที่ต้องการทราบชนิดมากที่สุดนั้นเอง

กำหนดให้ v′ เป็นคู่อันดับอันหนึ่ง $v' = (x_{v'}, \theta_{v'})$ ซึ่ง v′ เป็นสมาชิกของเซ็ตโปรโตไทป์อ้างอิง $v' \in V$ และคู่อันดับ (x',θ') แทนออบเจ็คที่ไม่ทราบชนิดดังนิยามข้างต้น(4.5) ภายใต้ข้อกำหนดที่กล่าวมาเราจะสามารถนิยามโปรโตไทป์อ้างอิงที่ใกล้กับออบเจ็คมากที่สุดได้ว่า "คู่อันดับ v' เป็นโปรโตไทป์อ้างอิงที่อยู่ใกล้กับออบเจ็ค x' มากที่สุด" ก็ต่อเมื่อ "ระยะห่างระหว่างตำแหน่ง x' กับตำแหน่ง $x_{v'}$ มีค่าน้อยกว่าหรือเท่ากับระยะห่างระหว่างตำแหน่ง x' กับตำแหน่งของทุกคู่อันดับที่อยู่ภายในเซ็ตโปรโตไทป์อ้างอิง" ซึ่งสามารถเขียนนิยามของโปรโตไทป์อ้างอิงที่ใกล้กับออบเจ็คมากที่สุดได้ดังสมการที่ (4.6)

ในความหมายเดียวกันเราสามารถนิยามได้อีกอย่างว่า"คู่อันดับ v' เป็นโปรโตไทป์อ้างอิงที่อยู่ใกล้กับออบเจ็ค x' มากที่สุด" ก็ต่อเมื่อ "ระยะห่างระหว่างตำแหน่ง x' กับตำแหน่ง $x_{v'}$ มีเท่ากับระยะทางที่สั้นที่ระหว่างตำแหน่ง x' กับตำแหน่งของทุกคู่อันดับที่อยู่ภายในเซ็ตโปรโตไทป์อ้างอิง" ซึ่งเราสามารถเขียนนิยามของโปรโตไทป์อ้างอิงที่ใกล้กับออบเจ็คมากที่สุดได้ดังสมการที่ (4.7)

เราสามารถเขียนนิยามของโปรโตไทป์ที่ใกล้กับออบเจ็คมากที่สุดได้ด้วยสมการที่ (4.6) และ (4.7) ซึ่งทั้งสองสมการเป็นสมการที่สมมูลกัน (มีความหมายเดียวกัน)

$$\text{nearestPrototype}(V,x') = (x_{v'},\theta_{v'}) \mid (x_{v'},\theta_{v'}) \in V \wedge \delta(x_{v'},x') \le \delta(x_i,x') \; for \; \forall i \in \{v1, v2, ..., vm\} \tag{4.6}$$

$$\text{nearestPrototype}(V,x') = (x_{v'},\theta_{v'}) \mid (x_{v'},\theta_{v'}) \in V \wedge \delta(x_{v'},x') = \min_{i=v1}^{vm} \delta(x_i,x') \tag{4.7}$$

นิยาม กฎ Nearest Neighbor Rule กล่าวว่า "กำหนดให้ออปเจ็คที่ไม่ทราบชนิด เป็นชนิดเดียวกันกับชนิดของโปรโตไทป์อ้างอิงตัวที่อยู่ใกล้กับออปเจ็คนั้นมากที่สุด" [1]

จากนิยามของโปรโตไทป์อ้างอิงที่ใกล้กับออบเจ็คมากที่สุดด้วยฟังก์ชันที่คืนค่าเป็นคู่อันดับที่แสดงถึงโปรโตไทป์ที่ใกล้กับออบเจ็คมากที่สุด ดังนั้นเราสามารถนิยามกฎ nearest neighbor rule ได้โดยใช้นิยามของโปรโตไทป์อ้างอิงที่ใกล้กับออบเจ็คมากที่สุดได้ดังสูตรที่ (4.8), (4.9) ซึ่งทั้งสองสูตรนั้นคือสูตรเดียวกันที่เขียนในรูปแบบที่แตกต่างกันเท่านั้น (โดยในที่นี้ใช้เครื่องหมาย "$\equiv$" ในการกำหนดกฎ (rule definition) โดยหมายความว่าสัญลักษณ์กฎด้านซ้ายเท่ากับข้อกำหนดด้านขวา และใช้เครื่องหมาย "$\mapsto$" แทนความหมายว่า "กำหนดค่าให้กับ")

$$NN(V,x') \equiv (categoryOf(\text{nearestPrototype}(V,x')) \mapsto \theta') \tag{4.8}$$

$$NN(V,x') \equiv ((\theta_{v'} \mid \text{nearestPrototype}(V,x')) \mapsto \theta')) \tag{4.9}$$

4.3 นิยามที่เกี่ยวกับความสอดคล้อง (Consistency Definition)

ในหัวข้อที่ผ่านมาได้แสดงให้ทราบถึงนิยามที่เกี่ยวกับกฎ nearest neighbor rule ไปแล้ว ในหัวข้อนี้เป็นการใช้นิยามที่ได้กล่าวไปแล้วนั้นมาสร้างเป็นนิยามที่เกี่ยวกับความสอดคล้อง ซึ่งสามารถแสดงได้ดังต่อไปนี้

นิยาม ความสอดคล้อง (Consistency)

กำหนดให้ O คือ เซ็ตของคู่อันดับออบเจ็คคลาส (object-class pairs) ซึ่งมีขนาด n

$$O = \{o_1, o_2, ..., o_n\} = \{(x_{o1}, \theta_{o1}), (x_{o2}, \theta_{o2}), ..., (x_{on}, \theta_{on})\}$$

และกำหนดให้ V คือ เซ็ตของคู่อันดับออบเจ็คคลาส (object-class pairs) ซึ่งมีขนาด m

$$V = \{v_1, v_2, ..., v_m\} = \{(x_{v1}, \theta_{v1}), (x_{v2}, \theta_{v2}), ..., (x_{vm}, \theta_{vm})\}$$

จากข้อกำหนดข้างต้นเราสามารถนิยามความสอดคล้องได้ว่า "เซ็ต V มีความสอดคล้องกับเซ็ต O" ก็ต่อเมื่อ "กฎ nearest neighbor rule ที่ใช้เซ็ต V เป็นเซ็ตโปรโตไทป์อ้างอิง สามารถแยกแยะออบเจ็คที่อยู่ภายในเซ็ต O ได้ถูกต้องทุกตัว" [14] เราสามารถเขียนนิยามของความสอดคล้องได้ดังสมการ (4.10) และ (4.11) ซึ่งทั้งสองสมการสมมูลกัน

$$V\ consistentTo\ O \leftrightarrow (NN(V,x_i) \text{ is TRUE } for\ \forall i \in \{o1, o2, ..., on\}) \tag{4.10}$$

$$V\ consistentTo\ O \leftrightarrow (\theta_i = \text{nearestPrototype}(V,x_i)\ for\ \forall i \in \{o1, o2, ..., on\}) \tag{4.11}$$

นิยาม ซับเซ็ตสอดคล้อง (Consistent Subset)

กำหนดให้ O และ V คือ เซ็ตของคู่อันดับออบเจ็คคลาส (object-class pairs) เราจะสามารถนิยามว่า "เซ็ต V เป็นซับเซ็ตสอดคล้องของเซ็ต O" ก็ต่อเมื่อ "เซ็ต V เป็นซับเซ็ตของเซ็ต O และเซ็ต V มีความสอดคล้องกับเซ็ต O" ซึ่งเราสามารถเขียนนิยามซับเซ็ตสอดคล้องในรูปประพจน์เชิงตรรกยะ (logical statement) ได้ดังนี้

$$(V \; consistentSubsetOf \; O) \leftrightarrow (V \subset O) \wedge (V \; consistentTo \; O) \qquad (4.12)$$

4.4 นิยามที่เกี่ยวกับการเลือกโปรโตไทป์และปัญหาการเลือกซับเซ็ตสอดคล้องที่มีขนาดเล็กที่สุด (Prototype Selection & Minimal Consistent Subset Selection Problem Definition)

จากที่ได้ทราบเกี่ยวกับการเลือกโปรโตไทป์และประเภทของการเลือกโปรโตไทป์กันไปแล้วในบทที่ 2 วิทยานิพนธ์มีขอบเขตที่สนใจอยู่เฉพาะในส่วนของการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (selection-condensing prototype selection) ดังนั้นเราจึงอธิบายเฉพาะส่วนการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงให้ชัดเจนอีกครั้ง ดังนี้

การเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิง (condensing prototype selection) คือ การกระทำเพื่อลดจำนวนโปรโตไทป์อ้างอิงของกฎ nearest neighbor rule ให้มีจำนวนน้อยลงมากที่สุดเท่าที่ทำได้ โดยที่ยังคงความสามารถในการแยกแยะข้อมูลให้ใกล้เคียงกับกฎ nearest neighbor rule ที่ใช้ชุดข้อมูลตั้งต้น (original dataset) เป็นโปรโตไทป์อ้างอิง ซึ่งการกระทำเพื่อให้ได้มาซึ่งโปรโตไทป์อ้างอิงใหม่นั้นอาจเป็นการเลือกตัวแทนมาจากชุดข้อมูลตั้งต้นหรือเป็นการสร้างโปรโตไทป์อ้างอิงขึ้นมาใหม่เองเลยก็ได้

ส่วนการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (selection-condensing prototype selection) นั้นมีข้อจำกัดเพิ่มขึ้นก็คือ การกระทำเพื่อให้ได้มาซึ่งโปรโตไทป์อ้างอิงใหม่นั้นต้องเป็นการเลือกมาจากชุดข้อมูลตั้งต้นเท่านั้น ซึ่งนั่นหมายความว่าเซ็ตโปรโตไทป์อ้างอิงซึ่งเป็นผลลัพท์ที่หาได้ใหม่จะต้องเป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้นเสมอนั่นเอง ดังนั้นเราสามารถอธิบายการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้นได้ดังนี้

การเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น คือ การเลือกออบเจ็คตัวแทนจากเซ็ตข้อมูลตั้งต้นไปเป็นโปรโตไทป์อ้างอิงให้กับกฎ

nearest neighbor rule ให้มีโปรโตไทป์อ้างอิงจำนวนน้อยที่สุด แต่ยังคงความสามารถในการแยกแยะข้อมูลให้ใกล้เคียงกับกฎ nearest neighbor rule ที่ใช้เซ็ตข้อมูลตั้งต้นเป็นโปรโตไทป์อ้างอิง

หากว่าเรายึดความสอดคล้องเป็นหลักเพื่อรับประกันว่าความสามารถในการแยกแยะข้อมูลของกฎ nearest neighbor rule ซึ่งใช้โปรโตไทป์ที่หามาได้ใหม่เป็นโปรโตไทป์อ้างอิงมีความสามารถในการแยกแยะข้อมูลใกล้เคียงกับกฎ nearest neighbor rule ที่ใช้ชุดข้อมูลตั้งต้นทั้งหมดเป็นโปรโตไทป์อ้างอิง แล้วเราจะสามารถอธิบายการเลือกโปรโตไทป์ที่ยึดหลักความสอดคล้องเพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้นได้ดังนี้

การเลือกโปรโตไทป์ที่ยึดหลักความสอดคล้องเพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น คือ การเลือกออบเจ็คตัวแทนจากเซ็ตข้อมูลตั้งต้นไปเป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule ให้มีโปรโตไทป์อ้างอิงจำนวนน้อยที่สุด โดยที่โปรโตไทป์อ้างอิงที่หามาได้ใหม่ต้องมีความสอดคล้องกับเซ็ตข้อมูลตั้งต้น

จากจุดนี้เราสามารถมองได้ว่าการเลือกโปรโตไทป์ที่ยึดหลักความสอดคล้องเพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้นนั้นเสมือนกับปัญหาที่เรียกว่า "ปัญหาการเลือกซับเซ็ตสอดคล้องที่มีขนาดเล็กที่สุด (minimal consistent subset selection problem)" นั่นเอง ซึ่งเราสามารถนิยามซับเซ็ตสอดคล้องที่เล็กที่สุดบนพื้นฐานของนิยามที่เราได้ตั้งไว้ก่อนแล้ว ได้ดังนี้

<u>นิยาม</u> ซับเซ็ตสอดคล้องที่เล็กที่สุด (Minimal Consistent Subset)

กำหนดให้ O คือ เซ็ตของคู่อันดับออบเจ็คคลาส (object-class pairs) ใดๆ และ V คือ เซ็ตของคู่อันดับออบเจ็คคลาส (object-class pairs) เช่นกัน

เซ็ต V จะเป็นซับเซ็ตสอดคล้องที่เล็กที่สุดของเซ็ต O ก็ต่อเมื่อ เซ็ต V จะเป็นซับเซ็ตสอดคล้องของเซ็ต O และ เซ็ต V มีขนาด(Cardinality) น้อยกว่าหรือเท่ากับ ขนาดของซับเซ็ตสอดคล้องที่เป็นไปได้ทั้งหมดของเซ็ต O เราสามารถเขียนนิยามซับเซ็ตสอดคล้องที่เล็กที่สุดได้ ดังนี้

$$\begin{aligned}&(V\ minimalConsistentSubsetOf\ O)\\ &\leftrightarrow (V\ consistentSubsetOf\ O)\ and\ (|V| \trianglelefteq |V'|\ \ for\ \forall V'\ consistentsubsetOf\ O)\end{aligned} \tag{4.13}$$

จากนิยามของซับเซ็ตสอดคล้องที่เล็กที่สุด (4.13) ซึ่งบอกว่า "เซ็ต V จะเป็นซับเซ็ตสอดคล้องที่เล็กที่สุดของเซ็ต O" ก็ต่อเมื่อ "เซ็ต V จะเป็นซับเซ็ตสอดคล้องของเซ็ต O และ เซ็ต V มีขนาด (Cardinality) น้อยกว่าหรือเท่ากับ ขนาดของซับเซ็ตสอดคล้องที่เป็นไปได้ทั้งหมดของเซ็ต O" เราสามารถอธิบายได้ว่า "เซ็ต V จะเป็นซับเซ็ตสอดคล้องที่เล็กที่สุดของเซ็ต O" ก็ต่อเมื่อ เซ็ต V เป็นไปตามเงื่อนไข 2 เงื่อนไข ดังนี้

เซ็ต V จะเป็นซับเซ็ตสอดคล้องที่เล็กที่สุดของเซ็ต O (V *minimalConsistentSubsetOf* O) ก็ต่อเมื่อ ($\leftrightarrow$) เซ็ต V เป็นไปตามเงื่อนไข 2 ข้อ นั่นคือ

1. เซ็ต V ต้องเป็นซับเซ็ตสอดคล้องของเซ็ต O (V *consistentSubsetOf* O) และ
2. เซ็ต V ต้องมีขนาดน้อยกว่าหรือเท่ากับซับเซ็ตสอดคล้องที่เป็นไปได้ทั้งหมดของเซ็ต O ($|V| \leq |V'|$ *for* $\forall V'$ *consistentsubsetOf* O)

โดยเงื่อนไขข้อที่ 1 นั้นเสมือนเป็นข้อจำกัด ซึ่งมีหน้าที่บังคับให้เซ็ต V ต้องอยู่ภายใต้ข้อจำกัดนี้ ส่วนเงื่อนไขข้อที่ 2 เสมือนเป็นข้อกำหนด ซึ่งกำหนดให้ทำการหาเซ็ต V ที่มีขนาดเล็กที่สุดเท่าที่จะเป็นไปได้ภายใต้เงื่อนไขบังคับข้อที่ 1 เราสามารถอธิบายปัญหาว่านี้เป็นปัญหาการหาค่าเหมาะสมที่สุดแบบมีเงื่อนไขบังคับ (constrained optimization problem) โดยเป็นปัญหาการหาค่าน้อยที่สุด (minimization problem) ที่มีฟังก์ชันก์เป้าหมาย (objective function) เป็น "ขนาดของเซ็ตโปรโตไทป์อ้างอิง V" และมีเงื่อนไขบังคับเป็น "เซ็ต V จะเป็นซับเซ็ตสอดคล้องของเซ็ต O" ดังนั้นเราสามารถเขียนปัญหาการหาซับเซ็ตซับเซ็ตสอดคล้องที่เล็กที่สุดให้อยู่ในรูปแบบปัญหาการหาค่าเหมาะสมที่สุดแบบมีเงื่อนไขบังคับ (constrained optimization problem) ได้ดังนี้

$$\begin{aligned} &\text{Minimize} \quad : |V| && \text{\#(Objective function)} \\ &\text{Subject to} \quad : V \textit{ consistentSubsetOf } O. && \text{\#(Constraints)} \end{aligned} \tag{4.14}$$

ถึงจุดนี้แล้วเราสามารถมองปัญหาการเลือกซับเซ็ตสอดคล้องที่มีขนาดเล็กที่สุดให้อยู่ในรูปปัญหาการหาค่าเหมาะสมที่สุดแบบมีเงื่อนไขบังคับ ซึ่งเราก็สามารถเปลี่ยนรูปของปัญหานี้ให้อยู่ในรูปแบบที่สามารถหาคำตอบได้ด้วยวิธีการเชิงแม่นตรง (exactly technique) เช่น โมเดลให้อยู่ในรูปปัญหากำหนดการเชิงจำนวนเต็ม (integer programming problem : IP) หรือในรูปปัญหากำหนดการเชิงจำนวนเต็มแบบไม่เชิงเส้น (integer nonlinear programming problem : INLP) เป็นต้น ซึ่งหากสนใจเทคนิคในการโมเดลปัญหาในอยู่ในรูปแบบที่สามารถหาคำตอบได้ด้วยวิธีการเชิงแม่นตรงสามารถศึกษาเพิ่มเติมได้จาก [19] หรือภาคผนวกส่วนงานวิจัยที่ได้รับการตีพิมพ์ในส่วนท้ายของวิทยานิพนธ์นี้ ซึ่งได้นำเสนอเทคนิคการโมเดลปัญหาการเลือกซับเซ็ตสอดคล้องที่มีขนาดเล็กที่สุดให้อยู่ในรูปปัญหากำหนดการเชิงจำนวนเต็มแบบไม่เชิงเส้น (integer nonlinear programming problem : INLP) และแสดงผลการทดลองหาคำตอบของปัญหาการเลือกซับเซ็ตสอดคล้องที่มีขนาดเล็กที่สุดนี้กับชุดข้อมูลดอกไอริชโดยใช้วิธีกำหนดการเชิงจำนวนเต็มแบบไม่เชิงเส้น

อย่างไรก็ดีถึงแม้ว่าปัญหาการเลือกซับเซ็ตสอดคล้องที่มีขนาดเล็กที่สุดจะสามารถโมเดลให้อยู่ในรูปแบบปัญหาที่สามารถหาคำตอบได้ด้วยวิธีการเชิงแม่นตรงซึ่งสามารถหาคำตอบที่แท้จริงของปัญหานี้ได้ก็ตาม แต่เพราะว่าปัญหาการเลือกซับเซ็ตสอดคล้องที่มีขนาดเล็กที่สุดนี้เป็นปัญหาเชิง

การจัดหมู่ที่ยาก (hard combinatorial problem) [3] ซึ่งวิธีการเชิงแม่นตรงเช่น integer programming technique หรือ integer nonlinear programming technique นั้นเป็นวิธีการที่หาคำตอบที่แท้จริงของปัญหาดังนั้นจึงต้องเสียเวลาในการทำงานนานมากเกินไป ซึ่งนั้นก็เป็นผลให้การหาคำตอบของปัญหาการเลือกซับเซ็ตสอดคล้องที่มีขนาดเล็กที่สุดโดยวิธีการเชิงแม่นตรงนั้นจึงใช้ได้ดีกับชุดข้อมูลที่มีขนาดเล็กหรือใช้ในการหาคำตอบที่แท้จริงซึ่งใช้ในการเปรียบเทียบเพื่อประเมินประสิทธิภาพของอัลกอลิทึมในทางทฤษฎีเท่านั้น

4.5 แนวคิดของวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มแบบมีการสอน

จากที่ได้ทราบแล้วว่าการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของชุดข้อมูลตั้งต้นนั้นเทียบได้กับความพยายามแก้ปัญเลือกซับเซ็ตสอดคล้องที่มีขนาดเล็กที่สุด ซึ่งปัญหาการเลือกซับเซ็ตสอดคล้องที่มีขนาดเล็กที่สุดนั้นเป็นปัญหาการจัดหมู่ที่ยาก (hard combinatorial problem) [3] ดังนั้นวิธีการเลือกโปรโตไทป์จึงไม่คิดหาคำตอบที่แท้จริงของปัญหาการเลือกซับเซ็ตสอดคล้องที่มีขนาดเล็กที่สุด แต่ต้องการเพียงคำตอบเชิงประมาณของปัญหานี้เท่านั้น นั่นคือการพยายามเลือกโปรโตไทป์อ้างอิงที่มีขนาดเล็กที่สุดเท่าที่จะทำได้แต่ยังคงมีความสอดคล้องกับชุดข้อมูลตั้งต้นอยู่โดยที่ใช้เวลาไม่มากจนเกินไปนัก ซึ่งวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มแบบมีการสอนก็จัดเป็นวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของชุดข้อมูลตั้งต้นแบบหนึ่งเช่นกัน ซึ่งวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มแบบมีการสอนได้มาจากแนวคิดหลักๆ 2 แนวคิด นั่นคือ แนวคิดความครอบคลุมและแนวคิดในการเลือกโปรโตไทป์โดยใช้คุณสมบัติความครอบคลุม ซึ่งสามารถนำเสนอได้ดังต่อไปนี้

4.5.1 แนวคิดความครอบคลุม

แนวคิดความครอบคลุมถือเป็นแนวคิดหลักอย่างหนึ่งของวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มแบบมีการสอน ซึ่งแนวคิดนี้ถูกนำมาใช้ประมาณปัญหาการเลือกซับเซ็ตสอดคล้องที่มีขนาดเล็กที่สุดให้อยู่ในรูปแบบที่ง่ายขึ้นและสามารถนำไปใช้ช่วยในการพิจารณาเลือกโปรโตไทป์ การนำคุณสมบัติของความครอบคลุมไปใช้ช่วยในการพิจารณาเลือกโปรโตไทป์จะนำเสนอในหัวข้อที่ 4.5.2 ส่วนหัวข้อนี้เป็นการนำเสนอนิยามที่เกี่ยวกับความครอบคลุมและการพิสูจน์คุณสมบัติที่ได้จากความครอบคลุม ซึ่งสามารถนำเสนอได้ดังต่อไปนี้

<u>นิยาม</u> คู่ลำดับต่างชนิดที่ใกล้ที่สุด

กำหนดให้ O คือ เซ็ตของคู่อันดับออบเจ็คคลาส (object-class pairs) ซึ่งมีขนาด n

$$O = \{o_1, o_2, ..., o_n\} = \{(x_{o1}, \theta_{o1}), (x_{o2}, \theta_{o2}), ..., (x_{on}, \theta_{on})\}$$

และกำหนดให้ u เป็นคู่ลำดับออบเจ็คคลาสอันหนึ่งซึ่งเป็นสมาชิกของเซ็ต O

$$u = (x_u, \theta_u) \; ; \; u \in O$$

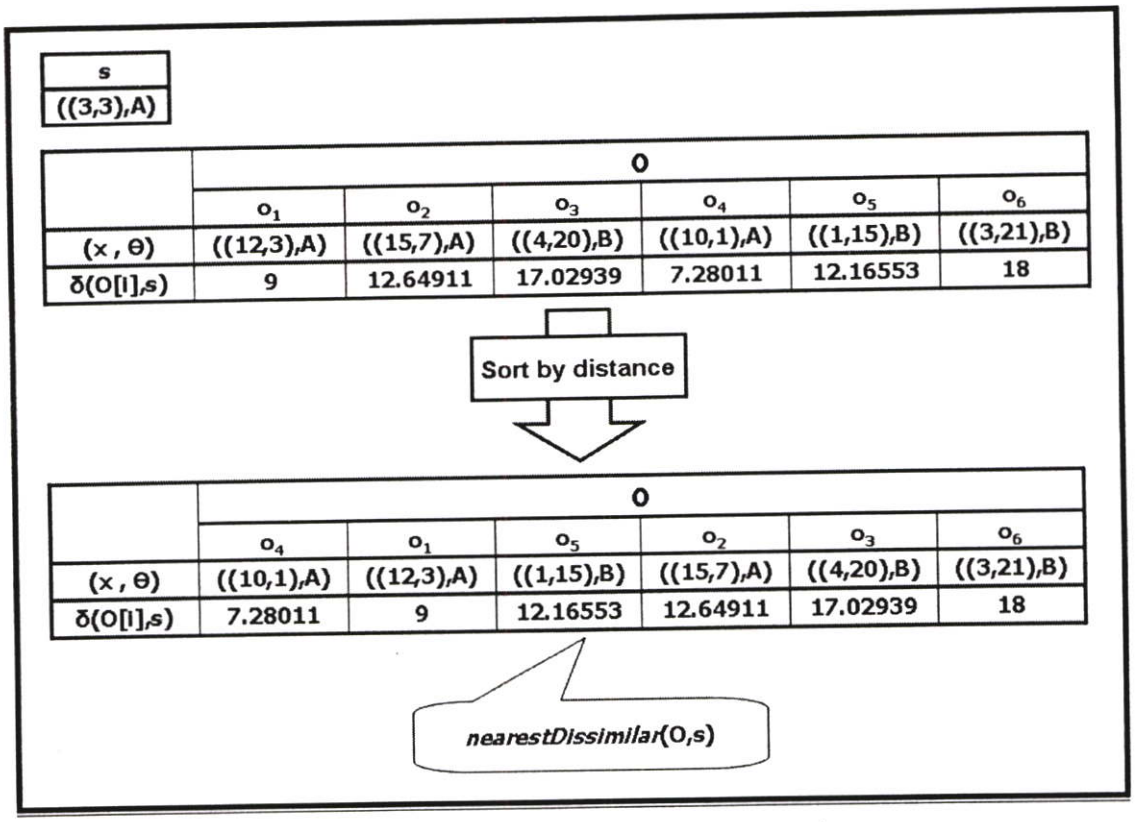
และกำหนดให้ $s = (x_s, \theta_s)$ คือ คู่ลำดับออบเจ็คคลาสอันหนึ่งซึ่งเราทราบทั้งค่าจุดพิกัดออบเจ็ค x_s และป้ายชื่อชนิด θ_s อยู่แล้ว เราจะสามารถนิยามคู่ลำดับต่างชนิดที่ใกล้ที่สุดได้ว่า

คู่อันดับต่างชนิดที่ใกล้คู่อันดับ s ที่สุดในเซ็ต O คือ คู่อันดับ u ซึ่งคู่อันดับ u นี้เป็นสมาชิกของเซ็ต O และคู่อันดับ u นี้มีชนิดต่างกับคู่อันดับ s และคู่อันดับ u นี้อยู่ใกล้คู่อันดับ s มากที่สุดเมื่อเทียบกับคู่อันดับอื่นๆในเซ็ต O ที่มีชนิดต่างกับคู่ลำดับ s

ด้วยข้อกำหนดข้างต้น เราสามารถเขียนนิยามคู่อันดับต่างชนิดที่ใกล้ที่สุดได้ดังนี้

$$\begin{aligned} &nearestDissimilar(O,s) \\ &= u \,|\, (u \in O) \wedge (\theta_u \neq \theta_s) \wedge (\delta(u,s) \leq \delta(o_i,s) \;\; for \forall i \,|\, ((i = o1...on) \wedge (\theta_i \neq \theta_s))) \end{aligned} \tag{4.15}$$

รูปที่ 4.1 แสดงตัวอย่างคู่อันดับต่างชนิดที่ใกล้คู่อันดับ s ที่สุดในเซ็ต O โดยรูปสี่เหลี่ยมด้านบนแสดงคู่อันดับออบเจ็คคลาส s ซึ่งมีจุดพิกัด(ออบเจ็ค)ที่ตำแหน่ง (3,3) และเป็นชนิดA ส่วนตารางบนแสดงถึงเซ็ตของคู่อันดับออบเจ็คคลาส O โดยในแถวที่สองแสดงชื่อของคู่อันดับที่มีอยู่ทั้งหมดในเซ็ต O แถวที่สามแสดงคู่อันดับออบเจ็คคลาสที่มีอยู่ทั้งหมดในเซ็ต O ส่วนในแถวที่สี่แสดงระยะห่างระหว่างแต่ละคู่อันดับที่อยู่ในเซ็ต O กับคู่อันดับ s เมื่อนำคู่อันดับที่อยู่ในเซ็ต O มาเรียงลำดับตามค่าของระยะห่างกับคู่อันดับ s (ค่าในแถวที่สาม)จากน้อยไปมาก จะสามารถแสดงได้ดังตารางด้านล่าง พิจารณาจากตารางด้านล่างจะเห็นได้ว่าคู่อันดับ o_5 : ((1,15),B) นั้นเป็นคู่อันดับที่เป็นสมาชิกของเซ็ต O และเป็นคู่อันดับที่เป็นชนิด B ซึ่งต่างกับชนิดของคู่อันดับ s (s เป็นชนิด A) และเป็นคู่อันดับที่อยู่ใกล้คู่อันดับ s มากที่สุดกว่าคู่อันดับอื่นๆในเซ็ต O ที่มีชนิดไม่เหมือนชนิดของคู่อันดับ s ซึ่งจะเห็นได้ว่าคู่อันดับ o_5 นั้นมีลักษณะที่เป็นไปตามนิยามที่ (4.15) ทั้งหมด ดังนั้นคู่อันดับ o_5 จึงเป็น "คู่อันดับต่างชนิดที่ใกล้คู่อันดับ s ที่สุดในเซ็ต O : $nearestDissimilar(O,s)$"



รูปที่ 4.1 แสดงตัวอย่างคู่อันดับต่างชนิดที่ใกล้คู่อันดับ s ที่สุดในเซ็ต O

นิยาม ความครอบคลุม

กำหนดให้ O คือ เซ็ตของคู่อันดับออบเจ็คคลาส (object-class pairs) ซึ่งมีขนาด n

$$O = \{o_1, o_2, ..., o_n\} = \{(x_{o1}, \theta_{o1}), (x_{o2}, \theta_{o2}), ..., (x_{on}, \theta_{on})\}$$

และกำหนดให้ o' เป็นคู่อันดับออบเจ็คคลาสซึ่งเป็นสมาชิกของเซ็ต O

$$o' = (x', \theta'); o' \in O$$

และกำหนดให้ o'' เป็นคู่อันดับออบเจ็คคลาสซึ่งเป็นสมาชิกของเซ็ต O

$$o'' = (x'', \theta''); o'' \in O$$

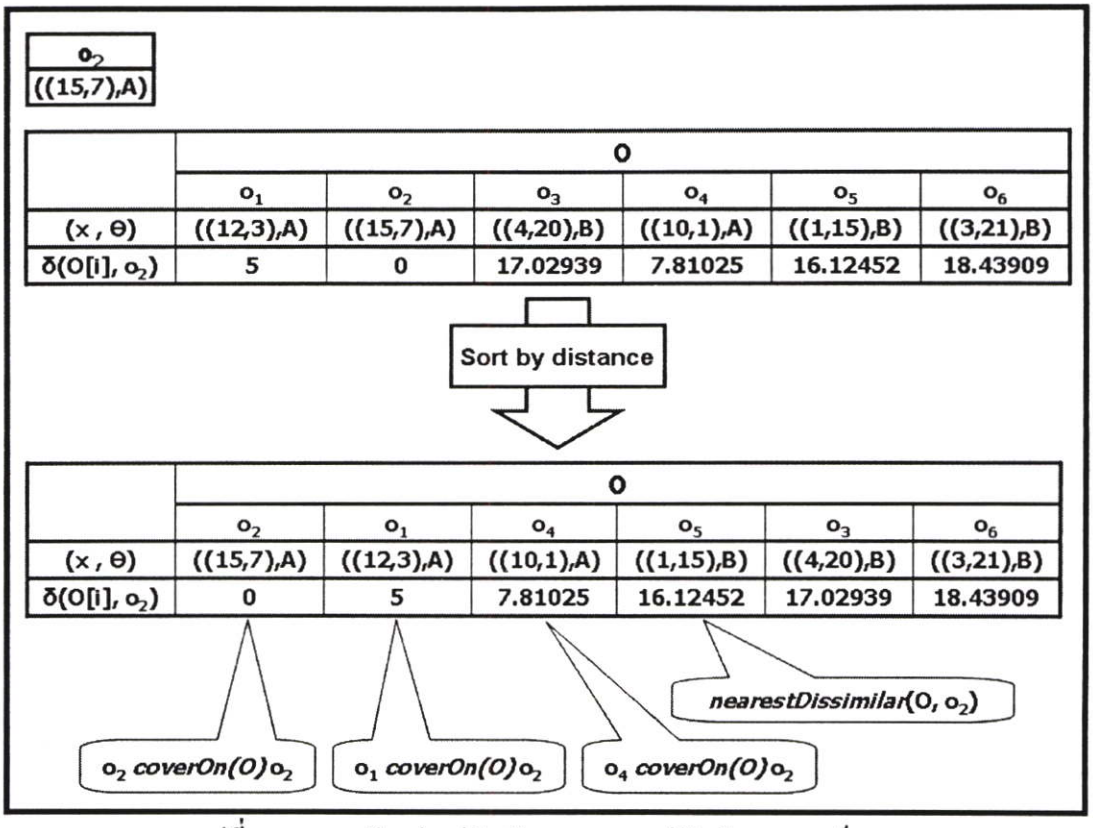
และกำหนดให้ δ แทนฟังก์ชันก็คำนวนระยะทางระหว่างคู่อันดับ 2 คู่อันดับ

เราจะสามารถนิยามความครอบคลุมได้ว่า

"คู่อันดับ o' ครอบคลุม คู่อันดับ o'' บนเซ็ต O" ก็ต่อเมื่อ "คู่อันดับ o' เป็นสมาชิกเซ็ต O" และ "คู่อันดับ o'' เป็นสมาชิกเซ็ต O" และ "คู่อันดับ o' อยู่ใกล้คู่อันดับ o'' มากกว่าคู่อันดับต่างชนิดที่ใกล้คู่อันดับ o'' ที่สุดในเซ็ต O"

ด้วยข้อกำหนดที่ตั้งไว้เราสามารถนิยามความครอบคลุมได้ดังประพจน์ที่ (4.16)

$$o' \text{ coverOn}(O)\ o'' \leftrightarrow (o' \in O) \wedge (o'' \in O) \wedge (\delta(o', o'') < \delta(nearestDissimilar(O, o''), o'') \quad (4.16)$$



รูปที่ 4.2 แสดงตัวอย่างคู่อันดับครอบคลุมคู่อันดับ o_2 บนเซ็ต O

รูปที่ 4.2 แสดงตัวอย่างของคู่อันดับที่มีความครอบคลุมคู่อันดับ o_2 บนเซ็ต O โดยรูปสี่เหลี่ยมด้านบนแสดงคู่อันดับออบเจ็คคลาส o_2 ซึ่งมีจุดพิกัด(ออบเจ็ค)ที่ตำแหน่ง (15,7) และเป็นชนิดA ส่วนตารางบนแสดงถึงเซ็ตของคู่อันดับออบเจ็คคลาส O โดยในแถวที่สองแสดงชื่อของคู่อันดับที่มีอยู่ทั้งหมดในเซ็ต O แถวที่สามแสดงคู่อันดับออบเจ็คคลาสที่มีอยู่ทั้งหมดในเซ็ต O ส่วนในแถวที่สี่แสดงระยะห่างระหว่างแต่ละคู่อันดับที่อยู่ในเซ็ต O กับคู่อันดับ o_2 เมื่อนำคู่อันดับที่อยู่ในเซ็ต O มาเรียงลำดับตามค่าของระยะห่างกับคู่อันดับ o_2 (ค่าในแถวที่สาม)จากน้อยไปมาก จะสามารถแสดงได้ดังตารางด้านล่าง พิจารณาจากตารางด้านล่างจะเห็นได้ว่าคู่อันดับ o_5 : ((1,15),B) เป็น "คู่อันดับต่างชนิดที่ใกล้คู่อันดับ o_2 ที่สุดในเซ็ต O : $nearestDissimilar(O,o_2)$" เพราะว่ามีลักษณะที่เป็นไปตามนิยามที่ (4.15) ครบทั้งหมด และถ้าหากพิจารณาที่คู่อันดับ o_4 : ((10,1),A) และคู่อันดับ o_2 จะเห็นได้ว่าคู่อันดับ o_4 เป็นสมาชิกของเซ็ต O และคู่อันดับ o_2 เป็นสมาชิกของเซ็ต O และคู่อันดับ o_4 อยู่ใกล้กับคู่อันดับ o_2 มากกว่าคู่อันดับ o_5 (ซึ่ง o_5 เป็น $nearestDissimilar(O,o_2)$) เพราะว่าเมื่อพิจารณาจากตารางด้านล่างของรูปที่4.2 จะเห็นว่าคู่อันดับ o_4 เรียงอยู่ในลำดับก่อนหน้าคู่อันดับ o_5 หลังจากพิจารณาเรียบร้อยแล้วจะเห็นได้ว่าคู่อันดับ o_4 และคู่อันดับ o_2 นั้นมีลักษณะที่เป็นไปตามนิยามที่ (4.16) ทั้งหมด ดังนั้นจึงสามารถกล่าวได้ว่า "คู่อันดับ o_4 ครอบคลุมคู่อันดับ o_2 บนเซ็ต O" ในทำนองเดียวกันนั้นเมื่อพิจารณาคู่อันดับอื่นๆในเซ็ต O ก็จะสามารถกล่าวเพิ่มเติมได้

อีกว่า "คู่อันดับ o_1 ครอบคลุมคู่อันดับ o_2 บนเซ็ต O" และ "คู่อันดับ o_2 ครอบคลุมคู่อันดับ o_2 บนเซ็ต O" เช่นกัน

นิยาม คุณสมบัติของความครอบคลุม

ในส่วนที่ผ่านมาได้นำเสนอถึงนิยามของความครอบคลุมกันไปแล้ว เนื้อหาส่วนนี้นำเสนอคุณสมบัติที่เกิดขึ้นเนื่องมาจากความครอบคลุม โดยเรานิยามขึ้นโดยยึดข้อกำหนดต่อไปนี้

กำหนดให้ O คือ เซ็ตของคู่อันดับออบเจ็คคลาสซึ่งมีขนาด n

$$O = \{o_1, o_2, ..., o_n\} = \{(x_{o1}, \theta_{o1}), (x_{o2}, \theta_{o2}), ..., (x_{on}, \theta_{on})\}$$

และกำหนดให้ V เป็น ซับเซ็ตอันหนึ่งของเซ็ต O

$$V \subset O$$

และกำหนดให้ v เป็นคู่อันดับออบเจ็คคลาสอันหนึ่งซึ่งเป็นสมาชิกของเซ็ต V : $(v \in V)$

และกำหนดให้ o เป็นคู่อันดับออบเจ็คคลาสอันหนึ่งซึ่งเป็นสมาชิกของเซ็ต O: $(o = (x, \theta) \in O)$

เราจะสามารถนิยามคุณสมบัติความครอบคลุมได้ว่า

"หากมีคู่อันดับ v ซึ่งคู่อันดับ v ครอบคลุมคู่อันดับ o บนเซ็ต O และคู่อันดับ v เป็นสมาชิกของเซ็ต V และเซ็ต V เป็นซับเซ็ตของเซ็ต O" แล้ว "โปรโตไทป์อ้างอิงในเซ็ต V ที่อยู่ใกล้คู่อันดับ o มากที่สุด จะมีชนิดเดียวกันกับ θ ซึ่งเป็นชนิดของคู่อันดับ o"

ด้วยข้อกำหนดที่ตั้งไว้เราสามารถนิยามความครอบคลุมระหว่างได้ดังประพจน์ที่ (4.17)

$$\exists v \bullet (v \text{ coverOn}(O)\ o) \wedge (v \in V) \wedge (V \subset O) \rightarrow (\theta = categoryOf(\text{nearestPrototype}(V, o))) \quad (4.17)$$

เพราะว่ากฎ nearest neighbor rule จะกำหนดให้คู่อันดับที่ต้องการทราบชนิดเป็นชนิดเดียวกันกับชนิดของโปรโตไทป์อ้างอิงที่ใกล้คู่อันดับที่ต้องการทราบชนิดมากที่สุด ดังนั้นเราจึงสามารถนิยามคุณสมบัติความครอบคลุมได้อีกประพจน์หนึ่งว่า

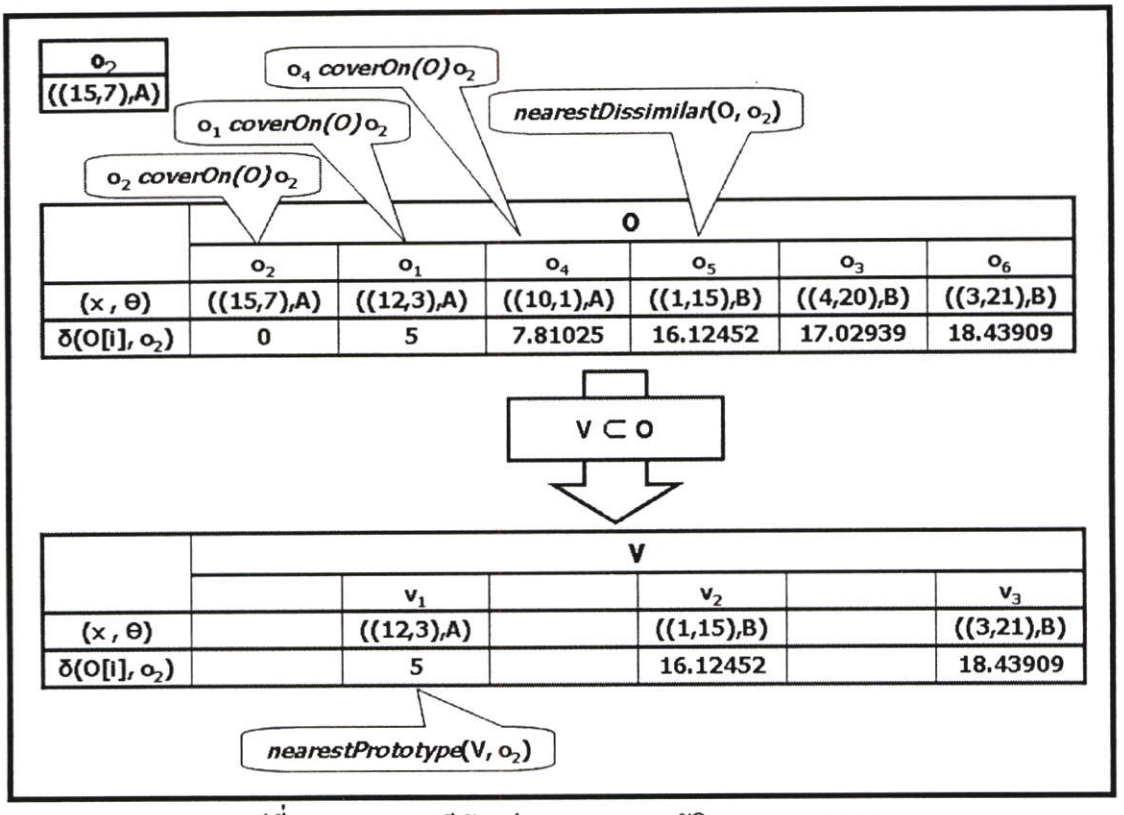
"หากมีคู่อันดับ v ซึ่งคู่อันดับ v ครอบคลุมคู่อันดับ o บนเซ็ต O และคู่อันดับ v เป็นสมาชิกของเซ็ต V และเซ็ต V เป็นซับเซ็ตของเซ็ต O" แล้ว "กฎ nearest neighbor rule ที่ใช้เซ็ต V เป็นเซ็ตโปรโตไทป์อ้างอิงจะแยกแยะคู่อันดับ o ได้อย่างถูกต้อง"

ด้วยข้อกำหนดที่ตั้งไว้เราสามารถนิยามความครอบคลุมระหว่างได้ดังประพจน์ที่ (4.18) (ซึ่งสมมูลกับประพจน์ที่ (4.17))

$$\exists v \bullet (v \text{ coverOn}(O)\ o) \wedge (v \in V) \wedge (V \subset O) \rightarrow (\text{NN}(V, o) \text{ is TRUE}) \quad (4.18)$$

ซึ่งคุณสมบัติของความครอบคลุมสามารถกล่าวอย่างง่ายๆได้ว่า

"หากเซ็ต V ซึ่งเป็นซับเซ็ตของเซ็ต O มีสมาชิกอย่างน้อยหนึ่งตัวที่ครอบคลุมคู่อันดับ o บนเซ็ต O" แล้ว "กฎ nearest neighbor rule ที่ใช้เซ็ต V เป็นเซ็ตโปรโตไทป์อ้างอิงจะแยกแยะคู่อันดับ o ได้อย่างถูกต้อง"



รูปที่ 4.3 แสดงกรณีตัวอย่างของคุณสมบัติความครอบคลุม

รูปที่ 4.3 แสดงกรณีตัวอย่างหนึ่งกรณีเพื่อให้เข้าใจถึงคุณสมบัติความครอบคลุม โดยรูปที่ 4.3 นี้แสดงเซ็ตคู่ลำดับออบเจ็คคลาส O ซึ่งเป็นเซ็ตเดียวกับในรูปที่ 4.2 โดยยกตารางด้านล่างของรูปที่ 4.2 มาเป็นตารางด้านบนของรูปที่ 4.3 ดังนั้นตารางด้านบนของรูปที่ 4.3 จึงแสดงถึงเซ็ตคู่ลำดับออบเจ็คคลาส O ที่มีการเรียงลำดับตามระยะห่างระหว่างแต่ละคู่อันดับกับคู่อันดับ o_2 และจะเห็นได้ว่าคู่อันดับ o_5 เป็น "คู่อันดับต่างชนิดที่ใกล้คู่อันดับ o_2 ที่สุดในเซ็ต O" ตามนิยามที่ (4.15) และหากพิจารณาตามนิยามที่ (4.16) จะสามารถกล่าวได้ว่า "คู่อันดับ o_4 ครอบคลุมคู่อันดับ o_2 บนเซ็ต O" และ "คู่อันดับ o_1 ครอบคลุมคู่อันดับ o_2 บนเซ็ต O" และ "คู่อันดับ o_2 ครอบคลุมคู่อันดับ o_2 บนเซ็ต O" ในส่วนของตารางด้านล่างในรูปที่ 4.3 แสดงถึงเซ็ตคู่ลำดับออบเจ็คคลาส V ซึ่งเป็นซับเซ็ตของเซ็ต O (เลือกสมาชิกของเซ็ต O มาเป็นสมาชิกของเซ็ต V) จากตารางด้านล่างจะเห็นว่าเซ็ต V มีคู่อันดับที่เป็นสมาชิกอยู่ 3 คู่อันดับคือ v_1, v_2 และ v_3 ซึ่งได้มาจากการคู่อันดับ o_1, o_5 และ o_6 ตามลำดับ จะเห็นว่ากรณีนี้เป็นไปตามพจน์แรกของนิยามของคุณสมบัติของความครอบคลุม (4.18)

เพราะเซ็ต V เป็นซับเซ็ตของเซ็ต O และมีคู่ลำดับ v_1 ซึ่งเป็นสมาชิกในเซ็ต V ที่ครอบคลุมคู่ลำดับ o_2 บนเซ็ต O (ทั้งนี้เพราะว่าคู่ลำดับ v_1 เท่ากับคู่ลำดับ o_1 และคู่ลำดับ o_1 นั้นครอบคลุมคู่ลำดับ o_2 บนเซ็ต O) ดังนั้นในกรณีนี้พจน์แรกของนิยาม(4.18) จึงเป็นจริง และเมื่อพิจารณาในตารางด้านล่างซึ่งแสดงเซ็ต V ที่มีลำดับเรียงตามระยะห่างระหว่างคู่ลำดับแต่ละตัวที่อยู่ใน V กับคู่อันดับ o_2 จะเห็นได้ว่าคู่อันดับ v_1 เป็นคู่อันดับที่อยู่ใกล้คู่อันดับ o_2 มากกว่าทุกคู่อันดับในเซ็ต V ดังนั้นคู่อันดับ v_1 จึงเป็นไปตามนิยามโปรโตไทป์อ้างอิงในเซ็ต V ที่ใกล้คู่อันดับ o_2 มากที่สุด (4.6) และคู่อันดับ v_1 เป็นชนิดA เหมือนกับชนิดของคู่อันดับ o_2 ดังนั้นกฎ nearest neighbor rule ที่ใช้เซ็ต V เป็นเซ็ตโปรโตไทป์อ้างอิงจึงสามารถแยกแยะคู่ลำดับ o_2 ได้อย่างถูกต้อง ซึ่งนั่นหมายความว่าในกรณีนี้พจน์หลังของนิยาม(4.18) จึงเป็นจริง เพราะฉะนั้นนิยามคุณสมบัติของความครอบคลุมจึงเป็นจริงในกรณีนี้

<u>พิสูจน์</u> คุณสมบัติของความครอบคุม

ในส่วนนี้นำเสนอเหตุผลที่สามารถยืนยันว่านิยามของคุณสมบัติของความครอบคุมนั้นมีความสมเหตุสมผลหรือเป็นสัจนิรันดร์ (tautology) โดยการพิสูจน์นั้นยึดข้อกำหนดต่อไปนี้

กำหนดให้ O คือ เซ็ตของคู่อันดับออบเจ็คคลาส (object-class pairs) ซึ่งมีขนาด n

$$O = \{o_1, o_2, ..., o_n\} = \{(x_{o1}, \theta_{o1}), (x_{o2}, \theta_{o2}), ..., (x_{on}, \theta_{on})\}$$

และกำหนดให้ V คือ เซ็ตของคู่อันดับออบเจ็คคลาสซึ่งเป็นซับเซ็ตของเซ็ต O

$$V \subset O$$

และกำหนดให้ v เป็นคู่อันดับออบเจ็คคลาสอันหนึ่งซึ่งเป็นสมาชิกของเซ็ต V $(v \in V)$

และกำหนดให้ o เป็นคู่อันดับออบเจ็คคลาสอันหนึ่งซึ่งเป็นสมาชิกของเซ็ต O $(o = (x, \theta) \in O)$

ประพจน์นิยามคุณสมบัติของความครอบคุมกล่าวว่า "หากมีคู่อันดับ v ซึ่งคู่อันดับ v ครอบคลุมคู่อันดับ o บนเซ็ต O และคู่อันดับ v เป็นสมาชิกของเซ็ต V และเซ็ต V เป็นซับเซ็ตของเซ็ต O" แล้ว "โปรโตไทป์อ้างอิงในเซ็ต V ที่อยู่ใกล้คู่อันดับ o มากที่สุด จะมีชนิดเดียวกันกับชนิด θ ซึ่งเป็นชนิดของคู่อันดับ o" ซึ่งก็คือประพจน์ (4.17)

$$\exists v \bullet (v \text{ coverOn}(O)\ o) \wedge (v \in V) \wedge (V \subset O) \rightarrow (\theta = categoryOf(\text{nearestPrototype}(V, o)))$$

สามารถพิสูจน์ว่าเป็นประพจน์ที่สมเหตุสมผลได้ดังนี้

จากข้อกำหนดข้างต้นที่กำหนดให้ O คือเซ็ตของคู่อันดับออบเจ็คคลาส (object-class pairs) ซึ่งมีขนาด n $O = \{o_1, o_2, ..., o_n\} = \{(x_{o1}, \theta_{o1}), (x_{o2}, \theta_{o2}), ..., (x_{on}, \theta_{on})\}$ ซึ่งสามารถแสดงได้ดังรูปที่ 4.4 โดยแถวที่ 2 ของตารางในรูปที่ 4.4 แสดงสัญลักษณ์แทนคู่อันดับตั้งแต่ o_1 ถึง o_n และในแถวที่ 3 แสดงคู่อันดับในเซ็ต O จำนวน n คู่

O			
o_1	o_2	...	o_n
(x_{o1}, Θ_{o1})	(x_{o2}, Θ_{o2})	...	(x_{on}, Θ_{on})

รูปที่ 4.4 แสดงเซ็ตของคู่อันดับออบเจ็คคลาส O ขนาด n ในรูปแบบตาราง

และจากข้อกำหนดที่กำหนดให้ o เป็นคู่อันดับอันหนึ่งซึ่งเป็นสมาชิกของเซ็ต O สามารถคู่อันดับ o แสดงได้ดังรูปที่ 4.5 ซึ่งแสดงถึง o เป็นคู่ลำดับอันหนึ่งที่มีค่าเป็น (x, θ)

o
(x, Θ)

รูปที่ 4.5 แสดงคู่อันดับออบเจ็คคลาส o

เมื่อนำคู่อันดับซึ่งเป็นสมาชิกเซ็ต O มาเรียงลำดับตามระยะห่างกับคู่อันดับ o จากน้อยไปมาก ได้เป็นลิสต์ลำดับ L (ordered list) ซึ่งสามารถนิยามลิสต์ L ได้ดังนี้

$$L = sort(O,o) = \langle (x_{l1},\theta_{l1}),...,(x_{li-1},\theta_{li-1}),(x_{li},\theta_{li}),...,(x_{lm},\theta_{lm}) \rangle \qquad (4.19)$$

ซึ่งเราสามารถแสดงลิสต์ L ในรูปแบบตารางให้เห็นชัดเจนมากยิ่งขึ้นได้ดังรูปที่ 4.6 โดยแถวที่ 2 ของตารางในรูปที่ 4.6 แสดงสัญลักษณ์แทนคู่อันดับตั้งแต่ l_1 ถึง l_n และแถวที่ 3 แทนคู่อันดับที่มีอยู่ที่หมดในลิสต์ L โดยคู่อันดับในลิสต์ L ที่อยู่ในลำดับต่ำ (ด้านซ้าย) จะอยู่ใกล้คู่อันดับ o มากกว่าหรือเท่ากับคู่อันดับในลิสต์ L ที่อยู่ในลำดับสูงกว่า (ด้านขวา) เสมอ

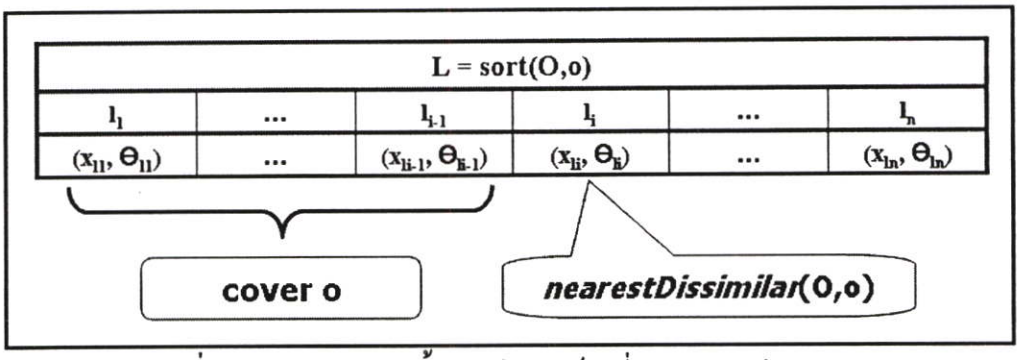
L = sort(O,o)					
l_1	...	l_{i-1}	l_i	...	l_n
(x_{l1}, Θ_{l1})	...	$(x_{li-1}, \Theta_{li-1})$	(x_{li}, Θ_{li})	...	(x_{ln}, Θ_{ln})

nearestDissimilar(O,o)

รูปที่ 4.6 แสดงลิสต์ L ซึ่งได้จากการเรียงลำดับสมาชิกของเซ็ต O ตามระยะห่างที่มีต่อคำอันดับ o

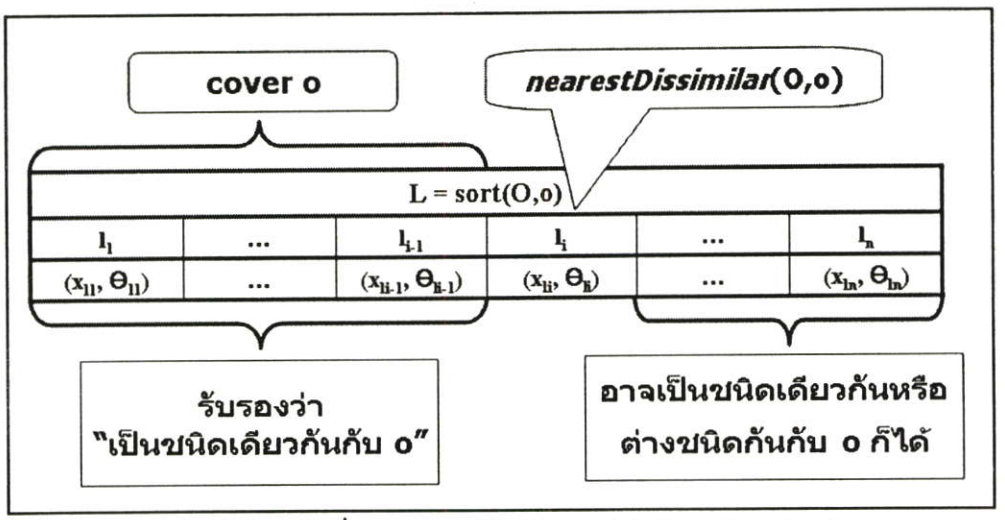
ในที่นี้เรากำหนดให้คู่อันดับที่อยู่ในลำดับที่ i ในลิสต์ L เป็น "คู่อันดับต่างชนิดที่ใกล้คู่อันดับ o ที่สุดในเซ็ต O : $nearestDissimilar(O,o)$" ซึ่งแสดงได้ดังรูปที่ 4.6 จากข้อกำหนดเหล่านี้เมื่อ

พิจารณาตามนิยามของความครอบคลุม (4.16) จะเห็นว่าคู่อันดับในตำแหน่งที่ 1 ถึง i-1 ในลิสต์ L จะสามารถครอบคลุมคู่อันดับ o บนเซ็ต O ได้เพราะว่า คู่อันดับ o เป็นสมาชิกเซ็ต O ตามข้อกำหนดที่ได้ไว้แต่แรก และคู่อันดับในตำแหน่งที่ 1 ถึง i-1 ในลิสต์L เป็นสมาชิกของเซ็ต O เพราะสมาชิกของลิสต์L ก็คือสมาชิกของเซ็ต O ที่นำมาเรียงลำดับใหม่เท่านั้น และคู่อันดับในตำแหน่งที่1 ถึง i-1 ในลิสต์ L อยู่ใกล้คู่อันดับ o มากกว่า $nearestDissimilar(O,o)$ เพราะ $nearestDissimilar(O,o)$ คือคู่ลำดับตำแหน่งที่ i ในลิสต์L ดังนั้นจึงสามารถกล่าวได้ว่า "คู่อันดับในตำแหน่งที่1 ถึง i-1 ในลิสต์ L ครอบคลุมคู่อันดับ o " ซึ่งสามารถแสดงได้ดังรูปที่ 4.7



รูปที่ 4.7 แสดงคู่อันดับทั้งหมดในลิสต์ L ที่ครอบคลุมคู่อันดับ o

คู่อันดับที่อยู่ในตำแหน่ง 1 ถึง i-1 ในลิสต์ L จะมีชนิดเหมือนกันกับคู่อันดับ o เสมอเพราะว่าคู่อันดับที่อยู่ในตำแหน่ง 1 ถึง i-1 ในลิสต์ L นั้นอยู่ในลำดับที่ต่ำกว่าคู่อันดับต่างชนิดที่ใกล้คู่อันดับ o ที่สุดในเซ็ต O (ซึ่งอยู่ในลำดับที่ i) แสดงว่าคู่อันดับที่อยู่ในตำแหน่ง 1 ถึง i-1 ในลิสต์ L อยู่ใกล้คู่อันดับ o มากกว่าคู่อันดับต่างชนิดที่ใกล้คู่อันดับ o ที่สุดในเซ็ต O ดังนั้นคู่อันดับที่อยู่ในตำแหน่ง 1 ถึง i-1 ในลิสต์ L จะมีชนิดเหมือนกันกับคู่อันดับ o เสมอนั่นเอง ส่วนคู่อันดับที่อยู่ในตำแหน่ง i+1 ถึง n ในลิสต์ L อาจจะมีชนิดเหมือนหรือต่างกับคู่อันดับ o ก็ได้ สามารถแสดงให้เห็นชัดเจนมากขึ้นดังรูปที่ 4.8



รูปที่ 4.8 แสดงให้เห็นว่าคู่อันดับที่ครอบคลุมคู่อันดับ o จะมีชนิดเหมือนกับชนิดของคู่อันดับ o เสมอ

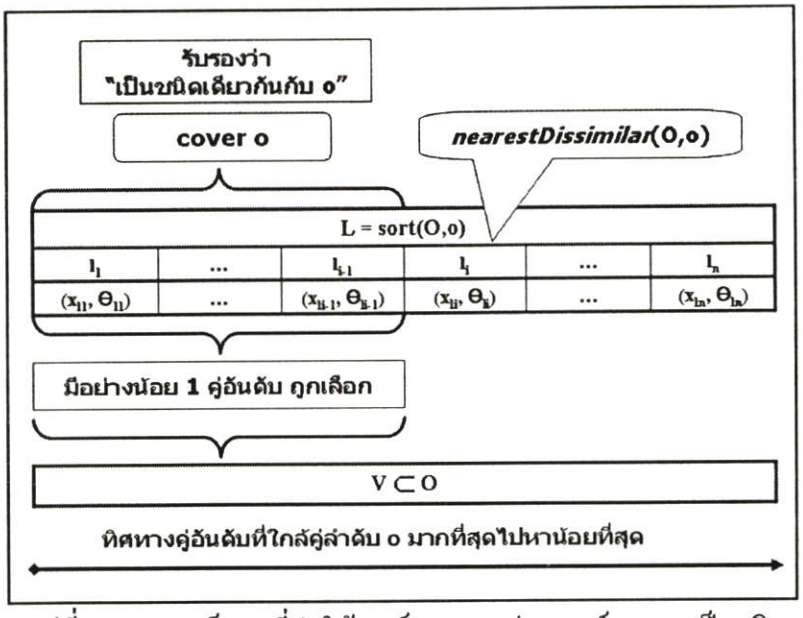
การพิสูจน์ว่าประพจน์ที่ (4.17) นั้นสมเหตุสมผลก็คือการแสดงให้เห็นว่า ในกรณีที่พจน์แรกของประพจน์ (4.17) เป็นจริงแล้วพจน์หลังของประพจน์จะเป็นจริงเสมอ ซึ่งพจน์แรกของประพจน์กล่าวว่า "มีคู่ลำดับที่ครอบคลุมคู่ลำดับ o บนเซ็ต O และคู่ลำดับนั้นเป็นสมาชิกของเซ็ต V และเซ็ต V เป็นซับเซ็ตของเซ็ต O"

$$\exists v \bullet (v \text{ coverOn}(O)\ o) \wedge (v \in V) \wedge (V \subset O)$$

ซึ่งเมื่อพิจารณาแล้วกรณีที่คำกล่าวนี้จะเป็นจริง ก็ต่อเมื่อเซ็ต V ต้องเป็นไปตาม 2 เงื่อนไขนี้

1. เซ็ต V เป็นซับเซ็ตของเซ็ต O
2. ภายในเซ็ต V ต้องมีคู่ลำดับสมาชิกอย่างน้อยหนึ่งตัวที่ครอบคลุมคู่ลำดับ o บนเซ็ต O

ซึ่งเซ็ต V ที่เป็นไปตามทั้งสองเงื่อนไขสามารถแสดงได้ดังรูปที่ 4.9



รูปที่ 4.9 แสดงเซ็ต V ที่ทำให้พจน์แรกของประพจน์ (4.17) เป็นจริง

จากนั้นเรามาพิจารณาที่พจน์หลังของพจน์หลังของประพจน์ (4.17) ซึ่งกล่าวว่า "โปรโตไทป์อ้างอิงในเซ็ต V ที่อยู่ใกล้คู่อันดับ o มากที่สุด จะมีชนิดเดียวกันกับชนิด θ ซึ่งเป็นชนิดของคู่อันดับ o "

$(\theta = categoryOf(\text{nearestPrototype}(V, o)))$

จากรูปที่ 4.9 แสดงให้เห็นว่ากลุ่มของคู่ลำดับที่ครอบคลุมคู่ลำดับ o บนเซ็ต O จะอยู่ใกล้คู่ลำดับ o มากกว่าคู่ลำดับอื่นๆในเซ็ต O ดังนั้นภายใต้เงื่อนไขที่เซ็ต V (ที่เป็นซับเซ็ตของเซ็ต O) มีคู่ลำดับที่ครอบคลุมคู่ลำดับ o บนเซ็ต O เป็นสมาชิกแล้ว คู่ลำดับในเซ็ต V ที่อยู่ใกล้คู่ลำดับ o มากที่สุดย่อมเป็นหนึ่งในกลุ่มคู่ลำดับที่ครอบคลุมคู่ลำดับ o บนเซ็ต O และจากที่กล่าวไปแล้วว่ากลุ่มคู่ลำดับที่ครอบคลุมคู่ลำดับ o บนเซ็ต O จะมีชนิดเดียวกันกับคู่ลำดับ o เสมอ ทำให้สามารถสรุปได้ว่าภายใต้เงื่อนไขที่เซ็ต V (ที่เป็นซับเซ็ตของเซ็ต O) มีคู่ลำดับที่ครอบคลุมคู่ลำดับ o บนเซ็ต O เป็นสมาชิกแล้ว คู่ลำดับในเซ็ต V ที่อยู่ใกล้คู่ลำดับ o มากที่สุดจะมีชนิดเดียวกันกับคู่ลำดับ o เสมอ ซึ่งจะเห็นได้ว่า ในกรณีที่พจน์แรกของประพจน์ (4.17) เป็นจริง แล้วพจน์หลังของประพจน์ (4.17) จะเป็นจริงเสมอ

ดังนั้นจึงสามารถกล่าวได้ว่า ประพจน์ที่ (4.17) มีความสมเหตุสมผล

หรือก็คือ นิยามคุณสมบัติของความครอบคลุมสมเหตุสมผล

4.5.2 แนวคิดในการเลือกโปรโตไทป์โดยใช้คุณสมบัติความครอบคลุม

ในหัวข้อที่แล้วได้นำเสนอแนวคิดความครอบคุมและคุณสมบัติที่เกิดขึ้นเพราะความครอบคลุม ในหัวข้อนี้เป็นการนำคุณสมบัติของความครอบคลุมมาใช้ในการสร้างเป็นโมเดลเชิงประมาณของปัญหาการหาซับเซ็ตสอดคล้องที่มีขนาดเล็กที่สุด และนำโมเดลที่ได้มาช่วยในการเลือกโปรโตไทป์อ้างอิงจากเซ็ตข้อมูลตั้งต้น

จากที่ได้ทราบแล้วว่าการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของชุดข้อมูลตั้งต้นนั้นเทียบได้กับการแก้ปัญหาซับเซ็ตสอดคล้องที่มีขนาดเล็กที่สุดเชิงประมาณที่ไม่ต้องการใช้การคำนวนไม่มากจนเกินไป เซ็ตอ้างอิงที่หาได้ไม่จำเป็นต้องเป็นซับเซ็ตที่มีขนาดเล็กที่สุดแท้จริงของเซ็ตของมูลตั้งต้น เพียงแต่ให้ได้ซับเซ็ตที่มีขนาดเล็กมากที่สุดเท่าที่จะทำได้ จากที่ได้กล่าวมาแล้วว่าปัญหาการหาซับเซ็ตซับเซ็ตสอดคล้องที่เล็กที่สุดสามารถแสดงให้อยู่ในรูปแบบปัญหาการหาค่าเหมาะสมที่สุดแบบมีเงื่อนไขบังคับ (constrained optimization problem) ได้ดังนี้ (จากสูตรที่ 4.14)

Minimize	: $\|V\|$	#(Objective function)
Subject to	: V *consistentSubsetOf* O.	#(Constraint)

ซึ่งในส่วนของเงื่อนไขบังคับ (Constraint) นั้นสามารถแยกได้ตามนิยามซับเซ็ตสอดคล้อง (4.12) ได้เป็น 2 เงื่อนไข ซึ่งสามารถเขียนปัญหาใหม่ได้ดังนี้

Minimize	: $\|V\|$	#(Objective function)	
Subject to	: $V \subset O$.	#(Constraint 1)	(4.20)
	V *consistentTo* O.	#(Constraint 2)	

ซึ่งในส่วนของเงื่อนไขบังคับที่ 2 (Constraint 2) นั้นคือ เซ็ต V ต้องสอดคล้องกับเซ็ต O ซึ่งจากนิยามความสอดคล้อง (4.10) ที่บอกว่า "เซ็ต V สอดคล้องกับเซ็ต O" ก็ต่อเมื่อ "กฎ nearest neighbor rule ที่ใช้เซ็ต V เป็นเซ็ตโปรโตไทป์อ้างอิง สามารถแยกแยะออบเจ็คที่อยู่ภายในเซ็ต O ได้ถูกต้องทุกตัว" เงื่อนไขบังคับที่ 2 จึงสามารถแยกเป็นเงื่อนไขได้จำนวน n เงื่อนไข (ในที่นี้ n ขนาดของเซ็ต $O = \{o_1, o_2, ..., o_n\}$) ซึ่งเราสามารถเขียนปัญหาใหม่ได้ดังนี้

Minimize	: $\|V\|$	#(Objective function)	
Subject to	: $V \subset O.$	#(Constraint 1)	
	$NN(V,o_1)$ is TRUE	#(Constraint 2)	(4.21)
	$NN(V,o_2)$ is TRUE	#(Constraint 3)	
	.....		
	$NN(V,o_n)$ is TRUE	#(Constraint n+1)	

จากนั้นเราจะทำการเปลี่ยนปัญหาการหาซับเซ็ตซับเซ็ตสอดคล้องที่เล็กที่สุดเป็นปัญหาการหาซับเซ็ตซับเซ็ตสอดคล้องที่เล็กที่สุดเชิงประมาณ โดยนำเอาคุณสมบัติของความครอบคลุมมาในการประมาณ โดยคุณสมบัติของความครอบคลุม (4.18) กล่าวว่า

"หากมีคู่อันดับ v ซึ่งคู่อันดับ v ครอบคลุมคู่อันดับ o บนเซ็ต O และคู่อันดับ v เป็นสมาชิกของเซ็ต V และเซ็ต V เป็นซับเซ็ตของเซ็ต O" แล้ว "กฎ nearest neighbor rule ที่ใช้เซ็ต V เป็นเซ็ตโปรโตไทป์อ้างอิงจะแยกแยะคู่อันดับ o ได้อย่างถูกต้อง"

$\exists v \bullet (v \text{ coverOn}(O)\ o) \wedge (v \in V) \wedge (V \subset O) \rightarrow (\text{NN}(V,o) \text{ is TRUE})$

เพราะว่าประพจน์นี้มีความสมเหตุสมผล ดังนั้นเมื่อคำกล่าวในพจน์แรกเป็นจริงแล้วคำกล่าวในพจน์หลังจะเป็นจริงเสมอ ดังนั้นเราจึงสามารถแทนเงื่อนไขที่ 2 ถึง n+1 ด้วยพจน์แรกของประพจนี้ เราจึงสารถเขียนปัญหาใหม่ได้ดังนี้

Minimize	: $\|V\|$	#(Objective function)	
Subject to	: $V \subset O.$	#(Constraint 1)	
	$\exists v \bullet (v \text{ coverOn}(O)\ o_1) \wedge (v \in V)$	#(Constraint 2)	(4.22)
	$\exists v \bullet (v \text{ coverOn}(O)\ o_2) \wedge (v \in V)$	#(Constraint 3)	
	.....		
	$\exists v \bullet (v \text{ coverOn}(O)\ o_n) \wedge (v \in V)$	#(Constraint n+1)	

ในปัญหา (4.22) จะสังเกตเห็นได้ว่าพจน์ $(V \subset O)$ ซึ่งควรจะมีอยู่เงื่อนไขที่ 2 ถึง n+1 สามารถตัดทิ้งได้ ทั้งนี้ก็เพราะมีเงื่อนไขที่ 1 ซึ่งกำหนดให้ $(V \subset O)$ อยู่แล้วพจน์อื่นๆที่มีความหมายบังคับเหมือนกันจึงซ้ำซ้อนและสามารถละทิ้งได้

ในฟังชันก์เป้าหมาย (objective function) ของปัญหา (4.22) คือ "ให้ได้เซ็ต V ที่มีขนาดเล็กที่สุด" และในเงื่อนไขบังคับที่ 1 (constraint 1) ของปัญหา (4.22) คือ "เซ็ต V เป็นซับเซ็ต O" ซึ่งนั่นหมายความว่าสมาชิกเซ็ต V ทุกตัวต้องเป็นสมาชิกเซ็ต O เสมอ ดังนั้นจึงสามารถอธิบายเป้าหมายและเงื่อนไขบังคับที่ 1 ว่าเป็นการเลือกสมาชิกของเซ็ต V จากสมาชิกในเซ็ต O ให้ได้เซ็ต V ที่มีขนาดเล็กที่สุด (การเลือกในที่นี้เป็นการสำเนาจากเซ็ต O ไปเซ็ต V ดังนั้นสมาชิกในเซ็ต O ไม่มีการเปลี่ยนแปลง) สามารถอธิบายปัญหา (4.22) ใหม่อีกครั้งเพื่อให้เข้าใจได้ง่ายยิ่งขึ้น ได้ดังตารางที่ 4.1

ตารางที่ 4.1 แสดงปัญหาการเลือกซับเซ็ตสอดคล้องที่มีขนาดเล็กที่สุดเชิงประมาณ

เป้าหมาย	"เลือกคู่อันดับของเซ็ต V จากคู่อันดับในเซ็ต O ให้ได้เซ็ต V ที่มีขนาดเล็กที่สุด"
เงื่อนไข	1. ในเซ็ต V ต้องมีคู่อันดับที่ครอบคลุมคู่ลำดับ o_1 บนเซ็ต O อย่างน้อย 1 ตัว 2. ในเซ็ต V ต้องมีคู่อันดับที่ครอบคลุมคู่ลำดับ o_2 บนเซ็ต O อย่างน้อย 1 ตัว ... n. ในเซ็ต V ต้องมีคู่อันดับที่ครอบคลุมคู่ลำดับ o_n บนเซ็ต O อย่างน้อย 1 ตัว

จากตารางที่ 4.1 จะเห็นได้ว่าปัญหานี้มีเงื่อนไขเหลือ n เงื่อนไขเพราะเงื่อนไขที่เซ็ต V ต้องเป็นซับเซ็ตของเซ็ต O ถูกตั้งเป็นเป้าหมายแล้ว จากนั้นพิจารณาปัญหาในตารางที่ 4.1 ว่ามีความหมายอย่างไรและต้องการคำตอบอะไร โดยเริ่มพิจารณาที่เป้าหมายของปัญหาซึ่งสามารถอธิบายความต้องการของปัญหาได้ว่า "ต้องการเลือกสำเนาคู่อันดับในเซ็ต O มาเก็บไว้ที่เซ็ต V ให้น้อยที่สุดเท่าที่จะทำได้โดยที่เซ็ต V ต้องเป็นไปตามเงื่อนไขที่มีอยู่ทั้งหมด" จากนั้นมาพิจารณาที่เงื่อนไขของปัญหาที่มีทั้งหมด n เงื่อนไขซึ่งมีรูปแบบเดียวกัน ซึ่งเราสามารถอธิบายเงื่อนไขทั้งหมดรวมกันได้ว่า "เซ็ต V ต้องมีคู่อันดับอย่างน้อยหนึ่งตัวที่ครอบคลุมคู่อันดับแต่ละตัวในเซ็ต O" นั่นหมายความว่า "สมาชิกแต่ละตัวในเซ็ต O ต้องถูกสมาชิกในเซ็ต V อย่างน้อยหนึ่งตัวครอบคลุมอยู่" ซึ่งเราสามารถพูดให้สั้นได้ว่า "เซ็ต V ต้องครอบคลุมสมาชิกในเซ็ต O ครบทั้งหมด" ดังนั้นเราสามารถอธิบายความหมายของปัญหาได้ดังนี้

> "ต้องการเลือกสำเนาสมาชิกในเซ็ต O มาเก็บไว้ที่เซ็ต V ให้น้อยที่สุดเท่าที่จะทำได้ โดยที่เซ็ต V ต้องครอบคลุมสมาชิกในเซ็ต O ครบทั้งหมด"

จากความหมายของปัญหาด้านบน หากว่าต้องการให้ได้เซ็ต V ที่มีขนาดเล็กตามเป้าหมายโดยที่เซ็ต V ครอบคลุมสมาชิกในเซ็ต O ครบทั้งหมดตามเงื่อนไข ก็ควรจะเลือกคู่อันดับที่มีความสามารถครอบคลุมคู่ลำดับอื่นได้จำนวนมากไปเป็นสมาชิกในเซ็ต V ซึ่งเราสามารถนิยามค่าความสามารถในการครอบคลุมได้ดังนี้

<u>นิยาม</u> ค่าความสามารถในการครอบคลุม

ค่าความสามารถในการครอบคลุมของคู่อันดับ o บนเซ็ต O คือ จำนวนคู่อันดับที่คู่อันดับ o ครอบคลุมได้บนเซ็ต O

เราสามารถยกตัวอย่างการหาค่าความสามารถในการครอบคลุมได้ดังนี้

กำหนดให้เซ็ต O มีสมาชิกทั้งหมด 7 ตัว ดังนี้

$O = \{ (x_1, A),\ (x_2, A),\ (x_3, A),\ (x_4, A),\ (x_5, B),\ (x_6, B),\ (x_7, B) \}$

ในที่นี้กำหนดให้ (ในความเป็นจริงค่าเหล่านี้ได้จากการคำนวนตามนิยามความครอบคลุม(4.16))

คู่ลำดับ $(x_1, A), (x_3, A)$ ครอบคลุมคู่ลำดับ (x_1, A) บนเซ็ต O

คู่ลำดับ $(x_1, A), (x_2, A), (x_3, A)$ ครอบคลุมคู่ลำดับ (x_2, A) บนเซ็ต O

คู่ลำดับ (x_3, A) ครอบคลุมคู่ลำดับ (x_3, A) บนเซ็ต O

คู่ลำดับ (x_4, A) ครอบคลุมคู่ลำดับ (x_4, A) บนเซ็ต O

คู่ลำดับ (x_5, B) ครอบคลุมคู่ลำดับ (x_5, B) บนเซ็ต O

คู่ลำดับ $(x_6, B), (x_7, B)$ ครอบคลุมคู่ลำดับ (x_6, B) บนเซ็ต O

คู่ลำดับ $(x_5, B), (x_7, B)$ ครอบคลุมคู่ลำดับ (x_7, B) บนเซ็ต O

จากนิยามค่าความสามารถในการครอบคลุมเราสามารถคำนวนค่าความสามารถในการครอบคลุมของแต่ละคู่อันดับบนเซ็ต O ได้ดังนี้

- ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_1, A) บนเซ็ต O เท่ากับ 2 เพราะคู่อันดับ (x_1, A) ครอบคลุมคู่อันดับ $(x_1, A), (x_2, A)$ บนเซ็ต O
- ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_2, A) บนเซ็ต O เท่ากับ 1 เพราะคู่อันดับ (x_2, A) ครอบคลุมคู่อันดับ (x_2, A) บนเซ็ต O
- ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_3, A) บนเซ็ต O เท่ากับ 3 เพราะคู่อันดับ (x_3, A) ครอบคลุมคู่อันดับ $(x_1, A), (x_2, A), (x_3, A)$ บนเซ็ต O
- ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_4, A) บนเซ็ต O เท่ากับ 1 เพราะคู่อันดับ (x_4, A) ครอบคลุมคู่อันดับ (x_4, A) บนเซ็ต O
- ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_5, B) บนเซ็ต O เท่ากับ 2 เพราะคู่อันดับ (x_5, B) ครอบคลุมคู่อันดับ $(x_5, B), (x_7, B)$ บนเซ็ต O
- ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_6, B) บนเซ็ต O เท่ากับ 1 เพราะคู่อันดับ (x_6, B) ครอบคลุมคู่อันดับ (x_6, B) บนเซ็ต O
- ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_7, B) บนเซ็ต O เท่ากับ 2 เพราะคู่อันดับ (x_7, B) ครอบคลุมคู่อันดับ (x_7, B) บนเซ็ต O

4.5.2.1 หลักการพิจารณาเลือกโปรโตไทป์

ในส่วนนี้นำเสนอหลักการในการเลือกโปรโตไทป์อ้างอิง (สมาชิกเซ็ต V) จากชุดข้อมูลตั้งต้น (เซ็ต O) โดยการนิยามค่าความสามารถในการครอบคลุมซึ่งคำนวนได้โดยใช้โมเดลเชิงประมาณของปัญหาซับเซ็ตสอดคล้องที่เราได้นำเสนอไปแล้ว มาช่วยในการพิจารณาว่าควรเลือกสมาชิกตัวใดในชุดข้อมูลตั้งต้นไปเป็นโปรโตไทป์อ้างอิง (สมาชิกเซ็ต V) โดยนำเสนอหลักการในการเลือกโปรโตไทป์ ได้ดังขั้นตอนต่อไปนี้

1. พิจารณาเลือกคู่อันดับที่มีค่าความสามารถในการครอบคลุมบนเซ็ต O มากที่สุดไปเป็นโปรโตไทป์อ้างอิง
2. คู่อันดับที่ถูกครอบคลุมโดยโปรโตไทป์อ้างอิงที่ได้เลือกไว้แล้วจะไม่ถูกนำไปคิดค่าความสามารถในการครอบคลุมบนเซ็ต O อีก
3. หากโปรโตไทป์อ้างอิงที่ได้เลือกไว้ยังไม่ครอบคลุมคู่ลำดับที่อยู่ในเซ็ต O ทั้งหมด ให้กลับไปพิจารณาเลือกโปรโตไทป์เพิ่มในขั้นตอนที่ 1 ใหม่อีกครั้ง

สามารถยกตัวอย่างการพิจารณาเลือกโปรโตไทป์โดยใช้ตัวอย่างเดียวกันกับตัวอย่างที่แสดงในหัวข้อนิยามค่าความสามารถในการครอบคลุมเพื่อให้ไม่ต้องอธิบายข้อกำหนดต่างๆซ้ำอีกครั้ง ซึ่งสามารถอธิบายการเลือกโปรโตไทป์ ตามขั้นตอนได้ดังนี้

1. พิจารณาเลือกคู่อันดับ (x_3, A) ไปเก็บเป็นโปรโตไทป์อ้างอิงในเซ็ต V เพราะค่าความสามารถในการครอบคลุมของคู่อันดับ (x_3, A) บนเซ็ต O มีค่าเท่ากับ 3 ซึ่งเป็นค่าที่มากที่สุด (การคำนวนค่าความสามารถในการครอบคลุมบนเซ็ต O แสดงไปแล้วในตัวอย่างของหัวข้อนิยามค่าความสามารถในการครอบคลุม)
2. คู่อันดับ (x_3, A) ครอบคลุมคู่อันดับ $(x_1, A), (x_2, A), (x_3, A)$ บนเซ็ต O ดังนั้นคู่อันดับ $(x_1, A), (x_2, A), (x_3, A)$ จะไม่ถูกนำไปคำนวนเป็นค่าความสามารถในการครอบคลุมบนเซ็ต O อีก ดังนั้นการคำนวนค่าความสามารถในการครอบคลุมบนเซ็ต O ในครั้งต่อไปจะคำนวนจากคำกล่าวด้านล่างนี้เท่านั้น

   คู่ลำดับ (x_4, A) ครอบคลุมคู่ลำดับ (x_4, A) บนเซ็ต O

   คู่ลำดับ (x_5, B) ครอบคลุมคู่ลำดับ (x_5, B) บนเซ็ต O

   คู่ลำดับ $(x_6, B), (x_7, B)$ ครอบคลุมคู่ลำดับ (x_6, B) บนเซ็ต O

   คู่ลำดับ $(x_5, B), (x_7, B)$ ครอบคลุมคู่ลำดับ (x_7, B) บนเซ็ต O
3. โปรโตไทป์อ้างอิงที่มีอยู่ไม่ครอบคลุมคู่ลำดับ (x_4, A), (x_5, B), (x_6, B), (x_7, B) กลับไปพิจารณาเลือกโปรโตไทป์เพิ่มในขั้นตอนที่ 1 ใหม่อีกครั้ง

1. พิจารณาเลือกโปรโตไทป์ใหม่อีกครั้ง โดยทำการคำนวนค่าความสามารถในการครอบคลุมของแต่ละคู่อันดับบนเซ็ต O ใหม่ (โดยไม่นับคู่อันดับที่ถูกครอบคลุมไปแล้ว) ได้ดังนี้

   - ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_1, A) บนเซ็ต O เท่ากับ 0

- ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_2, A) บนเซ็ต O เท่ากับ 0
- ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_3, A) บนเซ็ต O เท่ากับ 0
- ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_4, A) บนเซ็ต O เท่ากับ 1
- ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_5, B) บนเซ็ต O เท่ากับ 2
- ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_6, B) บนเซ็ต O เท่ากับ 1
- ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_7, B) บนเซ็ต O เท่ากับ 2

ดังนั้นจึงพิจารณาเลือกคู่อันดับ (x_5, B) ไปเก็บเป็นโปรโตไทป์อ้างอิงในเซ็ต V เพราะค่าความสามารถในการครอบคลุมของคู่อันดับ (x_5, B) บนเซ็ต O มีค่าเท่ากับ 2 ซึ่งเป็นค่าที่มากที่สุด (อาจจะเลือกคู่อันดับ (x_7, B) ก็ได้เพราะมีค่าความสามารถในการครอบคลุมบนเซ็ต O เท่ากัน)

2. คู่อันดับ (x_5, B) ครอบคลุมคู่ลำดับ $(x_5, B), (x_7, B)$ บนเซ็ต O ดังนั้นคู่อันดับ $(x_5, B), (x_7, B)$ จะไม่ถูกนำไปคำนวนเป็นค่าความสามารถในการครอบคลุมบนเซ็ต O อีก ดังนั้นการคำนวนค่าความสามารถในการครอบคลุมบนเซ็ต O ในครั้งต่อไปจะคำนวนจากคำกล่าวด้านล่างนี้เท่านั้น

   คู่ลำดับ (x_4, A) ครอบคลุมคู่ลำดับ (x_4, A) บนเซ็ต O

   คู่ลำดับ $(x_6, B), (x_7, B)$ ครอบคลุมคู่ลำดับ (x_6, B) บนเซ็ต O

3. โปรโตไทป์อ้างอิงที่มีอยู่ไม่ครอบคลุมคู่ลำดับ (x_4, A), (x_6, B) กลับไปพิจารณาเลือกโปรโตไทป์เพิ่มในขั้นตอนที่ 1 ใหม่อีกครั้ง

1. พิจารณาเลือกโปรโตไทป์ใหม่อีกครั้ง โดยทำการคำนวนค่าความสามารถในการครอบคลุมของแต่ละคู่อันดับบนเซ็ต O ใหม่ (โดยไม่นับคู่อันดับที่ถูกครอบคลุมไปแล้ว) ได้ดังนี้
   - ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_1, A) บนเซ็ต O เท่ากับ 0
   - ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_2, A) บนเซ็ต O เท่ากับ 0
   - ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_3, A) บนเซ็ต O เท่ากับ 0
   - ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_4, A) บนเซ็ต O เท่ากับ 1
   - ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_5, B) บนเซ็ต O เท่ากับ 0
   - ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_6, B) บนเซ็ต O เท่ากับ 1
   - ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_7, B) บนเซ็ต O เท่ากับ 1

   ดังนั้นจึงพิจารณาเลือกคู่อันดับ (x_4, A) ไปเก็บเป็นโปรโตไทป์อ้างอิงในเซ็ต V เพราะค่าความสามารถในการครอบคลุมของคู่อันดับ (x_4, A) บนเซ็ต O มีค่าเท่ากับ 2 ซึ่งเป็นค่าที่มากที่สุด (อาจจะเลือกคู่อันดับ (x_6, B) หรือ (x_7, B) ก็ได้เพราะมีค่าความสามารถในการครอบคลุมบนเซ็ต O เท่ากัน)

2. คู่อันดับ (x_4, A) ครอบคลุมคู่ลำดับ (x_4, A) บนเซ็ต O ดังนั้นคู่อันดับ (x_4, A) จะไม่ถูกนำไปคำนวนเป็นค่าความสามารถในการครอบคลุมบนเซ็ต O อีก ดังนั้นการคำนวนค่าความสามารถในการครอบคลุมบนเซ็ต O ในครั้งต่อไปจะคำนวนจากคำกล่าวด้านล่างนี้เท่านั้น

คู่ลำดับ $(x_6, B), (x_7, B)$ ครอบคลุมคู่ลำดับ (x_6, B) บนเซ็ต O

3. โปรโตไทป์อ้างอิงที่มีอยู่ไม่ครอบคลุมคู่ลำดับ (x_6, B) กลับไปพิจารณาเลือกโปรโตไทป์เพิ่มในขั้นตอนที่ 1 ใหม่อีกครั้ง

1. พิจารณาเลือกโปรโตไทป์ใหม่อีกครั้ง โดยทำการคำนวนค่าความสามารถในการครอบคลุมของแต่ละคู่อันดับบนเซ็ต O ใหม่ (โดยไม่นับคู่อันดับที่ถูกครอบคลุมไปแล้ว) ได้ดังนี้
   - ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_1, A) บนเซ็ต O เท่ากับ 0
   - ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_2, A) บนเซ็ต O เท่ากับ 0
   - ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_3, A) บนเซ็ต O เท่ากับ 0
   - ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_4, A) บนเซ็ต O เท่ากับ 0
   - ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_5, B) บนเซ็ต O เท่ากับ 0
   - ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_6, B) บนเซ็ต O เท่ากับ 1
   - ค่าความสามารถในการครอบคลุมของคู่อันดับ (x_7, B) บนเซ็ต O เท่ากับ 1

   ดังนั้นจึงพิจารณาเลือกคู่อันดับ (x_6, A) ไปเก็บเป็นโปรโตไทป์อ้างอิงในเซ็ต V เพราะค่าความสามารถในการครอบคลุมของคู่อันดับ (x_6, A) บนเซ็ต O มีค่าเท่ากับ 2 ซึ่งเป็นค่าที่มากที่สุด (อาจจะเลือกคู่อันดับ (x_7, B) ก็ได้เพราะมีค่าความสามารถในการครอบคลุมบนเซ็ต O เท่ากัน)
2. คู่อันดับ (x_6, A) ครอบคลุมคู่ลำดับ (x_6, A) บนเซ็ต O ดังนั้นคู่อันดับ (x_6, A) จะไม่ถูกนำไปคำนวนเป็นค่าความสามารถในการครอบคลุมบนเซ็ต O อีก
3. โปรโตไทป์อ้างอิงได้เลือกไว้ครอบคลุมคู่ลำดับที่อยู่ในเซ็ต O ทั้งหมดแล้ว จบการทำงาน

หลังจบการทำงานเซ็ตโปรโตไทป์อ้างอิง V จะมีสมาชิกเป็นคู่ลำดับ 4 คู่อันดับ นั่นคือ $(x_3, A), (x_5, B), (x_4, A), (x_6, A)$ ซึ่งครอบคลุมสมาชิกในเซ็ตข้อมูลตั้งต้น O ครบทุกตัว ดังนั้นจึงรับประกันได้ว่ากฎ nearest neighbor rule ที่ใช้เซ็ต V เป็นเซ็ตโปรโตไทป์อ้างอิงสามารถแยกแยะสมาชิกในเซ็ตข้อมูลตั้งต้น O ได้อย่างถูกต้องทั้งหมดตามนิยามคุณสมบัติของความครอบคลุมที่ได้กล่าวไปแล้ว

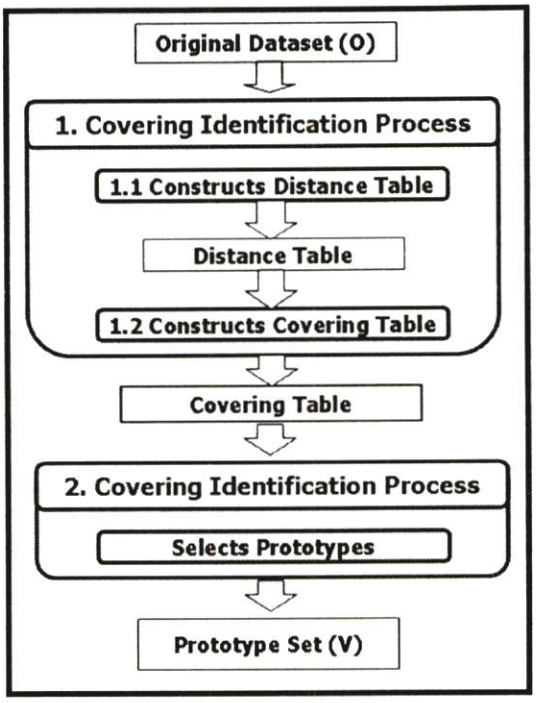
4.6 กระบวนการทำงานของวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน

ในส่วนนี้นำเสนอกระบวนการทำงานของการเลือกโปรโตไทป์โดยใช้วิธีการจัดกลุ่มข้อมูลแบบมีการสอน ซึ่งได้มาจากการนำแนวคิดต่างๆที่ได้นำเสนอไปก่อนหน้านี้มาสร้างเป็นระเบียบวิธีการให้สามารถทำงานได้จริงอย่างเป็นระบบ โดยกระบวนการทำงานสามารถแบ่งเป็นขั้นตอนหลักได้ดังนี้

1. ขั้นตอนการชี้เฉพาะคู่ลำดับที่ครอบคลุมแต่ละคู่อันดับบนเซ็ต O
   1.1 กระบวนการสร้างตารางระยะทาง
   1.2 กระบวนการสร้างตารางความครอบคลุม

2. ขั้นตอนการเลือกโปรโตไทป์ (ภายในมีกระบวนการเลือกโปรโตไทป์)

ซึ่งสามารถอธิบายลำดับการทำงานของวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนได้ดังรูปที่ 4.10 โดยเริ่มต้นเซ็ตข้อมูลตั้งต้นจะถูกส่งเข้าสู่ขั้นตอนการชี้เฉพาะคู่ลำดับที่ครอบคลุมแต่ละคู่อันดับบนเซ็ต O ซึ่งภายในประกอบด้วย 2 กระบวนการนั่นคือกระบวนการสร้างตารางระยะทางและกระบวนการสร้างตารางความครอบคลุม โดยกระบวนการสร้างตารางระยะทางจะนำคู่อันดับในเซ็ตข้อมูลตั้งต้นมาคำนวนระยะทางและเก็บค่าระยะทางที่ได้ทั้งหมดไว้ในตารางระยะทาง แล้วกระบวนการสร้างตารางความครอบคลุมจะนำตารางระยะทางที่ได้มาคำนวนความครอบคลุมบนเซ็ต O ของคู่ลำดับทั้งหมดและเก็บค่าที่ได้ไว้ในตารางความครอบคลุม หลังจากที่ได้ตารางความครอบคลุมเรียบร้อยแล้วก็เข้าสู่ขั้นตอนการเลือกโปรโตไทป์ซึ่งใช้หลักการเลือกโปรโตไทป์ที่ได้นำเสนอไปแล้วมาพิจารณาเลือกคู่อันดับในเซ็ตข้อมูลตั้งต้นไปเป็นโปรโตไทป์อ้างอิง

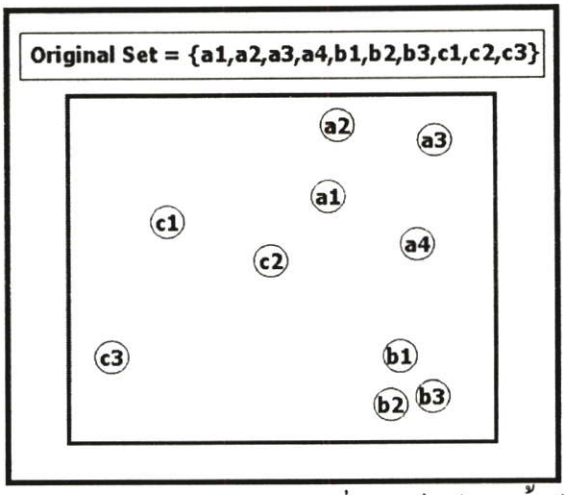


รูปที่ 4.10 แสดงขั้นตอนการทำงานหลักของวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน

4.6.1 กระบวนการสร้างตารางระยะทาง

กระบวนการสร้างตารางระยะทาง คือ กระบวนการคำนวนระยะทางระหว่าง 2 คู่อันดับที่เป็นไปได้ทั้งหมดในเซ็ตข้อมูลตั้งต้น แล้วนำค่าระยะทางที่ได้ทั้งหมดมาเก็บไว้ในตารางระยะทาง ซึ่งเมื่อเสร็จสิ้นกระบวนการนี้แล้วจะไม่จำเป็นต้องใช้ฟังก์ชันคำนวนระยะทางอีกต่อไป เพราะหากต้องการทราบระยะทางระหว่าง 2 คู่อันดับใดๆในเซ็ตข้อมูลตั้งต้นสามารถพิจารณาจากตารางระยะทางระยะทางได้ทันที

ตัวอย่างแสดงในรูปที่ 4.11 โดยกำหนดให้เซ็ตข้อมูลตั้งต้นมีสมาชิกทั้งหมด 11 คู่อันดับดังนี้ a1 = ((6.3,2.16),A), a2 = ((6.55,0.28),A), a3 = ((9.09,0.68),A), a4 = ((8.63,3.43),A), b1 = ((8.17,6.4),B), b2 = ((7.92,7.67),B), b3 = ((9.02,7.45),B), c1 = ((2.06,2.82),C), c2 = ((4.78,3.86),C), c3 = ((0.58,6.4),C) ซึ่งสามารถแสดงการวางตัวของคู่ลำดับทั้งหมดได้ดังรูปที่ 4.11 (ในที่นี้ค่าตำแหน่งนับจากด้านซ้ายบน) และเมื่อนำคู่ลำดับที่มีอยู่ทั้งหมดมาคำนวนหาระยะทางระหว่าง 2 คู่อันดับที่เป็นไปได้ทั้งหมดแล้วเก็บในตารางระยะทางซึ่งสามารถได้ดังตารางที่ 4.2



รูปที่ 4.11 แสดงการวางตัวของคู่อันดับที่มีในเซ็ตข้อมูลตั้งต้นทั้งหมด

ตารางที่ 4.2 แสดงตารางระยะทางของคู่อันดับที่มีในเซ็ตข้อมูลตั้งต้นทั้งหมด

	a1	a2	a3	a4	b1	b2	b3	c1	c2	c3
a1	0.00	1.90	3.16	2.65	4.63	5.74	5.95	4.29	2.28	7.12
a2	1.90	0.00	2.57	3.77	6.33	7.52	7.58	5.16	3.99	8.55
a3	3.16	2.57	0.00	2.79	5.79	7.09	6.77	7.35	5.36	10.3
a4	2.65	3.77	2.79	0.00	3.01	4.30	4.04	6.60	3.87	8.58
b1	4.63	6.33	5.79	3.01	0.00	1.29	1.35	7.08	4.24	7.59
b2	5.74	7.52	7.09	4.30	1.29	0.00	1.12	7.61	4.94	7.45
b3	5.95	7.58	6.77	4.04	1.35	1.12	0.00	8.36	5.56	8.51
c1	4.29	5.16	7.35	6.60	7.08	7.61	8.36	0.00	2.91	3.87
c2	2.28	3.99	5.36	3.87	4.24	4.94	5.56	2.91	0.00	4.91
c3	7.12	8.55	10.3	8.58	7.59	7.45	8.51	3.87	4.91	0.00

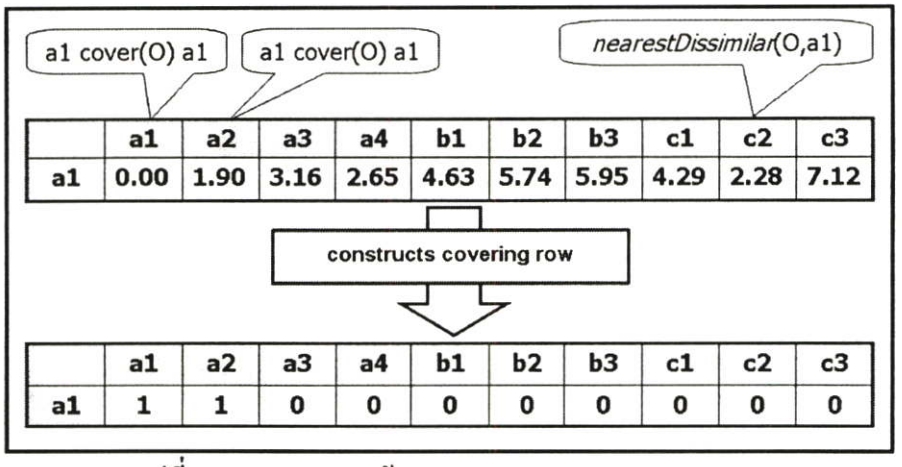
4.6.2 กระบวนการสร้างตารางความครอบคลุม

กระบวนการสร้างตารางความครอบคลุม คือ กระบวนการนำตารางระยะทางมาใช้พิจารณาว่าคู่ลำดับใดครอบคลุมคู่ลำดับใดบ้างบนเซ็ตข้อมูลตั้งต้น O จากนิยามความครอบคลุม (4.16) กล่าวว่า

"คู่อันดับ o' ครอบคลุม คู่อันดับ o'' บนเซ็ต O" ก็ต่อเมื่อ "คู่อันดับ o' เป็นสมาชิกเซ็ต O" และ "คู่อันดับ o'' เป็นสมาชิกเซ็ต O" และ "คู่อันดับ o' อยู่ใกล้คู่อันดับ o'' มากกว่าคู่อันดับต่างชนิดที่ใกล้คู่อันดับ o'' ที่สุดในเซ็ต O"

พิจารณาเฉพาะแถวแรกของตารางระยะทางซึ่งแสดงในตารางด้านบนของรูปที่ 4.12 จะเห็นได้ว่าคู่ลำดับ c2 มีชนิดต่างจากคู่ลำดับ a1 และคู่ลำดับ c2 อยู่ใกล้คู่ลำดับ a1 มากกว่าคู่ลำดับอื่นๆ ในเซ็ต V ที่มีชนิดต่างจากคู่ลำดับ a1 ดังนั้นคู่ลำดับ c2 จึงเป็นคู่ลำดับต่างชนิดที่อยู่ใกล้คู่ลำดับ a1 ที่สุดในเซ็ต V ตามนิยามคู่อันดับต่างชนิดที่ใกล้ที่สุด (4.15)

หลังจากที่ทราบว่าคู่ลำดับต่างชนิดที่อยู่ใกล้คู่ลำดับ a1 ที่สุดในเซ็ต O คือคู่ลำดับ c2 แล้ว พิจารณาหาคู่ลำดับในเซ็ต O ที่อยู่ใกล้คู่ลำดับ a1 มากกว่าคู่ลำดับ c2 จากแถวแรกของตารางความครอบคลุมแสดงให้เห็นว่าคู่ลำดับ a1 และ a2 อยู่ใกล้คู่ลำดับ a1 มากกว่าคู่ลำดับ c2 ดังนั้นคู่ลำดับ a1 ครอบคลุมคู่ลำดับ a1 บนเซ็ต O และคู่ลำดับ a2 ครอบคลุมคู่ลำดับ a1 บนเซ็ต O ตามนิยามความครอบคลุม (4.16) จากนั้นกำหนดค่าในตารางค่าความครอบคลุมในแถวที่ a1 และคอลัมน์ที่ a1 และ a2 ให้มีค่าเป็น "1" ส่วนตำแหน่งคอลัมน์อื่นในแถวที่ a1 กำหนดให้มีค่าเป็น "0" แสดงได้ดังตารางด้านล่างในรูปที่ 4.12



	a1	a2	a3	a4	b1	b2	b3	c1	c2	c3
a1	0.00	1.90	3.16	2.65	4.63	5.74	5.95	4.29	2.28	7.12

	a1	a2	a3	a4	b1	b2	b3	c1	c2	c3
a1	1	1	0	0	0	0	0	0	0	0

รูปที่ 4.12 แสดงการสร้างแถวของตารางความครอบคลุม

พิจารณาทุกแถวของตางรางระยะทางในลักษณะเดียวกับที่ได้กล่าวมาแล้ว จะสามารถสร้างตารางความครอบคลุมได้ดังตารางที่ 4.3

ตารางที่ 4.3 แสดงตารางความครอบคลุม

	a1	a2	a3	a4	b1	b2	b3	c1	c2	c3
a1	1	1	0	0	0	0	0	0	0	0
a2	1	1	1	1	0	0	0	0	0	0
a3	1	1	1	1	0	0	0	0	0	0
a4	1	0	1	1	0	0	0	0	0	0
b1	0	0	0	0	1	1	1	0	0	0
b2	0	0	0	0	1	1	1	0	0	0
b3	0	0	0	0	1	1	1	0	0	0
c1	0	0	0	0	0	0	0	1	1	1
c2	0	0	0	0	0	0	0	0	1	0
c3	0	0	0	0	0	0	0	1	1	1

4.6.3 กระบวนการเลือกโปรโตไทป์

กระบวนการเลือกโปรโตไทป์ คือ กระบวนการพิจารณาเลือกคู่อันดับที่อยู่ในเซ็ตข้อมูลตั้งต้น O ไปเป็นโปรโตไทป์อ้างอิงในเซ็ตโปรโทป์อ้างอิง V โดยอาศัยหลักการเลือกโปรโตไทป์ที่ได้กล่าวไปแล้วมาประยุกต์ใช้กับตารางค่าความครอบคลุม ซึ่งสามารถแสดงกระบวนการเลือกโปรโตไทปได้ดังนี้

คำนวนค่าความสามารถในการครอบคลุมของแต่ละคู่อันดับบนเซ็ต O ซึ่งมีค่าเท่ากับผลรวมตามแนวตั้งของแต่ละคอลัมน์ในตารางค่าความครอบคลุม สามารถแสดงค่าความสามารถของแต่ละคู่อันดับบนเซ็ต O ได้ดังแถวสุดท้ายของตารางภายในรูปที่ 4.13 จากนั้นพิจารณาเลือกคู่อันดับ a1 ไปเป็นโปรโตไทป์อ้างอิงในเซ็ต V เพราะค่าความสามารถในการครอบคลุมของคู่อันดับ a1 บนเซ็ต O มีค่าเท่ากับ 4 ซึ่งมากที่สุด

	a1	a2	a3	a4	b1	b2	b3	c1	c2	c3
a1	1	1	0	0	0	0	0	0	0	0
a2	1	1	1	1	0	0	0	0	0	0
a3	1	1	1	1	0	0	0	0	0	0
a4	1	0	1	1	0	0	0	0	0	0
b1	0	0	0	0	1	1	1	0	0	0
b2	0	0	0	0	1	1	1	0	0	0
b3	0	0	0	0	1	1	1	0	0	0
c1	0	0	0	0	0	0	0	1	1	1
c2	0	0	0	0	0	0	0	0	1	0
c3	0	0	0	0	0	0	0	1	1	1
+	4	3	3	3	3	3	3	2	3	2

เลือก a1 เป็นโปรโตไทป์

รูปที่ 4.13 แสดงตัวอย่างการพิจารณาเลือกคู่ลำดับ a1 เป็นโปรโตไทป์

คำนวนค่าความสามารถในการครอบคลุมของแต่ละคู่อันดับบนเซ็ต O ใหม่อีกครั้งโดยที่ไม่นับคู่อันดับที่เคยถูกโปรโตไทป์ครอบคลุมไปแล้ว ซึ่งมีค่าเท่ากับผลรวมตามแนวตั้งของแต่ละคอลัมน์ในตารางค่าความครอบคลุมที่ไม่นับแถวที่ a1,a2,a3,a4 ที่โปรโตไทป์ a1 ครอบคลุมไปแล้ว ซึ่งสามารถแสดงค่าความสามารถของแต่ละคู่อันดับบนเซ็ต O ได้ดังแถวสุดท้ายของตารางภายในรูปที่รูปที่ 4.14 จากนั้นพิจารณาเลือกคู่อันดับ b1 ไปเป็นโปรโตไทป์อ้างอิงในเซ็ต V เพราะค่าความสามารถในการครอบคลุมของคู่อันดับ b1 บนเซ็ต O มีค่าเท่ากับ 3 ซึ่งเป็นค่าที่มากที่สุด (หรืออาจเลือกคู่อันดับ b2,b3,c2 ก็ได้)

	a1	a2	a3	a4	b1	b2	b3	c1	c2	c3
a1	ถูกครอบคลุมแล้วโดยโปรโตไทป์ a1									
a2										
a3										
a4										
b1	0	0	0	0	1	1	1	0	0	0
b2	0	0	0	0	1	1	1	0	0	0
b3	0	0	0	0	1	1	1	0	0	0
c1	0	0	0	0	0	0	0	1	1	1
c2	0	0	0	0	0	0	0	0	1	0
c3	0	0	0	0	0	0	0	1	1	1
+	0	0	0	0	3	3	3	2	3	2

เลือก b1 เป็นโปรโตไทป์

รูปที่ 4.14 แสดงตัวอย่างการพิจารณาเลือกคู่ลำดับ b1 เป็นโปรโตไทป์

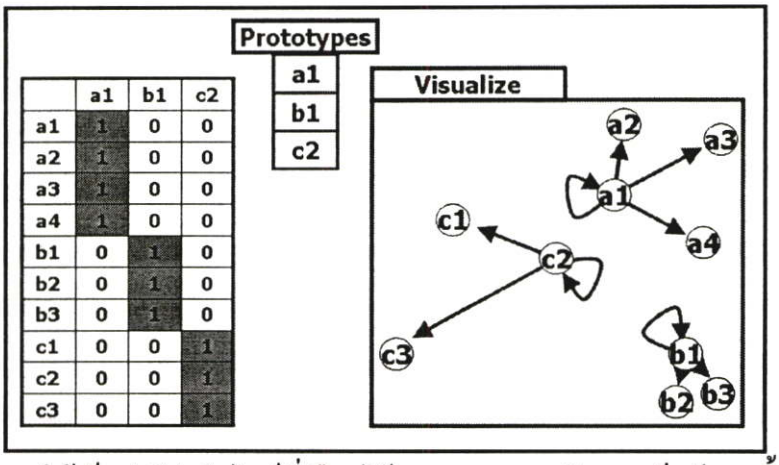
คำนวนค่าความสามารถในการครอบคลุมของแต่ละคู่อันดับบนเซ็ต O ใหม่อีกครั้งโดยที่ไม่นับคู่อันดับที่เคยถูกโปรโตไทป์ครอบคลุมไปแล้ว ซึ่งมีค่าเท่ากับผลรวมตามแนวตั้งของแต่ละคอลัมน์ในตารางค่าความครอบคลุมที่ไม่นับแถวที่ a1,a2,a3,a4,b1,b2,b3 ที่โปรโตไทป์ a1 และ b1 ครอบคลุมไปแล้ว ซึ่งสามารถแสดงค่าความสามารถของแต่ละคู่อันดับบนเซ็ต O ได้ดังแถวสุดท้ายของตารางภายในรูปที่รูปที่ 4.15 จากนั้นพิจารณาเลือกคู่อันดับ c2 ไปเป็นโปรโตไทป์อ้างอิงในเซ็ต V เพราะค่าความสามารถในการครอบคลุมของคู่อันดับ c2 บนเซ็ต O มีค่าเท่ากับ 3 ซึ่งเป็นค่าที่มากที่สุด

	a1	a2	a3	a4	b1	b2	b3	c1	c2	c3
a1	ถูกครอบคลุมแล้วโดยโปรโตไทป์ a1									
a2										
a3										
a4										
b1	ถูกครอบคลุมแล้วโดยโปรโตไทป์ b1									
b2										
b3										
c1	0	0	0	0	0	0	0	1	1	1
c2	0	0	0	0	0	0	0	0	1	0
c3	0	0	0	0	0	0	0	1	1	1
+	0	0	0	0	0	0	0	2	3	2

เลือก c2 เป็นโปรโตไทป์

รูปที่ 4.15 แสดงตัวอย่างการพิจารณาเลือกคู่ลำดับ c1 เป็นโปรโตไทป์

สิ้นสุดการทำงานเนื่องจากโปรโตไทป์ที่เลือกได้นั้นครอบคลุมคู่อันดับในเซ็ต O ทั้งหมดแล้ว ดังนั้นจึงรับได้ประกันว่า เมื่อกฎ nearest neighbor rule ใช้เซ็ต V เป็นเซ็ตโปรโตไทป์อ้างอิงแล้วจะสามารถแยกแยะสมาชิกในเซ็ตข้อมูลตั้งต้น O ได้อย่างถูกต้องทั้งหมด ซึ่งรูปที่ 4.16 แสดงให้เห็นว่าโปรโตไทป์ที่เลือกได้สามารถครอบคลุมสมาชิกของเซ็ตข้อมูลตั้งต้นทั้งหมด



	a1	b1	c2
a1	1	0	0
a2	1	0	0
a3	1	0	0
a4	1	0	0
b1	0	1	0
b2	0	1	0
b3	0	1	0
c1	0	0	1
c2	0	0	1
c3	0	0	1

รูปที่ 4.16 แสดงให้เห็นว่าโปรโตไทป์ที่เลือกได้ครอบคลุมสมาชิกของเซ็ตข้อมูลตั้งต้นทั้งหมด

บทที่ 5

ผลการทดลองและสรุปผลการทดลอง

ในบทนี้จะกล่าวถึงการทดลองเพื่อประเมินประสิทธิภาพของวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนซึ่งเป็นวิธีการที่วิทยานิพนธ์ฉบับนี้ได้นำเสนอ โดยในที่นี้ได้นำเสนอการทดลองเพื่อประเมินประสิทธิภาพของวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน 2 การทดลอง คือ การทดลองเลือกโปรโตไทป์และการทดลองแยกแยะข้อมูล

การทดลองเลือกโปรโตไทป์เป็นการทดลองเพื่อประเมินประสิทธิภาพในการลดจำนวนโปรโตไทป์อ้างอิงของวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน โดยการเปรียบเทียบจำนวนของโปรโตไทป์ผลลัพท์ที่เลือกได้ด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนกับจำนวนของโปรโตไทป์ผลลัพท์ที่เลือกได้ด้วยวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์แบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (สามารถศึกษารายละเอียดของวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์แบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้นในบทที่ 3) ซึ่งการทดลองและผลการทดลองเลือกโปรโตไทป์ได้นำเสนอในหัวข้อ 5.1

การทดลองแยกแยะข้อมูลเป็นการทดลองเพื่อประเมินประสิทธิภาพในการแยกแยะข้อมูลของกฎ nearest neighbor rule ที่ใช้โปรโตไทป์ที่เลือกได้ด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนเป็นโปรโตไทป์อ้างอิง โดยเปรียบเทียบกับประสิทธิภาพในการแยกแยะข้อมูลของกฎ nearest neighbor rule ที่ใช้ข้อมูลสำหรับใช้สอนทั้งหมดเป็นโปรโตไทป์อ้างอิงและเปรียบเทียบกับประสิทธิภาพในการแยกแยะข้อมูลของกฎ nearest neighbor rule ที่ใช้โปรโตไทป์ที่ได้มาจากการคำนวนด้วยวิธีการ learning vector quantization (LVQ) (สามารถศึกษารายละเอียดรายละเอียดของวิธีการ learning vector quantization (LVQ) เพิ่มเติมได้ในบทที่ 3) ซึ่งการทดลองและผลการทดลองแยกแยะข้อมูลได้นำเสนอในหัวข้อที่ 5.2

5.1 การทดลองเลือกโปรโตไทป์

การทดลองเลือกโปรโตไทป์ในที่นี้คือการทดลองเพื่อประเมินประสิทธิภาพในการลดจำนวนโปรโตไทป์ของวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน และจากที่ได้ทราบไปแล้วว่าวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนเป็นวิธีการเลือกโปรโตไทป์แบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้นวิธีการหนึ่ง ดังนั้นจึงสามารถทำการประเมินประสิทธิภาพในการลดจำนวนโปรโตไทป์ของวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนได้โดยการเปรียบจำนวนของโปรโตไทป์ผลลัพท์ที่เลือกได้ด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนกับจำนวนโปรโตไทป์ผลลัพท์ที่เลือกได้ด้วยวิธีการเลือกโปรโตไทป์แบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้นวิธีการอื่นๆ

ในหัวข้อนี้ได้นำเสนอผลการทดลองเลือกโปรโตไทป์โดยใช้วิธีการเลือกโปรโตไทป์แบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น 3 วิธี ซึ่งได้ถูกนำเสนอไปแล้วในบทที่ผ่านมา ดังนี้

1) วิธีการ condensed nearest neighbor rule (CNN) [14] โดยสามารถศึกษารายละเอียดและขั้นตอนการทำงานของวิธีการนี้เพิ่มเติม ได้ในบทที่ 3 หัวข้อที่ 3.2.1
2) วิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยการใช้ genetic algorithm (ในที่นี้เรียกชื่อย่อว่า "GA") [15] โดยสามารถศึกษารายละเอียดและขั้นตอนการทำงานของวิธีการนี้เพิ่มเติมได้ในบทที่ 3 หัวข้อที่ 3.2.2
3) วิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน ซึ่งเป็นวิธีการเลือกโปรโตไทป์ที่วิทยานิพนธ์ฉบับนี้ได้นำเสนอ (ในที่นี้เรียกชื่อย่อว่า "CLUSTER") โดยสามารถศึกษารายละเอียดและขั้นตอนการทำงานของวิธีการนี้เพิ่มเติมได้ในบทที่ 4

การทดลองเลือกโปรโตไทป์ได้ทดลองกับชุดข้อมูลจำนวน 6 ชุด คือ ชุดข้อมูลถั่วหลือง ชุดข้อมูลสวนสัตว์ ข้อมูลดอกไอริช ชุดข้อมูลไวน์ ชุดข้อมูลอีโคไล และชุดข้อมูลทิคแทคโท โดยการทดลองทั้งหมดนี้ทำบนเครื่องคอมพิวเตอร์ซีพียู Intel Pentium M ความเร็ว 1.73 GHz ความจุหน่วยความจำขนาด 1 GB

5.1.1 อัตราส่วนเปอร์เซ็นต์ของจำนวนโปรโตไทป์ที่เลือกได้ต่อขนาดเซ็ตข้อมูลตั้งต้น

หลักการประเมินนี้บ่งถึงความสามารถในการลดจำนวนโปรโตไทป์สำหรับอ้างอิงของวิธีการเลือกโปรโตไทป์แบบต่างๆ โดยการวัดค่าจากอัตราส่วนเปอร์เซ็นต์ของจำนวนโปรโตไทป์ที่เลือกได้โดยวิธีการนั้นๆต่อขนาดเซ็ตข้อมูลตั้งต้น ซึ่งสามารถแสดงได้ดังสมการนี้

$$\frac{|\,PrototypeSet\,|}{|\,OriginalDataSet\,|}\times 100 \quad (\%) \tag{5.1}$$

โดยที่

$|\,PrototypeSet\,|$ คือ ขนาดของเซ็ตโปรโตไทป์ที่เลือกได้โดยวิธีการนั้นๆ

$|\,OriginalDataset\,|$ คือ ขนาดของเซ็ตข้อมูลตั้งต้น

โดยที่ยิ่งค่าอัตราส่วนเปอร์เซ็นต์ของจำนวนโปรโตไทป์ที่เลือกได้ต่อขนาดเซ็ตข้อมูลตั้งต้นมีค่าน้อย ย่อมหมายถึงวิธีการเลือกโปรโตไทป์นั้นมีความสามารถในการลดจำนวนโปรโตไทป์ของชุดข้อมูลนี้ได้มาก ในทางตรงกันข้าม ยิ่งค่าอัตราส่วนเปอร์เซ็นต์ของจำนวนโปรโตไทป์ที่เลือกได้ต่อขนาดเซ็ตข้อมูลตั้งต้นมีค่ามาก ย่อมหมายถึงวิธีการเลือกโปรโตไทป์นั้นมีความสามารถในการลดจำนวนโปรโตไทป์ของชุดข้อมูลนี้ได้น้อย

5.1.2 ผลการทดลองเลือกโปรโตไทป์ของชุดข้อมูลถั่วเหลือง

การทดลองนี้ ทดลองบนชุดข้อมูลถั่วเหลือง โดยใช้ฟังก์ชันวัดระยะทางยูคลิดีน (Euclidean distance function)

5.1.2.1 ชุดข้อมูลถั่วเหลือง (Small Soybean Dataset)

ชุดข้อมูลนี้เป็นชุดข้อมูลขนาดเล็กที่เก็บข้อมูลตัวอย่างของถั่วเหลืองเพื่อใช้ในการวิเคราะห์เพื่อชี้เฉพาะชนิดของโรคที่เกิดขึ้นในถั่วเหลือง (Soybean Disease Diagnosis) ซึ่งได้มาจากแหล่งรวมชุดข้อมูลของ UCI Machine Learning Repository [18]

ชุดข้อมูลนี้มีลักษณะเฉพาะ(Feature) 35 มิติ (แอททริบิว)

ชุดข้อมูลนี้มีขนาด 47 ออบเจ็ค

ชุดข้อมูลนี้มีชนิดที่เป็นไปได้ทั้งหมด 4 ชนิด (ชนิดของโรคที่เกิดขึ้นในถั่วเหลือง)

5.1.2.2 ผลการทดลองที่ได้จากการใช้วิธีการ Condensed Nearest Neighbor Rule (CNN)

วิธีการนี้สามารถเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์ของชุดข้อมูลถั่วเหลืองอันนี้ ได้ผลลัพท์เป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องขนาด 7 จากเซ็ตข้อมูลถั่วเหลืองที่มีขนาด 47 โดยใช้เวลาในการทำงานประมาณ 0.015 วินาที

5.1.2.3 ผลการทดลองที่ได้จากการใช้วิธีการเลือกใช้เลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยการใช้ Genetic Algorithm (GA)

การทดลองของวิธีการนี้มีการกำหนดค่าคงที่อ้างอิงต่างๆ ไว้ดังนี้

- จำนวนประชากร (Population Size) $N_{pop} = 40$
- จำนวนรอบของเจนเนอเรชัน(Generation Number) $M_{gen} = 50$
- สัมประสิทธ์ถ่วงน้ำหนัก (weighted coefficient) $\alpha = 1/N\_OriginalDataset = 1/47 = 0.0213$
- ค่าความน่าจะเป็นที่แต่ละบิทโครโมโซมจะมีค่าเป็น 1 ในขั้นตอนการสุ่มสร้างโครโมโซมตั้งต้น $P_{ini} = 0.8$
- ค่าความน่าจะเป็นที่จะเกิดการกลายพันธ์ของแต่ละบิทโครโมโซม (Mutation Rate) $P_m = 0.015$

เนื่องจากการทำงานของวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยการใช้ Genetic Algorithm เป็นวิธีการเชิงเฟ้นสุ่มดังนั้นผลลัพท์ของแต่ละรอบของการทำงานจึงไม่แน่นอนเสมอไป ดังนั้นเราจึงทำการเก็บผลการทดลองจาก แต่ละรอบของการทำงานเป็นจำนวน 10 รอบ เพื่อหาค่าเฉลี่ย โดยผลการทดลองแต่ละรอบการทำงานมีดังตารางที่ 5.1

ตารางที่ 5.1 แสดงผลการทดลองใช้วิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยการใช้ Genetic Algorithm บนเซ็ตข้อมูลถั่วเหลืองเป็นจำนวน 10 รอบ

การทำงานรอบที่	จำนวนโปรโตไทป์	เวลา (วินาที)
1	6	0.250
2	6	0.343
3	6	0.343
4	6	0.250
5	6	0.265
6	6	0.328
7	5	0.250
8	4	0.312
9	6	0.281
10	6	0.343
	Min = 4	Average = 0.297
	Max = 6	
	Average = 5.7	

จากการทำงานตามวิธีการนี้เป็นจำนวน 10 รอบ วิธีการนี้สามารถเลือกโปรโตไทป์ของชุดข้อมูลถั่วเหลืองอันนี้

ได้ผลลัพท์ที่ดีที่สุดเป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องขนาด 4

ได้ผลลัพท์ที่แย่ที่สุดเป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องขนาด 6

ได้ผลลัพท์ที่เฉลี่ยเป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องขนาด 5.7

จากเซ็ตข้อมูลถั่วเหลืองที่มีจำนวน 47

เวลาในแต่ละรอบการทำงานประมาณ 0.297 วินาที

5.1.2.4 ผลการทดลองที่ได้จากวิธีการเลือกโปรโตไทป์โดยใช้วิธีการจัดกลุ่มข้อมูลแบบมีการสอน (ในที่นี้เรียกย่อว่า"CLUSTER")

วิธีการนี้สามารถเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์ของชุดข้อมูลถั่วเหลืองอันนี้ได้ผลลัพท์เป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องขนาด 9 จากเซ็ตข้อมูลถั่วเหลืองที่มีขนาด 47 โดยใช้เวลาในการทำงานประมาณ 0.015 วินาที

5.1.2.5 สรุปผลการทดลองของชุดข้อมูลถั่วเหลือง

เราสามารถสรุปรวมผลการทดลองได้โดยแสดงดังตารางที่ 5.2 ดังนี้

ตารางที่ 5.2 แสดงขนาดของเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องของชุดข้อมูลถั่วเหลืองที่เลือกได้โดยใช้วิธีการต่างๆพร้อมทั้งเวลาในการทำงานของวิธีการเหล่านั้น

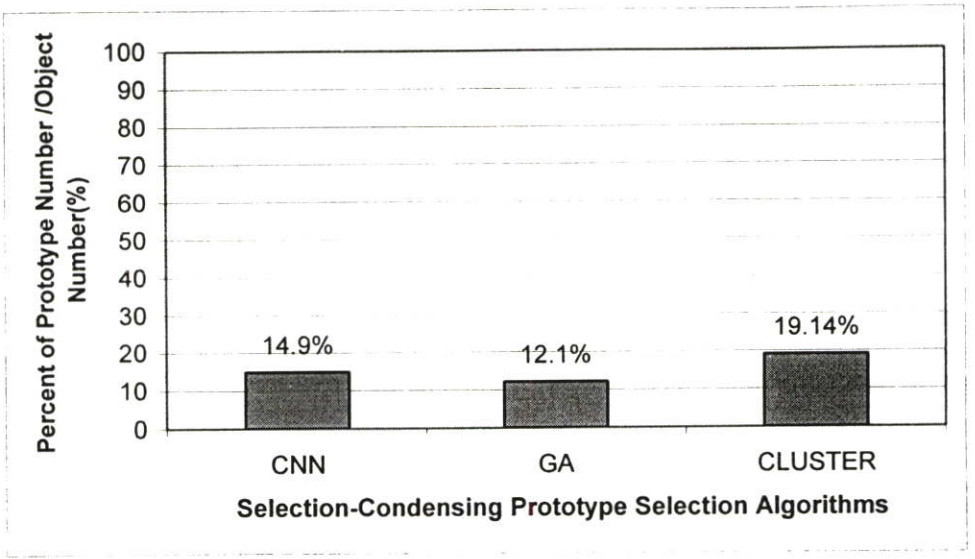
	ชุดข้อมูลถั่วเหลือง			
	CNN	**GA**		**CLUSTER**
จำนวนโปรโตไทป์	7	Min	4	9
		Max	6	
		$\bar{X}$	5.7	
เวลา (วินาที)	0.015	0.297		0.015

ชุดข้อมูลถั่วเหลืองนี้มีจำนวน 47 ออบเจ็ค เราสามารถคำนวนหาอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่เลือกได้จากแต่ละวิธีต่อขนาดของเซ็ตข้อมูลถั่วเหลือง ได้ดังนี้

ตารางที่ 5.3 แสดงอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่เลือกได้จากแต่ละวิธีต่อขนาดของเซ็ตข้อมูลถั่วเหลือง

	ชุดข้อมูลถั่วเหลือง			
	CNN	**GA**		**CLUSTER**
เปอร์เซ็นต์จำนวนโปรโตไทป์ที่เลือกได้ต่อขนาดเซ็ตตั้งต้น (%)	14.9	Min	8.5	19.14
		Max	12.8	
		$\bar{X}$	12.1	

เราสามารถแสดงอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์เลือกได้จากแต่ละวิธีต่อขนาดของเซ็ตข้อมูลถั่วเหลืองในรูปแบบกราฟ ซึ่งแสดงไว้ในรูปที่ 5.1 ดังนี้



รูปที่ 5.1 กราฟแสดงอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่เลือกได้จากแต่ละวิธีต่อขนาดของเซ็ตข้อมูลถั่วเหลือง

5.1.3 ผลการทดลองเลือกโปรโตไทป์ของของชุดข้อมูลสวนสัตว์

การทดลองนี้ ทดลองบนชุดข้อมูลสวนสัตว์ โดยใช้ฟังก์ชันวัดระยะทางยูคลิดีน (Euclidean distance function)

5.1.3.1 ชุดข้อมูลสวนสัตว์ (Zoo Dataset)

ชุดข้อมูลนี้เป็นชุดข้อมูลขนาดเล็กที่เก็บข้อมูลตัวอย่างสัตว์หลากหลายชนิด(เช่น ไก่ ควาย กบ งู เป็นต้น) โดยเก็บลักษณะเฉพาะ 16 ประการ (เช่น มีขนหรือไม่ จำนวนขา มีฟันหรือไม่ เป็นต้น) เพื่อใช้ในการวิเคราะห์เพื่อชี้เฉพาะประเภทของสัตว์เหล่านั้น (ในชุดข้อมูลได้แบ่งสัตว์ออกเป็น 7 ประเภท) ซึ่งได้มาจากแหล่งรวมชุดข้อมูลของ UCI Machine Learning Repository [18]

ชุดข้อมูลนี้มีลักษณะเฉพาะ(Feature) 16 มิติ (แอททริบิว) ซึ่งบางค่าในแอททริบิวเป็นค่าตัวเลขแสดงปริมาณของแอทริบิว บางค่าในแอททริบิวเป็นค่าบ่งชี้ว่าจริงหรือเท็จเท่านั้น

ชุดข้อมูลนี้มีขนาด 101 ออบเจ็ค

ชุดข้อมูลนี้มีชนิดที่เป็นไปได้ทั้งหมด 7 ชนิด (ชนิดของสัตว์)

5.1.3.2 ผลการทดลองที่ได้จากการใช้วิธีการ Condensed Nearest Neighbor Rule (CNN)

วิธีการนี้สามารถเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์ของของชุดข้อมูลสวนสัตว์อันนี้ ได้ผลลัพท์เป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องขนาด 15 จากเซ็ตข้อมูลสวนสัตว์ที่มีขนาด 101 โดยใช้เวลาในการทำงานประมาณ 0.015 วินาที

5.1.3.3 ผลการทดลองที่ได้จากการใช้วิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยการใช้ Genetic Algorithm (GA)

การทดลองของวิธีการนี้มีการกำหนดค่าคงที่อ้างอิงต่างๆ ไว้ดังนี้

- จำนวนประชากร (Population Size) $N_{pop} = 40$
- จำนวนรอบของเจนเนอเรชัน(Generation Number) $M_{gen} = 100$
- สัมประสิทธ์ถ่วงน้ำหนัก (weighted coefficient) $\alpha = 1/N\_OriginalDataset = 0.0099$
- ค่าความน่าจะเป็นที่แต่ละบิทโครโมโซมจะมีค่าเป็น 1 ในขั้นตอนการสุ่มสร้างโครโมโซมตั้งต้น $P_{ini} = 0.8$
- ค่าความน่าจะเป็นที่จะเกิดการกลายพันธ์ของแต่ละบิทโครโมโซม (Mutation Rate) $P_m = 0.015$

เนื่องจากการทำงานของวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยการใช้ Genetic Algorithm เป็นวิธีการเชิงแบบเฟ้นสุ่ม ดังนั้นผลลัพท์ของแต่ละรอบของการทำงานจึงไม่แน่นอนเสมอไป ดังนั้นเราจึงทำการเก็บผลการทดลองจาก แต่ละรอบของการทำงานเป็นจำนวน 10 รอบเพื่อหาค่าเฉลี่ย โดยผลการทดลองแต่ละรอบการทำงานมีดังตารางที่ 5.4

ตารางที่ 5.4 แสดงผลการทดลองใช้วิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยการใช้ Genetic Algorithm บนชุดข้อมูลสวนสัตว์เป็นจำนวน 10 รอบ

การทำงานรอบที่	จำนวนโปรโตไทป์	เวลา (วินาที)
1	13	0.531
2	13	0.546
3	13	0.468
4	14	0.562
5	13	0.515
6	11	0.546
7	13	0.468
8	14	0.437
9	13	0.484
10	12	0.546
	Min = 11	Average = 0.51
	Max = 14	
	Average = 12.9	

จากการทำงานตามวิธีการนี้เป็นจำนวน 10 รอบ วิธีการนี้สามารถเลือกหาเซ็ตของโปรโตไทป์ของชุดข้อมูลสวนสัตว์อันนี้

ได้ผลลัพท์ที่ดีที่สุดเป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องขนาด 11

ได้ผลลัพท์ที่แย่ที่สุดเป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องขนาด 14

ได้ผลลัพท์ที่เฉลี่ยเป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องขนาด 12.9

จากเซ็ตข้อมูลสวนสัตว์ที่มีจำนวน 101

เวลาในแต่ละรอบการทำงานประมาณ 0.51 วินาที

5.1.3.4 ผลการทดลองที่ได้จากวิธีการเลือกโปรโตไทป์โดยใช้วิธีการจัดกลุ่มข้อมูลแบบมีการสอน (ในที่นี้เรียกย่อว่า"CLUSTER")

วิธีการนี้สามารถเลือกหาเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องของชุดข้อมูลสวนสัตว์อันนี้ ได้ผลลัพท์เป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องขนาด 15 จากเซ็ตข้อมูลสวนสัตว์ที่มีขนาด 101 โดยใช้เวลาในการทำงานประมาณ 0.046 วินาที

5.1.3.5 สรุปผลการทดลองของชุดข้อมูลสวนสัตว์

เราสามารถสรุปรวมผลการทดลองได้โดยแสดงดังตารางที่ 5.5 ดังนี้

ตารางที่ 5.5 แสดงขนาดของเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องของชุดข้อมูลสวนสัตว์ที่เลือกได้โดยใช้วิธีการต่างๆพร้อมทั้งเวลาในการทำงานของวิธีการเหล่านั้น

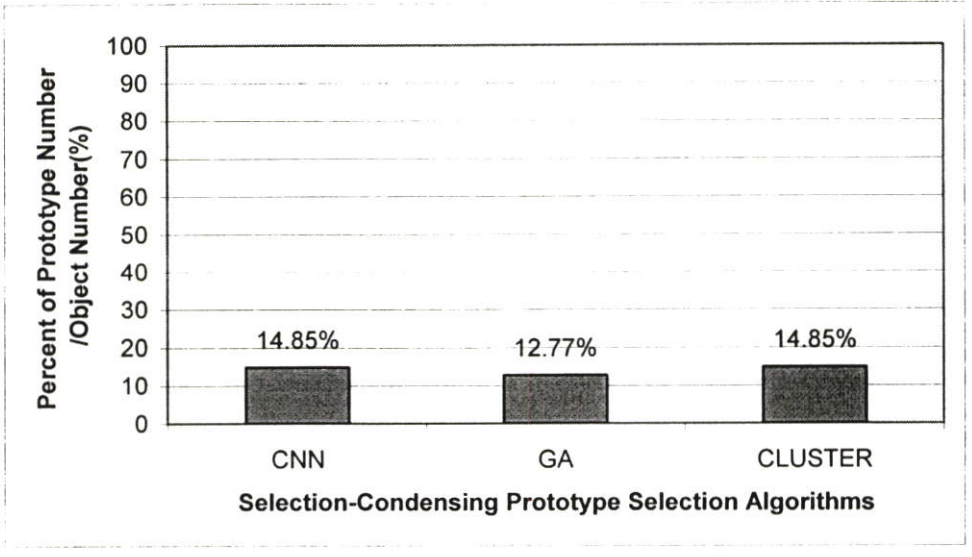
	ชุดข้อมูลสวนสัตว์			
	CNN	**GA**		**CLUSTER**
จำนวนโปรโตไทป์	15	Min	11	15
		Max	14	
		$\bar{X}$	12.9	
เวลา (วินาที)	0.015	0.51		0.046

ชุดข้อมูลสวนสัตว์นี้มีจำนวน 101 ออบเจ็ค เราสามารถคำนวนหาอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ได้จากแต่ละวิธีต่อขนาดของเซ็ตข้อมูลสวนสัตว์ ได้ดังตารางที่ 5.6

ตารางที่ 5.6 แสดงอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่เลือกได้จากแต่ละวิธีต่อขนาดของเซ็ตข้อมูลสวนสัตว์

	ชุดข้อมูลสวนสัตว์			
	CNN	GA		CLUSTER
เปอร์เซ็นต์จำนวนโปรโตไทป์ที่เลือกได้ต่อขนาดเซ็ตตั้งต้น (%)	14.85	Min	10.89	14.85
		Max	13.86	
		$\bar{X}$	12.77	

เราสามารถแสดงอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่เลือกได้จากแต่ละวิธีต่อขนาดของเซ็ตข้อมูลสวนสัตว์ในรูปแบบกราฟ ซึ่งแสดงไว้ในรูปที่ 5.2



รูปที่ 5.2 กราฟแสดงอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่เลือกได้จากแต่ละวิธีต่อขนาดของเซ็ตข้อมูลสวนสัตว์

5.1.4 ผลการทดลองเลือกโปรโตไทป์ของชุดข้อมูลดอกไอริช

การทดลองนี้ ทดลองบนชุดข้อมูลดอกไอริช โดยใช้ฟังก์ชันวัดระยะทางยูคลิดีน (Euclidean distance function)

5.1.4.1 ชุดข้อมูลดอกไอริช (Iris Dataset)

ชุดข้อมูลนี้เป็นชุดข้อมูลขนาดเล็กที่เก็บข้อมูลตัวอย่างของดอกไอริชเพื่อใช้ในการวิเคราะห์เพื่อชี้เฉพาะว่าดอกไอริชพันธุ์ไหน ซึ่งชุดข้อมูลดอกไอริชนี้เป็นชุดข้อมูลที่รู้จักอย่างแพร่หลายในแวดวงงานวิจัยโดยเฉพาะด้านการรู้จำรูปแบบข้อมูล(Pattern recognition) และแมชชีนเลอนนิง (Machine learning) และถูกเลือกใช้เป็นชุดข้อมูลสำหรับเปรียบสมรรถนะ (Benchmarking) ของวิธีการเลือกโปรโตไทป์ (Prototype selection) ชุดข้อมูลได้มาจากแหล่งรวมชุดข้อมูลของ UCI Machine Learning Repository [18]

ชุดข้อมูลนี้มีลักษณะเฉพาะ(Feature) 4 มิติ (แอททริบิว) ซึ่งแต่ละค่าในแอททริบิวเป็นค่าตัวเลขแสดงปริมาณของแอทริบิวนั้นๆ

ชุดข้อมูลนี้มีขนาด 150 ออบเจ็ค

ชุดข้อมูลนี้มีชนิดที่เป็นไปได้ทั้งหมด 3 ชนิด (พันธุ์ของดอกไอริช)

5.1.4.2 ผลการทดลองที่ได้จากการใช้วิธีการ Condensed Nearest Neighbor Rule (CNN)

วิธีการนี้สามารถเลือกหาเซ็ตของโปรโตไทป์ของชุดข้อมูลดอกไอริชอันนี้ ได้ผลลัพท์เป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่มีขนาด 24 จากเซ็ตข้อมูลดอกไอริชที่มีขนาด 150 โดยใช้เวลาในการทำงานประมาณ 0.031 วินาที

5.1.4.3 ผลการทดลองที่ได้จากการใช้วิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยการใช้ Genetic Algorithm (GA)

การทดลองของวิธีการนี้มีการกำหนดค่าคงที่อ้างอิงต่างๆ ไว้ดังนี้

- จำนวนประชากร (Population Size) $N_{pop} = 40$
- จำนวนรอบของเจนเนอเรชัน(Generation Number) $M_{gen} = 100$
- สัมประสิทธ์ถ่วงน้ำหนัก (weighted coefficient) $\alpha = 1/N\_OriginalDataset = 1/150 = 0.0067$
- ค่าความน่าจะเป็นที่แต่ละบิทโครโมโซมจะมีค่าเป็น 1 ในขั้นตอนการสุ่มสร้างโครโมโซมตั้งต้น $P_{ini} = 0.8$
- ค่าความน่าจะเป็นที่จะเกิดการกลายพันธ์ของแต่ละบิทโครโมโซม (Mutation Rate) $P_m = 0.015$

เนื่องจากการทำงานของวิธีการเลือกใช้เลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยการใช้ Genetic Algorithm เป็นวิธีการเชิงเฟ้นสุ่ม ดังนั้นผลลัพท์ของแต่ละรอบของการทำงานจึงไม่แน่นอนเสมอไป ดังนั้นเราจึงทำการเก็บผลการทดลองจาก แต่ละรอบของการทำงานเป็นจำนวน 10 รอบเพื่อหาค่าเฉลี่ย โดยผลการทดลองแต่ละรอบการทำงานมีดังตารางที่ 5.7

ตารางที่ 5.7 แสดงผลการทดลองใช้วิธีการเลือกใช้เลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยการใช้ Genetic Algorithm บนชุดข้อมูลดอกไอริชเป็นจำนวน 10 รอบ

การทำงานรอบที่	จำนวนโปรโตไทป์	เวลา (วินาที)
1	12	1.468
2	15	1.421
3	15	1.406
4	14	1.390
5	13	1.421
6	12	1.343
7	14	1.453
8	15	1.468
9	16	1.390
10	15	1.437
	Min = 12	Average = 1.420
	Max = 16	
	Average = 14.1	

จากการทำงานตามวิธีการนี้เป็นจำนวน 10 รอบ วิธีการนี้สามารถเลือกหาเซ็ตของโปรโตไทป์ของชุดข้อมูลดอกไอริชอันนี้

ได้ผลลัพท์ที่ดีที่สุดเป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่มีขนาด 12

ได้ผลลัพท์ที่แย่ที่สุดเป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่มีขนาด 16

ได้ผลลัพท์ที่เฉลี่ยเป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่มีขนาด 14.7

จากเซ็ตข้อมูลดอกไอริชที่มีจำนวน 150

เวลาในแต่ละรอบการทำงานประมาณ 1.42 วินาที

5.1.4.4 ผลการทดลองที่ได้จากวิธีการเลือกโปรโตไทป์โดยใช้วิธีการจัดกลุ่มข้อมูลแบบมีการสอน (ในที่นี้เรียกย่อว่า"CLUSTER")

วิธีการนี้สามารถเลือกหาเซ็ตของโปรโตไทป์ของชุดข้อมูลดอกไอริชอันนี้ ได้ผลลัพท์เป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่มีขนาด 17 จากเซ็ตข้อมูลดอกไอริชที่มีขนาด 150 โดยใช้เวลาในการทำงานประมาณ 0.031 วินาที

5.1.4.5 สรุปผลการทดลองของชุดข้อมูลดอกไอริช

เราสามารถสรุปรวมผลการทดลองได้โดยแสดงดังตารางที่ 5.8 ดังนี้

ตารางที่ 5.8 แสดงขนาดของเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องของชุดข้อมูลดอกไอริชที่เลือกได้โดยใช้วิธีการต่างๆพร้อมทั้งเวลาในการทำงานของวิธีการเหล่านั้น

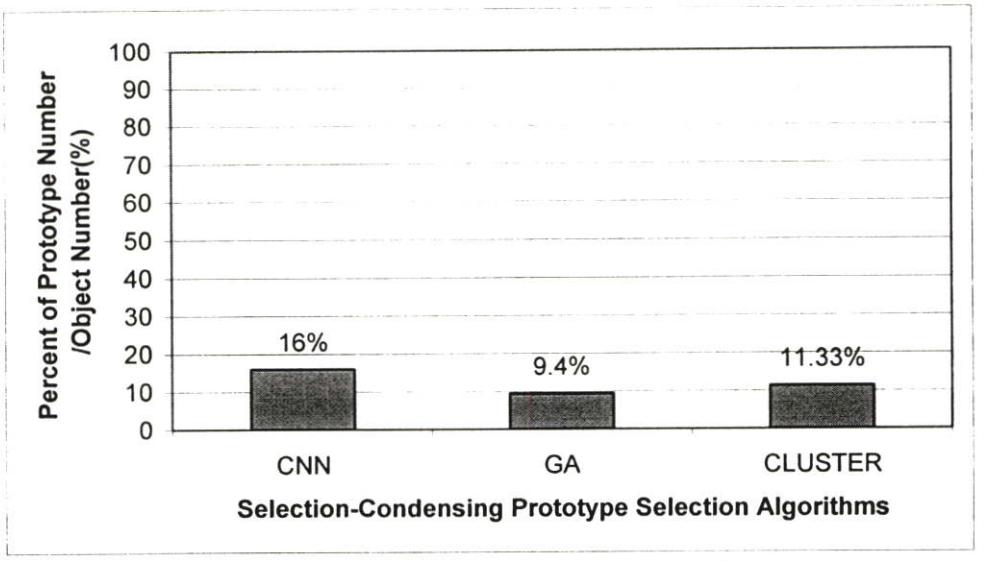
	ชุดข้อมูลดอกไอริช			
	CNN	**GA**		**CLUSTER**
จำนวนโปรโตไทป์	24	Min	12	17
		Max	16	
		$\bar{X}$	14.1	
เวลา (วินาที)	0.031	1.420		0.031

ชุดข้อมูลดอกไอริชนี้มีจำนวน 150 ออบเจ็ค เราสามารถคำนวนหาอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่เลือกได้จากแต่ละวิธีต่อขนาดของชุดข้อมูลดอกไอริช ได้ดังที่แสดงในตารางที่ 5.9

ตารางที่ 5.9 แสดงอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่เลือกได้จากแต่ละวิธีต่อขนาดของชุดข้อมูลดอกไอริช

	ชุดข้อมูลดอกไอริช			
	CNN	**GA**		**CLUSTER**
เปอร์เซ็นต์จำนวนโปรโตไทป์ที่เลือกได้ต่อขนาดเซ็ตตั้งต้น (%)	16	Min	8	11.33
		Max	10.67	
		$\bar{X}$	9.4	

เราสามารถแสดงอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่เลือกได้จากแต่ละวิธีต่อขนาดของเซ็ตข้อมูลดอกไอริชในรูปแบบกราฟ ซึ่งแสดงไว้ในรูปที่ 5.3



รูปที่ 5.3 กราฟแสดงอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่เลือกได้จากแต่ละวิธีต่อขนาดของเซ็ตข้อมูลดอกไอริช

5.1.5 ผลการทดลองเลือกโปรโตไทป์ของชุดข้อมูลไวน์

การทดลองนี้ ทดลองบนชุดข้อมูลไวน์ โดยใช้ฟังก์ชันวัดระยะทางยูคลิดีน (Euclidean distance function)

5.1.5.1 ชุดข้อมูลไวน์ (Wine Dataset)

ชุดข้อมูลนี้เป็นชุดข้อมูลขนาดเล็กที่เก็บข้อมูลตัวอย่างสารที่อยู่ไวน์เพื่อใช้ในการวิเคราะห์เพื่อชี้เฉพาะประเภทของไวน์ ชุดข้อมูลนี้ได้มาจากแหล่งรวมชุดข้อมูลของ UCI Machine Learning Repository [18]

ชุดข้อมูลนี้มีลักษณะเฉพาะ(Feature) 13 มิติ (แอททริบิว) ซึ่งแต่ละค่าในแอททริบิวเป็นค่าตัวเลขแสดงปริมาณของสารต่างที่อยู่ในไวน์

ชุดข้อมูลนี้มีขนาด 178 ออบเจ็ค

ชุดข้อมูลนี้มีชนิดที่เป็นไปได้ทั้งหมด 3 ชนิด (ชนิดของไวน์)

5.1.5.2 ผลการทดลองที่ได้จากการใช้วิธีการ Condensed Nearest Neighbor Rule (CNN)

วิธีการนี้สามารถเลือกหาเซ็ตของโปรโตไทป์ของชุดข้อมูลไวน์อันนี้ ได้ผลลัพท์เป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่มีขนาด 71 จากชุดข้อมูลไวน์ที่มีขนาด 178 โดยใช้เวลาในการทำงานประมาณ 0.093 วินาที

5.1.5.3 ผลการทดลองที่ได้จากการใช้วิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยการใช้ Genetic Algorithm (GA)

การทดลองของวิธีการนี้มีการกำหนดค่าคงที่อ้างอิงต่างๆ ไว้ดังนี้

- จำนวนประชากร (Population Size) $N_{pop} = 40$
- จำนวนรอบของเจนเนอเรชัน(Generation Number) $M_{gen} = 100$
- สัมประสิทธ์ถ่วงน้ำหนัก (weighted coefficient) $\alpha = 1/N\_OriginalDataset = 1/178 = 0.0056$
- ค่าความน่าจะเป็นที่แต่ละบิทโครโมโซมจะมีค่าเป็น 1 ในขั้นตอนการสุ่มสร้างโครโมโซมตั้งต้น $P_{ini} = 0.8$
- ค่าความน่าจะเป็นที่จะเกิดการกลายพันธ์ของแต่ละบิทโครโมโซม (Mutation Rate) $P_m = 0.015$

เนื่องจากการทำงานของวิธีการเลือกใช้เลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยการใช้ Genetic Algorithm เป็นวิธีการเชิงเฟ้นสุ่ม ดังนั้นผลลัพท์ของแต่ละรอบของการทำงานจึงไม่แน่นอนเสมอไป ดังนั้นเราจึงทำการเก็บผลการทดลองจาก แต่ละรอบของการทำงานเป็นจำนวน 10 รอบเพื่อหาค่าเฉลี่ย โดยผลการทดลองแต่ละรอบการทำงานมีดังตารางที่ 5.10

ตารางที่ 5.10 แสดงผลการทดลองใช้วิธีการเลือกใช้เลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยการใช้ Genetic Algorithm บนชุดข้อมูลไวน์เป็นจำนวน 10 รอบ

การทำงานรอบที่	จำนวนโปรโตไทป์	เวลา (วินาที)
1	63	1.265
2	65	1.265
3	63	1.296
4	64	1.281
5	62	1.312
6	62	1.296
7	62	1.343
8	63	1.343
9	62	1.296
10	62	1.234
	Min = 62	
	Max = 65	Average =1.293
	Average = 62.8	

จากการทำงานตามวิธีการนี้เป็นจำนวน 10 รอบ วิธีการนี้สามารถเลือกหาเซ็ตของโปรโตไทป์ของชุดข้อมูลไวน์อันนี้

ได้ผลลัพท์ที่ดีที่สุดเป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่มีขนาด 62

ได้ผลลัพท์ที่แย่ที่สุดเป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่มีขนาด 65

ได้ผลลัพท์ที่เฉลี่ยเป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่มีขนาด 62.8

จากเซ็ตข้อมูลชุดข้อมูลไวน์ที่มีจำนวน 178

เวลาในแต่ละรอบการทำงานประมาณ 1.3 วินาที

5.1.5.4 ผลการทดลองที่ได้จากวิธีการเลือกโปรโตไทป์โดยใช้วิธีการจัดกลุ่มข้อมูลแบบมีการสอน (ในที่นี้เรียกย่อว่า"**CLUSTER**")

วิธีการนี้สามารถเลือกหาเซ็ตของโปรโตไทป์ของชุดข้อมูลไวน์อันนี้ ได้ผลลัพท์เป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่มีขนาด 63 จากชุดข้อมูลไวน์ที่มีขนาด 178 โดยใช้เวลาในการทำงานประมาณ 0.093 วินาที

5.1.5.5 สรุปผลการทดลองของชุดข้อมูลไวน์

เราสามารถสรุปรวมผลการทดลองได้โดยแสดงดังตารางที่ 5.11 ดังนี้

ตารางที่ 5.11 แสดงขนาดของเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องของชุดข้อมูลไวน์ที่เลือกได้โดยใช้วิธีการต่างๆพร้อมทั้งเวลาในการทำงานของวิธีการเหล่านั้น

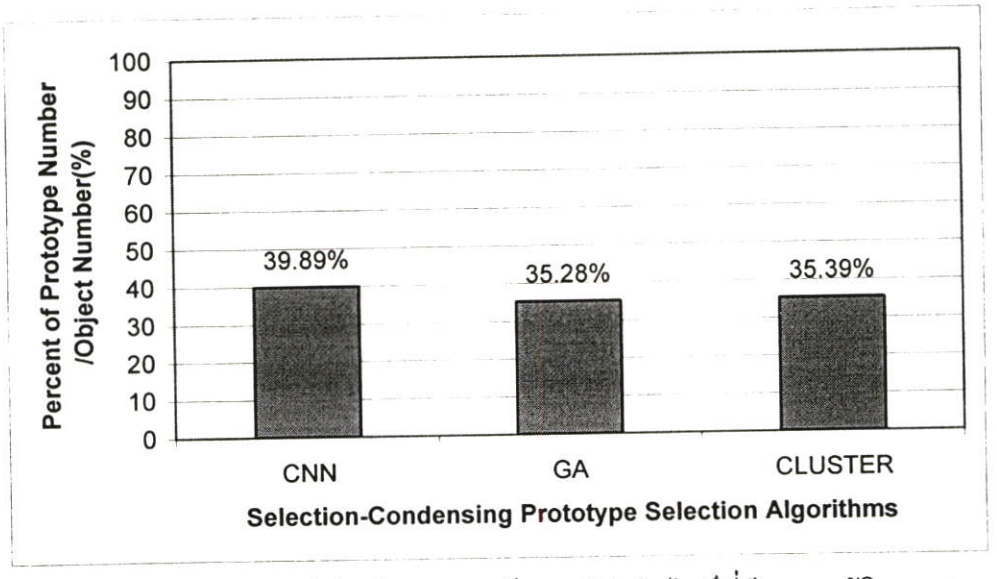
	ชุดข้อมูลไวน์			
	CNN	**GA**		**CLUSTER**
จำนวนโปรโตไทป์	71	Min	62	63
		Max	65	
		$\bar{X}$	62.8	
เวลา (วินาที)	0.093	1.293		0.093

ชุดข้อมูลไวน์นี้มีจำนวน 178 ออบเจ็ค เราสามารถคำนวนหาอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่เลือกได้จากแต่ละวิธีต่อขนาดของชุดข้อมูลไวน์ ได้ดังที่แสดงในตารางที่ 5.12

ตารางที่ 5.12 แสดงอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่เลือกได้จากแต่ละวิธีต่อขนาดของชุดข้อมูลไวน์

	ชุดข้อมูลไวน์			
	CNN	**GA**		**CLUSTER**
เปอร์เซ็นต์จำนวนโปรโตไทป์ที่เลือกได้ต่อขนาดเซ็ตตั้งต้น (%)	39.89	Min	34.83	35.39
		Max	36.52	
		$\bar{X}$	35.28	

เราสามารถแสดงอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่เลือกได้จากแต่ละวิธีต่อขนาดของชุดข้อมูลไวน์ในรูปแบบกราฟ ซึ่งแสดงไว้ในรูปที่ 5.4



รูปที่ 5.4 กราฟแสดงอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่เลือกได้จากแต่ละวิธีต่อขนาดของชุดข้อมูลไวน์

5.1.6 ผลการทดลองเลือกโปรโตไทป์ของชุดข้อมูลอีโคไล

การทดลองนี้ ทดลองบนชุดข้อมูลอีโคไล โดยใช้ฟังก์ชันวัดระยะทางยูคลิดีน (Euclidean distance function)

5.1.6.1 ชุดข้อมูลอีโคไล (E.coli Dataset)

ชุดข้อมูลนี้เป็นชุดข้อมูลขนาดเล็กได้มาจากแหล่งรวมชุดข้อมูลของ UCI Machine Learning Repository [18] ชุดข้อมูลนี้มีลักษณะเฉพาะ (Feature) 7 มิติ (แอททริบิว)

ชุดข้อมูลนี้มีขนาด 336 ออบเจ็ค

ชุดข้อมูลนี้มีชนิดที่เป็นไปได้ทั้งหมด 8 ชนิด

5.1.6.2 ผลการทดลองที่ได้จากการใช้วิธีการ Condensed Nearest Neighbor Rule (CNN)

วิธีการนี้สามารถเลือกหาเซ็ตของโปรโตไทป์ของชุดข้อมูลอีโคไลอันนี้ ได้ผลลัพท์เป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่มีขนาด 124 จากชุดข้อมูลอีโคไลที่มีขนาด 336 โดยใช้เวลาในการทำงานประมาณ 0.171 วินาที

5.1.6.3 ผลการทดลองที่ได้จากการใช้วิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยการใช้ Genetic Algorithm (GA)

การทดลองของวิธีการนี้มีการกำหนดค่าคงที่อ้างอิงต่างๆ ไว้ดังนี้

- จำนวนประชากร (Population Size) $N_{pop} = 40$
- จำนวนรอบของเจนเนอเรชัน(Generation Number) $M_{gen} = 200$
- สัมประสิทธ์ถ่วงน้ำหนัก (weighted coefficient) $\alpha = 1/N\_OriginalDataset = 0.003$
- ค่าความน่าจะเป็นที่แต่ละบิทโครโมโซมจะมีค่าเป็น 1 ในขั้นตอนการสุ่มสร้างโครโมโซมตั้งต้น $P_{ini} = 0.7$
- ค่าความน่าจะเป็นที่จะเกิดการกลายพันธ์ของแต่ละบิทโครโมโซม (Mutation Rate) $P_m = 0.015$

เนื่องจากการทำงานของวิธีการเลือกใช้เลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยการใช้ Genetic Algorithm เป็นวิธีการเชิงเฟ้นสุ่ม ดังนั้นผลลัพท์ของแต่ละรอบของการทำงานจึงไม่แน่นอนเสมอไป ดังนั้นเราจึงทำการเก็บผลการทดลองจาก แต่ละรอบของการทำงานเป็นจำนวน 10 รอบเพื่อหาค่าเฉลี่ย โดยผลการทดลองแต่ละรอบการทำงานมีดังตารางที่ 5.13

ตารางที่ 5.13 แสดงผลการทดลองใช้วิธีการเลือกใช้เลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยการใช้ Genetic Algorithm บนชุดข้อมูลไวน์เป็นจำนวน 10 รอบ

การทำงานรอบที่	จำนวนโปรโตไทป์	เวลา (วินาที)
1	117	4.593
2	113	4.562
3	109	4.562
4	116	4.515
5	112	4.578
6	114	4.531
7	106	4.515
8	103	4.671
9	108	4.609
10	118	4.500
	Min = 103	Average = 4.564
	Max = 118	
	Average = 111.6	

จากการทำงานตามวิธีการนี้เป็นจำนวน 10 รอบ วิธีการนี้สามารถเลือกหาเซ็ตของโปรโตไทป์ของชุดข้อมูลอีโคไลอันนี้

ได้ผลลัพท์ที่ดีที่สุดเป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่มีขนาด 103

ได้ผลลัพท์ที่แย่ที่สุดเป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่มีขนาด 118

ได้ผลลัพท์ที่เฉลี่ยเป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่มีขนาด 111.6

จากชุดข้อมูลอีโคไลที่มีจำนวน 336

เวลาในแต่ละรอบการทำงานประมาณ 4.564 วินาที

5.1.6.4 ผลการทดลองที่ได้จากวิธีการเลือกโปรโตไทป์โดยใช้วิธีการจัดกลุ่มข้อมูลแบบมีการสอน (ในที่นี้เรียกย่อว่า"**CLUSTER**")

วิธีการนี้สามารถเลือกหาเซ็ตของโปรโตไทป์ของชุดข้อมูลอีโคไลอันนี้ ได้ผลลัพท์เป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่มีขนาด 104 จากชุดข้อมูลอีโคไลที่มีขนาด 336 โดยใช้เวลาในการทำงานประมาณ 0.156 วินาที

5.1.6.5 สรุปผลการทดลองของชุดข้อมูลอีโคไล

เราสามารถสรุปรวมผลการทดลองได้โดยแสดงดังตารางที่ 5.14 ดังนี้

ตารางที่ 5.14 แสดงขนาดของเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องของชุดข้อมูลอีโคไลที่เลือกได้โดยใช้วิธีการต่างๆพร้อมทั้งเวลาในการทำงานของวิธีการเหล่านั้น

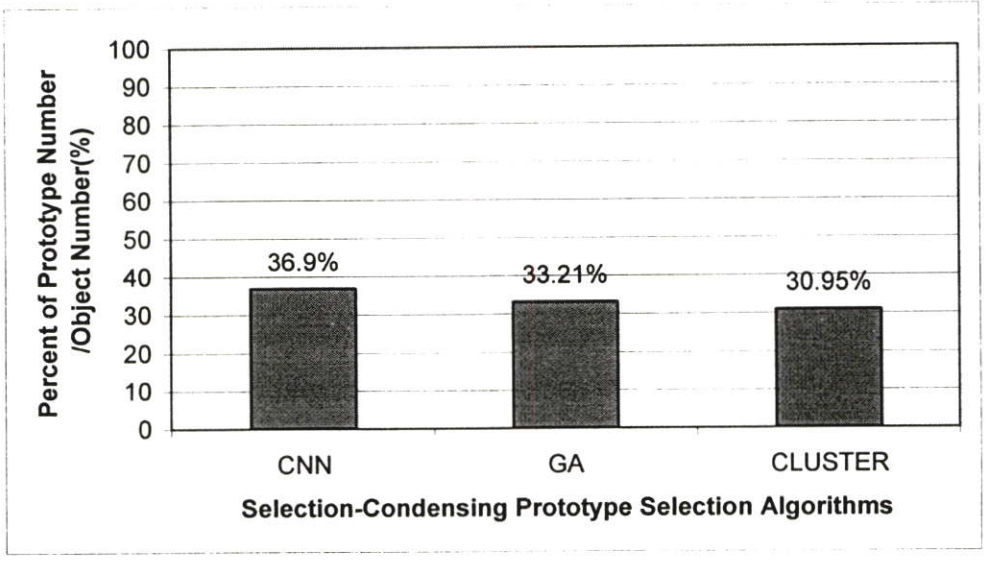
	ชุดข้อมูลอีโคไล			
	CNN	**GA**		**CLUSTER**
จำนวนโปรโตไทป์	124	Min	103	104
		Max	118	
		$\bar{X}$	111.6	
เวลา (วินาที)	0.171	4.564		0.156

ชุดข้อมูลอีโคไลนี้มีจำนวน 336 ออบเจ็ค เราสามารถคำนวนหาอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่เลือกได้จากแต่ละวิธีต่อขนาดของชุดข้อมูลอีโคไล ได้ดังที่แสดงในตารางที่ 5.15

ตารางที่ 5.15 แสดงอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่เลือกได้จากแต่ละวิธีต่อขนาดของชุดข้อมูลอีโคไล

	ชุดข้อมูลอีโคไล			
	CNN	GA		CLUSTER
เปอร์เซ็นต์จำนวนโปรโตไทป์ที่เลือกได้ต่อขนาดเซ็ตตั้งต้น (%)	36.90	Min	30.65	30.95
		Max	35.12	
		$\bar{X}$	33.21	

เราสามารถแสดงอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่เลือกได้จากแต่ละวิธีต่อขนาดของชุดข้อมูลอีโคไลในรูปแบบกราฟ ซึ่งแสดงไว้ในรูปที่ 5.5



รูปที่ 5.5 กราฟแสดงอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่เลือกได้จากแต่ละวิธีต่อขนาดของชุดข้อมูลอีโคไล

5.1.7 ผลการทดลองเลือกโปรโตไทป์ของชุดข้อมูลทิคแทคโท

การทดลองนี้ ทดลองบนชุดข้อมูลทิคแทคโท โดยใช้ฟังก์ชันวัดระยะทางยูคลิดีน (Euclidean distance function)

5.1.7.1 ชุดข้อมูลทิคแทคโท (Tic-Tac-Toe Dataset)

ชุดข้อมูลนี้เป็นชุดข้อมูลขนาดที่เก็บข้อมูลตัวอย่างของตำแหน่งของเกมส์ทิคแทคโทเพื่อใช้ในทำนายว่าเมื่อจบเกมส์จะแพ้หรือชนะ ชุดข้อมูลนี้ได้มาจากแหล่งรวมชุดข้อมูลของ UCI Machine Learning Repository [18]

ชุดข้อมูลนี้มีลักษณะเฉพาะ(Feature) 9 มิติ (แอททริบิว) ซึ่งแต่ละมิติก็คือตำแหน่งช่องสี่เหลี่ยมในเกมส์นั่นเอง ค่าที่เป็นไปได้ในแต่ละมิติคือ x, o, b (blank)

ชุดข้อมูลนี้มีขนาด 958 ออบเจ็ค

ชุดข้อมูลนี้มีชนิดที่เป็นไปได้ทั้งหมด 2 ชนิด (แพ้หรือชนะ)

5.1.7.2 ผลการทดลองที่ได้จากการใช้วิธีการ Condensed Nearest Neighbor Rule (CNN)

วิธีการนี้สามารถเลือกหาเซ็ตของโปรโตไทป์ของชุดข้อมูลทิคแทคโท ได้ผลลัพท์เป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่มีขนาด 135 จากชุดข้อมูลทิคแทคโทที่มีขนาด 958 โดยใช้เวลาในการทำงานประมาณ 0.515 วินาที

5.1.7.3 ผลการทดลองที่ได้จากการใช้วิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยการใช้ Genetic Algorithm (GA)

การทดลองของวิธีการนี้มีการกำหนดค่าคงที่อ้างอิงต่างๆ ไว้ดังนี้

- จำนวนประชากร (Population Size) $N_{pop} = 40$
- จำนวนรอบของเจนเนอเรชัน(Generation Number) $M_{gen} = 200$
- สัมประสิทธ์ถ่วงน้ำหนัก (weighted coefficient) $\alpha = 1/N\_OriginalDataset = 1/958 = 0.001$
- ค่าความน่าจะเป็นที่แต่ละบิทโครโมโซมจะมีค่าเป็น 1 ในขั้นตอนการสุ่มสร้างโครโมโซมตั้งต้น $P_{ini} = 0.7$
- ค่าความน่าจะเป็นที่จะเกิดการกลายพันธ์ของแต่ละบิทโครโมโซม (Mutation Rate) $P_m = 0.015$

เนื่องจากการทำงานของวิธีการเลือกใช้เลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยการใช้ Genetic Algorithm เป็นวิธีการเชิงเฟ้นสุ่ม ดังนั้นผลลัพท์ของแต่ละรอบของการทำงานจึงไม่แน่นอนเสมอไป ดังนั้นเราจึงทำการเก็บผลการทดลองจาก แต่ละรอบของการทำงานเป็นจำนวน 10 รอบเพื่อหาค่าเฉลี่ย โดยผลการทดลองแต่ละรอบการทำงานมีดังตารางที่ 5.16

ตารางที่ 5.16 แสดงผลการทดลองใช้วิธีการเลือกใช้เลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงโดยการใช้ Genetic Algorithm บนชุดข้อมูลทิดแทคโทเป็นจำนวน 10 รอบ

การทำงานรอบที่	จำนวนโปรโตไทป์	เวลา (วินาที)
1	215	26.765
2	206	26.750
3	228	27.406
4	213	26.812
5	214	26.921
6	218	26.750
7	216	26.765
8	227	27.796
9	212	26.875
10	210	26.765
	Min = 206	
	Max = 228	Average = 26.96
	Average = 215.9	

จากการทำงานตามวิธีการนี้เป็นจำนวน 10 รอบ วิธีการนี้สามารถเลือกหาเซ็ตของโปรโตไทป์ของชุดข้อมูลทิดแทคโทอันนี้

ได้ผลลัพท์ที่ดีที่สุดเป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่มีขนาด 206

ได้ผลลัพท์ที่แย่ที่สุดเป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่มีขนาด 228

ได้ผลลัพท์ที่เฉลี่ยเป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่มีขนาด 215.9

จากชุดข้อมูลทิดแทคโทที่มีจำนวน 958

เวลาในแต่ละรอบการทำงานประมาณ 26.96 วินาที

5.1.7.4 ผลการทดลองที่ได้จากวิธีการเลือกโปรโตไทป์โดยใช้วิธีการจัดกลุ่มข้อมูลแบบมีการสอน (ในที่นี้เรียกย่อว่า"CLUSTER")

วิธีการนี้สามารถเลือกหาเซ็ตของโปรโตไทป์ของชุดข้อมูลทิคแทคโทอันนี้ ได้ผลลัพท์เป็นเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่มีขนาด 85 จากชุดข้อมูลข้อมูลทิคแทคโทที่มีขนาด 958 โดยใช้เวลาในการทำงานประมาณ 0.906 วินาที

5.1.7.5 สรุปผลการทดลองของชุดข้อมูลทิคแทคโท

เราสามารถสรุปรวมผลการทดลองได้โดยแสดงดังตารางที่ 5.17 ดังนี้

ตารางที่ 5.17 แสดงขนาดของเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องของชุดข้อมูลทิคแทคโทที่เลือกได้โดยใช้วิธีการต่างๆพร้อมทั้งเวลาในการทำงานของวิธีการเหล่านั้น

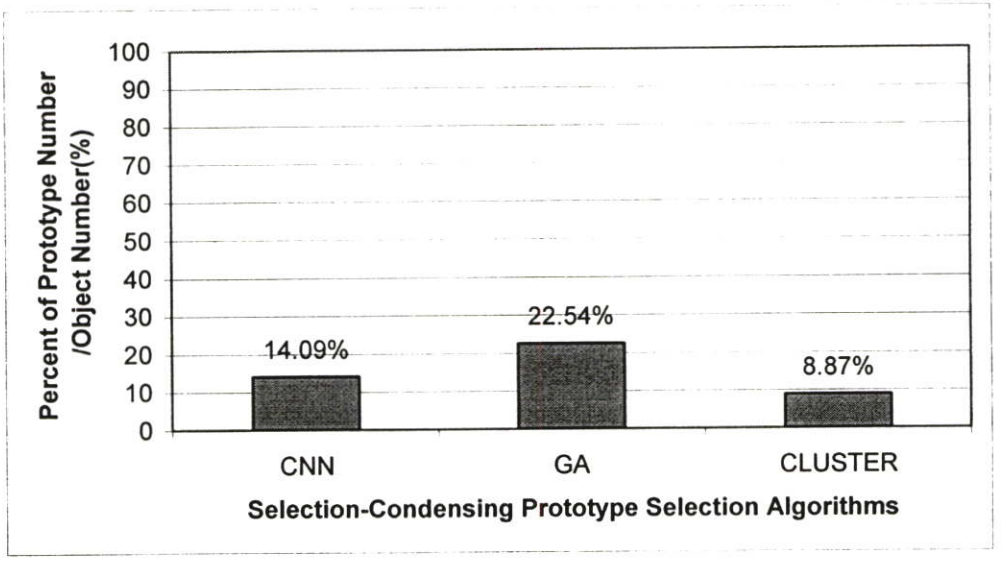
	ชุดข้อมูลทิคแทคโท			
	CNN	**GA**		**CLUSTER**
จำนวนโปรโตไทป์	135	Min	206	85
		Max	228	
		$\bar{X}$	215.9	
เวลา (วินาที)	0.515	26.96		0.906

ข้อมูลทิคแทคโทนี้มีจำนวน 958 ออบเจ็ค เราสามารถคำนวนหาอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่เลือกได้จากแต่ละวิธีต่อขนาดของชุดข้อมูลทิคแทคโท ได้ดังที่แสดงในตารางที่ 5.18

ตารางที่ 5.18 แสดงอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่เลือกได้จากแต่ละวิธีต่อขนาดของชุดข้อมูลทิคแทคโท

	ชุดข้อมูลทิคแทคโท			
	CNN	**GA**		**CLUSTER**
เปอร์เซ็นต์จำนวนโปรโตไทป์ที่เลือกได้ต่อขนาดเซ็ตตั้งต้น (%)	14.09	Min	21.50	8.87
		Max	23.80	
		$\bar{X}$	22.54	

เราสามารถแสดงอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่เลือกได้จากแต่ละวิธีต่อขนาดของชุดข้อมูลทิคแทคโทในรูปแบบกราฟ ซึ่งแสดงไว้ในรูปที่ 5.6



รูปที่ 5.6 กราฟแสดงอัตราเปอร์เซ็นต์ของขนาดเซ็ตของโปรโตไทป์ที่มีคุณสมบัติความสอดคล้องที่เลือกได้จากแต่ละวิธีต่อขนาดของชุดข้อมูลทิคแทคโท

5.1.8 สรุปผลการทดลองเลือกโปรโตไทป์

เราสามารถสรุปผลการทดลองเลือกโปรโตไทป์โดยใช้วิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (Selection-Condensing Prototype Selection Algorithm) บนชุดข้อมูลทั้งหมดได้ดังตารางที่ 5.19 ดังนี้

ตารางที่ 5.19 แสดงผลสรุปเปอร์เซ็นต์ของจำนวนโปรโตไทป์ต่อจำนวนข้อมูลตั้งต้นของชุดข้อมูลที่ได้ทำการทดลองไปทั้งหมด

ชุดข้อมูล	ขนาด	CNN(%)	GA(%)	CLUSTER(%)	Best Algorithm
ชุดข้อมูลถั่วเหลือง	47	14.90	12.10	19.14	GA
ชุดข้อมูลสวนสัตว์	101	14.85	12.77	14.85	GA
ชุดข้อมูลดอกไอริช	150	16.00	9.40	11.33	GA
ชุดข้อมูลไวน์	178	39.89	35.28	35.39	GA & CLUSTER
ชุดข้อมูลอีโคไล	336	36.90	33.21	30.95	CLUSTER
ชุดข้อมูลทิคแทคโท	958	14.09	22.54	8.87	CLUSTER

วิธีการเลือกโปรโตไทป์โดยใช้ Genetic Algorithms มีประสิทธิภาพในการลดจำนวนโปรโตไทป์อ้างอิงที่ดีมากสำหรับชุดข้อมูลที่มีขนาดเล็ก ดังในตารางที่ 5.19 จะเห็นว่าสำหรับข้อมูลที่มีขนาดเล็กตั้งแต่ชุดข้อมูลถั่วเหลืองซึ่งมีสมาชิก 47 ตัว ไปจนถึงชุดข้อมูลไวน์ซึ่งมีสมาชิก 178 ตัว วิธีการเลือกโปรโตไทป์โดยใช้ Genetic Algorithms มีประสิทธิภาพในการลดจำนวนโปรโตไทป์

อ้างอิงสูงที่สุดเมื่อเทียบกับวิธีการอื่น (วิธีการ CNN และวิธีการ CLUSTER) แต่ถ้าหากสังเกตให้ดีแล้วจะพบว่าประสิทธิภาพในการลดจำนวนโปรโตไทป์อ้างอิงของวิธีการเลือกโปรโตไทป์โดยใช้ Genetic Algorithms นั้นจะค่อยๆลดลงเมื่อใช้กับชุดข้อมูลที่มีขนาดใหญ่ขึ้น (สำหรับชุดข้อมูลไวน์ วิธีการ GA มีเปอร์เซ็นต์จำนวนโปรโตไทป์ต่อจำนวนข้อมูลตั้งต้นน้อยกว่ากับวิธีการ CLUSTER เพียง 0.11%) และสำหรับข้อมูลที่มีขนาดใหญ่มากขึ้นวิธีการเลือกโปรโตไทป์โดยใช้ Genetic Algorithms จะมีประสิทธิภาพในการลดจำนวนโปรโตไทป์อ้างอิงในระดับที่ไม่ดีนัก เช่น ในชุดข้อมูลทิคแทคโทวิธีการเลือกโปรโตไทป์โดยใช้ Genetic Algorithms มีเปอร์เซ็นต์จำนวนโปรโตไทป์ต่อจำนวนข้อมูลตั้งต้นถึง 22.54% (มากกว่าสองเท่าของวิธีการ CLUSTER) ซึ่งถือเป็นข้อเสียโดยธรรมชาติของวิธีการ GA ที่เป็นวิธีการค้นคำตอบวงกว้าง (global search algorithm) โดยการเฟ้นสุ่มเพื่อค้นหาคำตอบที่ดีจากปริภูมิที่เป็นไปได้ทั้งหมด เมื่อปริภูมิที่เป็นไปได้ทั้งหมดมีขนาดใหญ่ การค้นหาคำตอบที่ดีนั้นย่อมทำได้โดยยากลำบาก ดังที่ได้กล่าวไปแล้วในหัวข้อ 3.2.2.4

วิธีการ CNN มีประสิทธิภาพในการลดจำนวนโปรโตไทป์อ้างอิงในระดับปานกลาง โดยในชุดข้อมูลที่มีขนาดเล็ก (ชุดข้อมูลถั่วเหลืองจนถึงชุดข้อมูลไวน์) วิธีการ CNN มีประสิทธิภาพในการลดจำนวนโปรโตไทป์อ้างอิงด้อยกว่าวิธีการเลือกโปรโตไทป์โดยใช้ Genetic Algorithms แต่ก็ไม่ได้แตกต่างมากนักยังถือว่าอยู่ในระดับที่ใกล้เคียงกัน ส่วนในชุดข้อมูลที่มีขนาดใหญ่ขึ้น (ชุดข้อมูลอีโคไลและชุดข้อมูลทิคแทคโท) วิธีการ CNN ก็มีประสิทธิภาพในการลดจำนวนโปรโตไทป์อ้างอิงด้อยกว่าวิธีการ CLUSTER แต่ก็ถือว่าอยู่ในระดับที่ใกล้เคียงกัน

วิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน (CLUSTER) มีประสิทธิภาพในการลดจำนวนโปรโตไทป์อ้างอิงในระดับที่ค่อนข้างดีสำหรับข้อมูลขนาดเล็ก ดังในตาราง 5.19 จะเห็นว่าจากชุดข้อมูลสวนสัตว์ถึงชุดข้อมูลไวน์วิธีการ CLUSTER มีประสิทธิภาพในการลดจำนวนโปรโตไทป์อ้างอิงที่ใกล้เคียงกับวิธีการเลือกโปรโตไทป์โดยใช้ Genetic Algorithms และสำหรับชุดข้อมูลที่ขนาดใหญ่ขึ้น วิธีการ CLUSTER จะมีประสิทธิภาพในการลดจำนวนโปรโตไทป์อ้างอิงที่ดีมาก ดังในตาราง 5.19 จะเห็นว่าสำหรับชุดข้อมูลอีโคไลและชุดข้อมูลทิคแทคโท วิธีการ CLUSTER จะมีประสิทธิภาพในการลดจำนวนโปรโตไทป์อ้างอิงที่ดีมากที่สุดเมื่อเทียบวิธีการอื่นๆ และมีแนวโน้มที่ดีขึ้นสำหรับชุดข้อมูลที่มีขนาดใหญ่

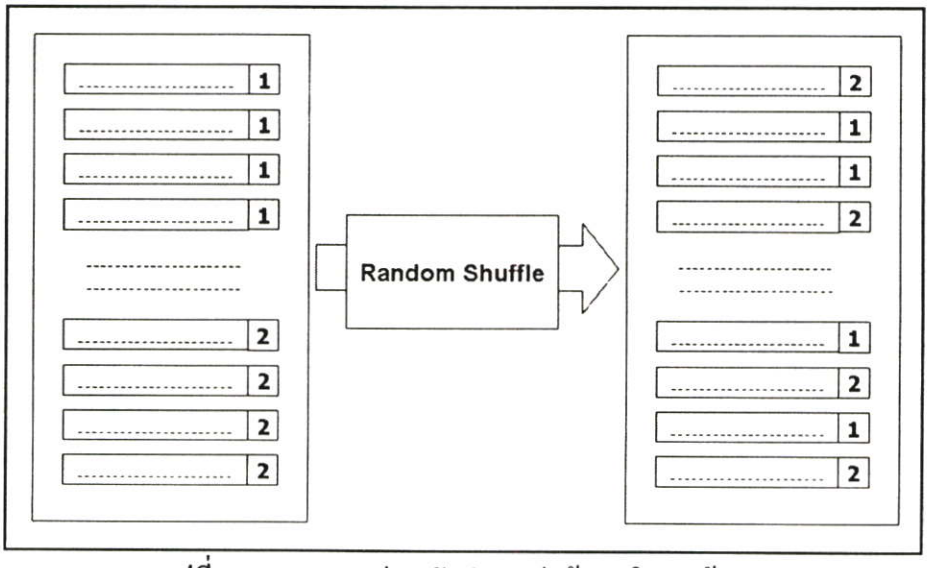
จากการทดลองเลือกโปรโตไทป์จะเห็นว่าวิธีการเลือกโปรโตไทป์โดยการจัดกลุ่มข้อมูลแบบมีการสอน (CLUSTER) มีประสิทธิภาพในการลดจำนวนโปรโตไทป์อ้างอิงอยู่ในระดับที่ดีถึงดีมากสำหรับชุดข้อมูลที่มีขนาดเล็กไปจนถึงชุดข้อมูลที่มีขนาดใหญ่ จึงถือได้ว่าวิธีการเลือกโปรโตไทป์โดยการจัดกลุ่มข้อมูลแบบมีการสอน (CLUSTER) เป็นวิธีการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์อ้างอิงแบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (Selection-Condensing Prototype Selection Algorithm) ที่มีประสิทธิภาพในการลดจำนวนโปรโตไทป์อ้างอิงที่ดีวิธีการหนึ่ง

5.2 การทดลองแยกแยะข้อมูล

หัวข้อนี้นำเสนอการทดลองแยกแยะข้อมูลซึ่งเป็นการทดลองเพื่อประเมินประสิทธิภาพในการแยกแยะข้อมูลของโปรโตไทป์ที่มาได้จากวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มแบบมีการสอน ซึ่งสามารถทำการประเมินได้โดยการเปรียบเทียบระหว่างประสิทธิภาพในการแยกแยะข้อมูลของกฎ nearest neighbor rule ที่ใช้โปรโตไทป์ที่เลือกได้ด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนเป็นโปรโตไทป์อ้างอิงและประสิทธิภาพในการแยกแยะข้อมูลของกฎ nearest neighbor rule ที่ใช้ข้อมูลสำหรับใช้สอนทั้งหมดเป็นโปรโตไทป์อ้างอิงและประสิทธิภาพในการแยกแยะข้อมูลของกฎ nearest neighbor rule ที่ใช้โปรโตไทป์ที่ได้มาจากการคำนวนด้วยวิธีการ learning vector quantization (LVQ) เป็นโปรโตไทป์อ้างอิง โดยสามารถอธิบายรายละเอียดของขั้นตอนการทำการทดลอง ได้ดังนี้

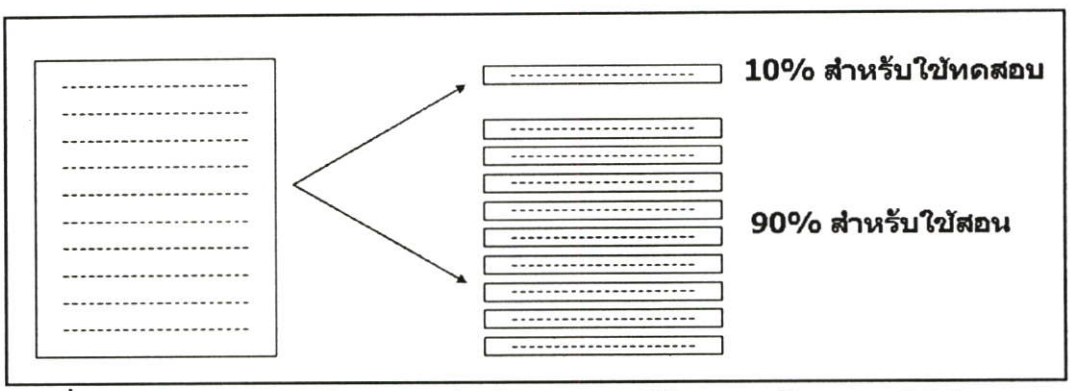
5.2.1 ขั้นตอนของการทดลองแยกแยะข้อมูล

1. นำชุดข้อมูลทั้งหมดมาสุ่มสลับลำดับ (random shuffle) ให้เป็นชุดข้อมูลที่ไม่เรียงลำดับตามชนิดและให้ข้อมูลทั้งหมดในชุดข้อมูลมีการกระจายตัวอย่างสม่ำเสมอ ทั้งนี้ก็เพราะชุดข้อมูลตั้งต้นโดยส่วนใหญ่มักมีการเรียงลำดับข้อมูลตามชนิดไว้อย่างเป็นระเบียบทำให้ข้อมูลในบางช่วงลำดับมีชนิดเดียวกันทั้งหมด ดังเช่นในรูปที่ 5.7 ด้านซ้ายแสดงชุดข้อมูลที่เรียงลำดับข้อมูลตามชนิด เมื่อนำไปสุ่มสลับลำดับแล้วจะได้ชุดข้อมูลที่ไม่เรียงลำดับตามชนิดดังแสดงในด้านขวาของรูปที่ 5.7



รูปที่ 5.7 แสดงการสุ่มสลับตำแหน่งข้อมูลในชุดข้อมูล

2. แบ่งชุดข้อมูลออกเป็น 10 ส่วนเท่าๆกัน (เรียกว่า "10-fold cross validation") โดยให้ข้อมูลแต่ละส่วนเขียนแทนด้วย Data1, Data2, ..., Data10 จากนั้นนำข้อมูล Data1, Data2, ..., Data10 มาแบ่งเป็น 2 ส่วน ส่วนหนึ่งเป็นข้อมูลสำหรับทดสอบและอีกส่วนหนึ่งเป็นข้อมูลสำหรับสอน โดยในการแบ่งครั้งแรกให้ Data1 เป็นข้อมูลสำหรับทดสอบ และให้ข้อมูลอีก 9 ส่วนที่เหลือ (Data2 ถึง Data10) เป็นข้อมูลสำหรับสอน ในการแบ่งครั้งที่สองให้ Data2 เป็นข้อมูลสำหรับทดสอบ และให้ข้อมูลอีก 9 ส่วนที่เหลือ (Data1, Data3 ถึง Data10) เป็นข้อมูลสำหรับสอน เมื่อพิจารณาแบ่งข้อมูลในลักษณะเดียวกันนี้จะสามารถแบ่งชุดข้อมูลตั้งต้นได้เป็นชุดข้อมูลสำหรับใช้ทดสอบและใช้สอนจำนวน 10 ชุด ในรูปที่ 5.8 แสดงตัวอย่างการแบ่งข้อมูลออกเป็น 10 ส่วนเท่าๆกันโดยให้หนึ่งส่วนเป็นข้อมูลสำหรับใช้ทดสอบและให้ส่วนที่เหลืออีกเก้าส่วนเป็นข้อมูลสำหรับใช้สอน



รูปที่ 5.8 แสดงการแบ่งชุดข้อมูลออกเป็นข้อมูลสำหรับใช้สอนและข้อมูลสำหรับใช้ทดสอบ

3. นำชุดข้อมูลสำหรับใช้ทดสอบและใช้สอนทั้ง 10 ชุด ใช้ในการทดสอบประสิทธิภาพในการแยกแยะข้อมูลของโปรโตไทป์ที่ได้มาจากแต่ละวิธี โดยมี 3 วิธีที่นำมาเปรียบเทียบกันดังต่อไปนี้ (ในที่นี้กำหนดให้กฎ nearest neighbor rule ใช้ฟังก์ชันยูคลิดีนดิสแทน(Euclidean distance function) เป็นฟังก์ชันวัดระยะทาง เพื่อการแยกแยะข้อมูล)
   - **วิธีที่ 1** ใช้ข้อมูลสำหรับใช้สอนทั้งหมดเป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule แล้วใช้กฎ nearest neighbor rule นี้ไปแยกแยะข้อมูลสำหรับใช้ทดสอบ แล้วบันทึกเปอร์เซ็นต์จำนวนข้อมูลที่แยกแยะได้ถูกต้องต่อจำนวนข้อมูลที่ใช้ทดสอบทั้งหมด
   - **วิธีที่ 2** เลือกโปรโตไทป์ของข้อมูลสำหรับใช้สอนด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน แล้วใช้โปรโตไทป์ที่เลือกได้เป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule จากนั้นใช้กฎ nearest neighbor rule นี้แยกแยะข้อมูลสำหรับใช้ทดสอบ แล้วบันทึกเปอร์เซ็นต์จำนวนข้อมูลที่แยกแยะได้ถูกต้องต่อจำนวนข้อมูลที่ใช้ทดสอบทั้งหมด

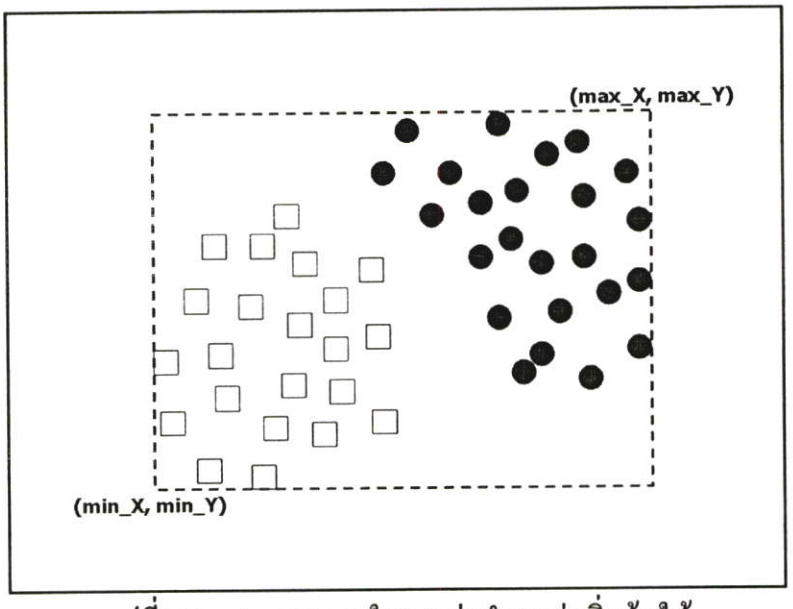
- **วิธีที่ 3** คำนวนหาตำแหน่งของโปรโตไทป์ของข้อมูลสำหรับใช้สอนด้วยวิธีการ learning vector quantization (LVQ) โดยกำหนดให้มีจำนวนโปรโตไทป์เท่ากับจำนวนโปรโตไทป์ที่หาได้ด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน แล้วใช้โปรโตไทป์ที่คำนวนได้เป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule จากนั้นใช้กฎ nearest neighbor rule นี้แยกแยะข้อมูลสำหรับใช้ทดสอบ แล้วบันทึกเปอร์เซ็นต์จำนวนข้อมูลที่แยกแยะได้ถูกต้องต่อจำนวนข้อมูลที่ใช้ทดสอบทั้งหมด

5.2.2 การกำหนดค่าพารามิเตอร์ของวิธีการ Learning Vector Quantization

วิธีการ learning vector quantization นั้นจำเป็นต้องมีการกำหนดค่าพารามิเตอร์ในการทำงาน ซึ่งโปรแกรมของวิธีการ learning vector quantization ที่ได้เขียนขึ้นมาเพื่อใช้ในการทดลองมีการกำหนดพารามิเตอร์ ดังต่อไปนี้

- กำหนดจำนวน codebook (โปรโตไทป์) ให้มีจำนวนเท่ากับจำนวนโปรโตไทป์ที่เลือกได้ด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน ทั้งนี้ก็เพราะว่าต้องการเปรียบเทียบความสามารถในการแยกแยะข้อมูลของโปรโตไทป์ที่ได้จากได้ทั้งสองวิธี ดังนั้นจึงต้องกำหนดให้โปรโตไทป์มีจำนวนเท่ากัน เช่น ถ้าหากวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนเลือกโปรโตไทป์ได้จำนวน 12 ตัว โดยโปรโตไทป์ 3 ตัวเป็นชนิด A และโปรโตไทป์ 6 ตัวเป็นชนิด B และโปรโตไทป์ 3 ตัวเป็นชนิด C ดังนั้นสามารถกำหนดให้วิธีการ learning vector quantization มี codebook 12 ตัว โดย codebook 3 ตัวเป็นชนิด A และ codebook 6 ตัวเป็นชนิด B และ codebook 3 ตัวเป็นชนิด C

- กำหนดตำแหน่งเริ่มต้นให้กับ codebook แต่ละตัวโดยการสุ่มให้มีตำแหน่งอยู่ภายในบริเวณที่เป็นไปได้ ดังรูปที่ 5.9 แสดงตำแหน่งของข้อมูลที่มีอยู่ทั้งหมด (ในที่นี้เป็นข้อมูล 2 มิติ) จะเห็นว่าตำแหน่งของข้อมูลจะอยู่ในบริเวณกรอบสี่เหลี่ยมเส้นประเท่านั้นดังนั้น การสุ่มตำแหน่งเริ่มต้นให้กับ codebook จะสุ่มให้อยู่ในบริเวณกรอบสี่เหลี่ยมเส้นประนี้เท่านั้น ซึ่งหมายความว่าการสุ่มค่าจะอยู่ในช่วงค่าน้อยที่สุดถึงมากที่สุดของแต่ละมิติเท่านั้น ซึ่งในรูป 5.9 การสุ่มในมิติ x จะอยู่ในช่วงที่ไม่น้อยกว่า min_X และไม่มากไปกว่า max_X และการสุ่มในมิติ y จะอยู่ในช่วงที่ไม่น้อยกว่า min_Y และไม่มากไปกว่า max_Y



รูปที่ 5.9 แสดงขอบเขตในการสุ่มตำแหน่งเริ่มต้นให้

- การกำหนดเงื่อนไขสิ้นสุดการทำงานของวิธี learning vector quantization ในการทดลองนี้กำหนดเงื่อนไขสิ้นสุดการทำงานด้วยจำนวนรอบ หมายความว่าเมื่อทำงานครบตามจำนวนรอบที่ได้กำหนดไว้ให้สิ้นสุดการทำงาน โดยในการทดลองนี้กำหนดให้จำนวนรอบการทำงานมีค่าเท่ากับ 100 เท่าของจำนวน codebook เช่น ถ้ามี codebook 15 ตัวจะกำหนดให้มีรอบการทำงานจำนวน 1500 รอบ เป็นต้น

- กำหนดให้ค่าอัตราการเรียนรู้ (learning rate: α) มีค่าเริ่มต้นที่ 0.1 เสมอ ($\alpha(0) = 0.1$) และกำหนดการลดค่าอัตราการเรียนรู้ (learning rate: α) เป็นแบบเชิงเส้น หมายความว่าค่าอัตราการเรียนรู้จะลดลงในปริมาณที่เท่ากันในทุกรอบการทำงานจนมีค่าเป็นศูนย์เมื่อครบจำนวนรอบการทำงานที่ได้กำหนดไว้ ซึ่งสามารถแสดงได้ดังนี้

$$\alpha(t) = \alpha(0) * \frac{(ITER - t)}{ITER} = 0.1 * \frac{(ITER - t)}{ITER} \tag{5.2}$$

โดย

t คือ รอบการทำงาน

$\alpha(t)$ คือ ค่าอัตราการเรียนรู้ในรอบการทำงานที่ t (ในรอบแรก $t = 0$ กำหนดให้ $\alpha(0) = 0.1$)

$ITER$ คือ จำนวนรอบการทำงานก่อนสิ้นสุดการทำงาน (ซึ่งในที่นี้กำหนดให้มีค่าเป็น 100 เท่าของจำนวน codebook)

5.2.3 ผลการทดลองแยกแยะข้อมูล

ในหัวข้อนี้แสดงผลของการทดลองเพื่อประเมินประสิทธิภาพในการแยกแยะข้อมูล โดยใช้ชุดข้อมูล 4 ชุดข้อมูล คือ ชุดข้อมูลดอกไอริช ชุดข้อมูลไวน์ ชุดข้อมูลอีโคไล และชุดข้อมูลทิคแทคโท ในการทดลอง ซึ่งสามารถแสดงผลการทดลองได้ดังต่อไปนี้

5.2.3.1 ผลการทดลองแยกแยะข้อมูลกับชุดข้อมูลดอกไอริช

ชุดข้อมูลดอกไอริชมีลักษณะเฉพาะ(Feature) 4 มิติ (แอททริบิว) และมีชนิดทั้งหมด 3 ชนิด และมีขนาด 150 ออบเจ็ค ทำการเตรียมข้อมูลโดยการแบ่งชุดข้อมูลดอกไอริชเป็นชุดข้อมูลสำหรับใช้สอนและสำหรับใช้ทดสอบจำนวน10 ชุด โดยแต่ละชุดข้อมูลจะมีข้อมูลสำหรับใช้ทดสอบจำนวน 15 ออบเจ็คและมีข้อมูลสำหรับใช้สอนจำนวน 135 ออบเจ็ค ซึ่งสามารถแสดงได้ดังตารางที่ 5.20

ตารางที่ 5.20 แสดงการแบ่งชุดข้อมูลดอกไอริชออกเป็นข้อมูลสำหรับใช้สอนและข้อมูลสำหรับใช้ทดสอบจำนวน 10 ชุด

ตารางการเตรียมชุดข้อมูลสำหรับใช้สอนและใช้ทดสอบ										
ข้อมูลชุดที่	1	2	3	4	5	6	7	8	9	10
ข้อมูลสำหรับสอน	135	135	135	135	135	135	135	135	135	135
ข้อมูลสำหรับทดสอบ	15	15	15	15	15	15	15	15	15	15

ผลการทดลองใช้ข้อมูลสำหรับใช้สอนทั้งหมดเป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule แล้วใช้กฎ nearest neighbor rule แยกแยะข้อมูลสำหรับใช้ทดสอบ สามารถแสดงได้ดังตารางที่ 5.21 ซึ่งแสดงให้เห็นว่าถ้าใช้โปรโตไทป์อ้างอิงเฉลี่ย 135 ตัวจะสามารถแยกแยะข้อมูลสำหรับใช้ทดสอบได้อย่างถูกต้องโดยเฉลี่ยประมาณ 96 เปอร์เซ็นต์

ตารางที่ 5.21 แสดงผลการทดลองใช้ข้อมูลสำหรับใช้สอนเป็นโปรโตไทป์อ้างอิง

ผลการทดลองใช้ข้อมูลสำหรับใช้สอนเป็นโปรโตไทป์อ้างอิง											
ข้อมูลชุดที่	1	2	3	4	5	6	7	8	9	10	เฉลี่ย
โปรโตไทป์อ้างอิง	135	135	135	135	135	135	135	135	135	135	135
เปอร์เซ็นต์การแยกแยะข้อมูลถูกต้อง	100	100	93.33	100	100	93.33	100	93.33	86.67	93.33	95.99

ผลการทดลองใช้โปรโตไทป์ของข้อมูลสำหรับใช้สอนเป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule (ซึ่งโปรโตไทป์ได้มาจากการวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน) แล้วใช้กฎ nearest neighbor rule นี้แยกแยะข้อมูลสำหรับใช้ทดสอบ สามารถแสดงผลการทดลองนี้ได้ดังตารางที่ 5.22 ซึ่งจะเห็นได้ว่าถ้าใช้โปรโตไทป์อ้างอิงที่เลือกด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนซึ่งมีจำนวนเฉลี่ย 15.2 ตัว แล้วจะสามารถแยกแยะข้อมูลสำหรับใช้ทดสอบได้อย่างถูกต้องโดยเฉลี่ยประมาณ 92.66 เปอร์เซ็นต์

ตารางที่ 5.22 แสดงผลการทดลองใช้โปรโตไทป์ที่เลือกด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนเป็นโปรโตไทป์อ้างอิง

ผลการทดลองใช้โปรโตไทป์ที่เลือกด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนเป็นโปรโตไทป์อ้างอิง											
ข้อมูลชุดที่	1	2	3	4	5	6	7	8	9	10	เฉลี่ย
โปรโตไทป์อ้างอิง	16	16	15	16	17	17	14	15	12	14	15.2
เปอร์เซ็นต์การแยกแยะข้อมูลถูกต้อง	100	100	93.33	100	93.33	80	93.33	93.33	86.67	86.67	92.66

ผลการทดลองใช้โปรโตไทป์ของข้อมูลสำหรับใช้สอนเป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule โดยที่โปรโตไทป์ของข้อมูลสำหรับใช้สอนได้มาจากการคำนวนด้วยวิธีการ learning vector quantization ซึ่งกำหนดให้มีจำนวน codebook (โปรโตไทป์) เท่ากับจำนวนโปรโตไทป์ที่หาได้ด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน(ในตารางที่ 5.22) เพื่อให้สามารถเปรียบเทียบกันได้ แล้วใช้กฎ nearest neighbor rule นี้แยกแยะข้อมูลสำหรับใช้ทดสอบ สามารถแสดงผลการทดลองได้ดังตารางที่ 5.23 ซึ่งจะเห็นได้ว่าถ้าใช้โปรโตไทป์อ้างอิงที่ได้จากการคำนวนด้วยวิธีการ learning vector quantization จำนวนเฉลี่ย 15.2 ตัว แล้วจะสามารถแยกแยะข้อมูลสำหรับใช้ทดสอบได้อย่างถูกต้องโดยเฉลี่ยประมาณ 92.66 เปอร์เซ็นต์

ตารางที่ 5.23 แสดงผลการทดลองใช้โปรโตไทป์ที่คำนวนด้วยวิธีการ learning vector quantization เป็นโปรโตไทป์อ้างอิง

ผลการทดลองใช้โปรโตไทป์ที่คำนวนด้วยวิธีการ Learning Vector Quantization เป็นโปรโตไทป์อ้างอิง											
ข้อมูลชุดที่	1	2	3	4	5	6	7	8	9	10	เฉลี่ย
โปรโตไทป์อ้างอิง	16	16	15	16	17	17	14	15	12	14	15.2
เปอร์เซ็นต์การแยกแยะข้อมูลถูกต้อง	100	86.67	93.33	100	93.33	93.33	93.33	86.67	86.67	93.33	92.66

5.2.3.2 สรุปผลการทดลองแยกแยะข้อมูลกับชุดข้อมูลดอกไอริช

ตารางที่ 5.24 แสดงการสรุปผลการทดลองเพื่อประเมินประสิทธิภาพในการแยกแยะข้อมูลกับชุดข้อมูลดอกไอริช การใช้ข้อมูลที่ใช้สอนทั้งหมดเป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule ซึ่งทำให้มีโปรโตไทป์อ้างอิงเป็นจำนวน 135 ตัวและมีประสิทธิภาพในการแยกแยะข้อมูลอยู่ที่ 95.99 เปอร์เซ็นต์ ส่วนการใช้โปรโตไทป์ที่เลือกได้ด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน (CLUSTER) เป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule สามารถลดจำนวนโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule เหลือเพียงเฉลี่ย 15.2 ตัว (เหลือประมาณ 11% ของจำนวนข้อมูลที่ใช้สอน) โดยที่ประสิทธิภาพในการแยกแยะข้อมูลลดลงมาอยู่ที่ 92.66 เปอร์เซ็นต์ ซึ่งมีการลดลงเพียง 3.33 เปอร์เซ็นต์เท่านั้น และเมื่อเปรียบเทียบการใช้โปรโตไทป์ที่ได้จากการคำนวนด้วยวิธีการ learning vector quantization (LVQ) เป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule โดยกำหนดให้มีจำนวนโปรโตไทป์เท่ากันกับโปรโตไทป์ที่เลือกได้ด้วยวิธีการจัดกลุ่มข้อมูลแบบมีการสอนนั้นคือเฉลี่ย 15.2 ตัว จะเห็นได้ว่าประสิทธิภาพในการแยกแยะข้อมูลของโปรโตไทป์ที่ได้จากทั้งสองวิธีมีค่าเท่ากัน ดังนั้นโปรโตไทป์ที่เลือกได้โดยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน (CLUSTER) จึงมีประสิทธิภาพในการแยกแยะข้อมูลอยู่ในระดับที่ดี

ตารางที่ 5.24 สรุปผลการทดลองแยกแยะข้อมูลกับชุดข้อมูลดอกไอริช

สรุปผลการทดลองแยกแยะข้อมูลกับชุดข้อมูลดอกไอริช			
	Train Dataset	**CLUSTER**	**LVQ**
โปรโตไทป์อ้างอิง	135	15.2	15.2
เปอร์เซ็นต์การแยกแยะข้อมูลถูกต้อง	95.99	92.66	92.66

5.2.3.3 ผลการทดลองแยกแยะข้อมูลกับชุดข้อมูลไวน์

ชุดข้อมูลไวน์มีลักษณะเฉพาะ(Feature) 13 มิติ มีชนิดทั้งหมด 3 ชนิด และมีขนาด 178 ออบเจ็ค ทำการเตรียมข้อมูลโดยแบ่งชุดข้อมูลไวน์เป็นชุดข้อมูลสำหรับใช้สอนและสำหรับใช้ทดสอบจำนวน10 ชุด โดยแต่ละชุดข้อมูลจะมีข้อมูลสำหรับใช้ทดสอบจำนวน 18 ออบเจ็คและมีข้อมูลสำหรับใช้สอนจำนวน 160 ออบเจ็ค นอกจากสองชุดข้อมูลสุดท้ายจะมีข้อมูลสำหรับใช้ทดสอบจำนวน 17 ออบเจ็คและมีข้อมูลสำหรับใช้สอนจำนวน 161 ออบเจ็ค ซึ่งสามารถแสดงได้ดังตารางที่ 5.25

ตารางที่ 5.25 แสดงการแบ่งชุดข้อมูลไวน์ออกเป็นข้อมูลสำหรับใช้สอนและข้อมูลสำหรับใช้ทดสอบจำนวน 10 ชุด

ตารางการเตรียมชุดข้อมูลสำหรับใช้สอนและใช้ทดสอบ										
ข้อมูลชุดที่	1	2	3	4	5	6	7	8	9	10
ข้อมูลสำหรับสอน	160	160	160	160	160	160	160	160	161	161
ข้อมูลสำหรับทดสอบ	18	18	18	18	18	18	18	18	17	17

ผลการทดลองใช้ข้อมูลสำหรับใช้สอนทั้งหมดเป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule แล้วใช้กฎ nearest neighbor rule แยกแยะข้อมูลสำหรับใช้ทดสอบ สามารถแสดงได้ดังตารางที่ 5.26 ซึ่งแสดงให้เห็นว่าถ้าใช้โปรโตไทป์อ้างอิงเฉลี่ย 160.2 ตัวจะสามารถแยกแยะข้อมูลสำหรับใช้ทดสอบได้อย่างถูกต้องโดยเฉลี่ยประมาณ 75.23 เปอร์เซ็นต์

ตารางที่ 5.26 แสดงผลการทดลองใช้ข้อมูลสำหรับใช้สอนเป็นโปรโตไทป์อ้างอิง

ผลการทดลองใช้ข้อมูลสำหรับใช้สอนเป็นโปรโตไทป์อ้างอิง											
ข้อมูลชุดที่	1	2	3	4	5	6	7	8	9	10	เฉลี่ย
โปรโตไทป์อ้างอิง	160	160	160	160	160	160	160	160	161	161	160.2
เปอร์เซ็นต์การแยกแยะข้อมูลถูกต้อง	72.22	83.33	88.89	83.33	61.11	61.11	83.33	77.78	88.24	52.94	75.23

ผลการทดลองใช้โปรโตไทป์ของข้อมูลสำหรับใช้สอนเป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule (ซึ่งโปรโตไทป์ได้มาจากการวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน) แล้วใช้กฎ nearest neighbor rule นี้แยกแยะข้อมูลสำหรับใช้ทดสอบสามารถแสดงผลการทดลองนี้ได้ดังตารางที่ 5.27 ซึ่งจะเห็นได้ว่าถ้าใช้โปรโตไทป์อ้างอิงที่เลือกด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนซึ่งมีจำนวนเฉลี่ย 58 ตัว แล้วจะสามารถแยกแยะข้อมูลสำหรับใช้ทดสอบได้อย่างถูกต้องโดยเฉลี่ยประมาณ 74.11 เปอร์เซ็นต์

ตารางที่ 5.27 แสดงผลการทดลองใช้โปรโตไทป์ที่เลือกด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนเป็นโปรโตไทป์อ้างอิง

ผลการทดลองใช้โปรโตไทป์ที่เลือกด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนเป็นโปรโตไทป์อ้างอิง											
ข้อมูลชุดที่	1	2	3	4	5	6	7	8	9	10	เฉลี่ย
โปรโตไทป์อ้างอิง	61	60	57	59	54	56	58	59	61	55	58
เปอร์เซ็นต์การแยกแยะข้อมูลถูกต้อง	72.22	83.33	88.89	88.89	55.56	61.11	77.78	72.22	88.24	52.94	74.11

ผลการทดลองใช้โปรโตไทป์ของข้อมูลสำหรับใช้สอนเป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule โดยที่โปรโตไทป์ของข้อมูลสำหรับใช้สอนได้มาจากการคำนวนด้วยวิธีการ learning vector quantization ซึ่งกำหนดให้มีจำนวน codebook (โปรโตไทป์) เท่ากับจำนวนโปรโตไทป์ที่หาได้ด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน(ในตารางที่ 5.27)

เพื่อให้สามารถเปรียบเทียบกันได้ แล้วใช้กฎ nearest neighbor rule นี้แยกแยะข้อมูลสำหรับใช้ทดสอบ สามารถแสดงผลการทดลองได้ดังตารางที่ 5.28 ซึ่งจะเห็นได้ว่าถ้าใช้โปรโตไทป์อ้างอิงที่ได้จากการคำนวนด้วยวิธีการ learning vector quantization จำนวนเฉลี่ย 15.2 ตัว แล้วจะสามารถแยกแยะข้อมูลสำหรับใช้ทดสอบได้อย่างถูกต้องโดยเฉลี่ยประมาณ 61.77 เปอร์เซ็นต์

ตารางที่ 5.28 แสดงผลการทดลองใช้โปรโตไทป์ที่คำนวนด้วยวิธีการ learning vector quantization เป็นโปรโตไทป์อ้างอิง

ผลการทดลองใช้โปรโตไทป์ที่คำนวนด้วยวิธีการ Learning Vector Quantization เป็นโปรโตไทป์อ้างอิง											
ข้อมูลชุดที่	1	2	3	4	5	6	7	8	9	10	เฉลี่ย
โปรโตไทป์อ้างอิง	61	60	57	59	54	56	58	59	61	55	58
เปอร์เซ็นต์การแยกแยะข้อมูลถูกต้อง	72.22	61.11	50	55.56	61.11	61.11	55.56	83.33	76.47	41.18	61.77

5.2.3.4 สรุปผลการทดลองแยกแยะข้อมูลกับชุดข้อมูลไวน์

ตารางที่ 5.29 แสดงการสรุปผลการทดลองเพื่อประเมินประสิทธิภาพในการแยกแยะข้อมูลกับชุดข้อมูลไวน์ การใช้ข้อมูลที่ใช้สอนทั้งหมดเป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule ซึ่งทำให้มีโปรโตไทป์อ้างอิงเป็นจำนวนเฉลี่ย 160.2 ตัว และมีประสิทธิภาพในการแยกแยะข้อมูลอยู่ที่ 75.23 เปอร์เซ็นต์ ส่วนการใช้โปรโตไทป์ที่เลือกได้ด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน (CLUSTER) เป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule สามารถลดจำนวนโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule เหลือเฉลี่ย 58 ตัว (เหลือประมาณ 37% ของจำนวนข้อมูลที่ใช้สอน) โดยที่ประสิทธิภาพในการแยกแยะข้อมูลลดลงมาอยู่ที่ 74.12 เปอร์เซ็นต์ ซึ่งมีการลดลงเพียง 1.11 เปอร์เซ็นต์เท่านั้น และเมื่อเปรียบเทียบการใช้โปรโตไทป์ที่ได้จากการคำนวนด้วยวิธีการ learning vector quantization (LVQ) เป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule โดยกำหนดให้มีจำนวนโปรโตไทป์เท่ากันกับโปรโตไทป์ที่เลือกได้ด้วยวิธีการจัดกลุ่มข้อมูลแบบมีการสอนนั้นคือเฉลี่ย 160.2 ตัว จะเห็นได้ว่าประสิทธิภาพในการแยกแยะข้อมูลของโปรโตไทป์ที่ได้จากวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน (CLUSTER) ค่อนข้างสูงกว่าประสิทธิภาพในการแยกแยะข้อมูลของโปรโตไทป์ที่ได้จากการคำนวนด้วยวิธีการ leaning vector quantization (LVQ) ดังนั้นโปรโตไทป์ที่เลือกได้โดยวิธีการเลือก

โปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน (CLUSTER) จึงมีประสิทธิภาพในการแยกแยะข้อมูลที่อยู่ในระดับที่ดี

ตารางที่ 5.29 สรุปผลการทดลองแยกแยะข้อมูลกับชุดข้อมูลไวน์

สรุปผลการทดลองแยกแยะข้อมูลกับชุดข้อมูลไวน์			
	Train Dataset	**CLUSTER**	**LVQ**
โปรโตไทป์อ้างอิง	160.2	58	58
เปอร์เซ็นต์การแยกแยะข้อมูลถูกต้อง	75.23	74.12	61.77

5.2.3.5 ผลการทดลองแยกแยะข้อมูลกับชุดข้อมูลอีโคไล

ชุดข้อมูลอีโคไลมีลักษณะเฉพาะ(Feature) 7 มิติ มีชนิดทั้งหมด 8 ชนิด และมีขนาด 336 ออบเจ็ค ทำการเตรียมข้อมูลโดยแบ่งชุดข้อมูลไวน์เป็นชุดข้อมูลสำหรับใช้สอนและสำหรับใช้ทดสอบจำนวน10 ชุด โดยแต่ละชุดข้อมูลจะมีข้อมูลสำหรับใช้ทดสอบจำนวน 34 ออบเจ็คและมีข้อมูลสำหรับใช้สอนจำนวน 302 ออบเจ็ค นอกจากสี่ชุดข้อมูลสุดท้ายจะมีข้อมูลสำหรับใช้ทดสอบจำนวน 33 ออบเจ็คและมีข้อมูลสำหรับใช้สอนจำนวน 303 ออบเจ็ค ซึ่งสามารถแสดงได้ดังตารางที่ 5.30

ตารางที่ 5.30 แสดงการแบ่งชุดข้อมูลอีโคไลออกเป็นข้อมูลสำหรับใช้สอนและข้อมูลสำหรับใช้ทดสอบจำนวน 10 ชุด

ตารางการเตรียมชุดข้อมูลสำหรับใช้สอนและใช้ทดสอบ										
ข้อมูลชุดที่	1	2	3	4	5	6	7	8	9	10
ข้อมูลสำหรับสอน	302	302	302	302	302	302	303	303	303	303
ข้อมูลสำหรับทดสอบ	34	34	34	34	34	34	33	33	33	33

ผลการทดลองใช้ข้อมูลสำหรับใช้สอนทั้งหมดเป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule แล้วใช้กฎ nearest neighbor rule แยกแยะข้อมูลสำหรับใช้ทดสอบ สามารถแสดงได้ดังตารางที่ 5.31 ซึ่งแสดงให้เห็นว่าถ้าใช้โปรโตไทป์อ้างอิงเฉลี่ย 302.4 ตัวจะสามารถแยกแยะข้อมูลสำหรับใช้ทดสอบได้อย่างถูกต้องโดยเฉลี่ยประมาณ 80.33 เปอร์เซ็นต์

ตารางที่ 5.31 แสดงผลการทดลองใช้ข้อมูลสำหรับใช้สอนเป็นโปรโตไทป์อ้างอิง

ผลการทดลองใช้ข้อมูลสำหรับใช้สอนเป็นโปรโตไทป์อ้างอิง											
ข้อมูลชุดที่	1	2	3	4	5	6	7	8	9	10	เฉลี่ย
โปรโตไทป์อ้างอิง	302	302	302	302	302	302	303	303	303	303	302.4
เปอร์เซ็นต์การแยกแยะข้อมูลถูกต้อง	85.29	85.29	82.35	79.41	76.47	79.41	75.76	81.81	81.81	75.76	80.33

ผลการทดลองใช้โปรโตไทป์ของข้อมูลสำหรับใช้สอนเป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule (ซึ่งโปรโตไทป์ได้มาจากการวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน) แล้วใช้กฎ nearest neighbor rule นี้แยกแยะข้อมูลสำหรับใช้ทดสอบสามารถแสดงผลการทดลองนี้ได้ดังตารางที่ 5.32 ซึ่งจะเห็นได้ว่าถ้าใช้โปรโตไทป์อ้างอิงที่เลือกด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนซึ่งมีจำนวนเฉลี่ย 93.5 ตัว แล้วจะสามารถแยกแยะข้อมูลสำหรับใช้ทดสอบได้อย่างถูกต้องโดยเฉลี่ยประมาณ 74.73 เปอร์เซ็นต์

ตารางที่ 5.32 แสดงผลการทดลองใช้โปรโตไทป์ที่เลือกด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนเป็นโปรโตไทป์อ้างอิง

ผลการทดลองใช้โปรโตไทป์ที่เลือกด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนเป็นโปรโตไทป์อ้างอิง											
ข้อมูลชุดที่	1	2	3	4	5	6	7	8	9	10	เฉลี่ย
โปรโตไทป์อ้างอิง	94	98	91	95	93	89	94	89	93	99	93.5
เปอร์เซ็นต์การแยกแยะข้อมูลถูกต้อง	70.59	76.47	76.47	64.71	73.53	79.41	75.76	72.73	78.79	78.79	74.73

ผลการทดลองใช้โปรโตไทป์ของข้อมูลสำหรับใช้สอนเป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule โดยที่โปรโตไทป์ของข้อมูลสำหรับใช้สอนได้มาจากการคำนวนด้วยวิธีการ learning vector quantization ซึ่งกำหนดให้มีจำนวน codebook (โปรโตไทป์) เท่ากับจำนวนโปรโตไทป์ที่หาได้ด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน(ในตารางที่ 5.32)

เพื่อให้สามารถเปรียบเทียบกันได้ แล้วใช้กฎ nearest neighbor rule นี้แยกแยะข้อมูลสำหรับใช้ทดสอบ สามารถแสดงผลการทดลองได้ดังตารางที่ 5.33 ซึ่งจะเห็นได้ว่าถ้าใช้โปรโตไทป์อ้างอิงที่ได้จากการคำนวนด้วยวิธีการ learning vector quantization จำนวนเฉลี่ย 93.5 ตัว แล้วจะสามารถแยกแยะข้อมูลสำหรับใช้ทดสอบได้อย่างถูกต้องโดยเฉลี่ยประมาณ 81.88 เปอร์เซ็นต์

ตารางที่ 5.33 แสดงผลการทดลองใช้โปรโตไทป์ที่คำนวนด้วยวิธีการ learning vector quantization เป็นโปรโตไทป์อ้างอิง

ผลการทดลองใช้โปรโตไทป์ที่คำนวนด้วยวิธีการ Learning Vector Quantization เป็นโปรโตไทป์อ้างอิง											
ข้อมูลชุดที่	1	2	3	4	5	6	7	8	9	10	เฉลี่ย
โปรโตไทป์อ้างอิง	94	98	91	95	93	89	94	89	93	99	93.5
เปอร์เซ็นต์การแยกแยะข้อมูลถูกต้อง	82.35	88.24	79.41	73.53	73.53	82.35	87.87	84.84	84.84	81.81	81.88

5.2.3.6 สรุปผลการทดลองแยกแยะข้อมูลกับชุดข้อมูลอีโคไล

ตารางที่ 5.34 แสดงการสรุปผลการทดลองเพื่อประเมินประสิทธิภาพในการแยกแยะข้อมูลกับชุดข้อมูลอีโคไล การใช้ข้อมูลที่ใช้สอนทั้งหมดเป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule ซึ่งทำให้มีโปรโตไทป์อ้างอิงเป็นจำนวนเฉลี่ย 302.4 ตัว และมีประสิทธิภาพในการแยกแยะข้อมูลอยู่ที่ 80.33 เปอร์เซ็นต์ ส่วนการใช้โปรโตไทป์ที่เลือกได้ด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน (CLUSTER) เป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule สามารถลดจำนวนโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule เหลือเฉลี่ย 93.5 ตัว (เหลือประมาณ 31% ของจำนวนข้อมูลที่ใช้สอน) โดยที่ประสิทธิภาพในการแยกแยะข้อมูลลดลงมาอยู่ที่ 74.73 เปอร์เซ็นต์ ซึ่งมีการลดลงเพียง 5.6 เปอร์เซ็นต์เท่านั้นซึ่งถือได้ว่าไม่มากนัก และเมื่อเปรียบเทียบการใช้โปรโตไทป์ที่ได้จากการคำนวนด้วยวิธีการ learning vector quantization (LVQ) เป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule โดยกำหนดให้มีจำนวนโปรโตไทป์เท่ากันกับโปรโตไทป์ที่เลือกได้ด้วยวิธีการจัดกลุ่มข้อมูลแบบมีการสอนนั้นคือเฉลี่ย 93.5 ตัว จะเห็นได้ว่าประสิทธิภาพในการแยกแยะข้อมูลของโปรโตไทป์ที่ได้จากการคำนวนด้วยวิธีการ leaning vector quantization (LVQ) มีค่าสูงถึง 81.88 เปอร์เซ็นต์ซึ่งสูงกว่าประสิทธิภาพในการแยกแยะข้อมูลโดยการใช้ข้อมูลที่ใช้สอนทั้งหมดเป็นโปรโตไทป์อ้างอิงและสูงกว่าประสิทธิภาพใน

การแยกแยะข้อมูลโดยใช้โปรโตไทป์ที่ได้จากวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน (CLUSTER) แต่อย่างไรก็ตามประสิทธิภาพในการแยกแยะข้อมูลก็ไม่ได้ต่างกันมากนัก (เพียง 7.15 เปอร์เซ็นต์) ซึ่งถือว่ามีประสิทธิภาพในการแยกแยะข้อมูลใกล้เคียงกัน

ตารางที่ 5.34 สรุปผลการทดลองแยกแยะข้อมูลกับชุดข้อมูลอีโคไล

สรุปผลการทดลองแยกแยะข้อมูลกับชุดข้อมูลอีโคไล			
	Train Dataset	**CLUSTER**	**LVQ**
โปรโตไทป์อ้างอิง	302.4	93.5	93.5
เปอร์เซ็นต์การแยกแยะข้อมูลถูกต้อง	80.33	74.73	81.88

5.2.3.7 ผลการทดลองแยกแยะข้อมูลกับชุดข้อมูลทิคแทคโท

ชุดข้อมูลทิคแทคโทมีลักษณะเฉพาะ(Feature) 9 มิติ (แอททริบิว) และมีชนิดทั้งหมด 2 ชนิด และมีขนาด 958 ออบเจ็ค ทำการเตรียมข้อมูลโดยการแบ่งชุดข้อมูลทิคแทคโทเป็นชุดข้อมูลสำหรับใช้สอนและสำหรับใช้ทดสอบจำนวน10 ชุด โดยแต่ละชุดข้อมูลจะมีข้อมูลสำหรับใช้ทดสอบจำนวน 96 ออบเจ็คและมีข้อมูลสำหรับใช้สอนจำนวน 862 ออบเจ็ค นอกจากสองชุดข้อมูลสุดท้ายที่มีข้อมูลสำหรับใช้ทดสอบจำนวน 95 ออบเจ็คและมีข้อมูลสำหรับใช้สอนจำนวน 863 ออบเจ็ค ซึ่งสามารถแสดงได้ดังตารางที่ 5.35

ตารางที่ 5.35 แสดงการแบ่งชุดข้อมูลทิคแทคโทออกเป็นข้อมูลสำหรับใช้สอนและข้อมูลสำหรับใช้ทดสอบจำนวน 10 ชุด

ตารางการเตรียมชุดข้อมูลสำหรับใช้สอนและใช้ทดสอบ										
ข้อมูลชุดที่	1	2	3	4	5	6	7	8	9	10
ข้อมูลสำหรับสอน	862	862	862	862	862	862	862	862	863	863
ข้อมูลสำหรับทดสอบ	96	96	96	96	96	96	96	96	95	95

ผลการทดลองใช้ข้อมูลสำหรับใช้สอนทั้งหมดเป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule แล้วใช้กฎ nearest neighbor rule แยกแยะข้อมูลสำหรับใช้ทดสอบ สามารถแสดงได้

ดังตารางที่ 5.36 ซึ่งแสดงให้เห็นว่าถ้าใช้โปรโตไทป์อ้างอิงเฉลี่ย 863 ตัวจะสามารถแยกแยะข้อมูลสำหรับใช้ทดสอบได้อย่างถูกต้องโดยเฉลี่ย100 เปอร์เซ็นต์

ตารางที่ 5.36 แสดงผลการทดลองใช้ข้อมูลสำหรับใช้สอนเป็นโปรโตไทป์อ้างอิง

ผลการทดลองใช้ข้อมูลสำหรับใช้สอนเป็นโปรโตไทป์อ้างอิง											
ข้อมูลชุดที่	1	2	3	4	5	6	7	8	9	10	เฉลี่ย
โปรโตไทป์อ้างอิง	862	862	862	862	862	862	862	862	863	863	862.2
เปอร์เซ็นต์การแยกแยะข้อมูลถูกต้อง	100	100	100	100	100	100	100	100	100	100	100

ผลการทดลองใช้โปรโตไทป์ของข้อมูลสำหรับใช้สอนเป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule (ซึ่งโปรโตไทป์ได้มาจากการวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน) แล้วใช้กฎ nearest neighbor rule นี้แยกแยะข้อมูลสำหรับใช้ทดสอบสามารถแสดงผลการทดลองนี้ได้ดังตารางที่ 5.37 ซึ่งจะเห็นได้ว่าถ้าใช้โปรโตไทป์อ้างอิงที่เลือกด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนซึ่งมีจำนวนเฉลี่ย 84.1 ตัว แล้วจะสามารถแยกแยะข้อมูลสำหรับใช้ทดสอบได้อย่างถูกต้องโดยเฉลี่ยประมาณ 95.4 เปอร์เซ็นต์

ตารางที่ 5.37 แสดงผลการทดลองใช้โปรโตไทป์ที่เลือกด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนเป็นโปรโตไทป์อ้างอิง

ผลการทดลองใช้โปรโตไทป์ที่เลือกด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอนเป็นโปรโตไทป์อ้างอิง											
ข้อมูลชุดที่	1	2	3	4	5	6	7	8	9	10	เฉลี่ย
โปรโตไทป์อ้างอิง	84	83	85	84	83	82	89	87	82	82	84.1
เปอร์เซ็นต์การแยกแยะข้อมูลถูกต้อง	94.79	98.95	95.83	95.83	95.83	92.71	95.83	97.92	94.74	91.58	95.4

ผลการทดลองใช้โปรโตไทป์ของข้อมูลสำหรับใช้สอนเป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule โดยที่โปรโตไทป์ของข้อมูลสำหรับใช้สอนได้มาจากการคำนวนด้วยวิธีการ learning vector quantization ซึ่งกำหนดให้มีจำนวน codebook (โปรโตไทป์) เท่ากับจำนวนโปรโตไทป์ที่หาได้ด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน(ในตารางที่ 5.37) เพื่อให้สามารถเปรียบเทียบกันได้ แล้วใช้กฎ nearest neighbor rule นี้แยกแยะข้อมูลสำหรับใช้ทดสอบ สามารถแสดงผลการทดลองได้ดังตารางที่ 5.38 ซึ่งจะเห็นได้ว่าถ้าใช้โปรโตไทป์อ้างอิงที่ได้จากการคำนวนด้วยวิธีการ learning vector quantization จำนวนเฉลี่ย 84.1 ตัว แล้วจะสามารถแยกแยะข้อมูลสำหรับใช้ทดสอบได้อย่างถูกต้องโดยเฉลี่ยประมาณ 95.2 เปอร์เซ็นต์

ตารางที่ 5.38 แสดงผลการทดลองใช้โปรโตไทป์ที่คำนวนด้วยวิธีการ learning vector quantization เป็นโปรโตไทป์อ้างอิง

ผลการทดลองใช้โปรโตไทป์ที่คำนวนด้วยวิธีการ Learning Vector Quantization เป็นโปรโตไทป์อ้างอิง											
ข้อมูลชุดที่	1	2	3	4	5	6	7	8	9	10	เฉลี่ย
โปรโตไทป์อ้างอิง	84	83	85	84	83	82	89	87	82	82	84.1
เปอร์เซ็นต์การแยกแยะข้อมูลถูกต้อง	95.83	95.83	96.88	95.83	93.75	93.75	95.83	95.83	94.74	93.68	95.12

5.2.3.8 สรุปผลการทดลองแยกแยะข้อมูลกับชุดข้อมูลทิคแทคโท

ตารางที่ 5.39 แสดงการสรุปผลการทดลองเพื่อประเมินประสิทธิภาพในการแยกแยะข้อมูลกับชุดข้อมูลทิคแทคโท การใช้ข้อมูลที่ใช้สอนทั้งหมดเป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule ซึ่งทำให้มีโปรโตไทป์อ้างอิงเป็นจำนวนเฉลี่ย 862.2 ตัว และมีประสิทธิภาพในการแยกแยะข้อมูล 100 เปอร์เซ็นต์ ส่วนการใช้โปรโตไทป์ที่เลือกได้ด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน (CLUSTER) เป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule สามารถลดจำนวนโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule เหลือเพียงเฉลี่ย 84.1 ตัว (เหลือประมาณ 10% ของจำนวนข้อมูลที่ใช้สอน) โดยที่ประสิทธิภาพในการแยกแยะข้อมูลลดลงมาอยู่ที่ 95.4 เปอร์เซ็นต์ ซึ่งมีการลดลงจากเดิม 4.6 เปอร์เซ็นต์เท่านั้น และเมื่อเปรียบเทียบการใช้โปรโตไทป์ที่ได้จากการคำนวนด้วยวิธีการ learning vector quantization (LVQ) เป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule โดยกำหนดให้มีจำนวนโปรโตไทป์เท่ากันกับโปรโตไทป์ที่

เลือกได้ด้วยวิธีการจัดกลุ่มข้อมูลแบบมีการสอนนั้นคือเฉลี่ย 84.1 ตัว จะเห็นได้ว่าประสิทธิภาพในการแยกแยะข้อมูลของโปรโตไทป์ที่ได้จากทั้งสองวิธีมีค่าใกล้เคียงกัน ดังนั้นโปรโตไทป์ที่เลือกได้โดยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มข้อมูลแบบมีการสอน (CLUSTER) จึงมีประสิทธิภาพในการแยกแยะข้อมูลอยู่ในระดับที่ดี

ตารางที่ 5.39 สรุปผลการทดลองแยกแยะข้อมูลกับชุดข้อมูลทิคแทคโท

สรุปผลการทดลองแยกแยะข้อมูลกับชุดข้อมูลทิคแทคโท			
	Train Dataset	**CLUSTER**	**LVQ**
โปรโตไทป์อ้างอิง	862.2	84.1	84.1
เปอร์เซ็นต์การแยกแยะข้อมูลถูกต้อง	100	95.4	95.12

5.2.4 สรุปผลการทดลองแยกแยะข้อมูล

เราสามารถสรุปผลการทดลองการแยกแยะข้อมูลกับชุดข้อมูลที่ได้ทำการทดลองไปทั้งหมด ได้ดังตารางที่ 5.40

ตารางที่ 5.40 สรุปผลการทดลองแยกแยะข้อมูลกับชุดข้อมูลที่ได้ทำการทดลองไปทั้งหมด

ชุดข้อมูลดอกไอริช		
	โปรโตไทป์อ้างอิง	**เปอร์เซ็นต์การแยกแยะข้อมูลถูกต้อง**
Train Dataset	135	95.99
CLUSTER	15.2	92.66
LVQ	15.2	92.66
ชุดข้อมูลไวน์		
	โปรโตไทป์อ้างอิง	**เปอร์เซ็นต์การแยกแยะข้อมูลถูกต้อง**
Train Dataset	160.2	75.23
CLUSTER	58	74.12
LVQ	58	61.77
ชุดข้อมูลอีโคไล		
	โปรโตไทป์อ้างอิง	**เปอร์เซ็นต์การแยกแยะข้อมูลถูกต้อง**
Train Dataset	302.4	80.33
CLUSTER	93.5	74.73
LVQ	93.5	81.88
ชุดข้อมูลทิคแทคโท		
	โปรโตไทป์อ้างอิง	**เปอร์เซ็นต์การแยกแยะข้อมูลถูกต้อง**
Train Dataset	862.2	100
CLUSTER	84.1	95.4
LVQ	84.1	95.12

จากผลการทดลองในตารางที่ 5.40 ได้แสดงให้เห็นว่าวิธีการเลือกโปรโตไทป์โดยการจัดกลุ่มข้อมูลแบบมีการสอนสามารถลดจำนวนของโปรโตไทป์อ้างอิงของกฎ nearest neighbor rule ได้เป็นจำนวนมาก โดยที่ ประสิทธิภาพในการแยกแยะข้อมูลของกฎ nearest neighbor rule ที่ใช้โปรโตไทป์ที่เลือกได้ด้วยวิธีการเลือกโปรโตไทป์โดยการจัดกลุ่มข้อมูลแบบมีการสอนเป็นโปรโตไทป์

อ้างอิงจะมีค่าใกล้เคียงกับประสิทธิภาพในการแยกแยะข้อมูลของของกฎ nearest neighbor rule ที่ใช้ข้อมูลสำหรับใช้สอนทั้งหมดเป็นโปรโตไทป์อ้างอิง

หากประเทียบประสิทธิภาพในการแยกแยะข้อมูลของกฎ nearest neighbor rule ที่ใช้โปรโตไทป์ที่เลือกได้ด้วยวิธีการเลือกโปรโตไทป์โดยการจัดกลุ่มข้อมูลแบบมีการสอนกับประสิทธิภาพในการแยกแยะข้อมูลของกฎ nearest neighbor rule ที่ใช้โปรโตไทป์ที่คำนวนได้โดยวิธีการ learning vector quantization เป็นโปรโตไทป์อ้างอิง โดยกำหนดให้มีจำนวนโปรโตไทป์ในการอ้างอิงเท่ากันจะเห็นได้ว่าประสิทธิภาพในการแยกแยะข้อมูลมีความใกล้เคียงกัน

วิธีการเลือกโปรโตไทป์โดยการจัดกลุ่มข้อมูลแบบมีการสอนจะเลือกโปรโตไทป์จากข้อมูลที่อยู่มีในชุดข้อมูลสำหรับใช้สอนเท่านั้น ส่วนวิธีการ learning vector quantization เป็นการคำนวนค่าของโปรโตไทป์ขึ้นมาใหม่โดยไม่ได้เลือกโปรโตไทป์จากข้อมูลที่มีอยู่ในชุดข้อมูลสำหรับใช้สอน ซึ่งทำให้วิธีการเลือกโปรโตไทป์โดยการจัดกลุ่มข้อมูลแบบมีการสอนสามารถใช้งานได้กับชุดข้อมูลประเภทที่ไม่สามารถคำนวนค่าเพื่อเลื่อนตำแหน่งของโปรโตไทป์ได้ แต่วิธีการ learning vector quantization ไม่สามารถใช้ได้ เพราะไม่สามารถคำนวนค่าโปรโตไทป์เพื่อเลื่อนหาตำแหน่งที่เหมาะสมได้

ประสิทธิภาพในการแยกแยะข้อมูลของกฎ nearest neighbor rule ที่ใช้โปรโตไทป์ที่เลือกได้ด้วยวิธีการเลือกโปรโตไทป์โดยการจัดกลุ่มข้อมูลแบบมีการสอนเป็นโปรโตไทป์อ้างอิงมีค่าใกล้เคียงกับประสิทธิภาพในการแยกแยะข้อมูลของของกฎ nearest neighbor rule ที่ใช้ข้อมูลสำหรับใช้สอนทั้งหมดเป็นโปรโตไทป์อ้างอิง และนอกจากนั้นประสิทธิภาพในการแยกแยะข้อมูลของกฎ nearest neighbor rule ที่ใช้โปรโตไทป์ที่เลือกได้ด้วยวิธีการเลือกโปรโตไทป์โดยการจัดกลุ่มข้อมูลแบบมีการสอนมีค่าใกล้เคียงกับประสิทธิภาพในการแยกแยะข้อมูลของของกฎ nearest neighbor rule ที่ใช้โปรโตไทป์ที่คำนวนได้โดยวิธีการ learning vector quantization เป็นโปรโตไทป์อ้างอิง ดังนั้นจึงถือได้ว่าโปรโตไทป์ที่เลือกได้ด้วยวิธีการเลือกโปรโตไทป์โดยการจัดกลุ่มข้อมูลแบบมีการสอนมีประสิทธิภาพในการแยกแยะข้อมูลอยู่ในระดับที่ดี

บทที่ 6
สรุปผลการวิจัยและข้อเสนอแนะ

6.1 สรุปผลการวิจัย

วิทยานิพนธ์ฉบับนี้ได้นำเสนอวิธีการในการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์แบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (Selection-Condensing Prototype Selection Algorithm) วิธีการหนึ่งซึ่งเรียกว่า "วิธีการเลือกโปรโตไทป์โดยใช้วิธีการจัดกลุ่มแบบมีการสอน" ซึ่งวิธีการนี้ช่วยในการแก้ปัญหาความต้องการหน่วยความจำในปริมาณมาก (high memory demands problem) และปัญหาความต้องการใช้การคำนวนในปริมาณสูง (high computational demands problem) ซึ่งเป็นปัญหาที่เกิดขึ้นกับวิธีการแยกแยะข้อมูลโดยใช้กฎ nearest neighbor rule ที่มีโปรโตไทป์อ้างอิงเป็นจำนวนมาก ซึ่งวิธีการที่ได้นำเสนอแก้ปัญหาดังกล่าวโดยการพิจารณาเลือกข้อมูลเพียงบางส่วนจากข้อมูลในชุดข้อมูลตั้งต้นไปเป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule แทนที่การใช้ข้อมูลในชุดข้อมูลตั้งต้นทั้งหมดเป็นโปรโตไทป์อ้างอิงให้กับกฎ nearest neighbor rule โดยที่รับรองว่าโปรโตไทป์อ้างอิงที่เลือกมาได้ใหม่จะมีคุณสมบัติความสอดคล้องกับเซ็ตข้อมูลตั้งต้นเสมอ

จากการทดลองเลือกโปรโตไทป์เพื่อประเมินประสิทธิภาพในการลดจำนวนโปรโตไทป์อ้างอิงของวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มแบบมีการสอน โดยทำการเปรียบเทียบจำนวนโปรโตไทป์อ้างอิงที่หาได้ด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มแบบมีการสอนกับจำนวนโปรโตไทป์อ้างอิงที่หาได้ด้วยวิธีการ CNN [14] และ GA [15] ซึ่งเป็นวิธีการในการเลือกโปรโตไทป์เพื่อลดจำนวนโปรโตไทป์แบบให้ผลลัพท์เป็นซับเซ็ตของเซ็ตข้อมูลตั้งต้น (Selection-Condensing Prototype Selection Algorithm) เหมือนกับวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มแบบมีการสอน และได้ทำการทดลองเลือกโปรโตไทป์กับชุดข้อมูลหลากหลายขนาดและมิติที่ได้มาจากแหล่งรวมชุดข้อมูลของ UCI [18] ซึ่งจากผลการทดลองได้แสดงให้เห็นว่าวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มแบบมีการสอนมีประสิทธิภาพในการลดจำนวนโปรโตไทป์อ้างอิงอยู่ในระดับที่ดีกับชุดข้อมูลที่มีขนาดและมิติที่หลากหลาย

จากการทดลองแยกแยะข้อมูลเพื่อประเมินประสิทธิภาพในการแยกแยะข้อมูลของโปรโตไทป์อ้างอิงที่หามาได้ด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มแบบมีการสอน โดยทำการเปรียบเทียบประสิทธิภาพในการแยกแยะข้อมูลของกฎ nearest neighbor rule ที่ใช้โปรโตไทป์อ้างอิงที่หามาได้ด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มแบบมีการสอนกับประสิทธิภาพในการแยกแยะข้อมูลของกฎ nearest neighbor rule ที่ใช้ข้อมูลสำหรับใช้สอนทั้งหมดเป็นโปรโตไทป์

อ้างอิงและประสิทธิภาพในการแยกแยะข้อมูลของกฎ nearest neighbor rule ที่ใช้โปรโตไทป์ที่คำนวนด้วยวิธีการ learning vector quantization ซึ่งจากผลการทดลองได้แสดงให้เห็นว่าประสิทธิภาพในการแยกแยะข้อมูลของโปรโตไทป์อ้างอิงที่หามาได้ด้วยวิธีการเลือกโปรโตไทป์โดยใช้การจัดกลุ่มแบบมีการสอนอยู่ในระดับที่ดี

6.2 ข้อเสนอแนะ

เนื่องด้วยวิธีการเลือกโปรโตไทป์โดยใช้วิธีการจัดกลุ่มแบบมีการสอนยึดหลักความสอดคล้องเพื่อใช้ในการพิจารณาเลือกโปรโตไทป์ ซึ่งนั้นหมายความว่าโปรโตไทป์อ้างอิงที่เลือกได้ต้องสามารถแยกแยะข้อมูลที่ใช้สอนได้อย่างถูกต้องทั้งหมด ซึ่งในบางกรณีแม้ว่าโปรโตไทป์อ้างอิงจะสามารถแยกแยะข้อมูลในชุดที่ใช้สอนได้อย่างถูกต้องทั้งหมดแต่อาจจะแยกแยะข้อมูลในชุดข้อมูลที่ใช้ทดสอบได้ไม่ดีก็ได้ ทั้งนี้ก็เพราะว่าการเลือกโปรโตไทป์นั้นยึดติดกับหลักความสอดคล้องว่าต้องแยกแยะข้อมูลที่ใช้สอนให้ถูกต้องเสมอ ซึ่งเป็นผลทำให้โปรโตไทป์อ้างอิงที่เลือกได้มีความจำเพาะในการแยกแยะข้อมูลกับชุดข้อมูลที่ใช้สอนมากจนเกินไป (Over Fitting) ทำให้โปรโตไทป์อ้างอิงที่หาได้ไม่รองรับการแยกแยะข้อมูลที่ใช้ทดสอบซึ่งเป็นข้อมูลที่อยู่นอกเหนือไปจากข้อมูลที่ใช้สอน ซึ่งด้วยเหตุนี้เป็นผลทำให้ในบางกรณีโปรโตไทป์อ้างอิงที่เลือกได้ด้วยวิธีการเลือกโปรโตไทป์โดยใช้วิธีการจัดกลุ่มแบบมีการสอนมีประสิทธิภาพในการแยกแยะข้อมูลได้ไม่ดีเท่าที่ควร

หนทางในการแก้ปัญหาความจำเพาะกับข้อมูลที่ใช้สอนมากเกินไป (Over Fitting) ก็คือ การหาหลักการใหม่เพื่อใช้ยึดแทนหลักความสอดคล้อง ซึ่งหลักการใหม่นี้จะต้องมีความยืดหยุ่นมากยิ่งขึ้น นั่นคือไม่จำเป็นว่าโปรโตไทป์อ้างอิงที่หาได้ใหม่จะต้องแยกแยะข้อมูลที่ใช้สอนได้อย่างถูกต้องทั้งหมด แต่ผ่อนปรนให้มีการแยกแยะข้อมูลที่ใช้สอนได้ผิดพลาดบางเล็กน้อย เพื่อให้โปรโตไทป์อ้างอิงที่หาได้ใหม่มีความสามารถในแยกแยะข้อมูลที่ครอบคลุมข้อมูลในวงกว้างมากยิ่งขึ้น (Generalization) ไม่จำเพาะเจาะจงกับข้อมูลที่ใช้สอนมากจนเกินไป

เอกสารอ้างอิง

[1] T.M. Cover and P.E. Hart, "Nearest Neighbor Pattern Classification ", IEEE Transactions on Information Theory, Vol. IT-13, No. 1, January 1967, pp. 21-27.

[2] R. A. Mollineda, F. J. Ferri, and E. Vidal. "Merged-based prototype selection for nearest neighbor classification", Proceedings of the 4th World Multiconference on Systemics, Cybernetics and Informatics (SCI2000), Orlando, USA, July 2000, pp. 640-645.

[3] V. Cerveron and F.J. Ferri, "Another move toward the minimum consistent subset: a tabu search approach to the condensed nearest neighbor rule", IEEE Transactions on Systems, Man and Cybernetics, Vol. 31, No. 3, 2001, pp. 408-413.

[4] "Statistical classification" [Online]. Available: http://en.wikipedia.org/wiki/Statistical_classification.

[5] Binay K. Bhattacharya, Ronald S. Poulsen, Godfried T. Toussaint, "Application of Proximity Graphs to Editing Nearest Neighbor Decision Rule", International Symposium on Information Theory, Santa Monica, 1981.

[6] A. K. Jain, R. P.W. Duin, J. Mao,"Statistical Pattern Recognition: A Review", IEEE Transaction on Pattern Analysis and Machine Intelligent, Vol. 22, No. 1, 2000, pp.4-37.

[7] R. O. Duda, P. E. Hart and D. G. Stork, Pattern Classification, 2nd ed, John Wiley & Sons, 2000.

[8] ชม กิ้มปาน, ทฤษฎีการจดจำรูปแบบ, ตำราชุดวิศวกรรมศาสตร์ สจล. , 1998, pp. 5-45.

[9] Quinlan. J. R., C4.5: Programs for Machine Learning, Morgan Kauffman, 1993.

[10] สมชาย ประสิทธิ์จูตระกูล, การออกแบบและวิเคราะห์อัลกอริทึม, ศูนย์เทคโนโลยีอิเล็คทรอนิกส์และคอมพิวเตอร์แห่งชาติ, 2544.

[11] Norbert Jankowski and Marek Grochowski, "Comparison of Instances Seletion Algorithms I. Algorithms Survey", ICAISC 2004, LNAI 3070, Springer-Verlag Berlin Heidelberg, 2004, pp. 598–603.

[12] Wilson D., "Asymptotic properties of nearest neighbor rules using edited data", IEEE Transactions on Systems, Man, and Cybernetics 2, 1972, pp. 408–421.

[13] Tomek I., "An experiment with the edited nearest-neighbor rule", IEEE Transactions on Systems, Man, and Cybernetics, 1976, pp. 448–452.

[14] P.E. Hart, "The Condensed Nearest Neighbor Rule", IEEE Transactions on Information Theory, Vol. 14, 1968, pp. 515-516.

[15] L.I. Kuncheva and J.C. Bezdek, "Nearest Prototype Classification: Clustering, Genetic Algorithms, or Random Search?", IEEE Transactions on Systems, Man and Cybernetics, Vol. 28, No. 1, 1998, pp. 160-164.

[16] J.C. Bezdek, T.R.Reichherzer, G.S.Lim, and Y.Attikiouzel, "Multiple-prototype classifier design", IEEE Transactions on Systems, Man and Cybernetics. B, vol.28, pp.67-79, Feb. 1998.

[17] T. Kohonen,"The self-organizing map", Proc. IEEE, vol 78, no. 9, pp.1464-1480, 1990.

[18] Department of Information and Computer Science, University of California. "UCI Machine Learning Repository" [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html. 1998.

[19] K. Kangkan, B. Kruatrachue, "Minimal Consistent Subset Selection as Interger Nonlinear Programming Problem", ISCIT 2006, October 2006.

[20] S. Haykin, Neural Network: A comprehensive Foundation 2nd ed, Prentice Hall, 1999, pp.466-473.

[21] D. W. Patterson, Artificial Neural Network: Theory and Applications, Prentice Hall, 1995, pp.377-386.

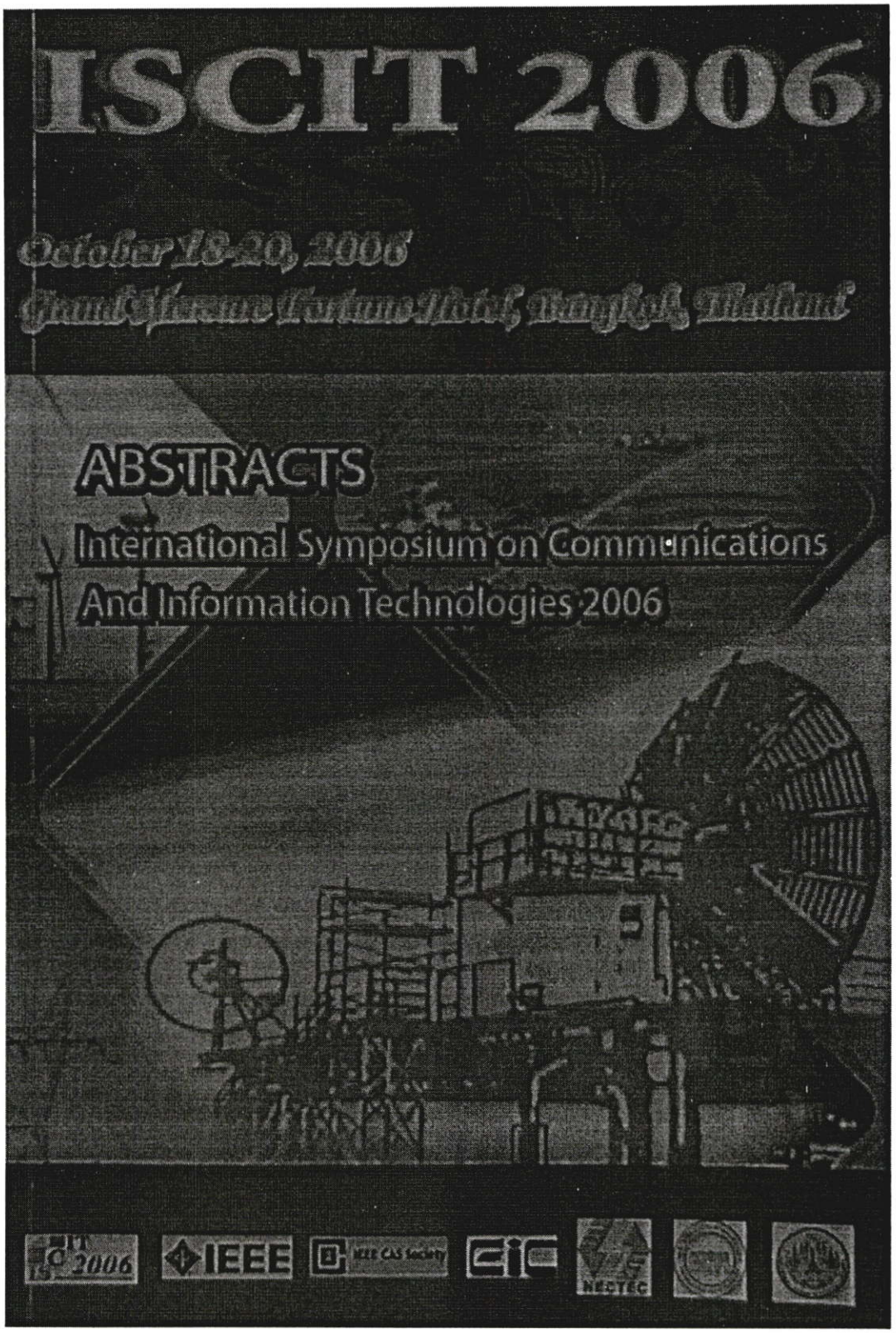
[22] N. Jankowski, M. Grochowski, "Comparison of Instance Selection Algorithms I. Algorithms Survey", ICAISC 2004, LNAI 3070, pp.598-603.

[23] Christian Borgelt, "Learning Vector Quantization Visualization" [Online]. Available: http://fuzzy.cs.uni-magdeburg.de/~borgelt/doc/lvqd. 2000.

[24] F. J. Ferri, R. A. Mollineda and E. Vidal, "An Experimental Comparison between Consistency-based and Adaptive Replacement Schemes", ICPR 2002, pp. 41-44.

ภาคผนวก

งานวิจัยที่ได้รับการตีพิมพ์


ISCIT 2006
October 18-20, 2006
Grand Prasur Fortune Hotel, Bangkok, Thailand
ABSTRACTS
International Symposium on Communications
And Information Technologies 2006
IEEE
NECTEC

Minimal Consistent Subset Selection as Integer Nonlinear Programming Problem

Kamonnat Kangkan* and Boontee Kruatrachue†
Department of Computer Engineering
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand
E-mail: kangkan.soranan@yahoo.com*, booontee@yahoo.com†

Abstract— **The minimal consistent subset selection is a solution of high computational demands problem of the nearest neighbor decision system. This paper presents a new approach that aims to make the problem more clearly by stating it as a constrained optimization problem, called "integer nonlinear programming problem (INLP)". In this context, we propose method that formulates the minimal consistent subset selection problem as 0-1 integer nonlinear programming problem. We show experimental result of the minimal consistent subset of "IRIS Dataset", obtained by solving its constrained optimization model. The results obtained suggest that the approach offers exactly optimal solution of the problem.**

Keywords – **Prototype selection, minimal consistent subset, consistency, nearest neighbor rule.**

I. INTRODUCTION

The nearest neighbor (NN) rule assigns an unclassified sample to the same class as the nearest of n stored, correctly classified samples (prototypes). In other words, given a collection of n reference points, a new point is assigned to the same class as its nearest neighbor. The k-nearest neighbor (k-NN) rule assigns an unclassified sample to the same class as the most frequently class of the k-nearest prototypes. Over the last 40 years, this simple classification rule has been intensively used in a broad range of pattern recognition applications. In contrast to its conceptual simplicity, the rule has a good behavior when applied to non-trivial problems. In fact, the k-NN rule is asymptotically optimal in the Bayes sense [1]. (The Bayes decision rule achieves minimum risk but, requires complete knowledge of the underlying statistics.)

From a practical point of view, however, the NN rule is not a prime candidate for many applications because it requires large memory space and expensive computation time. The k-NN rule consists of a search of prototypes given a particular distance definition. A trivial consequence of the large size of the sets of prototypes (to guarantee representativity and approach optimality) is the computational burden this searching problem implies. This constitutes one of the main drawbacks of the NN rules. Another very important drawback comes from the fact that the prototypes which are available may contain erroneously labeled or noisy prototypes which may lead to arbitrarily large deviations from the asymptotically optimal results.

Several different ways of overcoming these drawbacks have been proposed. In this context, prototype selection (PS) aims at modifying an initially given set of prototypes in order to reduce its size as well as to improve classification performance. From the point of view of this goal, PS methods can be divided into two different kinds of techniques which are usually referred to as editing and condensing. The main goal of editing techniques is to improve the performance of the resulting classifier by discarding outliers and cleansing the overlap among classes. Editing does not generally entail substantial reductions in size, but it usually produces well-clustered groups of homogeneous prototypes that lead to optimal 1-NN classification results. On the other hand, condensing algorithms try to find a significantly reduced set of prototypes whose 1-NN classification results are as close as possible to those obtained using all original prototypes [2].

According to the way in which prototypes are obtained, there is a separation between selection techniques [2],[3], [4],[5] in which the resulting prototypes are selected from the original set, and replacement techniques [6],[7],[8],[9] in which resulting prototypes are built and may be different from any prototype in the original set. Prototypes obtained by either of these methods can be referred to as S-prototypes and R-prototypes, respectively.

One of the first PS algorithms [2] was in fact a selection condensing algorithm in which S-prototypes are picked from the original set in order to ensure that the selected prototypes manage to correctly classify the entire original set. This property is referred to as consistency and can be applied to both S-prototypes and R-prototypes.

The consistency property is a key issue in many condensing algorithms proposed to date. A resulting set of prototypes is said to be consistent with an initial set if it can classify all initial prototypes using the 1-NN rule with no errors. In this way, the quality of the condensed sets depends on the quality of the original set, and reducing the size of the selected set is the main goal of recently proposed condensing algorithms of both types. Thus, the methods those try to find the consistent set with the smallest possible cardinality, called a minimal consistent subset (MCS), have been arisen. One of the earliest papers on this topic is by Hart [2], whose elegant method has been used as a basis for many subsequent modifications. Hart's iterative procedure selects elements sequentially and

ends up with a relatively large consistent set. A study by Dasarathy [3] presented a technique for finding a consistent set that he believed to be minimal. On the IRIS data set, however, Dasarathy's technique finds a 15-element consistent subset, whereas GA method [4] resulted in a 12-element consistent set and Tabu search method [5] resulted in a 10-element consistent set. Thus, there are some counterexamples to Dasarathy's conjecture that his MCS method is truly minimal. However, there is little another approach that tries to formalize the MCS problem or proposes methods that solve exactly the problem.

In fact, obtaining the minimal consistent subset of a given set is considered as a challenging problem especially in the case of S-prototypes, where it becomes a hard combinatorial problem [3]. Cause of the interest of the problem, we thus try to study in the problem and try to descript it more clearly and easily to understand by express it in form of constrained optimization problem, called "0-1 integer nonlinear programming problem". In this experiment, we used "OPBDP - A Davis-Putnam Based Enumeration Algorithm for Linear Pseudo-Boolean Optimization" [11] to solve our 0-1 integer nonlinear programming problem. OPBDP is an implementation in C++ of an implicit enumeration algorithm for solving (non)linear 0-1 (or pseudo-Boolean) optimization problems with integer coefficients. Thus, it suit, just enough, with our problem.

The rest of this paper is organized as follows. First, we describe the notions about MCS problem in Section II. Our method that models MCS problem as 0-1 INLP problem is later introduced in Section III. Section IV then presents the result of our method. The concluding comments of this paper are finally presented in Section V.

II. An Approach to Modeling Minimal Consistent Subset Selection Problem

A. General Notion

Given:

x is an instance of position type in metric space domain $\Re^d$; $x \in \Re^d$.

θ is an instance of category type in category domain $\mathbb{C}$; $\theta \in \mathbb{C} = \{class1, class2, ..., classC\}$.

δ is dissimilarity measure (distance function) on metric space $\Re^d$; *for any* $x \in \Re^d$.

B. Nearest Neighbor Notion

Given:

Original set of nearest neighbor classifier is a set of known position-category pair, with n cardinality, that used to train the nearest neighbor classifier.

$$O = \{o_1, o_2, ..., o_n\} = \{(x_{o1}, \theta_{o1}), (x_{o2}, \theta_{o2}), ..., (x_{on}, \theta_{on})\}$$

Reference set (prototype set) of nearest neighbor classifier is a set of known position-category pair, with m cardinality, that referenced by nearest neighbor classifier.

$$V = \{v_1, v_2, ..., v_m\} = \{(x_{v1}, \theta_{v1}), (x_{v2}, \theta_{v2}), ..., (x_{vm}, \theta_{vm})\}$$

A new pair (x, θ) is given, where only the measurement x is observable, The NN rule design to estimate category θ by utilizing information contained in the reference set.

Definition1: **nearest prototype to** x : We shall call $v' = (x', \theta') \in V$ a nearest prototype to position x if $\delta(x', x) \le \delta(x_i, x)$ *for* $\forall i; i \in \{v1, v2, ..., vm\}$.

Symbolized as,

$$nearestPrototype(x) = (x', \theta') \mid \delta(x', x) \le \delta(x_i, x); (x', \theta') \in V \text{ for } \forall i; i \in \{v1, v2, ..., vm\}.$$

Definition2: **1-NN rule**: "Assign category θ of new pair (x, θ) with same as the category of the nearest prototype to x."

$$\theta \Leftarrow NN(x).$$
$$\theta \Leftarrow categoryOf(nearestPrototype(x)).$$
$$\theta \Leftarrow \theta' \mid (x', \theta') = nearestPrototype(x).$$

C. Minimal Consistent Set Notion

Definition3: **Consistency Property**: Given V is a set of position-category pair and O is a set of position-category pair too. Set V will be have consistency property to set O if and only if the 1- nearest neighbor rule can classify all pair in set O with zero error rate, whereas using set V as its reference set.

$$V \; consistentTo \; O$$
$$\leftrightarrow \theta_i \; equalTo \; NN(x_i) \; for \; \forall i; i \in \{o1, o2, ..., on\};$$
$$V \; is \; the \; reference \; set.$$
$$\leftrightarrow \theta_i \; equalTo \; categoryOf(nearestPrototype(x_i))$$
$$for \; \forall i; i \in \{o1, o2, ..., on\}; V \; is \; the \; reference \; set.$$
$$\leftrightarrow \theta_i \; equalTo \; categoryOf((x', \theta') \mid \delta(x', x_i) \le d(x_j, x_i);$$
$$(x', \theta') \in V \; for \; \forall j; j \in \{v1, v2, ..., vm\})$$
$$for \; \forall i; i \in \{o1, o2, ..., on\}.$$

Definition4: **Consistent Subset**: Given set V is a subset of a set of position-category pair O ($V \subset O$). Then, Set V will be a consistent subset of set O if and only if set V has consistency property to set O.

$$(V \; consistentSubsetOf \; O) \leftrightarrow (V \subset O) \wedge (V \; consistentTo \; O)$$

Definition5: **Minimal Consistent Subset**: Set V will be Minimal Consistent Subset of set O if and only if V is a consistent subset of set O and it has a minimal cardinality.

$$MCS(O) = c' \mid cardinality(c') \le cardinality(c)\ where$$
$$c'\ consistentSubsetOf\ O\ for\ \forall c; c\ consistentsubsetOf\ O;$$

D. *Modeling Concept*

Minimal Consistent Subset Selection Problem is the selection of prototypes for the nearest neighbor rule which aims at obtaining a minimal subset of the original set that guarantees zero resubstitution error rates if used as the reference for the 1-NN rule.

The 1-NN rule can classify initial dataset with zero resubstitution error rates if and only if it can assign category of all pair in original set with the same category with its self.

Since, From Definition2, The 1-NN rule assigns category of unknown pair (x,θ) with the category of its nearest prototype.

Thus, in order to provide the 1-NN rule can assign category of the pair with its self class label. We must provide all original pairs with their nearest prototype that have same category (Consistency Property).

We can see Minimal Consistent Subset (Definition5) Selection Problem as the selection of prototypes for the nearest neighbor rule which aims at obtaining a minimal subset of the original set that guarantees all pairs in original set has the same class label with its nearest prototype 's class label.

Can formulate as format of "constrained optimization problem" like this

Minimize ***Objective function***:
Cardinality(Referenceset : V)
Subject to ***Constraint function***:
V consistentSubsetOf O

As same meaning as;

Minimize ***Objective function:***
"Amount of Prototypes"
Subject to ***Constraint function:***
"Each pair in original set must has same category with its nearest prototype's category, which reference set is subset of original set."

Given:

Definition6: **Ranking Object List to** x : is an ascent ordered list of position-category pair that ordered by distance value between position of each pair in the list and position x (with particular distance function).

$$L(x) = \{l_1, l_2, ..., l_p\}$$
$$= \{(x_{l1}, \theta_{l1}), (x_{l2}, \theta_{l2}), ..., (x_{lp}, \theta_{lp})\}$$
$$where\ \delta(x_{li}, x) \le \delta(x_{lj}, x) : i < j;$$
$$for\ \forall i, \forall j; i, j \in \{1, 2, ..., p\}.$$

With definition1 and definition6, we can give;
$nearestPrototype(x) \equiv firstPair(L(x))$;
, if reference set of nearest neighbor rule is equal to member set of ranking object list.

So that, to guarantee the nearest prototype to position x will have the same category with its category θ, we must accept the following rule.

Definition7: **Constraint Rule to** x : any pair (x_{lj}, θ_{lj}), in $L(x)$, can be selected to be the nearest prototype to the pair (x,θ) if and only if

1) pair (x_{lj}, θ_{lj}) and pair (x,θ) have the same class label $\theta_{lj} = \theta$; and
2) Any pairs (x_{li}, θ_{li}) where $\theta_{li} \neq \theta_{lj}$ and $i < j$; must not be selected to be prototypes.

, where the member set of the ranking object list $L(x)$ is equal to original set O.

So that, to guarantee the prototypes (reference set) will be consistent subset of original set, it must accept the constraint rule to all pair in original set. Then we can write the constraint function with this concept too.

III. The Modeling Method

A. *Method Procedure*

1) Stores initial dataset.
2) Constructs distance table by calculating all distance of 2 objects in initial dataset with given distance function.
3) Constructs ranking object table by constructs ranking object list of original set to all pair in original set and follows additional rules.
   - Object must be nearest to itself forever.(not cover overlap classification)
   - In case of many objects have the same distances to the object; object with the difference class to the object has higher priority (preceding in list).
4) Constructs constraint inequality by using the constraint rule and defines objective function with amount of prototypes.
5) Use any INLP Solver to solve this problem.

B. The example

1) Stores initial dataset that has aligned as in figure 1.

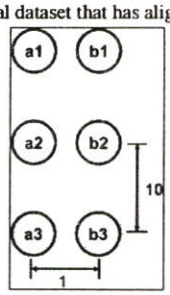


Figure 1
Original Dataset

Object a1, a2, a3 have class label A and object b1, b2, b3 have class label B.

2) Constructs distance table with Euclidean distance

Table 1
DISTANCE TABLE

TO	DISTANCE (FROM)					
	a1	**a2**	**a3**	**b1**	**b2**	**b3**
a1	00.000	10.000	20.000	01.000	10.050	20.025
a2	10.000	00.000	10.000	10.050	01.000	10.050
a3	20.000	10.000	00.000	20.025	10.050	01.000
a4	01.000	10.050	20.025	00.000	10.000	20.000
a5	10.050	01.000	10.050	10.000	00.000	10.000
a6	20.025	10.050	01.000	20.000	10.000	00.000

3) Constructs ranking object table by sorting object by its distance that get from distance table

Table 2
RANKING OBJECT TABLE

TO	RANKING OBJECT LIST					
a1	a1	b1	a2	b2	a3	b3
a2	a2	b2	a3	a1	b1	b3
a3	a3	b3	a2	b2	a1	b1
b1	b1	a1	b2	a2	b3	a3
b2	b2	a2	b1	b3	a1	a3
b3	b3	a3	b2	a2	b1	a1

4) Construct constraint inequality, as example
With constraint rule to a1 (1),
a1, a2, a3 can be prototypes of a1.
b1, b2, b3 can not be prototype of a1.
With constraint rule to a1 (2),
a1 can be a prototype of a1.
a2 can be a prototype of a1 if b1 is not prototype.
a3 can be a prototype of a1 if b1 and b2 are not prototypes.
Can write in logical form like this;
$f_{a1}(a1,a2,a3,b1,b2,b3) = (a1) \vee (a2 \wedge \neg b1) \vee (a3 \wedge \neg b1 \wedge \neg b2)$
Can write in integer nonlinear inequality like this;
$f_{a1}(a1,a2,a3,b1,b2,b3): (a1)+(a2)\times(1-b1)+(a3)\times(1-b1)\times(1-b2) \geq 1;$
$a1,a2,a3,b1,b2,b3 \in \{0,1\};$
Do in the same way for every object in initial dataset.
Can write in form of constrained optimization problem like this;

Minimize
objective function: $a1+a2+a3+b1+b2+b3;$
Subject to
constraint $\_a1: (a1)+(a2)\times(1-b1)+(a3)\times(1-b1)\times(1-b2) \geq 1;$
constraint $\_a2: (a2)+(a3)\times(1-b2)+(a1)\times(1-b2) \geq 1;$
constraint $\_a3: (a3)+(a2)\times(1-b3)+(a1)\times(1-b3)\times(1-b2) \geq 1;$
constraint $\_b1: (b1)+(b2)\times(1-a1)+(b3)\times(1-a1)\times(1-a2) \geq 1;$
constraint $\_b2: (b2)+(b1)\times(1-a2)+(b3)\times(1-a2) \geq 1;$
constraint $\_b3: (b3)+(b2)\times(1-a3)+(b1)\times(1-a3)\times(1-a2) \geq 1;$
where $a1,a2,a3,b1,b2,b3 \in \{0,1\};$

5) Use any INLP Solver to solve this problem. In this example, the solution is {a2, b2} (although, {a1, b1}, {a3, b3} can be solution too).

Even thought, this problem can formulate in form of integer linear programming problem (ILP) because of is in zero-one polynomial programming [16] problem form that can linearize it using any linearization method such as [17],[18] but in this context we think this non-linear form have more make sense than its diverse linear form.

IV. EXPERIMENTAL RESULTS

In this context, we use IRIS Dataset [10] to experimental confirm that this model can use to model the MCS Problem and the solution obtains the following consistent set.

The Iris problem consists of three classes of 50 four dimensional (4-D) vectors each, corresponding to three subspecies of iris flowers. This problem has often been used for benchmarking and, in particular, it has been considered for prototype selection algorithms [3],[4],[5],[8],[9].

Table 3
THREE TEN-PROTOTYPES CONSISTENT SUBSETS

CLASS 1				CLASS 2				CLASS 3			
5.1	3.8	1.6	0.2	6.3	2.5	4.9	1.5	6.3	2.9	5.6	1.8
				6.8	2.8	4.8	1.4	6	2.2	5	1.5
				6	2.9	4.5	1.5	6.3	2.7	4.9	1.8
				6	2.7	5.1	1.6	6.3	2.8	5.1	1.5
								5.8	2.7	5.1	1.9

Class 1				Class 2				Class 3			
5.1	3.8	1.6	0.2	6.3	2.5	4.9	1.5	5.8	2.7	5.1	1.9
				6	2.9	4.5	1.5	6.3	2.9	5.6	1.8
				6	2.7	5.1	1.6	6	2.2	5	1.5
				6.7	3.1	4.7	1.5	6.3	2.7	4.9	1.8
								6.3	2.8	5.1	1.5
Class 1				Class 2				Class 3			
5	3.3	1.4	0.2	6.3	2.5	4.9	1.5	6.3	2.9	5.6	1.8
				6.8	2.8	4.8	1.4	6	2.2	5	1.5
				6	2.9	4.5	1.5	6.3	2.7	4.9	1.8
				6	2.7	5.1	1.6	6.3	2.8	5.1	1.5
								5.8	2.7	5.1	1.9

We try to change option of the solver that aims to obtain difference solution. It obtains three difference consistent subset of Iris dataset. All three set have same cardinality that is 10.

The result obtains prototype number less than "minimal consistent set selection" procedure [3] that obtains 15-element consistent set and Genetic Algorithms or Random Search Method [4] that obtains 12-element consistent set and have same number as A Tabu Search Approach [5] that obtain 10-element consistent set.

V. Concluding Comments

This paper presents a new approach with aims to study minimal consistent subset selection problem more clearly. Although, we can find the solution in this case of IRIS dataset but in case of a bigger dataset these method is not feasible to directly solve the problem. However, the method can model MCS problem more clearly that lead easily adapt the problem to approximate method, for example if we use NUN concept [3] for approximate the MCS problem so that the model of approximate MCS problem has linearly constraint inequality then it can solve using any ILP algorithm or another view of this approximate problem had become the combinatorial problem that called "set covering problem" so that we can use any heuristic methods [12], [13],[14],[15] that can solve set covering problem to solve this approximate problem too or use Greedy Heuristic (voting) to solve this approximate problem like [3] too. We hope this paper be beneficial for anyone who interest this problem, any way.

References

[1] T.M. Cover and P.E. Hart, "Nearest Neighbor Pattern Classification ", IEEE Transactions on Information Theory, Vol. IT-13, No. 1, January 1967, pp. 21-27.

[2] P.E. Hart, "The Condensed Nearest Neighbor Rule", IEEE Transactions on Information Theory, Vol. 14, 1968, pp. 515-516.

[3] B.V. Dasarathy, "Minimal Consistent Set (MCS) Identification for Optimal Nearest Neighbor Decision Systems Design", IEEE Transactions on Systems, Man and Cybernetics, Vol. 24, No. 3, 1994, pp. 511-517.

[4] L.I. Kuncheva and J.C. Bezdek, "Nearest Prototype Classification: Clustering, Genetic Algorithms, or Random Search?", IEEE Transactions on Systems, Man and Cybernetics, Vol. 28, No. 1, 1998, pp. 160-164.

[5] V. Cerveron and F.J. Ferri, "Another move toward the minimum consistent subset: a tabu search approach to the condensed nearest neighbor rule", IEEE Transactions on Systems, Man and Cybernetics, Vol. 31, No. 3, 2001, pp. 408-413.

[6] C.L. Chang,"Finding prototypes for nearest neighbor classifiers", IEEE Trans. Comput., vol.C-23, pp.1179-1184, Nov.1974.

[7] T. Kohonen,"The self-organizing map", Proc. IEEE, vol 78, no. 9, pp.1464-1480, 1990.

[8] J.C. Bezdek,T.R.Reichherzer,G.S.Lim,and Y.Attikiouzel, "Multiple-prototype classifier design", IEEE Transactions on Systems, Man and Cybernetics. B, vol.28, pp.67-79, Feb. 1998.

[9] R. A. Mollineda, F. J. Ferri, and E. Vidal. "Merged-based prototype selection for nearest neighbor classification", In Proceedings of the 4th World Multiconference on Systemics, Cybernetics and Informatics (SCI2000), Orlando, USA, July 2000.

[10] E. Anderson, "The IRISses of the Gaspe Peninsula", Proc. Bull. Amer. IRIS Soc., Vol. 59, 1935, pp. 2-5.

[11] P. Barth. "A Davis-Putnam based enumeration algorithm for linear pseudo-boolean optimization", Technical Report MPI-I-95-2-003, Max-Planck-Institut f¨ur Informatik, January 1995. See http://domi no.mpi-sb.mpg.de/internet/reports.nsf/NumberView/1995-2-003.

[12] Li J. and Kwan R.S.K. (2004): "A Meta-heuristic with Orthogonal Experiment for the Set Covering Problem", Journal of Mathematical Modelling and Algorithms, Kluwer Academic Publishers, 3 (3): 263-283.

[13] Caprara, A., Fischetti, M. and Toth, P.: "A heuristic method for the set covering problem", Oper. Res. 47 (1999), 730–743.

[14] Beasley, J. E.: "A Lagrangian heuristic for set covering problems", Naval Res. Logist. 37 (1990), 151–164.

[15] Beasley, J. E. and Chu, P. C.: "A genetic algorithm for the set covering problem", European J. Oper. Res. 94 (1996), 392–404.

[16] Hamdy A. Taha , "Operations research : an introduction," 5th ed, New York : Macmillan, 1992, pp 333-335.

[17] Hamdy A. Taha, "A Balasian-based algorithm for zero-one polynomial programming.", Management Science,Vol 18, 1972, pp. B328-B343.

[18] R. Fortet. "Applications de l' algèbre de boole en recherche opérationelle", Revue Française de Recherche Opérationelle, 4:17–26, 1960.

ประวัติผู้เขียน

ชื่อ-นามสกุล นายกมลณัฐ กางการ (Kamonnat Kangkan)

ชื่อเดิม นายสรณัญช์ กางการ (Soranan Kangkan)

วันเดือนปีเกิด 9 พฤศจิกายน 2524

ที่อยู่ 294 หมู่ 1 ต.แวง อ.โพนทอง จ.ร้อยเอ็ด

ประวัติการศึกษา

พ.ศ. 2537	จบการศึกษาระดับประถมศึกษา จากโรงเรียนเมืองโพนทอง อ.โพนทอง จ.ร้อยเอ็ด
พ.ศ. 2540	จบการศึกษาระดับมัธยมศึกษาตอนต้น จากโรงเรียนโพนทองพัฒนาวิทยา อ.โพนทอง จ.ร้อยเอ็ด
พ.ศ. 2543	จบการศึกษาระดับมัธยมศึกษาตอนปลาย จากโรงเรียนสาธิต (ศึกษาศาสตร์) มหาวิทยาลัยขอนแก่น อ.เมือง จ.ขอนแก่น
พ.ศ. 2547	จบการศึกษาวิศวกรรมศาสตรบัณฑิต (วิศวกรรมสารสนเทศ) จากคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ลาดกระบัง กรุงเทพฯ